Note to readers:
Please ignore these
sidenotes; they're just
hints to myself for
preparing the index,
and they're often flaky!

KNUTH

THE ART OF COMPUTER PROGRAMMING

VOLUME 4 PRE-FASCICLE 5C

DANCING LINKS

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY

September 11, 2019

Internet
Stanford GraphBase
MMIX

Internet page `http://www-cs-faculty.stanford.edu/~knuth/taocp.html` contains current information about this book and related books.

See also `http://www-cs-faculty.stanford.edu/~knuth/sgb.html` for information about *The Stanford GraphBase*, including downloadable software for dealing with the graphs used in many of the examples in Chapter 7.

See also `http://www-cs-faculty.stanford.edu/~knuth/mmixware.html` for downloadable software to simulate the MMIX computer.

See also `http://www-cs-faculty.stanford.edu/~knuth/programs.html` for various experimental programs that I wrote while writing this material (and some data files).

Zeroth printing (revision 3), 11 September 2019

SHORT
Internet

PREFACE

With this issue we have terminated the section "Short Notes."
... It has never been "crystal clear" why a Contribution cannot be short,
just as it has occasionally been verified in these pages
that a Short Note might be long.

— ROBERT A. SHORT, *IEEE Transactions on Computers* (1973)

THIS BOOKLET contains draft material that I'm circulating to experts in the field, in hopes that they can help remove its most egregious errors before too many other people see it. I am also, however, posting it on the Internet for courageous and/or random readers who don't mind the risk of reading a few pages that have not yet reached a very mature state. *Beware:* This material has not yet been proofread as thoroughly as the manuscripts of Volumes 1, 2, 3, and 4A were at the time of their first printings. And alas, those carefully-checked volumes were subsequently found to contain thousands of mistakes.

Given this caveat, I hope that my errors this time will not be so numerous and/or obtrusive that you will be discouraged from reading the material carefully. I did try to make the text both interesting and authoritative, as far as it goes. But the field is vast; I cannot hope to have surrounded it enough to corral it completely. So I beg you to let me know about any deficiencies that you discover.

To put the material in context, this portion of fascicle 5 previews Section 7.2.2.1 of *The Art of Computer Programming*, entitled "Dancing links." It develops an important data structure technique that is suitable for *backtrack programming*, which is the main focus of Section 7.2.2. Several subsections (7.2.2.2, 7.2.2.3, etc.) will follow.

* * *

The explosion of research in combinatorial algorithms since the 1970s has meant that I cannot hope to be aware of all the important ideas in this field. I've tried my best to get the story right, yet I fear that in many respects I'm woefully ignorant. So I beg expert readers to steer me in appropriate directions.

Please look, for example, at the exercises that I've classed as research problems (rated with difficulty level 46 or higher), namely exercises 37, 56, 73, 183, 190, 351, 417; I've also implicitly mentioned or posed additional unsolved questions in the answers to exercises 38, 39(a), 64, 67, 108, 110, 144, 152, 156, 173, 215, 228, 247, 260, 262, 309, 323, 331, 346, 358, 369, 372, 374, 381, 408, 411, 414, 416, 425, 428, 445. Are those problems still open? Please inform me if you know of a solution to any of these intriguing questions. And of course if no solution is known today but you do make progress on any of them in the future, I hope you'll let me know.

I urgently need your help also with respect to some exercises that I made up as I was preparing this material. I certainly don't like to receive credit for things that have already been published by others, and most of these results are quite natural "fruits" that were just waiting to be "plucked." Therefore please tell me if you know who deserves to be credited, with respect to the ideas found in exercises 1, 2, 4, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 32, 34, 36, 54, 62, 66, 68, 69, 71, 83, 84, 85, 94, 95, 96, 97, 98, 99, 100, 109, 114, 117, 118, 125, 137(b,c,d), 140, 141(b), 142, 144, 145, 146, 147, 148, 152, 157, 158, 160, 162, 169, 170, 171, 175, 176, 177, 178, 184, 185, 187, 188, 191, 197, 201, 207, 212, 217, 249, 273, 275, 278, 280, 289, 294, 300, 301, 303(d), 307, 308, 309, 310, 311, 313, 317, 319, 320, 321, 328, 329, 330, 331, 339, 342, 346(a,b,c), 362, 363, 364, 365, 366, 367, 370, 371, 373, 376, 381, 383, 384, 386, 387, 392, 393, 394, 395, 401, 405, 411, 413, 424, 427, 429, 434, 438, 442, 443, 444, 445, 446, 448, 449. Furthermore I've cited unpublished work of Pemantle, Pittel, Randall, and Katz in answers 39, 262, and 394. Have any of those results ever appeared in print, to your knowledge?

Exercise 313 mentions a fascinating 12-iamond that deserves a standard name, but I've never seen it before. Please let me know if it is discussed somewhere in the literature — and if it has a better name than a "whirl," which is what I currently like to call it.

* * *

Special thanks are due to George Jelliss for answering dozens of historical queries, as well as to Nikolai Beluhov, James Dalgety, Matthias Engelhardt, Ira Gessel, Wei-Hwa Huang, Helmut Postl, Will Shortz, George Sicherman, Richard Stanley, and ... for their detailed comments on my early attempts at exposition. And I want to thank numerous other correspondents who have contributed crucial corrections.

* * *

I happily offer a "finder's fee" of $2.56 for each error in this draft when it is first reported to me, whether that error be typographical, technical, or historical. The same reward holds for items that I forgot to put in the index. And valuable suggestions for improvements to the text are worth 32¢ each. (Furthermore, if you find a better solution to an exercise, I'll actually do my best to give you immortal glory, by publishing your name in the eventual book:–)

In the preface to Volume 4B I plan to introduce the abbreviation *FGbook* for my book *Selected Papers on Fun & Games* (Stanford: CSLI Publications, 2011),

Pemantle
Pittel
Randall
Katz
whirl
Jelliss
Beluhov
Dalgety
Engelhardt
Gessel
Huang
Postl
Shortz
Sicherman
Stanley
FGbook

because I will be making frequent reference to it in connection with recreational problems.

Many of the concepts in this book are best illustrated by *puzzles*, because puzzles nicely encapsulate the principles that arise in problems of all kinds that we wish to solve. Indeed, many of the exercises ask the reader not only to *solve* a given puzzle, but also to design a *new* puzzle: Puzzle creation is itself an instructive art form. In such cases the answers to the puzzles in the answers can be found in Appendix E at the back of the book.

Cross references to yet-unwritten material sometimes appear as '00'; this impossible value is a placeholder for the actual numbers to be supplied later.

Happy reading!

Stanford, California D. E. K.
27 June 2019

puzzles
answers to the puzzles in the answers
Knuth

MPR
English words
Internet

Part of the Preface to Volume 4B

During the years that I've been preparing Volume 4, I've often run across basic techniques of probability theory that I would have put into Section 1.2 of Volume 1 if I'd been clairvoyant enough to anticipate them in the 1960s. Finally I realized that I ought to collect most of them together in one place, near the beginning of Volume 4B, because the story of these developments is too interesting to be broken up into little pieces scattered here and there.

Therefore this volume begins with a special section entitled "Mathematical Preliminaries Redux," and future sections use the abbreviation 'MPR' to refer to its equations and its exercises.

* * *

Several exercises involve the lists of English words that I've used in preparing examples. You'll need the data from

`http://www-cs-faculty.stanford.edu/~knuth/wordlists.tgz`

if you have the courage to work those exercises.

BARRIS
FIELDS
undoing
doubly linked list
LLINK
RLINK
garbage collection
List head
undeletion
delete

What a dance
do they do
Lordy, how I'm tellin' you!
— HARRY BARRIS, *Mississippi Mud* (1927)

Don't lose your confidence if you slip,
Be grateful for a pleasant trip,
And pick yourself up, dust yourself off, start all over again.
— DOROTHY FIELDS, *Pick Yourself Up* (1936)

7.2.2.1. Dancing links. One of the chief characteristics of backtrack algorithms is the fact that they usually need to *undo* everything that they *do* to their data structures. In this section we'll study some extremely simple link-manipulation techniques that modify and unmodify the structures with ease. We'll also see that these ideas have many, many practical applications.

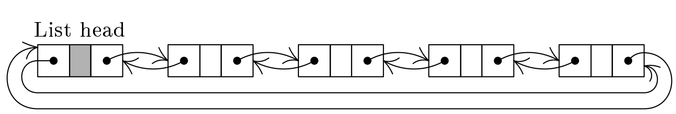
Suppose we have a doubly linked list, in which each node `X` has a predecessor and successor denoted respectively by `LLINK(X)` and `RLINK(X)`. Then we know that it's easy to delete `X` from the list, by setting

$$\texttt{RLINK(LLINK(X))} \leftarrow \texttt{RLINK(X)}, \qquad \texttt{LLINK(RLINK(X))} \leftarrow \texttt{LLINK(X)}. \tag{1}$$

At this point the conventional wisdom is to recycle node `X`, making it available for reuse in another list. We might also want to tidy things up by clearing `LLINK(X)` and `RLINK(X)` to Λ, so that stray pointers to nodes that are still active cannot lead to trouble. (See, for example, Eq. 2.2.5–(4), which is the same as (1) except that it also says '$\texttt{AVAIL} \Leftarrow \texttt{X}$'.) By contrast, the dancing-links trick resists any urge to do garbage collection. *In a backtrack application, we're better off leaving* `LLINK(X)` *and* `RLINK(X)` *unchanged.* Then we can undo operation (1) by simply setting

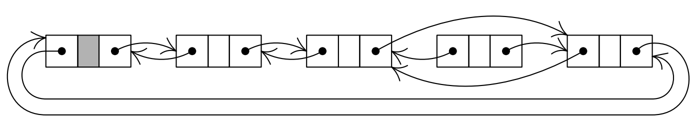
$$\texttt{RLINK(LLINK(X))} \leftarrow \texttt{X}, \qquad \texttt{LLINK(RLINK(X))} \leftarrow \texttt{X}. \tag{2}$$

For example, we might have a 4-element list, as in 2.2.5–(2):

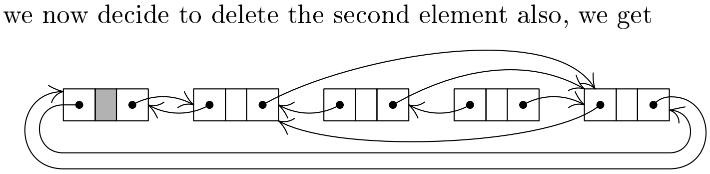


. (3)

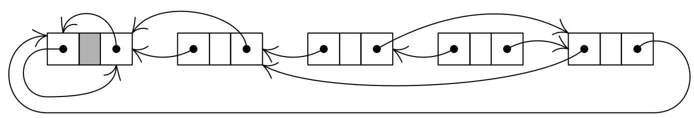
If we use (1) to delete the third element, (3) becomes


.

And if we now decide to delete the second element also, we get


.

Subsequent deletion of the final element, then the first, will leave us with this:



(4)

undelete
exact covering–
set partitioning, see exact covering
0s and 1s
sparse
options
items

The list is now empty, and its links have become rather tangled. (See exercise 1.) But we know that if we proceed to backtrack at this point, using (2) to undelete elements 1, 4, 2, and 3 in that order, we will magically restore the initial state (3). The choreography that underlies the motions of these pointers is fun to watch, and it explains the name "dancing links."

Exact cover problems. We will be seeing many examples where links dance happily and efficiently, as we study more and more examples of backtracking. The beauty of the idea can perhaps be seen most naturally in an important class of problems known as *exact covering*: We're given an $M \times N$ matrix A of 0s and 1s, and the problem is to find a subset of rows whose sum is exactly 1 in every column. For example, consider the 6×7 matrix

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}. \qquad (5)$$

Each row of A corresponds to a subset of a 7-element universe. A moment's thought shows that there's only one way to cover all seven of these columns with disjoint rows, namely by choosing rows 1, 4, and 5. We want to teach a computer how to solve such problems, when there are many, many rows and many columns.

Matrices of 0s and 1s appear frequently in combinatorial problems, and they help us to understand the relations between problems that are essentially the same although they appear to be different (see exercise 5). But inside a computer, we rarely want to represent an exact cover problem explicitly as a two-dimensional array of bits, because the matrix tends to be extremely sparse: There normally are very few 1s. Thus we'll use a different representation, essentially with one node in our data structure for each 1 in the matrix.

Furthermore, we won't even talk about rows and columns! Some of the exact cover problems we deal with already involve concepts that are called "rows" and "columns" in their own areas of application. Instead we will speak of *options* and *items*: Each option is a set of items; and *the goal of an exact cover problem is to find disjoint options that cover all the items.*

For example, we shall regard (5) as the specification of six options involving seven items. Let's name the items a, b, c, d, e, f, g; then the options are

$$\text{`}c\ e\text{'};\qquad \text{`}a\ d\ g\text{'};\qquad \text{`}b\ c\ f\text{'};\qquad \text{`}a\ d\ f\text{'};\qquad \text{`}b\ g\text{'};\qquad \text{`}d\ e\ g\text{'}. \qquad (6)$$

The first, fourth, and fifth options give us each item exactly once.

recursive algorithm
covering an item
active list of items
`LLINK`
`RLINK`
`ULINK`
`DLINK`
`TOP`

One of the nicest things about exact cover problems is that every tentative choice we make leaves us with a residual exact cover problem that is smaller — often substantially smaller. For example, suppose we try to cover item a in (6) by choosing the option '$a\ d\ g$': The residual problem has only two options,

$$\text{`}c\ e\text{'} \qquad \text{and} \qquad \text{`}b\ c\ f\text{'}, \tag{7}$$

because the other four involve the already-covered items. Now it's easy to see that (7) has no solution; therefore we can *remove* option '$a\ d\ g$' from (6). That leaves us with only one option for item a, namely '$a\ d\ f$'. And its residual,

$$\text{`}c\ e\text{'} \qquad \text{and} \qquad \text{`}b\ g\text{'}, \tag{8}$$

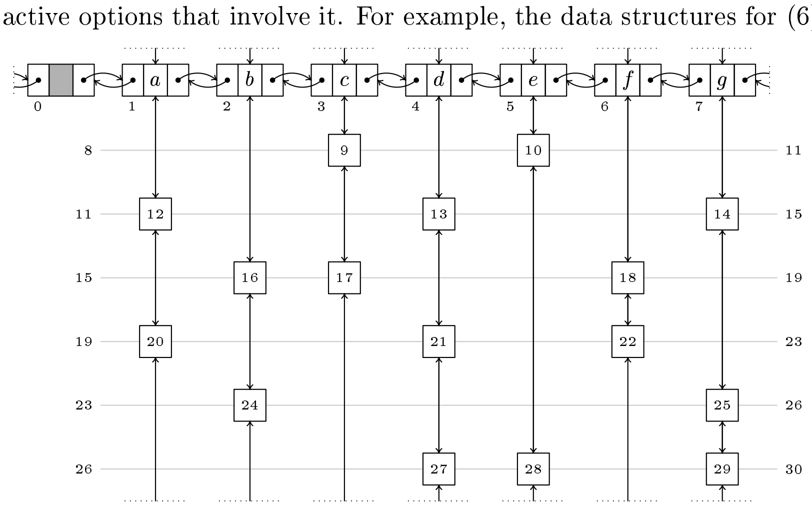
gives us the solution we were looking for.

Thus we're led naturally to a recursive algorithm that's based on the primitive operation of "covering an item": *To cover item i, we delete all of the options that involve i, from our database of currently active options, and we also delete i from the list of items that need to be covered.* The algorithm is simply this:

- Select an item i that needs to be covered; but terminate successfully if none are left (we've found a solution).
- If no active options involve i, terminate unsuccessfully (there's no solution). Otherwise cover item i. (9)
- For each just-deleted option O that involves i, one at a time, cover each item $j \neq i$ in O, and solve the residual problem.

(Everything that's covered must later be uncovered, of course, as we'll see.)

Interesting details arise when we flesh out this algorithm and look at appropriate low-level mechanisms. There's a doubly linked "horizontal" list of all items that need to be covered; and each item also has its own "vertical" list of all the active options that involve it. For example, the data structures for (6) are:



(10)

(In this diagram, doubly linked pointers "wrap around" at the dotted lines.) The horizontal list has `LLINK` and `RLINK` pointers; the vertical lists have `ULINK` and `DLINK`. Nodes of each vertical list also point to their list header via `TOP` fields.

spacer nodes
LEN

The top row of diagram (10) shows the initial state of the horizontal item list and its associated vertical headers. The other rows illustrate the six options of (6), which are represented by sixteen nodes within the vertical lists. Those options implicitly form horizontal lists, indicated by light gray lines; but their nodes *don't* need to be linked together with pointers, because the option lists don't change. We can therefore save time and space by allocating them sequentially. On the other hand, our algorithm does require an ability to traverse each option cyclically, in both directions. Therefore we insert *spacer nodes* between options. A spacer node x is identified by the condition $\texttt{TOP}(x) \le 0$; it also has

$$\begin{aligned}\texttt{ULINK}(x) &= \text{address of the first node in the option before } x;\\ \texttt{DLINK}(x) &= \text{address of the last node in the option after } x.\end{aligned} \qquad (11)$$

These conventions lead to the internal memory layout shown in Table 1. First come the records for individual items; those records have NAME, LLINK, and RLINK fields, where NAME is used in printouts. Then come the nodes, which have TOP, ULINK, and DLINK fields. The TOP field is, however, called LEN in the nodes that serve as item headers, because Algorithm X below uses those fields to store the lengths of the item lists. Nodes 8, 11, 15, 19, 23, 26, and 30 in this example are the spacers. Fields marked '—' are unused.

Table 1

THE INITIAL CONTENTS OF MEMORY CORRESPONDING TO (6) AND (10)

i:	0	1	2	3	4	5	6	7
NAME(i):	—	a	b	c	d	e	f	g
LLINK(i):	7	0	1	2	3	4	5	6
RLINK(i):	1	2	3	4	5	6	7	0
x:	0	1	2	3	4	5	6	7
LEN(x):	—	2	2	2	3	2	2	3
ULINK(x):	—	20	24	17	27	28	22	29
DLINK(x):	—	12	16	9	13	10	18	14
x:	8	9	10	11	12	13	14	15
TOP(x):	0	3	5	−1	1	4	7	−2
ULINK(x):	—	3	5	9	1	4	7	12
DLINK(x):	10	17	28	14	20	21	25	18
x:	16	17	18	19	20	21	22	23
TOP(x):	2	3	6	−3	1	4	6	−4
ULINK(x):	2	9	6	16	12	13	18	20
DLINK(x):	24	3	22	22	1	27	6	25
x:	24	25	26	27	28	29	30	
TOP(x):	2	7	−5	4	5	7	−6	
ULINK(x):	16	14	24	21	10	25	27	
DLINK(x):	2	29	29	4	5	7	—	

OK, we're ready now to spell out precisely what happens inside the computer's memory when Algorithm X wants to cover a given item i:

$$\text{cover}(i) = \begin{cases}\text{Set } p \leftarrow \texttt{DLINK}(i)\text{. (Here } p,\ l,\text{ and } r \text{ are local variables.)}\\ \text{While } p \ne i,\ \text{hide}(p),\text{ then set } p \leftarrow \texttt{DLINK}(p) \text{ and repeat.}\\ \text{Set } l \leftarrow \texttt{LLINK}(i),\ r \leftarrow \texttt{RLINK}(i),\\ \qquad \texttt{RLINK}(l) \leftarrow r,\ \texttt{LLINK}(r) \leftarrow l.\end{cases} \qquad (12)$$

$$\text{hide}(p) = \begin{cases} \text{Set } q \leftarrow p+1, \text{ and repeat the following while } q \neq p\text{:} \\ \quad \text{Set } x \leftarrow \texttt{TOP}(q),\ u \leftarrow \texttt{ULINK}(q),\ d \leftarrow \texttt{DLINK}(q); \\ \quad \text{if } x \le 0, \text{ set } q \leftarrow u \ (q \text{ was a spacer}); \\ \quad \text{otherwise set } \texttt{DLINK}(u) \leftarrow d,\ \texttt{ULINK}(d) \leftarrow u, \\ \qquad \texttt{LEN}(x) \leftarrow \texttt{LEN}(x) - 1,\ q \leftarrow q+1. \end{cases} \tag{13}$$

And — here's the point — those operations can readily be undone:

$$\text{uncover}(i) = \begin{cases} \text{Set } l \leftarrow \texttt{LLINK}(i),\ r \leftarrow \texttt{RLINK}(i), \\ \qquad \texttt{RLINK}(l) \leftarrow i,\ \texttt{LLINK}(r) \leftarrow i. \\ \text{Set } p \leftarrow \texttt{ULINK}(i). \\ \text{While } p \neq i, \text{ unhide}(p), \text{ then set } p \leftarrow \texttt{ULINK}(p) \text{ and repeat.} \end{cases} \tag{14}$$

$$\text{unhide}(p) = \begin{cases} \text{Set } q \leftarrow p-1, \text{ and repeat the following while } q \neq p\text{:} \\ \quad \text{Set } x \leftarrow \texttt{TOP}(q),\ u \leftarrow \texttt{ULINK}(q),\ d \leftarrow \texttt{DLINK}(q); \\ \quad \text{if } x \le 0, \text{ set } q \leftarrow d \ (q \text{ was a spacer}); \\ \quad \text{otherwise set } \texttt{DLINK}(u) \leftarrow q,\ \texttt{ULINK}(d) \leftarrow q, \\ \qquad \texttt{LEN}(x) \leftarrow \texttt{LEN}(x) + 1,\ q \leftarrow q-1. \end{cases} \tag{15}$$

We're careful here to do everything backwards, using operation (2) inside (14) and (15) to undelete in precisely the reverse order of the way that we'd previously used operation (1) inside (12) and (13) to delete. Furthermore, we're able to do this in place, without copying, by waltzing through the data structure at the same time as we're modifying it.

uncovering an item
hiding an option
unhiding an option
in situ changes
backtracking
Z
MRV

Algorithm X (*Exact cover via dancing links*). This algorithm visits all solutions to a given exact cover problem, using the data structures just described. It also maintains a list $x_0, x_1, \ldots, x_T$ of node pointers for backtracking, where T is large enough to accommodate one entry for each option in a solution.

X1. [Initialize.] Set the problem up in memory, as in Table 1. (See exercise 8.) Also set N to the number of items, Z to the last spacer address, and $l \leftarrow 0$.

X2. [Enter level l.] If $\texttt{RLINK}(0) = 0$ (hence all items have been covered), visit the solution that is specified by $x_0x_1 \ldots x_{l-1}$ and go to X8. (See exercise 13.)

X3. [Choose i.] At this point the items $i_1, \ldots, i_t$ still need to be covered, where $i_1 = \texttt{RLINK}(0)$, $i_{j+1} = \texttt{RLINK}(i_j)$, $\texttt{RLINK}(i_t) = 0$. Choose one of them, and call it i. (The MRV heuristic of exercise 9 often works well in practice.)

X4. [Cover i.] Cover item i using (12), and set $x_l \leftarrow \texttt{DLINK}(i)$.

X5. [Try x_l.] If $x_l = i$, go to X7 (we've tried all options for i). Otherwise set $p \leftarrow x_l + 1$, and do the following while $p \neq x_l$: Set $j \leftarrow \texttt{TOP}(p)$; if $j \le 0$, set $p \leftarrow \texttt{ULINK}(p)$; otherwise cover($j$) and set $p \leftarrow p + 1$. (This covers the items $\neq i$ in the option that contains x_l.) Set $l \leftarrow l + 1$ and return to X2.

X6. [Try again.] Set $p \leftarrow x_l - 1$, and do the following while $p \neq x_l$: Set $j \leftarrow \texttt{TOP}(p)$; if $j \le 0$, set $p \leftarrow \texttt{DLINK}(p)$; otherwise uncover($j$) and set $p \leftarrow p-1$. (This uncovers the items $\neq i$ in the option that contains x_l, using the reverse of the order in X5.) Set $i \leftarrow \texttt{TOP}(x_l)$, $x_l \leftarrow \texttt{DLINK}(x_l)$, and return to X5.

X7. [Backtrack.] Uncover item i using (14).

X8. [Leave level l.] Terminate if $l = 0$. Otherwise set $l \leftarrow l - 1$ and go to X6. ▮

Langford pairs
symmetry
n queens

The reader is strongly advised to work exercise 11 now — yes, now, really! — in order to experience the dance steps of this instructive algorithm. When the procedure terminates, all of the links will be restored to their original settings.

We're going to see lots of applications of Algorithm X, and similar algorithms, in this section. Let's begin by fulfilling a promise that was made on page 2 of Chapter 7, namely to solve the problem of *Langford pairs* efficiently by means of dancing links.

The task is to put $2n$ numbers $\{1, 1, 2, 2, \ldots, n, n\}$ into $2n$ slots $s_1 s_2 \ldots s_{2n}$, in such a way that exactly i numbers fall between the two occurrences of i. It illustrates exact covering nicely, because we can regard the n values of i and the $2n$ slots s_j as items to be covered. The allowable options for placing the i's are then

$$\text{`}i\ s_j\ s_k\text{'}, \qquad \text{for } 1 \le j < k \le 2n, \quad k = i + j + 1, \quad 1 \le i \le n; \qquad (16)$$

for example, when $n = 3$ they're

$$\text{`}1\, s_1\, s_3\text{'}, \text{`}1\, s_2\, s_4\text{'}, \text{`}1\, s_3\, s_5\text{'}, \text{`}1\, s_4\, s_6\text{'}, \text{`}2\, s_1\, s_4\text{'}, \text{`}2\, s_2\, s_5\text{'}, \text{`}2\, s_3\, s_6\text{'}, \text{`}3\, s_1\, s_5\text{'}, \text{`}3\, s_2\, s_6\text{'}. \quad (17)$$

An exact covering of all items is equivalent to placing each pair and filling each slot. Algorithm X quickly determines that (17) has just two solutions,

$$\text{`}3\, s_1\, s_5\text{'}, \text{`}2\, s_3\, s_6\text{'}, \text{`}1\, s_2\, s_4\text{'} \qquad \text{and} \qquad \text{`}3\, s_2\, s_6\text{'}, \text{`}2\, s_1\, s_4\text{'}, \text{`}1\, s_3\, s_5\text{'},$$

corresponding to the placements 3 1 2 1 3 2 and 2 3 1 2 1 3. Notice that those placements are mirror images of each other; exercise 15 shows how to save a factor of 2 and eliminate such symmetry, by omitting some of the options in (16).

With that change, there are exactly 326,721,800 solutions when $n = 16$, and Algorithm X needs about 1.13 trillion memory accesses to visit them all. That's pretty good — it amounts to roughly 3460 mems per solution, as the links whirl.

Of course, we've already looked at a backtrack procedure that's specifically tuned to the Langford problem, namely Algorithm 7.2.2L near the beginning of Section 7.2.2. With the enhancement of exercise 7.2.2–21, that one handles the case $n = 16$ somewhat faster, finishing after about 400 billion mems. But Algorithm X can be pleased that its general-purpose machinery isn't way behind the best custom-tailored method.

Secondary items. Can the classical problem of n queens also be formulated as an exact cover problem? Yes, of course! But the construction isn't quite so obvious. Instead of setting the problem up as we did in 7.2.2–(3), where we chose a queen placement for each row of the board, we shall now allow both rows and columns to participate equally when making the necessary choices.

There are n^2 options for placing queens, and we want exactly one queen in every row and exactly one in every column. Furthermore, we want *at most* one queen in every diagonal. More precisely, if x_{ij} is the binary variable that signifies a queen in row i and column j, we want

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } 1 \le j \le n; \qquad \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } 1 \le i \le n; \qquad (18)$$

$$\sum\{x_{ij} \mid 1 \le i, j \le n,\ i + j = s\} \le 1 \quad \text{for } 1 < s \le 2n; \tag{19}$$

$$\sum\{x_{ij} \mid 1 \le i, j \le n,\ i - j = d\} \le 1 \quad \text{for } -n < d < n. \tag{20}$$

slack variables
primary
secondary
N_1

The inequalities in (19) and (20) can be changed to equalities by introducing "slack variables" $u_2, \ldots, u_{2n}, v_{-n+1}, \ldots, v_{n-1}$, each of which is 0 or 1:

$$\sum\{x_{ij} \mid 1 \le i, j \le n,\ i + j = s\} + u_s = 1 \quad \text{for } 1 < s \le 2n; \tag{21}$$

$$\sum\{x_{ij} \mid 1 \le i, j \le n,\ i - j = d\} + v_d = 1 \quad \text{for } -n < d < n. \tag{22}$$

Thus we've shown that the problem of n nonattacking queens is equivalent to the problem of finding $n^2 + 4n - 2$ binary variables x_{ij}, u_s, v_d for which certain subsets of the variables sum to 1, as specified in (18), (21), and (22).

And that is essentially an exact cover problem, whose options correspond to the binary variables and whose items correspond to the subsets. The items are r_i, c_j, a_s, and b_d, representing respectively row i, column j, upward diagonal s, and downward diagonal d. The options are '$r_i\ c_j\ a_{i+j}\ b_{i-j}$' for queen placements, together with trivial options 'a_s' and 'b_d' to take up any slack.

For example, when $n = 4$ the n^2 placement options are

$$\begin{array}{llll}
`r_1\ c_1\ a_2\ b_0\text{'}; & `r_2\ c_1\ a_3\ b_1\text{'}; & `r_3\ c_1\ a_4\ b_2\text{'}; & `r_4\ c_1\ a_5\ b_3\text{'}; \\
`r_1\ c_2\ a_3\ b_{-1}\text{'}; & `r_2\ c_2\ a_4\ b_0\text{'}; & `r_3\ c_2\ a_5\ b_1\text{'}; & `r_4\ c_2\ a_6\ b_2\text{'}; \\
`r_1\ c_3\ a_4\ b_{-2}\text{'}; & `r_2\ c_3\ a_5\ b_{-1}\text{'}; & `r_3\ c_3\ a_6\ b_0\text{'}; & `r_4\ c_3\ a_7\ b_1\text{'}; \\
`r_1\ c_4\ a_5\ b_{-3}\text{'}; & `r_2\ c_4\ a_6\ b_{-2}\text{'}; & `r_3\ c_4\ a_7\ b_{-1}\text{'}; & `r_4\ c_4\ a_8\ b_0\text{'};
\end{array} \tag{23}$$

and the $4n - 2$ slack options (which contain just one item each) are

$$`a_2\text{'};\ `a_3\text{'};\ `a_4\text{'};\ `a_5\text{'};\ `a_6\text{'};\ `a_7\text{'};\ `a_8\text{'};\ `b_{-3}\text{'};\ `b_{-2}\text{'};\ `b_{-1}\text{'};\ `b_0\text{'};\ `b_1\text{'};\ `b_2\text{'};\ `b_3\text{'}. \tag{24}$$

Algorithm X will solve this small problem easily, although its treatment of the slacks is somewhat awkward (see exercise 16).

A closer look shows, however, that a slight change to Algorithm X will allow us to avoid slack options entirely! Let's divide the items of an exact cover problem into two groups: *primary* items, which must be covered *exactly* once, and *secondary* items, which must be covered *at most* once. If we simply modify step X1 so that only the primary items appear in the active list, everything will work like a charm. (Think about it.) In fact, the necessary changes to step X1 already appear in the answer to exercise 8.

Secondary items turn out to be extremely useful in applications. So let's redefine the exact cover problem, taking them into account: Henceforth we shall assume that an exact cover problem involves N distinct items, of which N_1 are primary and $N_2 = N - N_1$ are secondary. It is defined by a family of options, each of which is a subset of the items. *Every option must include at least one primary item.* The task is to find all subsets of the options that (i) contain every primary item exactly once, and (ii) contain every secondary item at most once.

(Options that are purely secondary are excluded from this new definition, because they will never be chosen by Algorithm X as we've refined it. If for some reason you don't like that rule, you can always go back to the idea of slack options. Exercise 19 discusses another interesting alternative.)

order of primary items
organ-pipe order
bitwise operations
diagonals
symmetry
pairwise ordering

The order in which primary items appear in Algorithm X's active list can have a significant effect on the running time, because the implementation of step X3 in exercise 9 selects the *first* item of minimum length. For example, if we consider the primary items of the n queens problem in the natural order r_1, c_1, r_2, c_2, …, r_n, c_n, queens tend to be placed at the top and left before we try to place them at the bottom and right. By contrast, if we use the "organ-pipe order" $r_{\lfloor n/2\rfloor+1}$, $c_{\lfloor n/2\rfloor+1}$, $r_{\lfloor n/2\rfloor}$, $c_{\lfloor n/2\rfloor}$, $r_{\lfloor n/2\rfloor+2}$, $c_{\lfloor n/2\rfloor+2}$, $r_{\lfloor n/2\rfloor-1}$, $c_{\lfloor n/2\rfloor-1}$, …, (r_1 or r_n), (c_1 or c_n), the queens are placed first in the center, where they prune the remaining possibilities more effectively. For example, the time needed to find all 14772512 ways to place 16 queens is 76 Gμ with the natural order, but only 40 Gμ with the organ-pipe order.

These running times can be compared with 9 Gμ, which we obtained using efficient bitwise operations with Algorithm 7.2.2W. Although that algorithm was specially tuned, the general-purpose dancing links technique runs only five times slower. Furthermore, the setup we've used here allows us to solve other problems that wouldn't be anywhere near as easy with ordinary backtrack. For example, we can limit the solutions to those that contain queens on each of the $2+4l$ longest diagonals, simply by regarding a_s and b_d as primary items instead of secondary, for $|n+1-s|\le l$ and $|d|\le l$. (There are 18048 such solutions when $n=16$ and $l=4$, found with organ-pipe order in 2.7 Gμ.)

We can also use secondary items to remove symmetry, so that most of the solutions are found only once instead of eight times (see exercises 22 and 23). The central idea is to use a *pairwise ordering* trick that works also in many other situations. Consider the following family of $2m$ options:

$$\alpha_j = \text{`}a\ x_0\ \ldots\ x_{j-1}\text{'} \quad\text{and}\quad \beta_j = \text{`}b\ x_j\text{'} \qquad \text{for } 0\le j<m, \tag{25}$$

where a and b are primary while x_0, x_1, …, x_{m-1} are secondary. For example, when $m=4$ the options are

$$\begin{array}{ll}
\alpha_0 = \text{`}a\text{'}; & \beta_0 = \text{`}b\ x_0\text{'};\\
\alpha_1 = \text{`}a\ x_0\text{'}; & \beta_1 = \text{`}b\ x_1\text{'};\\
\alpha_2 = \text{`}a\ x_0\ x_1\text{'}; & \beta_2 = \text{`}b\ x_2\text{'};\\
\alpha_3 = \text{`}a\ x_0\ x_1\ x_2\text{'}; & \beta_3 = \text{`}b\ x_3\text{'}.
\end{array}$$

It's not hard to see that there are exactly $\binom{m+1}{2}$ ways to cover both a and b, namely to choose α_j and β_k with $0\le j\le k<m$. For if we choose α_j, the secondary items x_0 through x_{j-1} knock out the options β_0, …, β_{j-1}.

This construction involves a total of $\binom{m+1}{2}$ entries in the α options and $2m$ entries in the β options. But exercise 20 shows that it's possible to achieve pairwise ordering with only $O(m\log m)$ entries in both α's and β's. For example, when $m=4$ it produces the following elegant pattern:

$$\begin{array}{ll}
\alpha_0 = \text{`}a\text{'}; & \beta_0 = \text{`}b\ y_1\ y_2\text{'};\\
\alpha_1 = \text{`}a\ y_1\text{'}; & \beta_1 = \text{`}b\ y_2\text{'};\\
\alpha_2 = \text{`}a\ y_2\text{'}; & \beta_2 = \text{`}b\ y_3\text{'};\\
\alpha_3 = \text{`}a\ y_3\ y_2\text{'}; & \beta_3 = \text{`}b\text{'}.
\end{array} \tag{26}$$

Progress reports. Many of the applications of Algorithm X take a long time, especially when we're using it to solve a tough problem that is breaking new ground. So we don't want to just start it up and wait with our fingers crossed, hoping that it will finish soon. We really want to watch it in action and see how it's doing. How many more hours will it probably run? Is it almost half done?

A simple amendment of step X2 will alleviate such worries. At the beginning of that step, as we enter a new node of the search tree, we can test whether the accumulated running time T has passed a certain threshold Θ, which is initially set to Δ. If $T \geq \Theta$, we print a progress report and set $\Theta \leftarrow \Theta + \Delta$. (Thus if $\Delta = \infty$, we get no reports; if $\Delta = 0$, we see a report at each node.)

The author's experimental program measures time in mems, so that he can obtain machine-independent results; and he usually takes $\Delta = 10\,\mathrm{G}\mu$. His program has two main ways to show progress, namely a long form and a short form. The long form gives full details about the current state of the search, based on exercise 12. For example, here's the first progress report that it displays when finding all solutions to the 16 queens problem as described above:

```
Current state (level 15):
 r8 c3 ab ba (4 of 16)
 c8 a8 bn r0 (1 of 13)
 r7 cb ai bj (7 of 10)
 r6 c4 aa bd (2 of 7)
   ⋮
 3480159 solutions, 10000000071 mems, and max level 16 so far.
```

(The computer's internal encoding for items is different from the conventions we have used; for example, '`r8 c3 ab ba`' stands for what we called 'r_9 c_4 a_{13} b_5'. The first choice at level 0 was to cover item `r8`, meaning to put a queen into row 9. The fourth of 16 options for that item has placed it in column 4. Then at level 1 we tried the first of 13 ways to cover item `c8`, meaning to put a queen into column 9. And so on. At each level, the leftmost item shown for the option being tried is the one that was chosen in step X3 for branching.)

That's the long form. The short form, which is the default, produces just one line for each state report:

```
10000000071mu: 3480159 sols, 4g 1d 7a 27 36 24 23 13 12 12 22 12 ... .19048
20000000111mu: 6604373 sols, 7g cd 6a 88 36 35 44 44 24 11 12 22 .43074
30000000052mu: 9487419 sols, bg cd 9a 68 37 35 24 13 12 12 .68205
40000000586mu: 12890124 sols, fg 6d aa 68 46 35 23 33 23 .90370
Altogether 14772512 solutions, 62296+45565990457 mems, 193032021 nodes.
```

Two-character codes are used to indicate the current position in the tree; for example, '`4g`' means branch 4 of 16, then '`1d`' means branch 1 of 13, etc. By watching these steadily increasing codes — it's fun! — we can monitor the action.

Each line in the short form ends with an estimate of how much of the tree has been examined, assuming that the search tree structure is fairly consistent. For instance, '`.19048`' means that we're roughly 19% done. If we're currently working at level l on choice c_l of t_l, this number is computed by the formula

$$\frac{c_0 - 1}{t_0} + \frac{c_1 - 1}{t_0 t_1} + \cdots + \frac{c_l - 1}{t_0 t_1 \ldots t_l} + \frac{1}{2 t_0 t_1 \ldots t_l}. \qquad (27)$$

node
search tree
backtree, see search tree
running time
threshold
progress report
author
mems
queens problem
completion ratio
ratio of completion

sudoku–
soduko: see sudoku
Garns
Number Place
Nikoli
pi, as random
0-origin indexing

Sudoku. A "sudoku square" is a 9×9 array that has been divided into 3×3 boxes and filled with the digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ in such a way that

- every row contains each of the digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ exactly once;
- every column contains each of the digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ exactly once;
- every box contains each of the digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ exactly once.

(Since there are nine cells in each row, each column, and each box, the words 'exactly once' can be replaced by 'at least once' or 'at most once', anywhere in this definition.) Here, for example, are three highly symmetrical sudoku squares:

(a)

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	3	4	5	6	7	8	9	1
5	6	7	8	9	1	2	3	4
8	9	1	2	3	4	5	6	7
3	4	5	6	7	8	9	1	2
6	7	8	9	1	2	3	4	5
9	1	2	3	4	5	6	7	8

; (b)

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	3	1	5	6	4	8	9	7
5	6	4	8	9	7	2	3	1
8	9	7	2	3	1	5	6	4
3	1	2	6	4	5	9	7	8
6	4	5	9	7	8	3	1	2
9	7	8	3	1	2	6	4	5

; (c)

1	2	3	4	5	6	7	8	9
5	6	4	8	9	7	2	3	1
9	7	8	3	1	2	6	4	5
6	4	5	9	7	8	3	1	2
7	8	9	1	2	3	4	5	6
2	3	1	5	6	4	8	9	7
8	9	7	2	3	1	5	6	4
3	1	2	6	4	5	9	7	8
4	5	6	7	8	9	1	2	3

. (28)

When the square has been only partially specified, the task of completing it — by filling in the blank cells — often turns out to be a fascinating challenge. Howard Garns used this idea as the basis for a series of puzzles that he called Number Place, first published in *Dell Pencil Puzzles & Word Games* #16 (May 1979), 6. The concept soon spread to Japan, where Nikoli Inc. gave it the name Su Doku (数独, "Unmarried Numbers") in 1984; and eventually it went viral. By the beginning of 2005, major newspapers had begun to feature daily sudoku puzzles. Today, sudoku ranks among the most popular recreations of all time.

Every sudoku puzzle corresponds to an exact cover problem that has a particularly nice form. Consider, for example, the following three instances:

(a)

		3		1				
4	1	5					9	
2		6	5			3		
5				8				9
	7		9				3	2
	3	8			4		6	
			2	6		4		3
			3					8
3	2				7	9	5	

; (b)

						3		
1			4					
						1		5
9								
					2	6		
				5	3			
	5		8					
			9				7	
	8	3					4	

; (c)

	3			1				
			4			1		
	5						9	
2						6		4
				3	5			
1								
4			6					
							5	
	9							

. (29)

(The clues in (29a) match the first 32 digits of π; but the clues in (29b) and (29c) disagree with π after awhile.) For convenience, let's number the rows, columns, and boxes from 0 to 8. Then every sudoku square $S = (s_{ij})$ corresponds naturally to the solution of a master exact cover problem whose $9 \cdot 9 \cdot 9 = 729$ options are

$$`p_{ij}\ r_{ik}\ c_{jk}\ b_{xk}' \quad \text{for } 0 \le i, j < 9,\ 1 \le k \le 9, \text{ and } x = 3\lfloor i/3 \rfloor + \lfloor j/3 \rfloor, \qquad (30)$$

and whose $4 \cdot 9 \cdot 9 = 324$ items are p_{ij}, r_{ik}, c_{jk}, b_{xk}. The reason is that option (30) is chosen with parameters (i, j, k) if and only if $s_{ij} = k$. Item p_{ij} must be covered by exactly one of the nine options that fill cell (i, j); item r_{ik} must be covered by exactly one of the nine options that put k in row i; ...; item b_{xk} must be covered by exactly one of the nine options that put k in box x. Got it?

My own motive for writing on the subject is partly to justify the appalling number of hours I have squandered solving Sudoku.
— BRIAN HAYES, in *American Scientist* (2006)

HAYES
naked single
MRV heuristic
Royle
McGuire

To find all sudoku squares that contain a given *partial* specification, we simply remove all of the items p_{ij}, r_{ik}, c_{jk}, b_{xk} that are already covered, as well as all of the options that involve any of those items. For example, (29a) leads to an exact cover problem with $4 \cdot (81 - 32) = 196$ items p_{00}, p_{01}, p_{03}, $\ldots$, r_{02}, r_{04}, r_{05}, $\ldots$, c_{01}, c_{06}, c_{07}, $\ldots$, b_{07}, b_{08}, b_{09}, $\ldots$; it has 146 options, beginning with 'p_{00} r_{07} c_{07} b_{07}' and ending with 'p_{88} r_{86} c_{86} b_{86}'. These options can be visualized by making a chart that shows every value that hasn't been ruled out:

	0	1	2	3	4	5	6	7	8
0	789	89	**3**	4678	**1**	2689	25678	2478	4567
1	**4**	**1**	**5**	678	237	2368	2678	**9**	67
2	**2**	89	**6**	**5**	479	89	**3**	1478	147
3	**5**	46	124	167	**8**	1236	17	147	**9**
4	16	**7**	14	**9**	5	156	158	**3**	**2**
5	19	**3**	**8**	17	257	**4**	157	**6**	157
6	1789	589	179	**2**	**6**	1589	**4**	17	**3**
7	1679	4569	1479	**3**	459	159	1267	127	**8**
8	**3**	**2**	14	148	4	**7**	**9**	**5**	16

(31)

The active list for item p_{00}, say, has options for values $\{7, 8, 9\}$; the active list for item r_{02} has options for columns $\{5, 6, 7\}$; the active list for item c_{01} has options for rows $\{4, 5, 6, 7\}$; and so on. (Indeed, sudoku experts tend to have charts like this in mind, implicitly or explicitly, as they work.)

Aha! Look at the lonely '5' in the middle! There's only one option for p_{44}; so we can promote that '5' to '**5**'. Hence we can also erase the other '5's that appear in row 4, column 4, or box 4. This operation is called "forcing a naked single."

And there's *another* naked single in cell $(8, 4)$. Promoting this one from '4' to '**4**' produces others in cells $(7, 4)$ and $(8, 2)$. Indeed, if the items p_{ij} have been placed first in step X1, Algorithm X will follow a merry path of forced moves that lead immediately to a complete solution of (29a), *entirely* via naked singles.

Of course sudoku puzzles aren't always this easy. For example, (29b) has only 17 clues, not 32; that makes naked singles less likely. (Puzzle (29b) comes from Gordon Royle's online collection of approximately 50,000 17-clue sudokus — all of which are essentially different, and all of which have a unique completion despite the paucity of clues. Royle's collection appears to be nearly complete: Whenever a sudoku fanatic encounters a 17-clue puzzle nowadays, that puzzle almost invariably turns out to be equivalent to one in Royle's list.)

A massive computer calculation, supervised by Gary McGuire and completed in 2012, has shown that *every uniquely solvable sudoku puzzle must*

bitwise
McGuire
Tugemann
Civario
naked singles
hidden single

contain at least 17 clues. We will see in Section 7.2.3 that exactly 5,472,730,538 nonisomorphic sudoku squares exist. McGuire's program examined each of them, and showed that comparatively few 16-clue subsets could possibly characterize it. About 16,000 subsets typically survived this initial screening; but they too were shown to fail. All this was determined in roughly 3.6 seconds per sudoku square, thanks to nontrivial and highly optimized bitwise algorithms. [See G. McGuire, B. Tugemann, and G. Civario, *Experimental Mathematics* **23** (2014), 190–217.]

The 17 clues of puzzle (29b) produce the following chart analogous to (31):

	0	1	2	3	4	5	6	7	8
0	245678	24679	2456789	12567	126789	156789	**3**	2689	246789
1	**1**	23679	256789	**4**	236789	56789	2789	2689	26789
2	234678	234679	246789	2367	236789	6789	**1**	2689	**5**
3	**9**	123467	1245678	167	14678	14678	24578	12358	123478
4	34578	1347	14578	17	14789	**2**	**6**	13589	134789
5	24678	12467	124678	167	**5**	**3**	24789	1289	124789
6	2467	**5**	124679	**8**	123467	1467	29	12369	12369
7	246	1246	1246	**9**	12346	1456	258	**7**	12368
8	267	**8**	**3**	12567	1267	1567	259	**4**	1269

(32)

This one has 307 options remaining — more than twice as many as before. Also, as we might have guessed, it has no naked singles. But it still reveals forced moves, if we look more closely! For example, column 3 contains only one instance of '3'; we can promote it to '**3**', and kill all of the other '3's in row 2 and box 1. This operation is called "forcing a hidden single."

Similarly, box 2 in (32) contains only one instance of '4'; and two other hidden singles are also present (see exercise 47). These forced moves cause other hidden singles to appear, and naked singles also arise soon. But after 16 forced promotions have been made, the low-hanging fruit is all gone:

	0	1	2	3	4	5	6	7	8
0	**5**	679	678	**2**	1678	16789	**3**	689	**4**
1	**1**	**3**	2678	**4**	678	**5**	78	2689	2679
2	24678	24679	24678	**3**	678	6789	**1**	2689	**5**
3	**9**	12467	1245678	167	14678	14678	478	12358	1237
4	**3**	147	14578	17	**9**	**2**	**6**	158	17
5	24678	12467	124678	167	**5**	**3**	478	1289	1279
6	467	**5**	**9**	**8**	1467	1467	**2**	136	136
7	246	1246	1246	**9**	**3**	146	**5**	**7**	**8**
8	67	**8**	**3**	**5**	**2**	167	**9**	**4**	16

(33)

naked pair
author
secondary

Algorithm X readily deduces (33) from (32), because it sees naked singles and hidden singles whenever an item p_{ij} or r_{ik} or c_{jk} or b_{xk} has only one remaining option, and because its data structures change easily as the links dance. But when state (33) is reached, the algorithm resorts to two-way branching, in this case looking first at case '7' of p_{16}, then backtracking later to consider case '8'.

A human sudoku expert would actually glance at (33) and notice that there's a more intelligent way to proceed, because (33) contains a "naked pair": Cells $(4,3)$ and $(4,8)$ both contain the same two choices; hence we're allowed to delete '1' and '7' wherever they appear elsewhere in row 4; and this will produce a naked '4' in column 1. Exercise 49 explores such higher-order deductions in detail.

Fancy logic that involves pairs and triples might well be preferable for earthlings, but simple backtracking works just fine for machines. In fact, Algorithm X finds the solution to (29b) after exploring a search tree with just 89 nodes, the first 16 of which led it directly to (33). (It spends about 250 kilomems initializing the data structures in step X1, then 50 more kilomems to complete the task. Much more time would have been needed if it had tried to look for complicated patterns in step X3.) Here's the solution that it discovers:

```
c33 b13 p23 r23 (1 of 1)
r13 c13 b03 p11 (1 of 1)
   . . .
c42 b72 p84 r82 (1 of 1)
p16 r18 c68 b28 (2 of 2)
b27 p18 r17 c87 (1 of 1)
   . . .
p85 r81 c51 b71 (1 of 1)
```

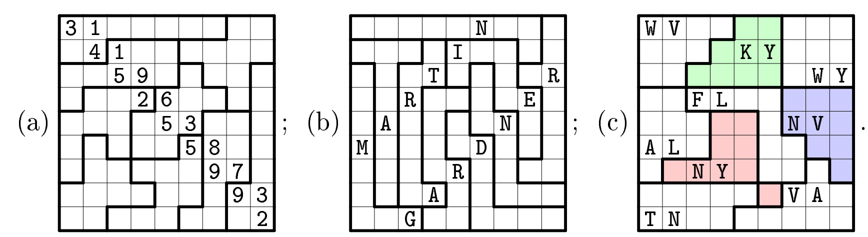
After it selects the correct value for column 6 of row 1, the rest is forced.

The dancing links method actually cruises to victory with amazing speed, on almost every known sudoku puzzle. Among several dozen typical specimens— seen by the author since 2005 in newspapers, magazines, books, and webpages worldwide, and subsequently presented to Algorithm X—roughly 70% were found to be solvable entirely by forced moves, based on naked or hidden singles, even though many of those puzzles had been rated 'diabolical' or 'fiendish' or 'torturous'! Only 10% of them led to a search tree exceeding 100 nodes, and none of the trees had more than 282 nodes. (See, however, exercise 52.)

It's interesting to consider what happens when the algorithm is weakened, so that its forced moves come only from naked singles, which are the easiest deductions for people to make. Suppose we classify the items r_{ik}, c_{jk}, and b_{xk} as *secondary*, leaving only p_{ij} as primary. (In other words, the specification will require *at most* one occurrence of each value k, in every row, every column, and every box, but it won't explicitly insist that every k should be covered.) The search tree for puzzle (29b) then grows to a whopping 41,877 nodes.

Finally, what about puzzle (29c)? That one has only 16 clues, so we know that it cannot have a unique solution. But those 16 clues specify only seven of the nine digits; they give us no way to distinguish 7 from 8. Algorithm X deduces, with a 129-node search tree, that only two solutions exist. (Of course those two are essentially the same; they're obtainable from each other by swapping $7 \leftrightarrow 8$.)

Puzzlists have invented many intriguing variations on the traditional sudoku challenge, several of which are discussed in the exercises below. Among the best are "jigsaw sudoku puzzles" (also known as "geometric sudoku," "polyomino sudoku," "squiggly sudoku," etc.), where the boxes have different shapes instead of simply being 3×3 subsquares. Consider, for example,



(34)

In puzzle (34a), which the author designed in 2017 with the help of Bob Harris, one can see for instance that there are only two places to put a '4' in the top row, because of the '4' in the next row. This puzzle is an exact cover problem just like (30), except that x is now a more complicated function of i and j. Similarly, Harris's classic puzzle (34b) [*Mathematical Wizardry for a Gardner* (2009), 55–57] asks us to put the letters {G, R, A, N, D, T, I, M, E} into each row, column, and irregularly shaped box. Again we use (30), but with the values of k running through letters instead of digits. [*Hint:* Cell $(0, 2)$ must contain 'A', because column 2 needs an 'A' somewhere.] Puzzle (34c), The United States Jigsaw Sudoku, is a masterpiece designed and posted online by Thomas Snyder in 2006. It brilliantly uses boxes in the shapes of West Virginia, Kentucky, Wyoming, Alabama, Florida, Nevada, Tennessee, New York (including Long Island), and Virginia — and its clues are postal codes! (See exercise 59.)

Jigsaw sudoku was invented by J. Mark Thompson, who began to publish such puzzles in 1996 [*GAMES World of Puzzles*, #14 (July 1996), 51, 67] under the name Latin Squares. At that time he had not yet heard about sudoku; one of the advantages of his puzzles over normal sudoku was the fact that they can be of any size, not necessarily 9×9. Thompson's first examples were 6×6.

The *solutions* to puzzles of this kind actually have an interesting prehistory: Walter Behrens, a pioneer in the applications of mathematics to agriculture, wrote an influential paper in 1956 that proposed using such patterns in empirical studies of crops that have been treated with various fertilizers [*Zeitschrift für landwirtschaftliches Versuchs- und Untersuchungswesen* **2** (1956), 176–193]. He presented dozens of designs, ranging from 4×4 to 10×10, including

(a)

1	5	4	2	3
2	4	3	5	1
3	1	2	4	5
4	3	5	1	2
5	2	1	3	4

; (b)

1	3	8	7	9	5	2
2	6	9	5	3	8	4
3	8	2	4	6	7	1
4	7	6	9	1	3	8
5	1	3	8	7	6	9
6	9	7	1	4	2	6
7	5	4	2	8	9	6
8	2	1	3	5	4	7
9	4	5	6	2	1	3

; (c)

1	5	8	6	4	3	9	7	2
2	6	9	5	7	8	1	3	4
3	7	4	1	9	2	5	6	8
4	9	2	8	6	7	3	1	5
5	3	7	4	2	1	6	8	9
6	8	1	3	5	9	2	4	7
7	2	6	9	1	4	8	5	3
8	4	5	2	3	6	7	9	1
9	1	3	7	8	5	4	2	6

. (35)

jigsaw sudoku puzzles
geometric sudoku
polyomino sudoku
squiggly sudoku
author
Harris
pi as src
Gardner
GRAND TIME puzzle
United States Jigsaw Sudoku
Snyder
Thompson
Latin Squares
Behrens
agriculture

Notice that Behrens's (35b) is actually 9×7, so its rows don't exhibit all 9 possibilities. He required only that no treatment number be repeated in any row or column. Notice also that his (35c) is actually a normal sudoku arrangement; this is the earliest known publication of what is now called a sudoku solution. Following a suggestion of F. Ragaller, Behrens called these designs "gerechte" ("equitable") latin squares or latin rectangles, because they assign neighborhood groupings to tracts of land that have been subjected to all n treatments.

All of his designs were partitions of rectangles into connected regions, each with n square cells. We'll see next that *that* idea actually turns out to have its own distinguished history of fascinating combinatorial patterns and recreations.

Polyominoes. A rookwise-connected region of n square cells is often called an *n-omino*, following a suggestion by S. W. Golomb [*AMM* **61** (1954), 675–682]. When $n = 1, 2, 3, \ldots$, Golomb's definition gives us monominoes, dominoes, trominoes, tetrominoes, pentominoes, hexominoes, and so on. In general, when n is unspecified, Golomb called such regions *polyominoes*.

We've already encountered small polyominoes, together with their relation to exact covering, in 7.1.4–(130). It's clear that a domino has only one possible shape. But there are two distinct species of trominoes, one of which is "straight" (1×3) and the other is "bent," occupying three cells of a 2×2 square. Similarly, the tetrominoes can be classified into five distinct types. (Can you draw all five, before looking at exercise 274? Tetris® players will have no trouble doing this.)

The most piquant polyominoes, however, are almost certainly the *pentominoes*, of which there are twelve. These twelve shapes have become the personal friends of millions of people, because they can be put together in so many elegant ways. Sets of pentominoes, made from finely crafted hardwoods or from brilliantly colored plastic, are readily available at reasonable cost. Every home really ought to have at least one such set — even though "virtual" pentominoes can easily be manipulated in computer apps — because there's no substitute for the strangely fascinating tactile experience of arranging these delightful physical objects by hand. Furthermore, we'll see that pentominoes have much to teach us about combinatorial computing.

If mounted on cardboard, [these pieces]
will form a source of perpetual amusement in the home.

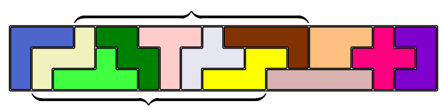
— HENRY E. DUDENEY, *The Canterbury Puzzles* (1907)

Which English nouns ending in -o *pluralize with* -s *and which with* -es*?*
If the word is still felt as somewhat alien, it takes -s,
while if it has been fully naturalized into English, it takes -es.
Thus, echoes, potatoes, tomatoes, dingoes, embargoes, *etc.*,
whereas Italian musical terms are altos, bassos, cantos, pianos, solos, *etc.*,
and there are Spanish words like tangos, armadillos, *etc.*
I once held a trademark on 'Pentomino(-es)', but I now prefer
to let these words be my contribution to the language as public domain.

— SOLOMON W. GOLOMB, letter to Donald Knuth (16 February 1994)

Behrens
sudoku solution
Ragaller
gerechte
Polyominoes
n-omino
Golomb
monominoes
dominoes
trominoes
tetrominoes
pentominoes
hexominoes
polyominoes
straight
bent
Tetris
DUDENEY
trademark
GOLOMB
Knuth

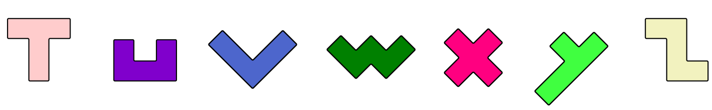
One of the first things we might try to do with twelve pieces of 5 cells each is to pack them into a rectangular box, either 6×10 or 5×12 or 4×15 or 3×20. The first three tasks are fairly easy; but a 3×20 box presents more of a challenge. Golomb posed this question in his article of 1954, without providing any answer. At that time he was unaware that Frans Hansson had already given a solution many years earlier, in an obscure publication called *The Problemist: Fairy Chess Supplement* **2**, 12 and 13 (June and August, 1935), problem 1844:



(36)

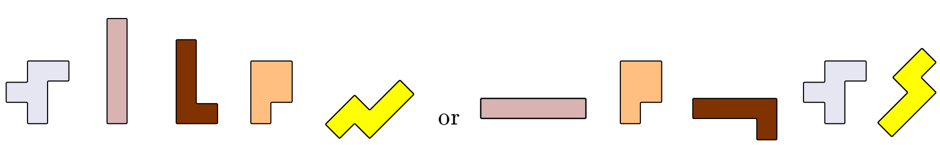
Hansson had in fact observed that the bracketed pieces "may also be rotated through two right angles, to give the only other possible solution."

This problem, and many others of a similar kind, can be formulated nicely in terms of exact covering. But before we do this, we need *names* for the individual pentomino shapes. Everybody agrees that seven of the pentominoes should be named after seven consecutive letters of the alphabet:



T U V W X Y Z

But two different systems of nomenclature have been proposed for the other five:



F I L P N or O P Q R S

(S. W. Golomb) (J. H. Conway)

where Golomb liked to think of the word 'Filipino' while Conway preferred to map the twelve pentominoes onto twelve consecutive letters. Conway's scheme tends to work better in computer programs, so we'll use it here.

The task of 3×20 pentomino packing is to arrange pentominoes in such a way that every piece name $\{\mathtt{O}, \mathtt{P}, \ldots, \mathtt{Z}\}$ is covered exactly once, and so is every cell ij for $0 \le i < 3$ and $0 \le j < 20$. Thus there are $12 + 3 \cdot 20 = 72$ items; and there's an option for each way to place an individual pentomino, namely

```
'O 00 01 02 03 04'
        ...
'O 2f 2g 2h 2i 2j'
'P 00 01 02 10 11'
        ...
'Z 0j 1h 1i 1j 2h'
```

(37)

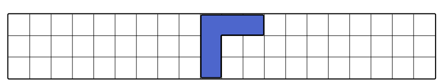
if we extend hexadecimal notation so that the "digits" $(\mathtt{a}, \mathtt{b}, \ldots, \mathtt{j})$ represent $(10, 11, \ldots, 19)$. In this list, pieces $(\mathtt{O}, \mathtt{P}, \ldots, \mathtt{Z})$ contribute respectively (48, 220,

Golomb
Hansson
Golomb
Conway
pentominoes, names of
L-pentomino, see Q-pentomino
F-pentomino, see R-pentomino
N-pentomino, see S-pentomino
packing
hexadecimal notation, extended

136, 144, 136, 72, 110, 72, 72, 18, 136, 72) options, making 1236 altogether. Exercise 266 explains how to generate all of the options for problems like this.

When Algorithm X is applied to (37), it finds eight solutions, because each of Hansson's arrangements is obtained with horizontal and/or vertical reflection. We can remove that symmetry by insisting that the V pentomino must appear in its 'Γ-like' orientation, as it does in (36), namely by removing all but 18 of its 72 options. (Do you see why? Think about it.) Without that simplification, a 32,644-node search tree finds 8 solutions in 146 megamems; with it, a 21,805-node search tree finds 2 solutions in 103 megamems.

A closer look shows that we can actually do much better. For example, one of the Γ-like options for V is '`V 09 0a 0b 19 29`', representing


; (38)

but this placement could never be used, because it asks us to pack pentominoes into the 27 cells at V's left. Many of the options for other pieces are similarly unusable, because (like (38)) they isolate a region whose area isn't a multiple of 5.

In fact, if we remove all such options, only 728 of the original 1236 potential placements remain; they include respectively (48, 156, 132, 28, 128, 16, 44, 16, 12, 4, 128, 16) placements of (`O`, `P`, ..., `Z`). That gives us 716 options, when we remove also the 12 surviving placements for V that make it non-'Γ'. When Algorithm X is applied to this reduced set, the search tree for finding all solutions goes down to 1243 nodes, and the running time is only 4.5 megamems.

(There's also a slightly better way to remove the symmetry: Instead of insisting that piece V looks like 'Γ' we can insist that piece X lies in the left half, and that piece Z hasn't been "flipped over." This implies that there are (16, 2, 8) potential placements for (`V`, `X`, `Z`), instead of (4, 4, 16). The resulting search tree has just 1128 nodes, and the running time is 4.0 Mμ.)

Notice that we could have begun with a weaker formulation of this problem: We could merely have asked for pentomino arrangements that use each piece *at most* once, while covering each cell ij exactly once. That would be essentially the same as saying that the piece names $\{$`O`,`P`,...,`Z`$\}$ are *secondary* items instead of primary. Then the original set of 1236 options in (37) would have led to a search tree with 61,851 nodes, and a runtime of 291 Mμ. Dually, we could have kept the piece names primary but made the cell names secondary; that would have yielded a 1,086,521,921-node tree, with a runtime of 2.94 Tμ! These statistics are curiously *reversed*, however, with respect to the reduced set of 716 options obtained by discarding cases like (38): Then piece names secondary yields 19306 nodes (68 Mμ); cell names secondary yields 11656 nodes (37 Mμ).

In the early days of computing, pentomino problems served as useful benchmarks for combinatorial calculations. Programmers didn't have the luxury of large random-access memory until much later; therefore techniques such as dancing links, in which more than a thousand options are explicitly listed and manipulated, were unthinkable at the time. Instead, the options for each piece were implicitly generated on-the-fly as needed, and there was no incentive to use

Hansson
symmetry removal
V pentomino
remove the symmetry
flipped over
secondary
benchmarks
history

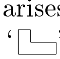
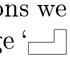
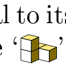
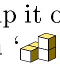
fancy heuristics while backtracking. Each branch of the search was essentially based on the available ways to cover the first cell ij that hadn't yet been occupied.

We can simulate the behavior of those historic methods by running Algorithm X without the MRV heuristic and simply setting $i \leftarrow$ `RLINK(0)` in step X3. An interesting phenomenon now arises: If the cells ij are considered in their natural order — first `00`, then `01`, ..., then `0j`, then `10`, ..., finally `2j` — the search tree has 1.5 billion nodes. (There are 29 ways to cover `00`; if we choose '`00 01 02 03 04 O`' there are 49 ways to cover `05`; and so on.) But if we consider the 20×3 problem instead of 3×20, so that the cells ij for $0 \le i < 20$ and $0 \le j < 3$ are processed in order `00`, `01`, `02`, `10`, ..., `2j`, the search tree has just 71191 nodes, and all eight solutions are found very quickly. (This speedup is mostly due to having a better "focus," which we'll discuss later.) Again we see that a small change in problem setup can have enormous ramifications.

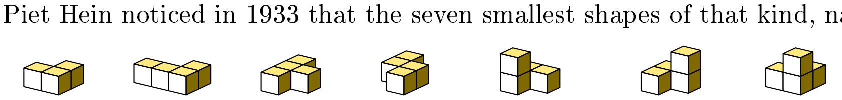
The best of these early programs were highly tuned, written in assembly language with ingenious uses of macro instructions. Memwise, they were therefore superior to Algorithm X on smallish problems. [See J. G. Fletcher, *CACM* **8**, 10 (October 1965), cover and 621–623; N. G. de Bruijn, *FGbook* pages 465–466.] But the MRV heuristic eventually wins, as problems get larger.

Exercises 268–323 discuss some of the many intriguing and instructive problems that arise when we explore the patterns that can be made with pentominoes and similar families of planar shapes. Several of these problems are indeed large — beyond the capabilities of today's machines.

Polycubes. And if you think two-dimensional shapes are fun, you'll probably enjoy three dimensions even more! A *polycube* is a solid object formed by taking one or more $1 \times 1 \times 1$ cubes and joining them face-to-face. We call them monocubes, dicubes, tricubes, tetracubes, pentacubes, etc.; but we *don't* call them "n-cubes" when they're made from n unit cubies, because mathematicians have reserved that term for n-dimensional objects.

A new situation arises when $n = 4$. In two dimensions we found it natural to regard the tetromino '' as identical to its mirror image '', because we could simply flip it over. But the tetracube '' is noticeably different from its mirror reflection '', because we can't change one into the other without going into the fourth dimension. Polycubes that differ from their mirror images are called *chiral*, a word coined by Lord Kelvin in 1893 when he studied chiral molecules.

The simplest polycubes are *cuboids* — also called "rectangular parallelepipeds" by people who like long names — which are like bricks of size $l \times m \times n$. But things get particularly interesting when we consider noncuboidal shapes. Piet Hein noticed in 1933 that the seven smallest shapes of that kind, namely



1: bent 2: ell 3: tee 4: skew 5: L-twist 6: R-twist 7: claw , (39)

can be put together to form a $3 \times 3 \times 3$ cube, and he liked the pieces so much that he called them *Soma*. Notice that the first four pieces are essentially planar, while the other three are inherently three-dimensional. The twists are chiral.

MRV heuristic
focus
assembly language
macro instructions
Fletcher
de Bruijn
MRV heuristic
Polycubes
monocubes
dicubes
tricubes
tetracubes
pentacubes
n-cubes
cubie: A $1 \times 1 \times 1$ cube inside a larger box
tetromino
flip it over
chiral
Kelvin
cuboids
parallelepipeds
bricks
Hein
bent tromino
ell tetromino
tee tetromino
skew tetromino
twist tetracube
claw tetracube
Soma

Martin Gardner wrote about the joys of Soma in *Scientific American* **199**, 3 (September 1958), 182–188, and it soon became wildly popular: More than two million SOMA® cubes were sold in America alone, after Parker Brothers began to market a well-made set together with an instruction booklet written by Hein.

Gardner
Parker Brothers
HEIN
Patent
canonical
factoring

> *A minimum number of blocks of simple form are employed. . . . Experiments and calculations have shown that from the set of seven blocks it is possible to construct approximately the same number of geometrical figures as could be constructed from twenty-seven separate cubes.*
>
> — PIET HEIN, *United Kingdom Patent Specification 420,349* (1934)

The task of packing these seven pieces into a cube is easy to formulate as an exact cover problem, just as we did when packing pentominoes. But this time we have 24 3D-rotations of the pieces to consider, instead of 8 2D-rotations and/or 3D-reflections; so exercise 324 is used instead of exercise 266 to generate the options of the problem. It turns out that there are 688 options, involving 34 items that we can call 1, 2, ..., 7, 000, 001, ..., 222. For example, the first option

$$\text{‘1 000 001 010’} \tag{40}$$

characterizes one of the 144 potential ways to place the "bent" piece 1.

Algorithm X needs just 407 megamems to find all 11,520 solutions to this problem. Furthermore, we can save most of that time by taking advantage of symmetry: Every solution can be rotated into a unique "canonical" solution in which the "ell" piece 2 has not been rotated; hence we can restrict that piece to only six placements, namely (000, 010, 020, 100), (001, 011, 021, 101), ..., (102, 112, 122, 202) — all shifts of each other. This restriction removes $138 = \frac{23}{24} \cdot 144$ options, and the algorithm now finds the 480 canonical solutions in just 20 megamems. (These canonical solutions form 240 mirror-image pairs.)

Factoring an exact cover problem. In fact, we can simplify the Soma cube problem much further, so that all of its solutions can actually be found by hand in a reasonable time, by *factoring* the problem in a clever way.

Let's observe first that any solution to an exact cover problem automatically solves infinitely many *other* problems. Going back to our original formulation in terms of an $m \times n$ matrix $A = (a_{ij})$, the task is to find all sets of rows whose sum is 1 in every column, namely to find all binary vectors $x_1 \ldots x_m$ such that $\sum_{i=1}^{m} x_i a_{ij} = 1$ for $1 \le j \le n$. Therefore if we set $b_i = \alpha_1 a_{i1} + \cdots + \alpha_n a_{in}$ for $1 \le i \le m$, where $(\alpha_1, \ldots, \alpha_n)$ is any n-tuple of coefficients, the vectors $x_1 \ldots x_m$ will also satisfy $\sum_{i=1}^{m} x_i b_i = \alpha_1 + \cdots + \alpha_n$. By choosing the α's intelligently we may be able to learn a lot about the possibilities for $x_1 \ldots x_m$.

For example, consider again the 6×7 matrix A in (5), and let $\alpha_1 = \cdots = \alpha_7 = 1$. The sum of the entries in each row of A is either 2 or 3; thus we're supposed to cover 7 things, by burying either 2 or 3 at a time. Without knowing anything more about the detailed structure of A, we can conclude immediately that there's only one way to obtain a total of 7, namely by selecting $2 + 2 + 3$! Furthermore, only rows 1 and 5 have 2 as their sum; we *must* choose them.

Now here's a more interesting challenge: "Cover the 64 cells of a chessboard with 21 straight trominoes and one monomino." This problem corresponds to a big matrix that has $96+64$ rows and $1+64$ columns,

$$\begin{array}{c}
\begin{array}{cccccccccccccccccccccccccccc}
\texttt{M} & \texttt{00} & \texttt{01} & \texttt{02} & \texttt{03} & \texttt{04} & \texttt{05} & \texttt{06} & \texttt{07} & \texttt{10} & \texttt{11} & \texttt{12} & \texttt{13} & \texttt{14} & \texttt{15} & \texttt{16} & \texttt{17} & \texttt{20} & \texttt{21} & \texttt{22} & \texttt{23} & \texttt{24} & & \texttt{74} & \texttt{75} & \texttt{76} & \texttt{77}
\end{array} \\
\begin{pmatrix}
0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\ldots&0&0&0&0\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&\ldots&0&0&0&0\\
0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\ldots&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&\ldots&0&0&0&0\\
&&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&&&\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\ldots&0&1&1&1\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\ldots&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\ldots&0&0&0&0\\
&&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&&&\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&\ldots&0&0&0&1
\end{pmatrix}
\end{array}, \qquad (41)$$

where the first 96 rows specify all possible ways to place a tromino and the other 64 rows specify the possibilities for the monomino. Column ij represents cell (i,j); column '$\texttt{M}$' means "monomino."

The three cells (i,j) covered by a straight tromino always lead to distinct values of $(i-j) \bmod 3$. Therefore, if we add up the 22 columns of (41) for which $(i-j) \bmod 3 = 0$, we get 1 in each of the first 96 rows, and 0 or 1 in the other 64 rows. We're supposed to get a total of 22 in the chosen rows; hence the monomino has to go into a cell (i,j) with $i \equiv j$ (modulo 3).

A similar argument, using $i+j$ instead of $i-j$, shows that the monomino must also go into a cell with $i+j \equiv 1$ (modulo 3). Therefore $i \equiv j \equiv 2$. *We've proved that there are only four possibilities for* (i,j), namely $(2,2)$, $(2,5)$, $(5,2)$, $(5,5)$. [Golomb made this observation in his 1954 paper that introduced polyominoes, after "coloring" the cells of a chessboard with three colors. The general notion of factoring includes all such coloring arguments as special cases.]

Our proof that (38) is an impossible pentomino placement can also be regarded as an instance of factorization. The residual problem, if (38) is chosen, has a total of either 0 or 5 in the first 27 columns of each remaining row of the associated matrix. Therefore we can't achieve a total of 27 from those rows.

Consider now a three-dimensional problem [J. Slothouber and W. Graatsma, *Cubics* (1970), 108–109]: Can six $1 \times 2 \times 2$ cuboids be packed into a $3 \times 3 \times 3$ box? This is the problem of choosing six rows of the 36×27 matrix

$$\begin{array}{c}
\begin{array}{ccccccccccccccccccccccccccc}
\texttt{000} & \texttt{001} & \texttt{002} & \texttt{010} & \texttt{011} & \texttt{012} & \texttt{020} & \texttt{021} & \texttt{022} & \texttt{100} & \texttt{101} & \texttt{102} & \texttt{110} & \texttt{111} & \texttt{112} & \texttt{120} & \texttt{121} & \texttt{122} & \texttt{200} & \texttt{201} & \texttt{202} & \texttt{210} & \texttt{211} & \texttt{212} & \texttt{220} & \texttt{221} & \texttt{222}
\end{array} \\
\begin{pmatrix}
1&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
&&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&&\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1
\end{pmatrix}
\end{array}, \qquad (42)$$

in such a way that all of the column sums are ≤ 1.

chessboard
straight trominoes
monomino
Golomb
coloring arguments
Slothouber
Graatsma
square tetracubes

Soma cube
checkerboard coloring

The 27 cubies (i,j,k) of a $3\times3\times3$ cube fall into four classes, depending on how many of its coordinates have the middle value 1:

$$\begin{array}{ll} \text{A vertex cubie has no 1s.} & \left(\binom{3}{0}2^3 = 8 \text{ cases.}\right) \\ \text{An edge cubie has one 1.} & \left(\binom{3}{1}2^2 = 12 \text{ cases.}\right) \\ \text{A face cubie has two 1s.} & \left(\binom{3}{2}2^1 = 6 \text{ cases.}\right) \\ \text{A central cubie has three 1s.} & \left(\binom{3}{3}2^0 = 1 \text{ case.}\right) \end{array} \qquad (43)$$

Every symmetry of the cube preserves these classes.

Imagine placing four new columns v, e, f, c at the right of (42), representing the number of vertex, edge, face, and central cubies of a placement. Then 24 of the rows will have $(v,e,f,c) = (1,2,1,0)$, and the other 12 rows will have $(v,e,f,c) = (0,1,2,1)$. If we choose a rows of the first kind and b rows of the second kind, this factorization tells us that we must have

$$a \ge 0,\ \ b \ge 0,\ \ a+b = 6,\ \ a \le 8,\ \ 2a+b \le 12,\ \ a+2b \le 6,\ \ b \le 1. \qquad (44)$$

That's more than enough to prove that $b = 0$ and $a = 6$, and thus to find the essentially unique way to pack those six cuboids.

(We could paraphrase this argument as follows, making it more impressive by concealing the low-level algebra that inspired it: "Each $1\times2\times2$ cuboid occupies at least one face cubie. So each of them must be placed on a different face.")

With these examples in mind, we're ready now to apply factorization to the Soma cube. The possible (v,e,f,c) values for pieces 1 through 7 in (39) are:

$$\begin{array}{ll} \text{Piece 1:} & (0,1,1,1),\ (0,0,2,1),\ (0,1,2,0),\ (0,2,1,0),\ (1,1,1,0),\ (1,2,0,0). \\ \text{Piece 2:} & (0,1,2,1),\ (0,2,2,0),\ (1,2,1,0),\ (2,2,0,0). \\ \text{Piece 3:} & (0,0,3,1),\ (0,2,1,1),\ (0,3,1,0),\ (2,1,1,0). \\ \text{Piece 4:} & (0,1,2,1),\ (1,2,1,0). \\ \text{Piece 5:} & (0,1,2,1),\ (0,2,2,0),\ (1,1,1,1),\ (1,2,1,0). \\ \text{Piece 6:} & (0,1,2,1),\ (0,2,2,0),\ (1,1,1,1),\ (1,2,1,0). \\ \text{Piece 7:} & (0,2,1,1),\ (0,0,3,1),\ (1,1,2,0),\ (1,3,0,0). \end{array} \qquad (45)$$

(This is actually much more information than we need, but it doesn't hurt.)

Looking only at the totals for v, we see that we must have

$$(0 \text{ or } 1) + (0,\ 1,\text{ or } 2) + (0 \text{ or } 2) + (0 \text{ or } 1) + (0 \text{ or } 1) + (0 \text{ or } 1) + (0 \text{ or } 1)\ =\ 8;$$

and the *only* way to achieve this is via

$$(0 \text{ or } 1) + (1 \text{ or } 2) + 2 + (0 \text{ or } 1) + (0 \text{ or } 1) + (0 \text{ or } 1) + (0 \text{ or } 1)\ =\ 8,$$

thus eliminating several options for pieces 2 and 3. More precisely, *piece 2 must touch at least one vertex; piece 3 must be placed along an edge.*

Looking next at the totals for $v+f$, which are the "black" cubies if we color them alternately black and white with black in the corners, we must also have

$$(1 \text{ or } 2) + 2 + 3 + 2 + 2 + 2 + (1 \text{ or } 3)\ =\ 14;$$

Guy, Michael
Guy
Conway
Berlekamp
Conway
Guy
symmetry

and the only way to achieve this is with two from piece 1 and one from piece 7: *Piece 1 must occupy two black cubies, and piece 7 must occupy just one.*

We have therefore eliminated 200 of the 688 options in the list that begins with (40). And we also know that exactly five of the pieces 1, 2, 4, 5, 6, 7 occupy as many of the corner vertices as they individually can. This extra information can be encoded by introducing 13 new primary items

`*, 1+, 1-, 2+, 2-, 4+, 4-, 5+, 5-, 6+, 6-, 7+, 7-` (46)

and six new options

```
'* 1+ 2- 4- 5- 6- 7-'
'* 1- 2+ 4- 5- 6- 7-'
'* 1- 2- 4+ 5- 6- 7-'
'* 1- 2- 4- 5+ 6- 7-'      (47)
'* 1- 2- 4- 5- 6+ 7-'
'* 1- 2- 4- 5- 6- 7+'
```

and by appending p`+` or p`-` to each of piece p's options that do or don't touch the most corners. For example, this new set of $6 + 488$ options for the Soma cube problem includes the following typical ways to place various pieces:

```
'1 000 001 011 1+'
'1 001 011 101 1-'
'2 000 001 002 010 2+'
'2 000 001 011 021 2-'
'3 000 001 002 011'
'4 000 001 011 012 4+'
'4 000 011 111 121 4-'
'5 000 001 010 110 5+'
'5 001 010 011 101 5-'
'6 000 001 010 101 6+'
'6 001 010 011 110 6-'
'7 000 001 010 100 7+'
'7 001 010 011 111 7-'
```

As before, Algorithm X finds 11,520 solutions; but now it needs only 108 megamems to do so. Each of the new options is used in at least 21 of the solutions, hence we've removed all of the "fat" in the original set. [This instructive analysis of Soma is due to M. J. T. Guy, R. K. Guy, and J. H. Conway in 1961. See Berlekamp, Conway, and Guy, *Winning Ways*, second edition (2004), 845–847.]

To reduce the number of solutions, using symmetry, we can force piece 3 to occupy the cells $\{000, 001, 002, 011\}$ (thus saving a factor of 24), and we can remove all options for piece 7 that use a cell ijk with $k = 2$ (saving an additional factor of 2). From the remaining 455 options, Algorithm X needs just 2 megamems to generate all 240 of the essentially distinct solutions.

The seven Soma pieces are amazingly versatile, and so are the other polycubes of small sizes. Exercises 325–345 explore some of their remarkable properties, together with historical references.

color-controlled–
word search
hidden mathematicians+
mathematicians+
color codes

Color-controlled covering. *Take a break!* Before reading any further, please spend a minute or two solving the "word search" puzzle in Fig. 71. Comparatively mindless puzzles like this one provide a low-stress way to sharpen your word-recognition skills. It can be solved easily — for instance, by making eight passes over the array — and the solution appears in Fig. 72 on the next page.

Fig. 71. Find the mathematicians*:

Put ovals around the following names where they appear in the 15×15 array shown here, reading either forward or backward or upward or downward, or diagonally in any direction. After you've finished, the leftover letters will form a hidden message. (The solution appears on the next page.)

ABEL	HENSEL	MELLIN
BERTRAND	HERMITE	MINKOWSKI
BOREL	HILBERT	NETTO
CANTOR	HURWITZ	PERRON
CATALAN	JENSEN	RUNGE
FROBENIUS	KIRCHHOFF	STERN
GLAISHER	KNOPP	STIELTJES
GRAM	LANDAU	SYLVESTER
HADAMARD	MARKOFF	WEIERSTRASS

O	T	H	E	S	C	A	T	A	L	A	N	D	A	U
T	S	E	A	P	U	S	T	H	O	R	S	R	O	F
T	L	S	E	E	A	Y	R	R	L	Y	H	A	P	A
E	P	E	A	R	E	L	R	G	O	U	E	M	S	I
N	N	A	R	R	C	V	L	T	R	T	A	A	M	A
I	T	H	U	O	T	E	K	W	I	A	N	D	E	M
L	A	N	T	N	B	S	I	M	I	C	M	A	A	W
L	G	D	N	A	R	T	R	E	B	L	I	H	C	E
E	R	E	C	I	Z	E	C	E	P	T	N	E	D	Y
M	E	A	R	S	H	R	H	L	I	P	K	A	T	H
E	J	E	N	S	E	N	H	R	I	E	O	N	E	T
H	S	U	I	N	E	B	O	R	F	E	W	N	A	R
T	M	A	R	K	O	F	F	O	F	C	S	O	K	M
P	L	U	T	E	R	P	F	R	O	E	K	G	R	A
G	M	M	I	N	S	E	J	T	L	E	I	T	S	G

Our goal in this section is not to discuss how to *solve* such puzzles; instead, we shall consider how to *create* them. It's by no means easy to pack those 27 names into the box in such a way that their 184 characters occupy only 135 cells, with eight directions well mixed. How can that be done with reasonable efficiency?

For this purpose we shall extend the idea of exact covering by introducing *color codes*. Let's imagine that each cell ij of the array is to be "colored" with one of the letters $\{\mathtt{A}, \ldots, \mathtt{Z}\}$. Then the task of creating such a puzzle is essentially to choose from among a vast set of options

```
'ABEL 00:A 01:B 02:E 03:L'
'ABEL 00:A 10:B 20:E 30:L'
'ABEL 00:A 11:B 22:E 33:L'
'ABEL 00:L 01:E 02:B 03:A'
        . . . . .
'WEIERSTRASS e4:S e5:S e6:A e7:R e8:T e9:S ea:R eb:E ec:I ed:E ee:W'
```
(48)

in such a way that the following conditions are satisfied:

i) Exactly one option must be chosen for each of the 27 mathematicians' names.
ii) The chosen options must give consistent colors to each of the 15×15 cells ij.

---

* The journal *Acta Mathematica* celebrated its 21st birthday by publishing a special *Table Générale des Tomes 1–35*, edited by Marcel Riesz (Uppsala: 1913), 179 pp. It contained a complete list of all papers published so far in that journal, together with portraits and brief biographies of all the authors. The 27 mathematicians mentioned in Fig. 71 are those who were subsequently mentioned in Volumes 1, 2, or 3 of *The Art of Computer Programming* — except for people like `MITTAG-LEFFLER` or `POINCARÉ`, whose names contain special characters.

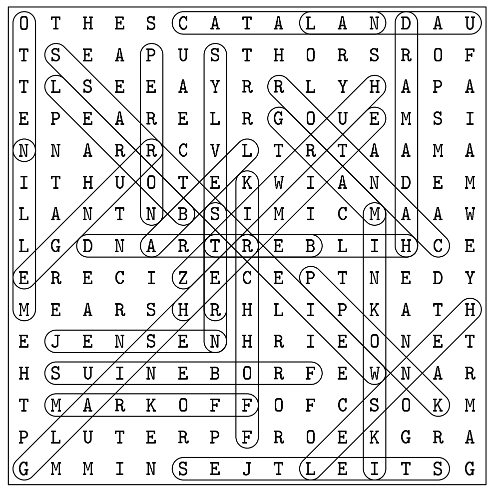


Fig. 72. Solution to the puzzle of the hidden mathematicians (Fig. 71). Notice that the central letter `R` actually participates in six different names:

```
BERTRAND
GLAISHER
HERMITE
HILBERT
KIRCHHOFF
WEIERSTRASS
```

The `T` to its left participates in five.

Here's what the leftover letters say:

> These authors of early papers in *Acta Mathematica* were cited years later in *The Art of Computer Programming*.

interactively
exact covering with colors
XCC
items
options
required items
elective items
colon

There also are informal constraints: It's desirable to have many shared letters between names, and to intermix the various directions, so that the puzzle has plenty of variety and perhaps a few surprises. But conditions (i) and (ii) are the important criteria for a computer to consider; the auxiliary informalities are best handled interactively, with human guidance.

Notice that the color constraints (ii) are significantly different from the name constraints (i). Several *distinct* options are allowed to specify the color of the *same* cell, as long as those specifications don't conflict with each other.

Let us therefore define a new problem, *exact covering with colors*, or XCC for short. As before, we're given a set of items, of which N_1 are primary and $N - N_1$ are secondary. We're also given a family of M options, each of which includes at least one primary item. The new rule is that a *color* is assigned to the secondary items of each option. The new task is to find all choices of options such that

i) every primary item occurs exactly once; and
ii) every secondary item has been assigned at most one color.

The primary items are required; the secondary items are elective.

Color assignments are denoted by a colon; for example, '`00:A`' in (48) means that color `A` is assigned to the secondary item `00`. When a secondary item of an option is *not* followed by a colon, it is implicitly assigned a *unique* color, which doesn't match the color of any other option. Therefore the ordinary exact cover problems that we've been studying so far, in which secondary items don't explicitly receive colors but cannot be chosen in more than one option, are just special cases of the XCC problem, even though nothing about color was mentioned.

A tremendous variety of combinatorial problems can be expressed readily in the XCC framework. And there's good news: The dancing links technique works beautifully with such problems! Indeed, we will see that this considerably more general problem can be solved with only a few small extensions to Algorithm X.

The nodes of Algorithm X have just three fields: `TOP`, `ULINK`, and `DLINK`. We now add a fourth field, `COLOR`; this field is set to the positive value c when the node represents an item that has explicitly been assigned color c. Consider, for example, the following toy problem with three primary items $\{\text{p}, \text{q}, \text{r}\}$ and two secondary items $\{\text{x}, \text{y}\}$, where the options are

```
'p q x y:A';
'p r x:A y';
'p x:B';
'q x:A';
'r y:B'.
```
(49)

Table 2 shows how it would be represented in memory, extending the conventions of Table 1. Notice that COLOR $= 0$ when no color has been specified. The COLOR fields of the header nodes (nodes 1–5 in this example) need not be initialized because they're never examined. The COLOR fields of the spacer nodes (nodes 6, 11, 16, 19, 22, 25) are unimportant, except that they must be nonnegative.

Table 2

THE INITIAL CONTENTS OF MEMORY CORRESPONDING TO (49)

i:	0	1	2	3	4	5	6
NAME(i):	—	p	q	r	x	y	—
LLINK(i):	3	0	1	2	6	4	5
RLINK(i):	1	2	3	0	5	6	4
x:	0	1	2	3	4	5	6
LEN(x):	—	3	2	2	4	3	0
ULINK(x):	—	17	20	23	21	24	—
DLINK(x):	—	7	8	13	9	10	10
x:	7	8	9	10	11	12	13
TOP(x):	1	2	4	5	−1	1	3
ULINK(x):	1	2	4	5	7	7	3
DLINK(x):	12	20	14	15	15	17	23
COLOR(x):	0	0	0	A	0	0	0
x:	14	15	16	17	18	19	20
TOP(x):	4	5	−2	1	4	−3	2
ULINK(x):	9	10	12	12	14	17	8
DLINK(x):	18	24	18	1	21	21	2
COLOR(x):	A	0	0	0	B	0	0
x:	21	22	23	24	25		
TOP(x):	4	−4	3	5	−5		
ULINK(x):	18	20	13	15	23		
DLINK(x):	4	24	3	5	—		
COLOR(x):	A	0	0	B	0		

It's easy to see how these COLOR fields can be used to get the desired effect: When an option is chosen, we "purify" any secondary items that it names, by effectively removing all options that have conflicting colors. One slightly subtle point arises, because we don't want to waste time purifying a list that has already been culled. The trick is to set COLOR(x) $\leftarrow -1$ in every node x that's already known to have the correct color, except in nodes that have already been hidden.

Thus we want to upgrade the original operations cover(i) and hide(p) in (12) and (13), as well as their counterparts uncover(i) and unhide(p) in (14) and (15), in order to incorporate color controls. The changes are simple:

cover′(i) is like cover(i), but it calls hide′(p) instead of hide(p); (50)

hide′(p) is like hide(p), but it ignores node q when `COLOR`(q) < 0; (51)

uncover′(i) is like uncover(i), but it calls unhide′(p) instead of unhide(p); (52)

unhide′(p) is like unhide(p), but it ignores node q when `COLOR`(q) < 0. (53)

Our colorful algorithm also introduces two new operations and their inverses:

$$\text{commit}(p,j) = \begin{cases} \text{If } \mathtt{COLOR}(p) = 0, \text{ cover}'(j); \\ \text{if } \mathtt{COLOR}(p) > 0, \text{ purify}(p). \end{cases} \tag{54}$$

$$\text{purify}(p) = \begin{cases} \text{Set } c \leftarrow \mathtt{COLOR}(p),\ i \leftarrow \mathtt{TOP}(p),\ q \leftarrow \mathtt{DLINK}(i). \\ \text{While } q \neq i, \text{ do the following and set } q \leftarrow \mathtt{DLINK}(q)\text{:} \\ \quad\quad \text{if } \mathtt{COLOR}(q) = c, \text{ set } \mathtt{COLOR}(q) \leftarrow -1; \\ \quad\quad \text{otherwise hide}'(q). \end{cases} \tag{55}$$

$$\text{uncommit}(p,j) = \begin{cases} \text{If } \mathtt{COLOR}(p) = 0, \text{ uncover}'(j); \\ \text{if } \mathtt{COLOR}(p) > 0, \text{ unpurify}(p). \end{cases} \tag{56}$$

$$\text{unpurify}(p) = \begin{cases} \text{Set } c \leftarrow \mathtt{COLOR}(p),\ i \leftarrow \mathtt{TOP}(p),\ q \leftarrow \mathtt{ULINK}(i). \\ \text{While } q \neq i, \text{ do the following and set } q \leftarrow \mathtt{ULINK}(q)\text{:} \\ \quad\quad \text{if } \mathtt{COLOR}(q) < 0, \text{ set } \mathtt{COLOR}(q) \leftarrow c; \\ \quad\quad \text{otherwise unhide}'(q). \end{cases} \tag{57}$$

Otherwise Algorithm C is almost word-for-word identical to Algorithm X.

Algorithm C (*Exact covering with colors*). This algorithm visits all solutions to a given XCC problem, using the same conventions as Algorithm X.

C1. [Initialize.] Set the problem up in memory, as in Table 2. (See exercise 8.) Also set N to the number of items, Z to the last spacer address, and $l \leftarrow 0$.

C2. [Enter level l.] If `RLINK`(0) $= 0$ (hence all items have been covered), visit the solution that is specified by $x_0x_1\ldots x_{l-1}$ and go to C8. (See exercise 13.)

C3. [Choose i.] At this point the items $i_1, \ldots, i_t$ still need to be covered, where $i_1 =$ `RLINK`(0), $i_{j+1} =$ `RLINK`(i_j), `RLINK`(i_t) $= 0$. Choose one of them, and call it i. (The MRV heuristic of exercise 9 often works well in practice.)

C4. [Cover i.] Cover item i using (50), and set $x_l \leftarrow$ `DLINK`(i).

C5. [Try x_l.] If $x_l = i$, go to C7 (we've tried all options for i). Otherwise set $p \leftarrow x_l + 1$, and do the following while $p \neq x_l$: Set $j \leftarrow$ `TOP`(p); if $j \leq 0$, set $p \leftarrow$ `ULINK`(p); otherwise commit(p,j) and set $p \leftarrow p+1$. (This covers the items $\neq i$ in the option that contains x_l.) Set $l \leftarrow l+1$ and return to C2.

C6. [Try again.] Set $p \leftarrow x_l - 1$, and do the following while $p \neq x_l$: Set $j \leftarrow$ `TOP`(p); if $j \leq 0$, set $p \leftarrow$ `DLINK`(p); otherwise uncommit(p,j) and set $p \leftarrow p-1$. (This uncovers the items $\neq i$ in the option that contains x_l, using the reverse order.) Set $i \leftarrow$ `TOP`(x_l), $x_l \leftarrow$ `DLINK`(x_l), and return to C5.

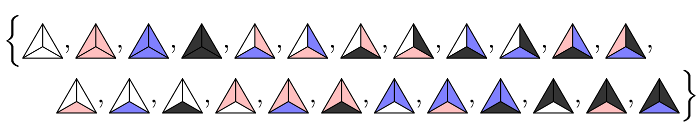
C7. [Backtrack.] Uncover item i using (52).

C8. [Leave level l.] Terminate if $l = 0$. Otherwise set $l \leftarrow l-1$ and go to C6. ▌

cover′
hide′
uncover′
unhide′
patching and updating an algorithm
MRV

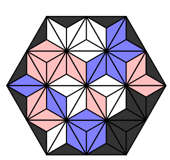
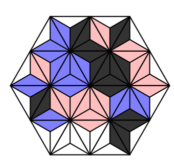
Algorithm C applies directly to several problems that we've already discussed in previous sections. For example, it readily generates word rectangles, as well as intriguing patterns of words that have more intricate structure (see exercises 87–93). We can use it to find all de Bruijn cycles, and their two-dimensional counterparts (see exercises 94–97). The extra generality of exact covering options also invites us to impose additional constraints for special applications. Furthermore, Algorithm C facilitates experiments with the tetrad tiles that we studied in Section 2.3.4.3 (see exercises 120 and 121).

The great combinatorialist P. A. MacMahon introduced several families of colorful geometric patterns that continued to fascinate him throughout his life. For example, in U.K. Patent 3927 of 1892, written with J. R. J. Jocelyn, he considered the 24 different triangles that can be made with four colors on their edges,



(58)

and showed two ways in which they could be arranged to form a hexagon with matching colors at adjacent edges and with solid colors on the outer boundary:

(a)  (b)  (59)

(Notice that chiral pairs, like  and  in (58), are considered to be distinct; MacMahon's tiles can be rotated, but they can't be "flipped over.")

Four suitable colours are black, white, red, and blue,
as they are readily distinguishable at night.
— P. A. MACMAHON, *New Mathematical Pastimes* (1921)

Let's assume that the boundary is supposed to be all white, as in pattern (59b). There are millions of ways to satisfy this condition; but every really distinct solution is counted 72 times, because the hexagon has 12 symmetries under rotation and reflection, and because the three nonwhite colors can be permuted in $3! = 6$ ways. We can remove the hexagon symmetries by fixing the position of the all-white triangle (see exercise 119). And the color symmetries can be removed by using an interesting extension of Algorithm C, which reduces the number of solutions by a factor of $d!$ when the options are symmetrical with respect to d colors (see exercise 122). In this way all of the solutions — can you guess how many? — can actually be found with only 5.2 Gμ of computation (see exercise 126).

MacMahon went on to study many other matching problems with these triangles, as well as with similar sets of tiles that are based on squares, hexagons, and other shapes. He also considered three-dimensional arrangements of colored cubes, which are supposed to match where they touch. Exercises 127–148 are devoted to some of the captivating questions that have arisen from this work.

word rectangles
de Bruijn cycles
tetrad tiles
MacMahon
Patent
Jocelyn
superdominoes
chiral pairs
MACMAHON
rotation
reflection
dihedral group
color symmetries
colored cubes

XCC
constraint satisfaction problem
CSP
Wainwright
partridge puzzle
cubes, numbers of the form n^3
squares, numbers of the form n^2
Christmas

Introducing multiplicity. We've now seen from numerous examples that Algorithm C — which extends Algorithm X and solves arbitrary XCC problems — is enormously versatile. In fact, there's a sense in which *every* constraint satisfaction problem is a special case of an XCC problem (see exercise 100).

But we can extend Algorithm C even further, again without substantial changes, so that it goes well beyond the original notion of exact covering. For example, let's consider Robert Wainwright's "partridge puzzle" (1981), which was inspired by the well-known fact that the sum of the first n cubes is a perfect square:

$$1^3 + 2^3 + \cdots + n^3 \;=\; N^2, \quad \text{where } N = 1 + 2 + \cdots + n. \tag{60}$$

Wainwright wondered if this relation could be verified geometrically, by taking one square of size 1×1, two squares of size 2×2, $\ldots$, n squares of size $n \times n$, and packing them all into a big square of size $N \times N$. (We know from exercise 1.2.1–8 that $4k$ squares of each size $k \times k$ can be packed into a $2N \times 2N$ square. But Wainwright hoped for a more direct corroboration of (60).) He proved the task impossible for $2 \le n \le 5$, but found a perfect packing when $n = 12$; so he thought of the 12 days of Christmas, and named his puzzle accordingly (see exercise 154).

This partridge puzzle is easily expressed in terms of options that involve n items $\#k$ for $1 \le k \le n$, as well as N^2 items ij for $0 \le i, j < N$. By analogy with what we did with pentominoes in (37), the options are

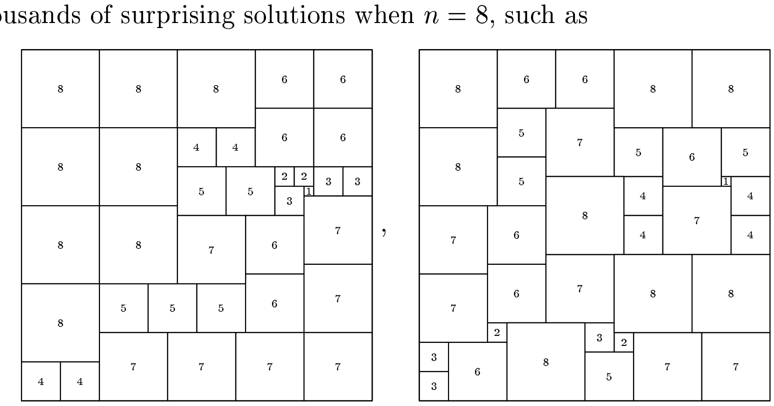
$$\text{`}\#k\ ij\ i(j{+}1)\ \ldots\ i(j{+}k{-}1)\ (i{+}1)j\ (i{+}1)(j{+}1)\ \ldots\ (i{+}k{-}1)(j{+}k{-}1)\text{'} \tag{61}$$

for $1 \le k \le n$ and $0 \le i, j \le N - k$. (Exactly $(N + 1 - k)^2$ options involve $\#k$, and each of them names $1 + k^2$ items.) For example, the options when $n = 2$ are

'#1 00', '#1 01', '#1 02', '#1 10', '#1 11', '#1 12', '#1 20', '#1 21', '#1 22',
'#2 00 01 10 11', '#2 01 02 11 12', '#2 10 11 20 21', '#2 11 12 21 22'.

As before, we want to cover each of the N^2 cells ij exactly once. But there's a difference: We now want to cover primary item $\#k$ exactly k times, not just once.

That's a rather big difference. But in Algorithm M below, we'll see that the dancing links approach can handle it. For example, that algorithm can show that the partridge puzzle has no perfect packings for $n = 6$ or $n = 7$; but it finds thousands of surprising solutions when $n = 8$, such as



(62)

When we first defined exact cover problems, near the beginning of this section, we considered $M \times N$ matrices of 0s and 1s, such as (5). In matrix terms, the task was to find all subsets of the rows whose sum is $11\ldots1$. Algorithm M is going to do much more: It will find all subsets of rows whose sum is $v_1v_2\ldots v_N$, where $v_1v_2\ldots v_N$ is *any* desired vector of multiplicities.

In fact, Algorithm M will go further yet, by allowing *intervals* $[u_j \mathinner{..} v_j]$ to be prescribed for each multiplicity. It will actually solve the general *MCC problem*, "multiple covering with colors," which is defined as follows: There are N items, of which N_1 are primary and $N - N_1$ are secondary. Each primary item j for $1 \le j \le N_1$ is assigned an interval $[u_j \mathinner{..} v_j]$ of permissible values, where $0 \le u_j \le v_j$ and $v_j > 0$. There also are M options, each of which includes at least one primary item. A color is assigned to the secondary items of each option; a "blank" color is understood to represent a unique color that appears nowhere else. The task is to find all subsets of options such that

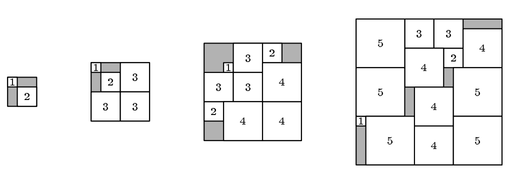
i) each primary item j occurs at least u_j times and at most v_j times;

ii) every secondary item has been assigned at most one color.

Thus every XCC problem is the special case $u_j = v_j = 1$ of an MCC problem.

Indeed, the MCC problem is *extremely* general! For example, its special case $u_j = 1$ and $v_j = M$, without secondary items, is the classical not-necessarily-exact *cover problem*, in which we simply require each item to appear in *at least* one option. Section 7.2.2.6 will be entirely devoted to the cover problem.

Let's confine our attention here to a few more examples of the MCC problem, in preparation for Algorithm M. In the first place, we can tackle a refined version of Wainwright's partridge puzzle: "Pack at most k squares of size $k \times k$, for $1 \le k \le n$, into an $N \times N$ square, without overlapping, so that as many as possible of the N^2 cells are covered." (As before, $N = 1+2+\cdots+n$.) We know from (62) that the entire square can be covered when $n = 8$; but smaller cases are another story. Solutions for $2 \le n \le 5$ are readily found by hand:



(63)

And to prove that every packing for $n = 5$ must leave at least 13 cells vacant, Algorithm M will show that the MCC problem (61) has no solutions when items #1, #2, #3 are respectively given the multiplicities $[0 \mathinner{..} 13]$, $[0 \mathinner{..} 2]$, $[0 \mathinner{..} 3]$ instead of 1, 2, 3. Exercise 157 constructs optimum packings when $n = 6$ and $n = 7$, thereby settling all small cases of the partridge puzzle.

Next, let's consider an MCC problem of quite a different kind: "Place m queens so that they control all cells of an $n \times n$ chessboard." (The classic 5-queens problem — which should be distinguished from the '5 queens problem' considered earlier — is the special case $m = 5$, $n = 8$.) Exercise 7.1.4–241 discusses the history of this problem, which goes back to a remarkable book by de Jaenisch (1863).

intervals
multiple covering with colors
primary item
secondary items
cover problem
Wainwright
partridge puzzle
chessboard
5-queens problem
n queens problem
queen domination problem
de Jaenisch

We can solve it, MCC-wise, by introducing $n^2 + 1$ primary items, namely the pairs ij for $0 \le i, j < n$ and the special item #, together with n^2 options:

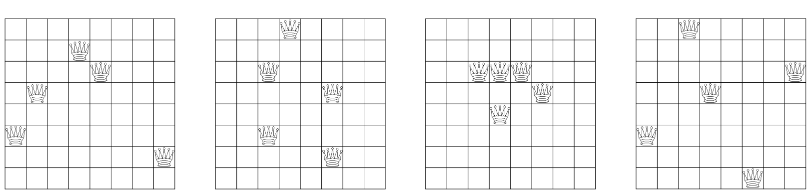
$$\text{`}\#\ ij\ i_1j_1\ i_2j_2\ \ldots\ i_tj_t\text{'} \quad \text{for } 0 \le i, j < n, \tag{64}$$

where i_1j_1, i_2j_2, …, i_tj_t are the cells attacked by a queen on ij. Each cell ij is assigned the multiplicity $[1\,..\,m]$; item # gets multiplicity m.

From this specification Algorithm M will readily find all 4860 solutions to the 5-queens problem, after 13 gigamems of computation. For example, it begins with 22 ways to cover the corner cell 00. If it puts a queen there, it has 22 ways to cover cell 17; and so on. The branching factor at each step tends to decrease rapidly after three queens have been placed.

The beauty of the MCC setup in (64) is that we can solve many related problems by making simple changes to the specifications. For example, by retaining only the 36 options for $1 \le i, j \le 6$, we could find the 284 solutions that place no queens at the edges of the board. Or by removing the 16 options for $2 \le i, j \le 5$, we would discover that exactly 880 of the solutions place no queens near the middle. Exactly 88 solutions avoid the central two rows and the central two columns. Exactly 200 solutions put all five queens on "black" cells (with $i + j$ even). Exactly 90 avoid the upper left and lower right quadrants. Exactly 2 solutions (can you find them?) place all five queens in the top half of the board.

By changing the multiplicities in the bottom row from $[1\,..\,5]$ to 1, we get 18 solutions for which every cell in that row is attacked just once. Or, changing the central 16 multiplicities to $[2\,..\,5]$ yields 48 solutions for which every cell near the center is attacked at least twice. Changing all the cell multiplicities to $[1\,..\,4]$ reduces the number of solutions from 4860 to 3248; changing them all to $[1\,..\,3]$ reduces it to 96. Exercise 161 illustrates several of the less obvious possibilities.



(65)

The examples of MCC problems that we've seen so far have involved primary items only. Secondary items, and their color controls, add new dimensions and extend the range of applications enormously. Consider, for example, the *word rectangles* that we investigated briefly in Section 7.2.2. Here's a 4×5 word rectangle that uses only nine distinct letters of the alphabet:

```
L A B E L
A B I D E
S L A I N .
T E S T S
```

Can we find one that uses only *eight* distinct letters, while sticking to common words? (More precisely, is there such a rectangle whose columns are chosen from the most common 1000 four-letter words of English, and whose rows belong to `WORDS`(2000), the curated collection from the Stanford GraphBase?)

Secondary items
color controls
word rectangles
English
`WORDS`(2000)
Stanford GraphBase

nonsharp preference heuristic
recursively

The answer is yes, and in fact there are six solutions:

```
S T R U T   E A S E D   W A D E D   R A D A R   L L A M A   S C A R S
T E A S E   A G I L E   A R E N A   A R E N A   E A G E R   C O C O A
E A T E N   S E N S E   S E E D S   S E A T S   S T E A M   A R R A Y    (66)
P R E S S   E D G E D   H A R S H   H A R S H   T E S T S   R E E D S
```

One way to find them is to set up an MCC problem in which the primary items are A_0, A_1, A_2, A_3, D_0, D_1, D_2, D_3, D_4, #A, #B, …, #Z, #; they all have multiplicity 1 except that # has multiplicity 8. There also are secondary items A, B, …, Z, and ij for $0 \le i < 4$, $0 \le j < 5$. The letter-counting is handled by $2 \cdot 26$ short options:

‘#A A:0’, ‘#A A:1 #’; ‘#B B:0’, ‘#B B:1 #’; …; ‘#Z Z:0’, ‘#Z Z:1 #’. (67)

Then each legal 5-letter word $c_1c_2c_3c_4c_5$ yields four options, ‘A_i $i0$:c_1 $i1$:c_2 $i2$:c_3 $i3$:c_4 $i4$:c_5 c_1:1 c_2:1 c_3:1 c_4:1 c_5:1’, for $0 \le i < 4$; each legal 4-letter word $c_1c_2c_3c_4$ yields five options, ‘D_j $0j$:c_1 $1j$:c_2 $2j$:c_3 $3j$:c_4 c_1:1 c_2:1 c_3:1 c_4:1’, for $0 \le j < 5$. (Letters that occur more than once in a word are listed only once.)

For example, one of the options chosen for the first solution in (66) is ‘A_3 30:P 31:R 32:E 33:S 34:S P:1 R:1 E:1 S:1’; it forces the options ‘#P P:1 #’, ‘#R R:1 #’, ‘#E E:1 #’, ‘#S S:1 #’ to be chosen too, thus contributing four to the number of chosen options that contain #.

By the way, when Algorithm M is applied to these options, it’s important to use the “nonsharp preference heuristic” discussed in exercise 10 and its answer. Otherwise the algorithm will foolishly make binary branches on the items #A, …, #Z, before trying out actual words. A 1000-way branch on D_0 is much better than a 2-way branch on #Q, in this situation.

***A new dance step.** In order to implement multiplicity, we need to update the data structures in a new way. Suppose, for example, that there are five options available for some primary item p, and suppose they are represented in nodes a, b, c, d, and e. Then p’s vertical list of active options has the following links:

x:	p	a	b	c	d	e	
ULINK(x):	e	p	a	b	c	d	(68)
DLINK(x):	a	b	c	d	e	p	

If the multiplicity of p is 3, there are $\binom{5}{3} = 10$ ways to choose three of the five options; but we do *not* want to make a 10-way branch! Instead, each branch of Algorithm M below chooses only the *first* of the options that will appear in the solution. Then it reduces the problem recursively; the reduced problem will have a shorter list for p, from which two further options should be selected. Since we must choose either a, b, or c as the first option, the algorithm will therefore begin with a 3-way branch. For example, if b is chosen to be first, the reduced problem will ask for two of options $\{c, d, e\}$ to be chosen eventually.

The algorithm will recursively find all solutions to that reduced problem. But it won’t necessarily begin by branching again on the *same* item, p; some other item, q, might well have become more significant. For instance, the choice

tweaking
FT[l]
first tweaks
untweaking
link manipulations

of b might have assigned color values that make $\mathtt{LEN}(q) \le 1$. (The choice of b might also have made c, d, and/or e illegal.)

The main point is that, after we've chosen the first of three options for p in the original problem, we have *not* "covered" p as we did in Algorithms X and C. Item p remains on an equal footing with all other active items of the reduced problem, so we need to modify (68) accordingly.

The operation of reducing the problem by removing an option from an item list, in the presence of multiplicity, is called "tweaking" that option. For example, just after the algorithm has chosen b as the first option for p, it will have tweaked both a and b. This operation is deceptively simple:

$$\text{tweak}(x,p) \;=\; \begin{cases}\text{hide}'(x) \text{ and set } d \leftarrow \mathtt{DLINK}(x),\ \mathtt{DLINK}(p) \leftarrow d,\\ \mathtt{ULINK}(d) \leftarrow p,\ \mathtt{LEN}(p) \leftarrow \mathtt{LEN}(p) - 1.\end{cases} \qquad (69)$$

(See (51). We will tweak(x,p) only when $x = \mathtt{DLINK}(p)$ and $p = \mathtt{ULINK}(x)$.) Notice that tweaking x does more than hiding x, but it does less than covering p.

Eventually the algorithm will have tried each of a, b, and c as p's first option, and it will want to backtrack and undo the tweaking. The actions tweak(a,p), tweak(b,p), tweak(c,p) will have clobbered most of the original uplinks in (68):

$$\begin{array}{rcccccc} x\text{:} & p & a & b & c & d & e \\ \mathtt{ULINK}(x)\text{:} & e & p & p & p & p & d \\ \mathtt{DLINK}(x)\text{:} & d & b & c & d & e & p \end{array} \qquad (70)$$

Unfortunately, this residual data isn't sufficient for us to restore the original state, because we've lost track of node a. But if we had recorded the value of a when we began, we would be in good shape, because a pointer to node a together with the DLINKs in (70) would now lead us to node b, then to c, and then to d.

Therefore the algorithm maintains an array $\mathtt{FT}[l]$, to remember the locations of the "first tweaks" that were made at every level l. And it adds a new dance step, "untweaking," to its repertoire of link manipulations:

$$\text{untweak}(l) \;=\; \begin{cases}\text{Set } a \leftarrow \mathtt{FT}[l],\ p \leftarrow \bigl(a \le N?\ a\text{: } \mathtt{TOP}(a)\bigr),\ x \leftarrow a,\ y \leftarrow p;\\ \text{set } z \leftarrow \mathtt{DLINK}(p),\ \mathtt{DLINK}(p) \leftarrow x,\ k \leftarrow 0;\\ \text{while } x \ne z, \text{ set } \mathtt{ULINK}(x) \leftarrow y \text{ and } k \leftarrow k+1,\\ \qquad \text{unhide}'(x), \text{ and set } y \leftarrow x,\ x \leftarrow \mathtt{DLINK}(x);\\ \text{finally set } \mathtt{ULINK}(z) \leftarrow y \text{ and } \mathtt{LEN}(p) \leftarrow \mathtt{LEN}(p) + k.\end{cases} \qquad (71)$$

(See exercise 163. This computation relies on a surprising fact proved in exercise 2(a), namely that unhiding can safely be done in the same order as hiding.)

The same mechanism can be used when the specified multiplicity is an *interval* instead of a single number. For example, suppose item p in the example above is required to occur in either 2, 3, or 4 options, not exactly 3. Then the first option chosen must be a, b, c, or d; and the reduced problem will ask p to occur in either 1, 2, or 3 of the options that remain. Eventually the algorithm will resort to untweaking, after a, b, c, and d have all been tweaked and explored.

Similarly, if p's multiplicity has been specified to be either 0, 1, or 2, the algorithm below will tweak each of a through e in turn. It will also run though all solutions that omit *all* of p's options, before finally untweaking and backtracking.

A special case arises, however, when p's multiplicity has been specified to be either 0 or 1. In such cases we're not allowed to choose options b, c, d, or e after option a has been chosen. Therefore it's important to invoke cover$'(p)$, as in Algorithm C, instead of hiding one option at a time. (See (50).) The individual options of p are then tweaked, to remove them one by one from the active list; this tweaking uses the special operation tweak$'(x,p)$, which is like tweak(x,p) in (69) except that it omits the operation hide$'(x)$, because hiding was already done when p was covered. Finally, the case of 0-or-1 multiplicity eventually concludes by invoking the routine untweak$'(l)$, which is like untweak(l) in (71) except that (i) it omits unhide$'(x)$, and (ii) it calls uncover$'(p)$ after restoring $\texttt{LEN}(p)$.

tweak$'$
untweak$'$
MRV
LEN
BOUND
SLACK

We're ready now to write Algorithm M, except that we need a way to represent the multiplicities in the data structures. For this purpose every primary item has two new fields, SLACK and BOUND. Suppose the desired multiplicity of p is in the interval $[u\,..\,v]$, where $0 \le u \le v$ and $v \ne 0$; Algorithms X and C correspond to the case $u = v = 1$. Then we set

$$\texttt{SLACK}(p) \leftarrow v - u, \qquad \texttt{BOUND}(p) \leftarrow v \tag{72}$$

when the algorithm begins. The value of $\texttt{SLACK}(p)$ will never be changed. But $\texttt{BOUND}(p)$ will decrease dynamically, as we reduce the problem, so that we will never choose more options for p than its current bound.

Algorithm M (*Covering with multiplicities and colors*). This algorithm visits all solutions to a given MCC problem, extending Algorithms X and C.

M1. [Initialize.] Set the problem up in memory as in step C1 of Algorithm C, with the addition of multiplicity specifications (72). Also set N to the number of items, N_1 to the number of primary items, Z to the last spacer address, and $l \leftarrow 0$.

M2. [Enter level l.] If $\texttt{RLINK}(0) = 0$ (hence all items have been covered), visit the solution that is specified by $x_0x_1\ldots x_{l-1}$ and go to M9. (See exercise 164.)

M3. [Choose i.] At this point the items $i_1, \ldots, i_t$ still need to be covered, where $i_1 = \texttt{RLINK}(0)$, $i_{j+1} = \texttt{RLINK}(i_j)$, $\texttt{RLINK}(i_t) = 0$. Choose one of them, and call it i. (The MRV heuristic of exercise 166 often works well in practice.) If the branching degree θ_i is zero (see exercise 166), go to M9.

M4. [Prepare to branch on i.] Set $x_l \leftarrow \texttt{DLINK}(i)$ and $\texttt{BOUND}(i) \leftarrow \texttt{BOUND}(i) - 1$. If $\texttt{BOUND}(i)$ is now zero, cover item i using (50). If $\texttt{BOUND}(i) \ne 0$ or $\texttt{SLACK}(i) \ne 0$, set $\texttt{FT}[l] \leftarrow x_l$.

M5. [Possibly tweak x_l.] If $\texttt{BOUND}(i) = \texttt{SLACK}(i) = 0$, go to M6 if $x_l \ne i$, to M8 if $x_l = i$. (That case is like Algorithm C.) Otherwise if $\texttt{LEN}(i) \le \texttt{BOUND}(i) - \texttt{SLACK}(i)$, go to M8 (list i is too short). Otherwise if $x_l \ne i$, call tweak(x_l, i) (see (69)), or tweak$'(x_l, i)$ if $\texttt{BOUND}(i) = 0$. Otherwise if $\texttt{BOUND}(i) \ne 0$, set $p \leftarrow \texttt{LLINK}(i)$, $q \leftarrow \texttt{RLINK}(i)$, $\texttt{RLINK}(p) \leftarrow q$, $\texttt{LLINK}(q) \leftarrow p$.

M6. [Try x_l.] If $x_l \ne i$, set $p \leftarrow x_l + 1$, and do the following while $p \ne x_l$: Set $j \leftarrow \texttt{TOP}(p)$; if $j \le 0$, set $p \leftarrow \texttt{ULINK}(p)$; otherwise if $j \le N_1$, set $\texttt{BOUND}(j) \leftarrow \texttt{BOUND}(j) - 1$, $p \leftarrow p + 1$, and cover$'(j)$ if $\texttt{BOUND}(j)$ is now 0; otherwise commit(p, j) and set $p \leftarrow p + 1$. (This loop covers or partially covers the items $\neq i$ in the option that contains x_l.) Set $l \leftarrow l + 1$ and return to M2.

M7. [Try again.] Set $p \leftarrow x_l - 1$, and do the following steps while $p \neq x_l$: Set $j \leftarrow$ `TOP`(p); if $j \leq 0$, set $p \leftarrow$ `DLINK`(p); otherwise if $j \leq N_1$, set `BOUND`$(j) \leftarrow$ `BOUND`$(j) + 1$, $p \leftarrow p - 1$, and uncover$'(j)$ if `BOUND`(j) is now 1; otherwise uncommit(p, j) and set $p \leftarrow p - 1$. (This loop uncovers the items $\neq i$ in the option that contains x_l, using the reverse order.) Set $x_l \leftarrow$ `DLINK`(x_l) and return to M5.

M8. [Restore i.] If `BOUND`(i) = `SLACK`$(i) = 0$, uncover item i using (52). Otherwise call untweak(l) (see (71)), or untweak$'(l)$ if `BOUND`$(i) = 0$. Set `BOUND`$(i) \leftarrow$ `BOUND`$(i) + 1$.

M9. [Leave level l.] Terminate if $l = 0$. Otherwise set $l \leftarrow l - 1$. If $x_l \leq N$, set $i \leftarrow x_l$, $p \leftarrow$ `LLINK`(i), $q \leftarrow$ `RLINK`(i), `RLINK`$(p) \leftarrow$ `LLINK`$(q) \leftarrow i$, and go to M8. (That reactivates i.) Otherwise set $i \leftarrow$ `TOP`(x_l) and go to M7. ▮

***Analysis of Algorithm X.** Now let's get quantitative, and see what we can actually *prove* about the running time of these algorithms.

For simplicity, we'll ignore color constraints and look only at Algorithm X, as it finds all solutions to an exact cover problem, where the problem is specified in terms of an $M \times N$ matrix A of 0s and 1s such as (5).

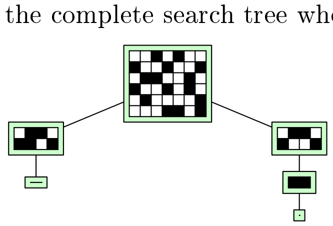
We'll assume that the problem is *strict*, in the sense that no two rows of the matrix are identical, and no two columns of the matrix are identical. For if two or more rows or columns are equal, we need keep only one of them; it's easy to relate all solutions of the original problem A to the solutions of this reduced problem A'. (See exercise 179.)

Our first goal will be to find an upper bound on the number of nodes in the search tree, as a function of the number of rows of A (the number of options). This upper bound grows exponentially, because the exact cover problem can have lots of solutions; but we'll see that it can't actually be extremely large.

For this purpose we'll define the *doomsday function* $D(n)$, which will have the property that the search tree for every strict exact cover problem with n options has at most $D(n)$ nodes, when Algorithm X uses the MRV heuristic in step X3.

The search tree has a root node labeled with the original matrix A, and its other nodes are defined recursively: When a node at level l is labeled with a subproblem for which step X3 makes a t-way branch, that node has t subtrees, whose roots are labeled by the reduced problem that remains after step X4 has covered item i and after step X5 has optionally covered one or more other items j, for t different choices of x_l.

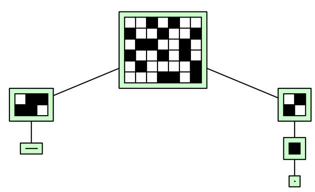
Here, for example, is the complete search tree when A is the matrix of (5):



Analysis of algorithms–exact cover problem
0s and 1s
strict
doomsday function
recursively

strict
MRV
doomsday function

(Each matrix and submatrix in this diagram has been framed with a light-gray border. The node at bottom left illustrates a 0×1 submatrix, where the algorithm had to backtrack because it had no way to cover the remaining column. The node at bottom right illustrates a 0×0 submatrix, which happens to be a solution to the 1×2 problem above it.) We can, if we like, *reduce* all of the submatrices by eliminating repeated columns, although Algorithm X doesn't do this; then we get *strict* exact cover problems at every node of the search tree:



(73)

A t-way branch implies that the matrix A has a certain structure. We know that there's some column, say $i_1 = i$, that has 1 in exactly t rows, say $o_1, \ldots, o_t$, and that *every* column contains at least t 1s. When we branch on row o_p, for $1 \le p \le t$, the reduced problem that defines the pth subtree will retain all but s_p of the rows of A, where s_p is the number of rows that intersect o_p. We can order the rows so that $s_1 \le \cdots \le s_t$. For example, in (73) we have $t = 2$ and $s_1 = s_2 = 4$.

A nice thing now happens: There's always a unique index $0 \le t' \le t$ such that

$$s_p \;=\; t+p-1, \qquad \text{for } 1 \le p \le t'. \tag{74}$$

That is, either $s_1 > t$ and $t' = 0$; or $s_1 = t$ and $t' = 1$ and either ($t = 1$ or $s_2 > t+1$); or $s_1 = t$ and $s_2 = t+1$ and $t' = 2$ and either ($t = 2$ or $s_3 > t+2$); or $\ldots$; or $s_1 = t$ and $s_2 = t+1$ and $\ldots$ and $s_t = 2t-1$ and $t' = t$.

Suppose, for example, that $t = 4$ and $s_1 = 4$; we must prove that $s_2 \ge 5$. Since $s_1 = 4$, row o_1 doesn't intersect any rows except $\{o_1, \ldots, o_t\}$; consequently option o_1 consists of the *single* item 'i_1'. Hence option o_2 must contain at least two items, '$i_1\ i_2\ \ldots$', otherwise the problem wouldn't be strict. This new item appears in at least 4 options, however, one of which is different from o_1. Option o_2 therefore intersects 5 or more options (including itself). QED.

Similarly, if $t = 4$ and $s_1 = 4$ and $s_2 = 5$, exercise 180 proves that $s_3 \ge 6$, and indeed it proves that even more is true. For example, if $t = t' = 4$, so that $(s_1, s_2, s_3, s_4) = (4, 5, 6, 7)$ as demanded by (74), exercise 180 proves the existence of options o_5, o_6, o_7 that have a particularly simple form:

$$\begin{aligned} &o_1 = \text{`}i_1\text{'};\ \ o_2 = \text{`}i_1\ i_2\text{'};\ \ o_3 = \text{`}i_1\ i_2\ i_3\text{'};\ \ o_4 = \text{`}i_1\ i_2\ i_3\ i_4\text{'};\\ &\qquad o_5 = \text{`}i_2\ i_3\ i_4\ \ldots\text{'};\ \ o_6 = \text{`}i_3\ i_4\ \ldots\text{'};\ \ o_7 = \text{`}i_4\ \ldots\text{'}. \end{aligned} \tag{75}$$

Okay, we're ready now to construct the promised "doomsday function" $D(n)$. It starts out very tame,

$$D(0) \;=\; D(1) \;=1; \tag{76}$$

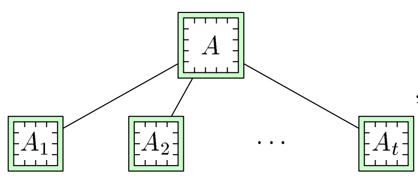
and for convenience we set $D(n) = -\infty$ if $n < 0$. When $n \ge 2$ the definition is

$$D(n) = \max\bigl\{d(n,t,t') \bigm| 1 \le t < n \text{ and } 0 \le t' \le t\bigr\}, \tag{77}$$

where $d(n,t,t')$ is an upper bound for the size of the search tree over all n-option strict exact cover problems whose parameters (74) are t and t'. One such bound, Eppstein author birthday

$$d(n,t,0) \;=\; 1 + t\cdot D(n-t-1), \tag{78}$$

handles the case $t'=0$, because the search tree in that case is a t-way branch



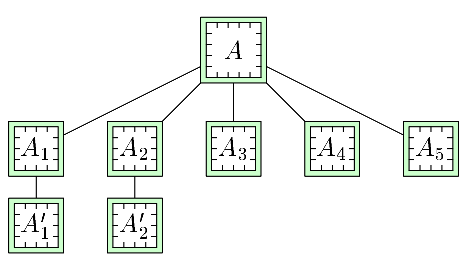
, (79)

and each subproblem A_p has at most $n-t-1$ options.

The formula for $t'>0$ is more intricate, and less obvious:

$$d(n,t,t') \;=\; t' + t'\cdot D(n-t-t'+1) + (t-t')\cdot D(n-t-t'-1),\;\; \text{if } 1\le t'\le t. \tag{80}$$

It can be justified by the structure theory of exercise 180, using the fact that each of the first $t'-1$ branches is immediately followed by a 1-way branch. For example, the search tree looks like this when $t=5$ and $t'=3$:



(81)

Here A_1' is the only way to cover i_2 in A_1, and A_2' is the only way to cover i_3 in A_2. The strict problems A_1', A_2', and A_3 have at most $n-7$ options; A_4 and A_5 have at most $n-9$. Therefore (81) has at most $3+3D(n-7)+2D(n-9)$ nodes.

With an easy computer program, (76), (78), and (80) lead to the values

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$D(n)$ =	1	1	2	4	5	6	10	13	17	22	31	41	53	69	94	125	165	213	283	377	501	661

and it turns out that the maximum is attained *uniquely* when $t=4$ and $t'=0$, for all $n\ge 19$. Hence we have $D(n)=1+4D(n-5)$ for all sufficiently large n; and in fact exercise 181 exhibits a simple formula that expresses $D(n)$ exactly.

Theorem E. *The search tree of a strict exact cover problem with n options has $O(4^{n/5})=O(1.31951^n)$ nodes; it might have as many as $\Omega(7^{n/8})=\Omega(1.27537^n)$.*

Proof. The upper bound follows from exercise 181; the lower bound follows from the family of problems in exercise 182. ▮

[David Eppstein presented this theorem to the author as a birthday greeting(!); see `11011110.github.io/blog/2008/01/10/analyzing-algorithm-x.html`.]

So far we've simply been analyzing the number of nodes in Algorithm X's search tree. But some nodes might cost much more than others, because they might remove unusually many options from the currently active lists.

Therefore let's probe deeper, by studying the number of *updates* that Algorithm X makes to its data structures, namely the number of times that it uses operation (1) to remove an element from a doubly linked list. (This is also the number of times that it will eventually use operation (2) to *restore* an element.) More precisely, the number of updates is the number of times cover(i) is called, plus the number of times that hide(p) sets `LEN`(x) ← `LEN`(x) − 1. (See (12) and (13).) The total running time of Algorithm X, measured in mems, usually turns out to be roughly 13 times the total number of updates that it makes.

It's instructive to analyze the number of updates that are made when solving the "extreme" exact cover problems, which arise when there are n items and $2^n - 1$ options: Such problems have the most solutions and the most data, because *every* nonempty subset of the items is an option. The solutions to these extreme problems are precisely the *set partitions* — the ϖ_n possible ways to partition the items into disjoint blocks, which we studied in Section 7.2.1.5. For example, when $n = 3$ the options are '1', '2', '1 2', '3', '1 3', '2 3', '1 2 3', and there are $\varpi_3 = 5$ solutions: '1', '2', '3'; '1', '2 3'; '1 2', '3'; '1 3', '2'; '1 2 3'.

Any given item can be covered in 2^{n-1} ways; and if we cover it with an option of size k, we're left with the extreme problem on the remaining $n - k$ items. Algorithm X therefore advances 2^{n-1} times from level 0 to level 1, after which it essentially calls itself recursively. And at level 0 it performs a certain number of updates, say v_n, regardless of what strategy is used in step X3 to choose an item for branching. Therefore it makes a total of x_n updates, where

$$x_n \;=\; v_n + \binom{n-1}{0}x_{n-1} + \binom{n-1}{1}x_{n-2} + \cdots + \binom{n-1}{n-1}x_0. \qquad (82)$$

The solution to this recurrence is $x_0 = v_0$, $x_1 = v_0 + v_1$, $x_2 = 2v_0 + v_1 + v_2$, and in general $x_n = \sum_{k=0}^{n} a_{nk}v_k$, where the matrix (a_{nk}) is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & \ldots \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 & \ldots \\ 5 & 3 & 1 & 1 & 0 & 0 & 0 & \ldots \\ 15 & 9 & 4 & 1 & 1 & 0 & 0 & \ldots \\ 52 & 31 & 14 & 5 & 1 & 1 & 0 & \ldots \\ 203 & 121 & 54 & 20 & 6 & 1 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & \ldots \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & \ldots \\ -1 & -2 & -1 & 1 & 0 & 0 & 0 & \ldots \\ -1 & -3 & -3 & -1 & 1 & 0 & 0 & \ldots \\ -1 & -4 & -6 & -4 & -1 & 1 & 0 & \ldots \\ -1 & -5 & -10 & -10 & -5 & -1 & 1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix}^{-1}, \qquad (83)$$

with rows and columns numbered from 0. The numbers a_{n0} in the left column, which solve (82) when $v_n = \delta_{n0}$, are the familiar Bell numbers ϖ_n; they enumerate the leaves in the search tree. The numbers a_{n1} in the next column, which solve (82) when $v_n = \delta_{n1}$, are the Gould numbers $\widehat{\varpi}_n$; they enumerate set partitions whose last block or "tail" is a singleton, when the blocks of the partition are ordered by their least elements. In general, a_{nk} for $k > 0$ is the number of set

updates
set partitions
ϖ_n
recursively
Bell numbers
Gould numbers
last block of a set partition
tail of a set partition

partitions whose tail has size k. [See H. W. Gould and J. Quaintance, *Applicable Analysis and Discrete Mathematics* **1** (2007), 371–385.]

Exercise 186 proves that the actual number of updates at level 0 is

$$v_n \;=\; \bigl((9n-27)4^n - (8n-32)3^n + (36n-36)2^n + 72 - 41\delta_{n0}\bigr)/72; \qquad (84)$$

and exercise 187 exploits relationships between the sequences $\langle a_{nk}\rangle$ to show that

$$x_n \;=\; 22\varpi_n + 12\widehat{\varpi}_n - (\tfrac{2}{3}n-1)3^n - \tfrac{5}{2}n2^n - 12n - 5 - 12\delta_{n1} - 18\delta_{n0}. \qquad (85)$$

Asymptotically, $\widehat{\varpi}_n/\varpi_n$ converges rapidly to the "Euler–Gompertz constant"

$$\hat{g} = \int_0^\infty \frac{e^{-x}\,dx}{1+x} = 0.59634\,73623\,23194\,07434\,10784\,99369\,27937\,60741+ \qquad (86)$$

(see exercise 188). Thus $x_n \approx (22+12\hat{g})\varpi_n \approx 29.156\varpi_n$, and we've proved that *Algorithm X performs approximately 29.156 updates per solution to the extreme exact cover problem, on average.* That's encouraging: One might suspect that the list manipulations needed to deal with 2^n options of average length n would cost substantially more, but the dancing-links approach turns out to be within a constant factor of Section 7.2.1.5's highly tuned methods for set partitions.

***Analysis of matching problems.** Among the simplest exact cover problems are the ones whose options don't contain many items. For example, a so-called X2C problem ("exact cover with 2-sets") is the special case where every option has exactly 2 items; an X3C problem has 3 items per option; and so on. We've seen in (30) that sudoku is an X4C problem.

Let's take a close look at the simplest case, the X2C problems. Despite their simplicity, we'll see that such problems actually include many cases of interest. Every X2C problem corresponds in an obvious way to a graph G, whose vertices v are the items and whose edges $u \mathrel{-\!\!-} v$ are the pairs of items that occur together in an option '$u\ v$'. In these terms the X2C problem is the classical task of finding a perfect matching, namely a set of edges that contains each vertex exactly once.

We shall study efficient algorithms for perfect matching in Section 7.5.5 below. But an interesting question faces us now, in the present section: How well does our general-purpose Algorithm X compare to the highly tuned special-purpose algorithms that have been developed especially for matching in graphs?

Suppose, for example, that G is the complete graph K_{2q+1}. In other words, suppose that there are $n = 2q+1$ items $\{0, 1, \ldots, 2q\}$, and that there are $m = \binom{2q+1}{2} = (2q+1)q$ options '$i\ j$' for $0 \le i < j \le 2q$. This problem clearly has no solution, because we can't cover an odd number of points with 2-element sets! But Algorithm X won't know this (unless we give it a hint by factoring the problem appropriately). Thus it's interesting to see how long Algorithm X will spin its wheels before giving up on this problem.

In fact the analysis is easy: No matter what item i is chosen in step X3, the algorithm will split nicely into $2q$ branches, one for each option '$i\ j$' with $j \ne i$. And each of those branches will be equivalent to the matching problem on the remaining $2q-1$ items; the remaining options will, in fact, be equivalent to the complete graph K_{2q-1}. Thus the search tree will have $2q$ nodes at depth 1,

Quaintance
Gompertz
Euler–Gompertz constant
X2C problem
X3C problem
sudoku
X4C problem
perfect matching
matching
complete graph
factoring

$(2q)(2q-2)$ nodes at depth 2, …, and $(2q)(2q-2)\ldots(2) = 2^q q!$ nodes at depth q. Backtracking will occur at the latter nodes, which are leaves because they correspond to impossible matching in the graph K_1.

How long does this process take? A closer look (see exercise 193) shows that the total number of updates to the data structure will satisfy the recurrence

$$U(2q+1) \;=\; 1+2q+4q^2+2qU(2q-1), \quad \text{for } q>0; \qquad U(1)=1. \tag{87}$$

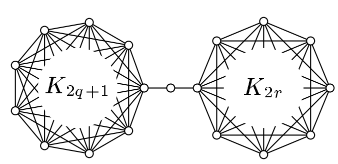
Consequently (see exercise 194) the number of updates needed by Algorithm X to discover that K_{2q+1} has no perfect matching turns out to be less than 8.244 times the number of leaves.

There's better news when Algorithm X is presented with the complete graph K_{2q}, because this problem has solutions — lots of them. Indeed, it's easy to see that K_{2q} has exactly $(2q-1)(2q-3)\ldots(3)(1) = (2q)!/(2^q q!)$ perfect matchings. For example, K_8 has $7\cdot5\cdot3\cdot1 = 105$ of them. The total number of updates in this case satisfies a recurrence similar to (87):

$$U(2q) \;=\; 1-2q+4q^2+(2q-1)U(2q-2), \quad \text{for } q>0; \qquad U(0)=0. \tag{88}$$

And exercise 194 proves that this is less than 10.054 updates per matching found.

Armed with these facts, we can work out what happens when the graph



(89)

is presented to Algorithm X. (This graph has $2q+2r+2$ vertices.) The result, which is revealed in exercise 195, is both instructive and bizarre.

A 2D MATCHING problem — also called bipartite matching, and 2DM for short — is the special case of an X2C problem in which every option has the form '$X_j\ Y_k$', where the items $\{X_1,\ldots,X_n\}$ and $\{Y_1,\ldots,Y_n\}$ are disjoint sets. Higher-dimensional matching is defined similarly; sudoku is actually a case of 4DM.

Let's round out our analyses of matching by considering the *bounded permutation problem*: "Given a sequence of positive integers $a_1\ldots a_n$, find all permutations $p_1\ldots p_n$ of $\{1,\ldots,n\}$ such that $p_j \le a_j$ for $1\le j\le n$." We can assume that $a_1\le\cdots\le a_n$, because $p_1\ldots p_n$ is a permutation; we can also assume that $a_j \ge j$, otherwise there are no solutions; and we can assume without loss of generality that $a_n\le n$. This is easily seen to be a 2DM problem, having exactly $a_1+\cdots+a_n$ options, namely '$X_j\ Y_k$' for $1\le j\le n$ and $1\le k\le a_j$.

Suppose we branch first on X_1. Then each of the a_1 subproblems is easily seen to be essentially a bounded permutation problem with n decreased by 1, and with $a_1\ldots a_n$ replaced by $(a_2-1)\ldots(a_n-1)$. Thus a recursive analysis applies, and again we find that the dancing links algorithm does rather well. For example, if $a_j=\min(j+1,n)$ for $1\le j\le n$, there are 2^{n-1} solutions, and Algorithm X performs only about 12 updates per solution. If $a_j=\min(2j,n)$ for $1\le j\le n$, there are $\lfloor\frac{n+1}{2}\rfloor!\,\lfloor\frac{n+2}{2}\rfloor!$ solutions, and Algorithm X needs only about $4e-1\approx 9.87$ updates per solution. Exercise 196 has the details.

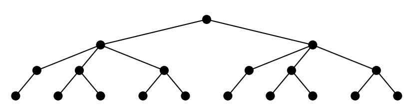
leaves
recurrence
2D MATCHING
bipartite matching
2DM
3D MATCHING
3DM
4DM
bounded permutation problem
permutations

***Maintaining a decent focus.** Some backtrack algorithms waste time by trying to solve two or more loosely related problems at once. Consider, for example, the 2DM problem with 7 items $\{0,1,\ldots,6\}$ and the following 13 options:

focus–
MRV
search tree
exponential behavior

$$\text{‘0 1’, ‘0 2’, ‘1 4’, ‘1 5’, ‘1 6’, ‘2 4’, ‘2 5’, ‘2 6’, ‘3 4’, ‘3 5’, ‘3 6’, ‘4 5’, ‘4 6’.} \qquad (90)$$

Algorithm X, using its MRV heuristic, will branch on item 0, choosing either ‘0 1’ or ‘0 2’; then it will be faced with a three-way branch; and it will evenually conclude that there's no solution, after implicitly traversing a 19-node search tree,



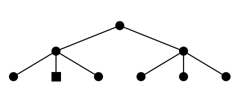
. (91)

We get an extreme example of bad focus if we take n independent copies of problem (90), with $7n$ items $\{k0, k1, \ldots, k6\}$ and $13n$ options ‘$k0$ $k1$’, ‘$k0$ $k2$’, …, ‘$k4$ $k6$’, for $0 \le k < n$: The algorithm will begin with 2-way branches on each of 00, 10, …, $(n-1)0$; then it will show that each of the 2^n resulting subproblems is unsolvable, making 3-way branches as it begins to study each one. Its search tree, before giving up, will have $10 \cdot 2^n - 1$ nodes. By contrast, if we had somehow forced the algorithm to keep its attention on the very first copy of (90) (the case $k = 0$), instead of using the MRV heuristic, it would have concluded that there are no solutions after backtracking through only 19 nodes.

Similarly, the simple exact cover problem on items $\{0,1,\ldots,5\}$ with options

$$\text{‘0 1’, ‘0 2’, ‘1 3 4’, ‘1 3 5’, ‘1 4 5’, ‘2 3 4’, ‘2 3 4 5’, ‘2 4 5’, ‘3 4’, ‘3 5’, ‘4 5’,} \qquad (92)$$

has a search tree with 9 nodes, one of which is a solution:

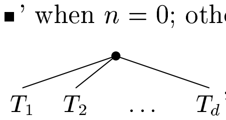


. (93)

Taking n independent copies of (92) gives us an exact cover problem with a unique solution, whose search tree via Algorithm X and MRV has $8 \cdot 2^n - 7$ nodes. But if the algorithm had been able to focus on one problem at a time, it would have discovered the solution with a search tree of only $8n + 1$ nodes.

From a practical standpoint, it must be admitted that the exponential behavior of these badly focused toy examples is worrisome only when n is larger than 30 or so, because 2^n is not scary for modern computers when n is small. Still, we can see that a well-focused approach can give significant advantages. So it will be useful to understand how Algorithm X and its cousins behave in general, when the input actually consists of two independent problems.

Let's pause a minute to define the search tree precisely. Given an $m \times n$ matrix A of 0s and 1s, the search tree T of its associated exact cover problem is simply a solution node ‘■’ when $n = 0$; otherwise T is



, $d \ge 0$, (94)

direct sum
dominant node
Langford pairs
n queens problem
minimally dominant

where the item chosen for branching in step X3 has d options, and T_k is the search tree for the reduced problem after the items of the kth option have been removed. (With the MRV heuristic, d is the minimum length of all active item lists, and we choose the leftmost item having this value of d.)

The exact cover problem that we get when trying to solve two independent problems given by matrices A and A' is the problem that corresponds to the direct sum $A \oplus A'$ (see Eq. 7–(40)). Therefore if T and T' are the corresponding search trees, we will write $T \oplus T'$ for the search tree of $A \oplus A'$, under the MRV heuristic. (That tree depends only on T and T', not on any other aspects of A or A'.) If either T or T' is simply a solution node, the rule is simple:

$$T \oplus \blacksquare = T; \qquad \blacksquare \oplus T' = T'. \tag{95}$$

Otherwise T and T' have the form (94), and we have

$$T \oplus T' = \begin{cases} \overset{\bullet}{T_1 \oplus T' \;\; T_2 \oplus T' \;\; \cdots \;\; T_d \oplus T'}, & \text{if } d \le d'; \\[2ex] \overset{\bullet}{T \oplus T'_1 \;\; T \oplus T'_2 \;\; \cdots \;\; T \oplus T'_{d'}}, & \text{if } d > d'. \end{cases} \tag{96}$$

Dear reader, please work exercise 202 — which is very easy! — before reading further. That exercise will help you to understand the definition of $T \oplus T'$; and you'll also see that every node of $T \oplus T'$ is associated with an ordered pair $\alpha\alpha'$, where α and α' are nodes of T and T', respectively. These ordered pairs are the key to the structure of $T \oplus T'$: If α and α' appear at levels $l > 0$ and $l' > 0$ of their trees, so that they are reached from the roots by the respective paths $\alpha_0 — \alpha_1 — \cdots — \alpha_l = \alpha$ and $\alpha'_0 — \alpha'_1 — \cdots — \alpha'_{l'} = \alpha'$, then the parent of $\alpha\alpha'$ in $T \oplus T'$ is either $\alpha_{l-1}\alpha'$ or $\alpha\alpha'_{l'-1}$. Consequently every ancestor α_k of α in T, for $0 \le k \le l$, occurs in an ancestor $\alpha_k\alpha'_{k'}$ of $\alpha\alpha'$ in $T \oplus T'$, for some $0 \le k' \le l'$.

Let $\deg(\alpha)$ be the number of children that node α has in a search tree, except that we define $\deg(\alpha) = \infty$ when α is a solution node. (Equivalently, $\deg(\alpha)$ is the minimum length of an item list, taken over all active items in the subproblem that corresponds to node α. If α is a solution, there are no active items, hence the minimum is infinite.) Let's call α a *dominant node* if its degree exceeds the degree of all its proper ancestors. The root node is always dominant, and so is every solution node. For example, (91) has three dominant nodes, and (93) has four.

In these terms, exercise 205 proves a significant fact about direct sums:

Lemma D. *Every node of $T \oplus T'$ corresponds to an ordered pair $\alpha\alpha'$ of nodes belonging to T and T'. Either α or α' is dominant in its tree, or both are.* ▮

Lemma D is good news, focuswise, because the search trees that arise in practice tend to have comparatively few dominant nodes. In such cases the MRV heuristic manages to keep the search reasonably well focused, because $T \oplus T'$ isn't too large. For example, the search trees for Langford pairs, or for the n queens problem, are "minimally dominant": Only their root nodes and their solutions dominate; elsewhere the branching degrees don't reach new heights.

Y pentominoes
Haselgrove
symmetry reduction
sharp preference
local equivalence–

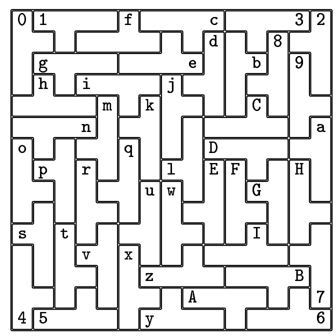


Fig. 73. A 15 × 15 square can be tiled with Y pentominoes, by setting up an exact cover problem with one item for each cell and one option for each placement of a Y. (To eliminate the 8-fold symmetry, only 5 of the 40 options for occupying the center cell were permitted.) Algorithm X's first solution, shown here, was found by branching successively on the possible ways to fill the cells marked `0`, `1`, . . . , `9`, `a`, . . . , `z`, `A`, . . . , `I`.

Let's look now at a non-contrived example. Figure 73 illustrates a somewhat surprising way to pack 45 Y pentominoes into a 15 × 15 square. [Such tilings were first found in 1973 by J. Haselgrove, at a time when perfect Y-packings were known only for rectangles whose area was even. Her program first ruled out all rectangles of odd area less than 225, as well as the case 9×25, before discovering a 15×15 solution. See *JRM* **7** (1974), 229.] Notice that the first eight pentominoes in Fig. 73, those marked `0` through `7`, were placed in or next to the four corners — thus flirting dangerously with the possibility that the algorithm might be trying to solve four independent problems at once! Luckily, the subsequent choices were able to gain and retain focus, because hard-to-fill cells almost always kept popping up near the recent activity. A five-way branch was needed only when placing the pentominoes marked `8`, `b`, `e`, `g`, `h`, and `C` in the solution shown.

Focus can sometimes be improved by explicitly preferring some items to others, based on their names; see the "sharp preference" heuristic of exercise 10.

Exercise 207 discusses another approach, an experimental modification of Algorithm X, which attempts to improve focus in situations like Fig. 73 by allowing a user to specify the importance of recent activity. The ideas are interesting, but so far they haven't led to any spectacular successes.

Exploiting local equivalence. A close look at Fig. 73 reveals another phenomenon that is often present in exact cover problems: The tiles marked `8` and `b`, near the upper right corner, form an 'H' shape, which could be reflected left-right to yield another valid tiling. In fact there are three other such H's in the picture; therefore Fig. 73 actually represents $2^4 = 16$ different solutions to the problem, although those solutions are "locally" equivalent.

It turns out that the 15 × 15 tiling problem in Fig. 73 has exactly 212 mutually incongruent solutions, each of which can be rotated and/or reflected to make a set of eight that are congruent to each other; and each of those solutions contains at least two H's. Algorithm X needs just 92 Gμ to find them all. But we can do even better, because of H-equivalence: A slight extension to the options of the exact cover problem will produce only the solutions for which every 'H' has just one of its two forms — and so does every '⌶', namely every 90° rotation of an 'H'. (See exercise 208.) This modified problem has just 16 solutions, which are obtained with only 26 Gμ of computation and compactly represent all 212.

bipair: A pair of pairs
secondary item
perfect matching
complete bipartite graph
permutations
4-cycle

In general, an exact cover might contain four distinct options α, β, α', β' for which α and β are disjoint, α' and β' are disjoint, and

$$\alpha + \beta \;=\; \alpha' + \beta'. \tag{97}$$

(The '+' sign here is like '$\cup$'; it stands for addition of binary vectors, when options are rows of a 0–1 matrix.) In such cases we say that $(\alpha, \beta; \alpha', \beta')$ is a *bipair*. Whenever $(\alpha, \beta; \alpha', \beta')$ is a bipair, every solution that contains both α and β leads to another solution that contains both α' and β', and vice versa. Thus we can avoid considering half of all such solutions if we exclude one of those alternatives. And it's easy to do that: For example, to exclude all cases that contain both α' and β', we simply introduce a new secondary item, and append it to options α' and β'.

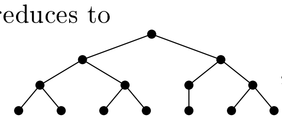
To illustrate this idea, let's apply it to the unsolvable toy problem (90). That problem has many bipairs, but we'll consider only two of them,

$$(\text{‘0 1’}, \text{‘2 4’}; \text{‘0 2’}, \text{‘1 4’}) \quad \text{and} \quad (\text{‘0 1’}, \text{‘2 5’}; \text{‘0 2’}, \text{‘1 5’}). \tag{98}$$

To avoid solutions that contain both '0 1' and '2 4', as well as those that contain both '0 2' and '1 5', we introduce secondary items A and B, and we extend four of the options (90) to

$$\text{‘0 1 A’}, \quad \text{‘0 2 B’}, \quad \text{‘1 5 B’}, \quad \text{‘2 4 A’}. \tag{99}$$

Then the search tree (91) reduces to


, (100)

and the former focusing problems disappear.

But wait, you say. Both of the bipairs in (98) involve the options '0 1' and '0 2'. Why is it legal to prefer different halves of those overlapping bipairs? Isn't it possible that we might "paint ourselves into a corner," if we allow ourselves to make arbitrary decisions about each of several interrelated bipairs?

That's a good question. Indeed, bad decisions *can* lead to trouble. Consider, for example, the problem of perfect matching on the complete bipartite graph $K_{3,3}$, which can be coded as an X2C with the nine options '$x\,X$' for $x \in \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}$ and $X \in \{\mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$. (The problem of perfect matching on $K_{n,n}$ is equivalent to finding the permutations of n elements; thus $K_{3,3}$ has $3! = 6$ perfect matchings.)

Every bipair ('t u', 'v w'; 't w', 'u v') in a perfect matching problem is equivalent to a 4-cycle t — u — v — w — t in the given graph. And if we disallow the right halves of the six bipairs

('x Y', 'y X'; 'x X', 'y Y') ('x Y', 'y Z'; 'x Z', 'y Y')
('y Y', 'z X'; 'y X', 'z Y') ('y Y', 'z Z'; 'y Z', 'z Y')
('z Y', 'x X'; 'z X', 'x Y') ('z Y', 'x Z'; 'z Z', 'x Y')

we obtain nine options that have no solution:

'x X A',	'y X B',	'z X C',	
'x Y C D',	'y Y A E',	'z Y B F',	(101)
'x Z E',	'y Z F',	'z Z D'.	

Fortunately, however, there's always a safe and easy way to proceed. We can assign an arbitrary (but fixed) ordering to the set of all options. Then, if for every bipair $(\alpha,\beta;\alpha',\beta')$ we always choose the half that contains $\min(\alpha,\beta,\alpha',\beta')$, the choices will be consistent.

More precisely, we can express every bipair in the canonical form

$$(\alpha,\beta;\alpha',\beta') \qquad \alpha<\beta,\ \alpha<\alpha',\text{ and }\alpha'<\beta', \tag{102}$$

with respect to any fixed ordering of the options. An exact covering is called *strong*, with respect to a set of such canonical bipairs, if its options don't include both α' and β' for any bipair in that set.

Theorem S. *If an exact cover problem has a solution, it has a strong solution.*

Proof. Every solution Σ corresponds to a binary vector $x=x_1\ldots x_M$, where $x_j=[\text{option } j \text{ is in } \Sigma]$. If Σ isn't strong, with respect to a given set of canonical bipairs, it violates at least one of those bipairs, say $(\alpha,\beta;\alpha',\beta')$. Thus there are indices j, k, j', k' with $j<k$, $j<j'$, and $j'<k'$ such that α, β, α', β' are respectively the jth, kth, j'th, k'th options, and such that $x_{j'}=x_{k'}=1$. By (97), $x_j=x_k=0$; and we obtain another solution Σ' by setting $x'_j\leftarrow x'_k\leftarrow 1$, $x'_{j'}\leftarrow x'_{k'}\leftarrow 0$, otherwise $x'_i=x_i$. This vector x' is lexicographically greater than x; so we'll eventually obtain a strong solution by repeating the process. ▮

In particular, we're allowed to exclude both '0 1' and '2 4', as well as both '0 2' and '1 5', with respect to the bipairs (98), because we can choose an ordering in which options '1 4' and '2 5' precede the other options '0 1', '0 2', '1 5', '2 4'.

Another convenient way to make consistent choices among related bipairs is based on ordering the primary items, instead of the options. (See exercise 212).

It's interesting to apply this theory to the problem of perfect matching in the complete graph K_{2q+1}. We showed in (87) above that Algorithm X needs a *long* time — $\Omega(2^q q!)$ mems — to discover that this problem has no solution. But bipairs come to the rescue.

Indeed, K_{2q+1} has lots of bipairs, $\Theta(q^4)$ of them. A straightforward application of Theorem S, using the natural order '0 1' < '0 2' < $\cdots$ < '$(2q-1)$ $2q$' on the $\binom{2q+1}{2}$ options, solves the problem in $\Theta(q^4)$ mems, by using just $\Theta(q^3)$ of the bipairs. And a more clever way to order the options allows us to solve it in only $\Theta(q^2)$ mems, using just $\Theta(q^2)$ well-chosen bipairs. The search tree can in fact be reduced to just $2q+1$ nodes — which is optimum! Exercise 215 explains all.

***Preprocessing the options.** Sometimes the input to an XCC problem can be greatly simplified, because we can eliminate many of its options and/or items. The general idea of "preprocessing," which transforms one combinatorial problem into an equivalent but hopefully simpler one, is an important paradigm, which is often called *kernelization* for reasons that we shall discuss later.

Algorithm P below is a case in point. It takes any sequence of items and options that would be acceptable to Algorithm X or to Algorithm C, and produces another such sequence with the same number of solutions. Any solution of the new problem can in fact be converted to a solution of the original one, if desired.

strong
strong exact covering
binary vector
lexicographically greater
complete graph
preprocessing–
kernelization

generic option
color controls
blocked
forces
factoring
data structures

The algorithm is based entirely on two general principles, used repeatedly until they no longer apply:

- An option can be removed if it blocks all uses of some primary item.
- An item can be removed if some primary item always forces it to appear.

More precisely, let o be a generic option '$i_1\ i_2[{:}c_2]\ \ldots\ i_t[{:}c_t]$', where i_1 is primary and the other $t-1$ items might have color controls. When Algorithm C deals with option o, it covers i_1 in step C4 and commits the other items in step C5, thereby removing all options that aren't compatible with o. If this process causes some primary item p to lose its last remaining option, we say that p is "blocked" by o. In such a case o is useless, and we can remove it. For example, '$d\ e\ g$' can be removed from (6), because it blocks a; '1 s_4 s_6' can be removed from (17), because it blocks s_3; then '1 s_1 s_3' and '2 s_2 s_5' also go away, because they block s_4.

That was the first principle mentioned above, the one that removes options. The item-removing principle is similar, but more dramatic when it applies: Let p be a primary item, and suppose that p's options all contain an uncolored instance of some other item, i. In such a case we say that p "forces" i; and we can remove item i, because p must be covered in every solution and it carries i along. For example, a forces d in (6). Hence we can remove item d, shortening the second and fourth options to just '$a\ g$' and '$a\ f$'. Further simplifications now arise.

These two principles, blocking and forcing, are by no means a complete catalog of transformations that could be used to preprocess exact cover problems. For example, they are incapable of discovering the fact that (38) is a useless option in the pentomino problem, nor do they discover the simplifications that we deduced by factoring the Soma cube problem. (See the discussion before (46).) Exercise 219 discusses yet another way to discard superfluous options.

A "perfect" and "complete" preprocessor would in fact be able to recognize *any* problem that has at most one solution. We can't hope to achieve that, so we've got to stop somewhere. We shall limit ourselves to the removal of blocking and forcing, because those transformations can be done in polynomial time, and because no other easily recognizable simplifications are apparent.

Algorithm P discovers all such simplifications by systematically traversing the given items and options, using the same data structures that were enjoyed by Algorithm C. It cycles through all items i, trying first to remove i by studying what happens when i is covered. If that fails, it studies what happens when the items of options that begin with i are committed. It needs some small variations of the former 'cover' and 'hide' operations (compare with (12)–(15), (50)–(53)):

$$\text{cover}''(i) = \begin{cases} \text{Set } p \leftarrow \texttt{DLINK}(i)\text{. While } p \neq i, \\ \quad \text{hide}''(i) \text{ unless } \texttt{COLOR}(p) \neq 0, \\ \quad \text{then set } p \leftarrow \texttt{DLINK}(p) \text{ and repeat.} \end{cases} \tag{103}$$

$$\text{hide}''(p) = \begin{cases} \text{Do operation hide}(p)\text{; but also set } S \leftarrow x, \\ \quad \text{whenever } \texttt{LEN}(x) \text{ has been set to 0 and } x \leq N_1. \end{cases} \tag{104}$$

$$\text{uncover}''(i) = \begin{cases} \text{Set } p \leftarrow \texttt{ULINK}(i)\text{. While } p \neq i, \\ \quad \text{unhide}(i) \text{ unless } \texttt{COLOR}(p) \neq 0, \\ \quad \text{then set } p \leftarrow \texttt{ULINK}(p) \text{ and repeat.} \end{cases} \tag{105}$$

Langford pairs
n queens problem
MacMahon's triangle problem
pentominoes
one-sided

Algorithm P (*Preprocessing for exact covering*). This algorithm reduces a given XCC problem until no instances of blocking or forcing are present. It uses the data structures of Algorithm C, together with new global variables C and S.

P1. [Initialize.] Set the problem up in memory, as in step C1 of Algorithm C. (Again there are N items, of which N_1 are primary.) Also set $C \leftarrow 1$. If there's an item $i \le N_1$ with $\texttt{LEN}(i) = 0$, go to P9.

P2. [Begin a round.] If $C = 0$, go to P10. Otherwise set $C \leftarrow 0$, $i \leftarrow 1$.

P3. [Is item i active?] If $i = N$, return to P2. Otherwise if $\texttt{LEN}(i) = 0$, go to P8.

P4. [Cover i.] Set $S \leftarrow 0$. Use (103) to cover item i. Then go to P7 if $S \ne 0$; otherwise set $x \leftarrow \texttt{DLINK}(i)$.

P5. [Try x.] If x isn't the leftmost remaining node of its option, go to P6. Otherwise use the method of exercise 220 to test whether this option blocks some primary item. If so, set $C \leftarrow 1$, $\texttt{TOP}(x) \leftarrow S$, and $S \leftarrow x$.

P6. [Try again.] Set $x \leftarrow \texttt{DLINK}(x)$, and return to P5 if $x \ne i$. Otherwise uncover item i using (105); use the method of exercise 221 to delete all options that were stacked in step P5; and go to P8.

P7. [Remove item i.] Uncover item i (which is forced by the primary item S). Then use the method of exercise 222 to delete or shorten every option that uses item i. Finally, set $C \leftarrow 1$, $\texttt{DLINK}(i) \leftarrow \texttt{ULINK}(i) \leftarrow i$, $\texttt{LEN}(i) \leftarrow 0$.

P8. [Loop on i.] Set $i \leftarrow i + 1$ and return to P3.

P9. [Collapse.] Set $N \leftarrow 1$ and delete all options. (The problem is unsolvable.)

P10. [Finish.] Output the reduced problem, whose items are those for which $\texttt{LEN}(i) > 0$ or $i = N = 1$, and terminate. (See exercise 223.) ▮

How effective is Algorithm P? Well, sometimes it spins its wheels and finds absolutely nothing to simplify. For example, the options (16) for n Langford pairs contain no instances of blocking or forcing when $n > 5$. Neither do the options for the n queens problem when $n > 3$. There's no "excess fat" in those specifications. In MacMahon's triangle problem (exercise 126), Algorithm P needs just 20 megamems to remove 576 of the 1537 options; but the options that it removes don't really matter, because Algorithm C traverses exactly the same search tree, with or without them.

We do gain 10% when we try to pack pentominoes into a 6×10 box (exercise 271): Without preprocessing, Algorithm X needs 4.11 Gμ to discover all 2339 solutions to that classic task. But Algorithm P needs just 0.19 Gμ to remove 235 of the 2032 options, after which Algorithm X finds the same 2339 solutions in 3.52 Gμ; so the total time has been reduced to 3.71 Gμ. The similar problem of packing the *one-sided* pentominoes into a 6×15 box has an even bigger payoff: It has 3308 options without preprocessing, and 15.5 Tμ are needed to process them. But after preprocessing—which costs a mere 260 Mμ—there are 3157 options, and the running time has decreased to 13.1 Tμ.

The simplifications discovered by Algorithm P for those pentomino problems involve only blocking (see exercise 225). But more subtle reductions occur in the

Y pentomino problem of Fig. 73. For example, cell 20 is forced by cell 10, in that problem; and in round 2, cell 00 is forced by cell 22. In round 4, cell 61 is blocked by the option '50 51 52 53 62' — a surprising discovery! Unfortunately, however, those clever reductions have little effect on the overall running time.

Preprocessing really shines on the problem of exercise 114, which asks for all sudoku solutions that are self-equivalent when reflected about their main diagonal. In this case Algorithm P is presented with 5410 options that involve intricate color controls, on 585 primary items and 90 secondary items. It rapidly reduces them to just 2426 options, on 506 primaries and 90 secondaries; and Algorithm C needs only 287 Gμ to process the reduced options and to discover the 30,286,432 solutions. That's 7.5 times faster than the 2162 Gμ it would have needed without reduction.

Thus, preprocessing is a mixed bag. It might win big, or it might be a waste of time. We can hedge our bets by allocating a fixed budget — for instance by deciding that Algorithm P will be allowed to run at most a minute or so. Its data structures are in a "safe" state at the beginning of step P3; therefore we can jump from there directly to step P10 if we don't want to run to completion.

Of course, preprocessing can also be applied to the subproblems that arise in the midst of a longer computation. A careful balancing of different strategies might be the key to solving problems that are especially tough.

Minimum-cost solutions. Many of the exact cover problems that we've been studying have few solutions, if any. In such cases our joy is to discover the rare gems. But in many other cases the problems have solutions galore; and for such problems we've focused our attention so far on the task of minimizing the amount of time per solution, assuming that all of the solutions are interesting.

A new perspective arises when each option of our problem has been assigned a nonnegative *cost*. Then it becomes natural to seek solutions of minimum cost. And ideally we'd like to do this without examining very many of the high-cost solutions at all; they're basically useless, but a low-cost solution might be priceless.

Fortunately there's a reasonably simple way to modify our algorithms, so that they will indeed find minimum-cost solutions rather quickly. But before we look at the details of those modifications, it will be helpful to look at several examples of what is possible.

Consider, for instance, the problem of Langford pairs from this point of view. We observed near the very beginning of Chapter 7 that there are $2L_{16} =$ 653,443,600 ways to place the 32 numbers $\{1, 1, 2, 2, \ldots, 16, 16\}$ into an array $a_1 a_2 \ldots a_{32}$ so that exactly i entries lie between the two occurrences of i, for $1 \le i \le 16$. And we claimed that the pairing displayed in 7–(3), namely

$$2\ 3\ 4\ 2\ 1\ 3\ 1\ 4\ 16\ 13\ 15\ 5\ 14\ 7\ 9\ 6\ 11\ 5\ 12\ 10\ 8\ 7\ 6\ 13\ 9\ 16\ 15\ 14\ 11\ 8\ 10\ 12, \qquad (106)$$

is one of 12,016 solutions that maximize the sum $\Sigma_1 = \sum_{k=1}^{32} k a_k$. Consequently the *reverse* of that pairing, namely

$$12\ 10\ 8\ 11\ 14\ 15\ 16\ 9\ 13\ 6\ 7\ 8\ 10\ 12\ 5\ 11\ 6\ 9\ 7\ 14\ 5\ 15\ 13\ 16\ 4\ 1\ 3\ 1\ 2\ 4\ 3\ 2, \qquad (107)$$

Y pentomino
sudoku solutions
self-equivalent
minimum-cost sols–
cost
Langford pairs
reverse of a string

is one of 12,016 solutions that *minimize* Σ_1. We noted in (16) above that Langford pairs are the solutions to a simple exact cover problem, whose options '$i\ s_j\ s_k$' represent the assignments $a_j = i$ and $a_k = i$. Therefore, if we associate the cost $\$(ji + ki)$ with option '$i\ s_j\ s_k$', the minimum-cost solutions will be precisely the Langford pairings that minimize Σ_1. (See exercise 226.)

16 queens problem
n queens
distance, in a plane

One way to minimize the total cost is, of course, to visit all solutions and to compute the individual sums. But there's a better way: The min-cost variant of Algorithm X below, which we shall call Algorithm X$^\$$, finds a solution of cost \$3708 and proves its minimality after only 60 gigamems of computation. That's more than 36 times faster than the use of plain vanilla Algorithm X, which needs 2.2 *teramems* to run through the full set of solutions.

Moreover, Algorithm X$^\$$ doesn't stop there. It actually will compute the K solutions of least cost, for any given value of K. For example, if we take $K = 12500$, it needs just 70 gigamems to discover the 12,016 solutions of cost \$3708, together with 484 solutions of the next-lowest cost (which happens to be \$3720).

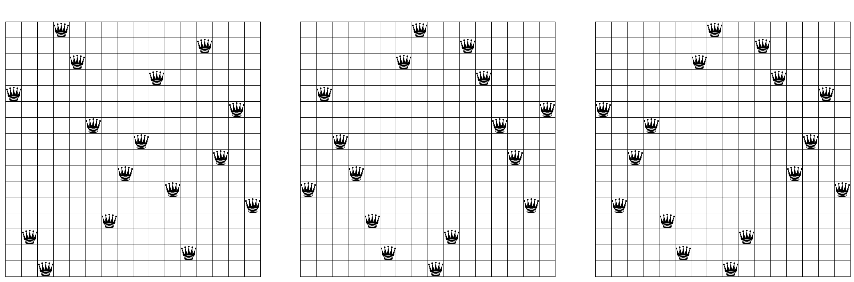
The news is even better when we try to minimize $\Sigma_2 = \sum_{k=1}^{32} k^2 a_k$ instead of Σ_1. Algorithm X$^\$$ needs just 28 Gμ to prove that the minimum Σ_2 is \$68880. And better yet is the fact, obtained in only 10 Gμ, that the minimum of $\sum_{k=1}^{32} k a_k^2$ is \$37552, obtainable *uniquely* by the remarkable pairing

$$16\ 14\ 15\ 9\ 6\ 13\ 5\ 7\ 12\ 10\ 11\ 6\ 5\ 9\ 8\ 7\ 14\ 16\ 15\ 13\ 10\ 12\ 11\ 8\ 4\ 1\ 3\ 1\ 2\ 4\ 3\ 2, \qquad (108)$$

which also happens to minimize both Σ_1 and Σ_2. (See exercise 229.)

Another classic combinatorial task, the 16 queens problem, provides another instructive example. We know from previous discussions that there are exactly 14,772,512 ways to place 16 nonattacking queens on a 16×16 board. We also know that Algorithm X needs about 40 Gμ of computation to visit them all, when we give it options like (23).

Let's suppose that the cost of placing a queen in cell (i, j) is the *distance* from that cell to the center of the board. (If we number the rows and columns from 1 to 16, that distance $d(i, j)$ is $\sqrt{(i - 17/2)^2 + (j - 17/2)^2}$; it varies from $d(8, 8) = 1/\sqrt{2}$ to $d(1, 1) = 15/\sqrt{2}$.) Thus we want to concentrate the queens near the center as much as possible, although many of them must lie at or near the edges because there must be one queen in each row and one queen in each column.



(a) (b) (c)

Fig. 74. Optimum solutions to the 16 queens problem, placing them (a) as close as possible to the center, or (b, c) as far as possible from it.

Figure 74(a) shows how to minimize the total cost — and this answer actually turns out to be unique, except for rotation and reflection. Similarly, Figs. 74(b) and 74(c) show the placements that *maximize* the cost. (Curiously, those solutions are obtainable from each other by reflecting the middle eight rows, without changing the top four or the bottom four.) Algorithm X$^\$$, with $K = 9$, discovers and proves the optimality of those placements in just (a) 3.7 Gμ; (b, c) 0.8 Gμ.

The modifications that convert Algorithm X to Algorithm X$^\$$ also convert Algorithm C into Algorithm C$^\$$. Therefore we can find minimum-cost solutions to XCC problems, which go well beyond ordinary exact cover problems.

For example, here's a toy problem that now becomes tractable: *Put ten different 5-digit prime numbers into the rows and columns of a* 5×5 *array, in such a way that their product is as small as possible.* (A 5-digit prime number is one of the 8363 primes between 10007 and 99991, inclusive.) One such "prime square," made up entirely of primes that are less than 30000, is

$$\begin{bmatrix} 2&1&2&1&1\\ 2&0&1&0&1\\ 1&1&0&0&3\\ 1&1&0&6&9\\ 1&1&1&1&3 \end{bmatrix}. \qquad (109)$$

To set this up as an XCC problem, introduce ten primary items $\{a_1, a_2, a_3, a_4, a_5\}$ and $\{d_1, d_2, d_3, d_4, d_5\}$ that represent "across" and "down," together with 25 secondary items ij for $1 \le i, j \le 5$ that represent cells of the array, together with 8363 additional secondary items $p_1p_2p_3p_4p_5$, one for eligible prime $p = p_1p_2p_3p_4p_5$. The options for placing p in row i or column j are then

$$\begin{array}{l} `a_i\ \ i1{:}p_1\ \ i2{:}p_2\ \ i3{:}p_3\ \ i4{:}p_4\ \ i5{:}p_5\ \ p_1p_2p_3p_4p_5\text{';}\\ `d_j\ \ 1j{:}p_1\ \ 2j{:}p_2\ \ 3j{:}p_3\ \ 4j{:}p_4\ \ 5j{:}p_5\ \ p_1p_2p_3p_4p_5\text{'.} \end{array} \qquad (110)$$

For example, 'a_4 41:1 42:1 43:0 44:6 45:9 11069' enables the prime 11069 in (109).

This is a good example where preprocessing is helpful, because the primes that are usable in a_1 and d_1 must not contain a 0; furthermore, the primes that are usable in a_5 and d_5 must contain only the digits $\{1, 3, 7, 9\}$. Algorithm P discovers those facts on its own, without being told anything special about number theory. It reduces the 83630 options of (110) to only 62900; and those reductions provide useful clues for the choices of items on which to branch.

The Monte Carlo estimate of exercise 86 tells us that there are roughly 6×10^{14} different ways to fit ten primes into a 5×5 array — a vast number. We probably don't need to look at too many of those possibilities, yet it isn't easy to decide which of them can safely be left unexamined.

To minimize the product of the primes, we assign the cost $\$(\ln p)$ to each of the options in (110). (This works because the logarithm of a product is the sum of the logarithms of the factors.) More precisely, we use the cost $\$\lfloor C \ln p \rfloor$, where C is large enough to make truncation errors negligible, but not large enough to cause arithmetic overflow, because Algorithm C$^\$$ wants all costs to be integers.

Every solution has the same cost as its transpose. Thus we can get the best five prime squares by asking Algorithm C$^\$$ to compute the $K = 10$ least-cost

XCC problems
prime square
preprocessing
Monte Carlo
truncation errors
arithmetic overflow
transpose

preprocessing
USA
contiguous states
planar graph
exact cover problem
exterior cost

solutions, each of which occurs twice. Here they are, with the best at the left:

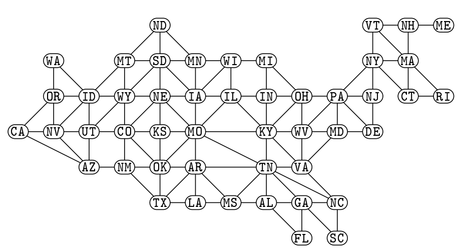
$$\begin{bmatrix}1&1&1&1&3\\1&0&1&0&3\\1&1&0&0&3\\3&1&7&6&9\\1&1&1&7&1\end{bmatrix}\begin{bmatrix}1&1&1&1&3\\1&0&1&0&3\\1&1&0&0&3\\3&1&7&7&1\\1&1&1&9&7\end{bmatrix}\begin{bmatrix}1&1&1&1&3\\1&0&0&0&7\\1&1&0&0&3\\3&1&6&9&9\\1&1&1&1&7\end{bmatrix}\begin{bmatrix}1&1&1&1&3\\1&0&0&0&7\\1&1&0&0&3\\3&1&6&6&3\\1&1&1&7&7\end{bmatrix}\begin{bmatrix}1&1&1&1&3\\1&0&0&0&7\\1&1&0&0&3\\3&1&6&6&3\\1&1&1&9&7\end{bmatrix}. \qquad (111)$$

The running time, 440 Gμ, would have been 1270 Gμ without preprocessing; so the 280 Gμ spent in preprocessing paid off. But the five greatest-cost solutions,

$$\begin{bmatrix}9&9&9&8&9\\8&8&9&9&7\\9&8&6&8&9\\9&9&7&9&3\\9&9&9&9&1\end{bmatrix}\begin{bmatrix}9&9&9&8&9\\8&9&8&9&9\\9&6&7&9&9\\9&8&7&3&7\\9&9&9&9&1\end{bmatrix}\begin{bmatrix}9&9&9&8&9\\8&8&9&9&7\\9&8&8&9&7\\9&9&5&7&1\\9&9&9&7&1\end{bmatrix}\begin{bmatrix}9&9&9&8&9\\8&8&9&9&7\\9&8&8&9&7\\9&9&5&8&1\\9&9&9&9&1\end{bmatrix}\begin{bmatrix}9&9&9&8&9\\8&8&9&9&7\\9&8&8&9&7\\9&9&5&7&7\\9&9&9&7&1\end{bmatrix} \qquad (112)$$

(greatest at the right), can be found in just 22 Gμ, without preprocessing.

Let's turn now from purely mathematical problems to some "organic" scenarios that are more typical of the real world. The USA's 48 contiguous states



(113)

define an interesting planar graph that has already supplied us with a variety of instructive examples. This graph G has 48 vertices and 105 edges. Suppose we want to partition it into eight connected subgraphs of six vertices each. What's the minimum number of edges whose removal will do that?

Well, exercise 7.2.2–76 has told us how to list all of the connected subsets of six states, and there happen to be 11505 of them. That gives us 11505 options for an exact cover problem on 48 items, whose solutions are precisely the potential partitions of interest. The total number of solutions turns out to be 4,536,539; and Algorithm X is able to visit them all, at a cost of 807 gigamems.

But let's try to do better, using Algorithm X$^\$$. Every induced subgraph $G|U$ has an *exterior cost*, which is the number of edges from U to vertices not in U. When we partition a graph by removing edges, every such edge contributes to the exterior cost of two of the components that remain; hence the number of removed edges is exactly half the sum of the exterior costs. The best partition therefore corresponds to the minimum-cost solution to our exact cover problem, if we assign the exterior costs to each option. For example, one of the 11505 options is

'`ND SD NE KS OK TX`', (114)

and we assign a cost of \$19 to that option.

Algorithm X$^\$$ now obligingly finds, in just 3.2 gigamems, that the optimum solution costs \$72. Hence we get the desired partition by removing only 36 edges.

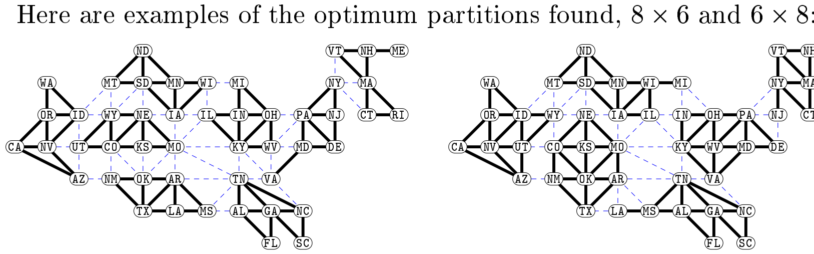
New England
census data
population

Before we look at the answer, let's stare at the problem a bit longer, because we still haven't discovered the best way to solve it! A closer examination shows that option (114) is useless, because it could never actually appear in any solution: It cuts the graph into two pieces, with 11 states to the left and 31 states to the right. (We encountered a similar situation earlier in (38).) In fact, 4961 of the 11505 options turn out to be unusable, for essentially the same reason. The state of Maine (`ME`), for example, belongs of 25 connected subgraphs of order 6; but we can easily see that the only way to get `ME` into the final partition is to group it with the other five states of New England (`NH`, `VT`, `MA`, `CT`, `RI`). Exercise 242 explains how to detect and reject the useless options quickly.

The remaining problem, which has 6544 options, is solved by Algorithm X in 327 Gμ and by Algorithm X$^{\$}$ in just 1046 Mμ.

Essentially the same methods will partition the graph nicely into six connected clusters of order eight. This time the exact cover problem has 40520 options after reduction, and a total of 4,177,616 solutions. But Algorithm X$^{\$}$ needs less than 2 Gμ to determine the minimum cost, which is \$54.

Here are examples of the optimum partitions found, 8×6 and 6×8:



(115)

In each case the optimum can actually be achieved in two ways: On the left, one could swap the affiliations of `VA` and `WV`; on the right, a more complicated cyclic shuffle (`MI NE LA VA`) could be used.

It's also instructive to solve a different kind of problem, namely to use census data and to partition G based on the *population* of each state. For example, let's try to find eight connected clusters that each contain nearly the same number of people. The total population, P, of the 48 states was officially 306084180 in 2010. We want each cluster to represent $P/8$ people, or as close to $P/8$ as we can get. That's about 38 million people per cluster.

The algorithm of exercise 242 will find and reduce all connected subgraphs whose total population x satisfies $L \le x < U$, for any given bounds L and U. If we take $L > \lfloor P/9 \rfloor$ (which is about 34 million) and $U \le \lceil P/7 \rceil$ (which is about 44 million), those candidate subgraphs will define an exact cover problem for which every solution uses exactly eight options, because $9x > P$ and $7x < P$.

That algorithm proves that G contains 1,926,811 connected sets of states whose population lies in $[34009354\,..\,43726312)$; and it prunes away 1,571,057 of them, leaving 355,754. But that's overkill. This problem has enough flexibility that its final solution can be expected to contain only sets whose population is quite close to 38 million. Therefore we might as well restrict ourselves to the range $[37000000\,..\,39000000)$ instead. There are 34,111 such options; surely they should be enough to solve our problem.

Well, that's very plausible, but unfortunately it doesn't work: Those 34,111 options have no solution, because Algorithm X can't use them to cover `NY` (New York)! Notice that `NY` is an articulation point of G. The population of New York is about 19.4 million, and the combined population of the six New England states is about 14.3 million. Whatever option covers New York had better cover all of New England too, otherwise New England is stranded. So that makes $19.4 + 14.3 = 33.7$ million people. New York's only other neighbors are New Jersey (8.8 million) and Pennsylvania (12.7 million); adding either of them will put us over 42 million.

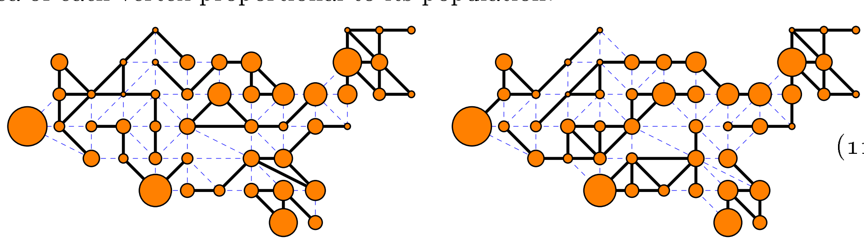
articulation point
New York
New England
pi as source

So we're clearly not going to be able to cover New York with a cluster that's close to the desired 38 million. We'll either need a lightweight one (New York plus New England) or a heavyweight one (with New Jersey too). Let's throw those two options in with the other 34,111.

Notice that this problem is quite different from the others we've been discussing, because its options vary greatly in size. One of the options contains just one state, `CA` (California), whose population is the largest (37.3 million); others contain up to fifteen states, almost spanning the continent from `DE` to `NV`.

Now we assign the cost $\$(x^2)$ to each option with population x, because the minimum-cost solutions will then minimize the squared deviations $(x_1 - P/8)^2 + \cdots + (x_8 - P/8)^2$. (See exercise 243.) This works well; and Algorithm X$^\$$ needs only 3.3 gigamems to find the optimum solution below. The seven options not involving New York all contain between 37.3 and 38.1 million people.

A similar analysis, partitioning into six equipopulated clusters instead of eight, gives in 1.1 Gμ a minimum-cost solution whose six populations are all in the range [50650000 . . 51150000]. Both solutions are illustrated here, with the area of each vertex proportional to its population:



(116)

In both cases the solution is unique. (And in both cases, let's face it, the solution is also pretty weird. Partitions like this could only be concocted by a computer. Exercise 246 discusses approximate solutions that are less eccentric.)

***Implementing the min-cost cutoffs.** OK, we've now seen lots of reasons why Algorithms X$^\$$ and C$^\$$ are desirable. But how exactly can we obtain those algorithms by extending Algorithm X and C? It will suffice to describe Algorithm C$^\$$.

The mission of Algorithm C$^\$$ is to find the K min-cost solutions. More precisely, it should discover K solutions whose total cost is as small as possible, with the understanding that different solutions might have the same cost. Let's imagine, for example, a problem that has exactly ten solutions, and that their costs are \$3, \$1, \$4, \$1, \$5, \$9, \$2, \$6, \$5, \$3, in the order that Algorithm C would discover them. Algorithm C$ won't differ from Algorithm C until it has found K solutions, because those K might turn out to be the best. After K are known, however, it will be harder to please: It will accept a new solution only if that solution is better than one of the best K it knows. Thus if $K = 3$, say, the accepted solutions will have costs \$3, \$1, \$4, \$1, \$2; Algorithm C$ won't find the other five.

To implement that behavior, we maintain a `BEST` table, which contains the K least costs known so far. That table is "heap ordered," with

$$\texttt{BEST}[\lfloor j/2\rfloor] \geq \texttt{BEST}[j] \qquad \text{for } 1 \leq \lfloor j/2\rfloor < j \leq K \tag{117}$$

(see Eq. 5.2.3–(3)). In particular, `BEST[1]` will be the greatest of the least K costs, and we call it the *cutoff value*, T. Algorithm C$ will reject any solution whose cost is T or more. Initially, $\texttt{BEST}[j] = \infty$ for $1 \leq j \leq K$; then every new solution of cost $c < T$ will be "sifted" into the `BEST` table as in Algorithm 5.2.3H. The successive cutoff values in the example above, if $K = 3$, would therefore be ∞, ∞, 4, 3, 3, 3, 2, 2, 2, 2. If $K = 4$ they'd be ∞, ∞, ∞, 4, 4, 4, 3, 3, 3, 3.

Algorithm C$ adds a `COST` field to every node, thereby making each node 64 bits larger than before. Step C1$ stores the cost of every option, assumed to be a nonnegative integer, in each node belonging to that option.

The costs in every list of options created by step C1$ are *ordered*, so that

$$\texttt{COST}(x) \leq \texttt{COST}(y) \qquad \text{if } y = \texttt{DLINK}(x), \tag{118}$$

whenever neither x nor y is a header node. Therefore, if p is primary and belongs to t options, we have $\texttt{COST}(x_1) \leq \texttt{COST}(x_2) \leq \cdots \leq \texttt{COST}(x_t)$, where $x_1 = \texttt{DLINK}(p)$, $x_{j+1} = \texttt{DLINK}(x_j)$ for $1 \leq j < t$, and $p = \texttt{DLINK}(x_t)$. This fact will allow us to ignore options that are too expensive to be part of a min-cost solution.

For this purpose we generalize the basic operations of covering, purifying, uncovering, and unpurifying (see (50)–(57)), by including a *threshold* parameter ϑ: Their loops in Algorithm C$ now say 'While $q \neq i$ and $\texttt{COST}(q) < \vartheta$' instead of simply 'While $q \neq i$'. We also change the uncovering and unpurifying operations, so that they now go downward using `DLINK`s instead of upward using `ULINK`s. Furthermore, we make the unhiding operation of (15) go from left to right, with q increasing, just as hiding does in (13). (These conventions clearly flout the rules by which we established the validity of dancing links in the first place! But we're lucky, because they're justified by the theory in exercise 2.)

At level l of the search, Algorithm C$ has constructed a partial solution, consisting of l options represented by nodes $x_0 \ldots x_{l-1}$. Let C_l be their total cost. In step C4$ we set $x_l \leftarrow \texttt{DLINK}(i)$, then cover item i using the threshold value $\vartheta_0 = T - C_l - \texttt{COST}(x_l)$. (Item i will have been chosen so that $\vartheta_0 > 0$.) The covering process will now proceed faster than before, if ϑ_0 is fairly low, because it won't bother to hide options that could not be in an accepted solution. We need to remember the value of ϑ_0, so that exactly the same threshold will be used when backtracking; therefore step C4$ sets $\texttt{THO}[l] \leftarrow \vartheta_0$, and step C7$ uses $\texttt{THO}[l]$ as the threshold for uncovering item i, where `THO` is an auxiliary array.

The cutoff value T decreases as computation proceeds. Therefore the threshold $\vartheta = T - C_l - \texttt{COST}(x_l)$ used in step C5$ for covering and purification might

BEST
heap ordered
cutoff value
COST
covering
purifying
uncovering
unpurifying
threshold
unhiding
hiding
dancing links
backtracking

be different each time. Step C5$^\$$ should go directly to C7$^\$$ if $\vartheta \le 0$. Otherwise it sets TH[l] $\leftarrow \vartheta$ in that step, and uses TH[l] for undoing in step C6$^\$$, where TH is another auxiliary array.

Step C3$^\$$, which chooses the item on which to branch at level l, is of course crucially important. If some primary item i has no options, or if the cost of its least expensive option is so high that it can't lead to a solution better than we've already found, step C3$^\$$ should jump immediately to step C8$^\$$. Otherwise, many strategies are worthy of investigation, and there's room here to discuss only the method that was used in the author's experiments: Good results were obtained by choosing an i with the fewest not-too-costly options, as in the MRV heuristic. In case of ties, the author's implementation chose an i whose least expensive option cost the *most*. (That item must be covered sooner or later, so there's no way to avoid paying that much. We probably have a better chance of reaching a cutoff quickly if we maximize our chances of failure.) Exercise 248 has full details.

Many applications of Algorithm C$^\$$ have special features that allow us to prune unproductive branches from the search tree long before they would be cut off by the methods discussed so far. For example, every option in our "square of primes" problem has exactly one primary item (see (110)). In such cases, we know that every solution obtained by extending $x_0 \ldots x_{l-1}$ must cost at least

$$C_l + \mathtt{COST}(\mathtt{DLINK}(i_1)) + \cdots + \mathtt{COST}(\mathtt{DLINK}(i_t)), \qquad (119)$$

because of (118), where $i_1, \ldots, i_t$ are the primary items still active. If this total is T or more, step C3$^\$$ can proceed immediately to step C8$^\$$.

Similarly, in the n queens problem, every option has exactly two primary items, one of the form R_i and one of the form C_j. The active items $i_1, \ldots, i_t$ must therefore contain $t/2$ of each. Let C_R and C_C be $\sum \mathtt{COST}(\mathtt{DLINK}(i_j))$, summed over those types. If *either* $C_l + C_\mathrm{R}$ *or* $C_l + C_\mathrm{C}$ is $\ge T$, step C3$^\$$ can jump to C8$^\$$.

In our first problem for the contiguous USA, every option has exactly 6 items. (See (114).) Hence the number of active items, t, is always a multiple of 6. Exercise 249 presents a nice algorithm to find the least possible cost of $t/6$ future options; step C3$^\$$ of Algorithm C$^\$$ uses that method to find early cutoffs.

The Langford pair problem has options with three items, one of which is a digit. The pentomino problems have options with six items, one of which is a piece name. In both cases, Algorithm C$^\$$ can obtain suitable lower bounds for early cutoffs by *combining* the strategies already mentioned. (See exercise 250.)

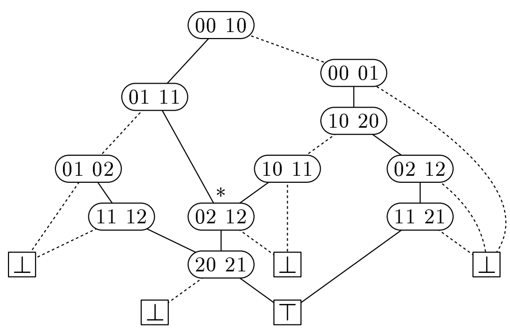
Finally, Algorithm C$^\$$ uses one other important technique: It gains traction by *preprocessing the costs*. Notice that *if p is a primary item, and if the cost of every option that includes p is c or more, we could decrease the cost of all those options by c, without changing the set of min-cost solutions.* That's true because p is going to appear exactly once in every solution. We can think of c as an unavoidable *tax* or "cover charge," which must be paid "up front."

In general there are many ways to preprocess the costs without changing the underlying problem. Properly transformed costs can help the algorithm's heuristics to make much more intelligent choices. Exercise 247 discusses the simple method that was used for step C1$^\$$ in the author's experiments discussed earlier.

author
MRV heuristic
square of primes
n queens problem
contiguous USA
Langford pair
pentomino
preprocessing the costs
marginal costs
tax
author

ZDDs
bipair

***Dancing with ZDDs.** The solutions visited by Algorithm X in step X2 can be represented naturally in the form of a decision tree, as we discussed in Section 7.1.4. For example, here's a decision tree for the solutions to the problem of covering the eight cells of a 3×3 board with four dominoes, after the corner cell 22 has been removed:



(120)

This diagram uses the standard ZDD conventions: Every branch node names an option. A solid line means that the option is taken, while a broken line means that it is not. The terminal nodes $\bot$ and $\top$ represent failure and success. This problem has four solutions, corresponding to the four paths from the root to $\top$.

We learned in Section 7.1.4 that ZDDs can readily be manipulated, and that a small ZDD can sometimes characterize a large family of solutions. If we're lucky, we can save a huge amount of time and energy by simply generating an appropriate ZDD instead of visiting the solutions one by one.

Such economies arise when the same subproblem occurs repeatedly. For example, two branches come together in (120) at the node marked '$*$'; this happens because the problem that remains after placing two dominoes '00 10' and '01 11' is the same as the residual problem after placing '00 01' and '10 11'. "We've been there and done that." Hence we needn't recapitulate our former actions, if we've already built a subZDD to remember what we did. (Those two pairs of domino placements form a "bipair," as discussed earlier; but the ZDD idea is considerably more general and powerful.)

Let's look more closely at the underlying details. The exact cover problem solved by (120) has eight items 00, 01, 10, 02, 11, 20, 12, 21, representing cells to be covered; and it has the following ten options, representing domino placements:

$$\begin{array}{ccccc} 1{:}\,00\ 01 & 3{:}\,01\ 02 & 5{:}\,02\ 12 & 7{:}\,10\ 20 & 9{:}\,11\ 21 \\ 2{:}\,00\ 10 & 4{:}\,01\ 11 & 6{:}\,10\ 11 & 8{:}\,11\ 12 & 10{:}\,20\ 21 \end{array} \tag{121}$$

The ZDD (120) is internally represented as a sequence of branching instructions,

$$\begin{array}{llll} I_{12} = (\bar{2}?\,8{:}\,11), & I_9 = (\bar{8}?\,0{:}\,2), & I_6 = (\bar{5}?\,0{:}\,5), & \\ I_{11} = (\bar{4}?\,10{:}\,3), & I_8 = (\bar{1}?\,0{:}\,7), & I_5 = (\bar{9}?\,0{:}\,1), & I_3 = (\bar{5}?\,0{:}\,2), \\ & & & I_2 = (\overline{10}?\,0{:}\,1), \\ I_{10} = (\bar{3}?\,0{:}\,9), & I_7 = (\bar{7}?\,4{:}\,6), & I_4 = (\bar{6}?\,0{:}\,3), & \end{array} \tag{122}$$

where 0 and 1 stand for $\bot$ and $\top$. (See, for instance, 7.1.4–(8).) "If we don't take option 2, go to instruction 8; but if we do take it, continue with instruction 11."

A few modifications to Algorithm X will transform it from a solution-visiting method into a constructor of ZDDs. In fact, color controls can be handled too:

Algorithm Z (*Dancing with ZDDs*). Given an XCC problem as in Algorithm C, this algorithm outputs a free ZDD for the sets of options that satisfy it. The ZDD instructions $\{I_2, \ldots, I_s\}$ have the form $(\bar{o}_j?\ l_j\colon h_j)$ illustrated in (122), and I_s is the root. (But if the problem has no solutions, the algorithm terminates with $s = 1$, and the root is 0.) The data structures of Algorithm C are extended by a "memo cache" consisting of signatures `S[`j`]` and ZDD pointers `Z[`j`]`. Algorithm C's table of choices $x_0x_1\ldots$ is joined by two new auxiliary tables $m_0m_1\ldots$ and $z_0z_1\ldots$, indexed by the current level l.

Z1. [Initialize.] Set the problem up in memory, as in step C1 of Algorithm C. Also set N to the number of items, Z to the last spacer address, $l \leftarrow 0$, `S[0]` $\leftarrow 0$, `Z[0]` $\leftarrow 1$, $m \leftarrow 1$, $s \leftarrow 1$.

Z2. [Enter level l.] Form a "signature" σ that characterizes the current subproblem (see below). If $\sigma =$ `S[`t`]` for some t (this is a "cache hit"), set $\zeta \leftarrow$ `Z[`t`]` and go to Z8. Otherwise set `S[`m`]` $\leftarrow \sigma$, $m_l \leftarrow m$, $z_l \leftarrow 0$, and $m \leftarrow m+1$.

Z3. [Choose i.] At this point items $i_1, \ldots, i_t$ still need to be covered, as in step C3 of Algorithm C. Choose one of them, and call it i.

Z4. [Cover i.] Cover item i using (12), and set $x_l \leftarrow$ `DLINK(`i`)`.

Z5. [Try x_l.] If $x_l = i$, go to Z7. Otherwise set $p \leftarrow x_l + 1$, and do the following while $p \neq x_l$: Set $j \leftarrow$ `TOP(`p`)`; if $j \le 0$, set $p \leftarrow$ `ULINK(`p`)`; otherwise commit(p, j) and set $p \leftarrow p+1$. Set $l \leftarrow l+1$ and return to Z2.

Z6. [Try again.] Set $p \leftarrow x_l - 1$, and do the following while $p \neq x_l$: Set $j \leftarrow$ `TOP(`p`)`; if $j \le 0$, set $o \leftarrow 1-j$ and $p \leftarrow$ `DLINK(`p`)`; otherwise uncommit(p, j) and set $p \leftarrow p-1$. If $\zeta \neq 0$, set $s \leftarrow s+1$, output $I_s = (\bar{o}?\ z_l\colon \zeta)$, and set $z_l \leftarrow s$. Set $i \leftarrow$ `TOP(`x_l`)`, $x_l \leftarrow$ `DLINK(`x_l`)`, and return to Z5.

Z7. [Backtrack.] Uncover item i using (14). Then set `Z[`m_l`]` $\leftarrow z_l$ and $\zeta \leftarrow z_l$.

Z8. [Leave level l.] Terminate if $l = 0$. Otherwise set $l \leftarrow l-1$ and go to Z6. ▮

Important: The 'commit' and 'uncommit' operations in steps Z5 and Z6 should modify (54)–(57), by calling cover(j), hide(q), uncover(j), and unhide(q) instead of cover$'(j)$, hide$'(q)$, uncover$'(j)$, and unhide$'(q)$. These changes cause every step of Algorithm Z to be slightly different from the corresponding step of Algorithm C. (Yet only step Z2 has changed substantially.)

Exercise 253 shows that a few more changes will make Algorithm Z compute the total number of solutions, instead of (or in addition to) outputting a ZDD.

The keys to Algorithm Z's success are the *signatures* computed in step Z2. This computation is easy if there are no secondary items: The signature σ is then simply a bit vector of length N, containing 1 in every position i where item i is still active. The computation is, however, somewhat subtler in the presence of secondary items; exercise 254 has the details.

It's instructive to analyze some special cases. For example, suppose Algorithm Z is asked to find the perfect matchings of the complete graph K_N. This

XCC problem
memo cache
signature
cache hit
commit
uncommit
cover$'(j)$
hide$'(q)$
uncover$'(j)$
unhide$'(q)$
perfect matchings
matchings
complete graph

Fibonacci number

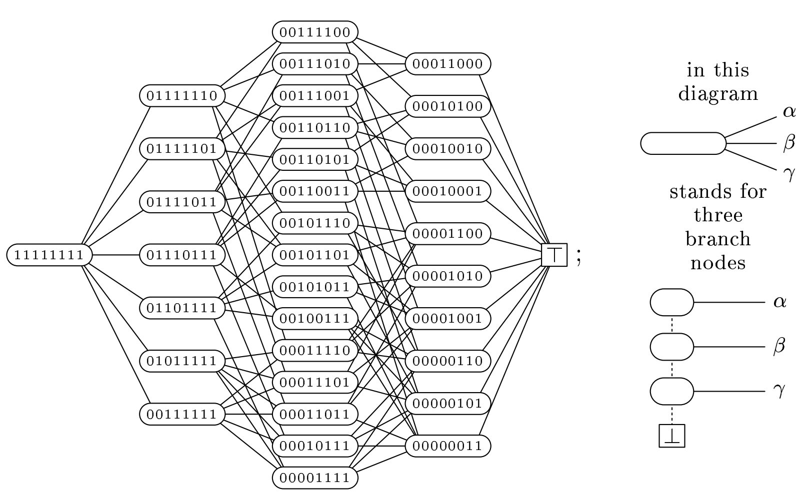
problem has N primary items $\{1,\ldots,N\}$, and $\binom{N}{2}$ options '$j\ k$' for $1 \le j < k \le N$. We noted earlier, in the discussion preceding (87), that every item list on level l has exactly $N-1-2l$ options, regardless of the choices made in step Z3. If we always choose the smallest uncovered item, step Z2 will compute exactly $\binom{N-l}{l}$ different signatures on level l, namely the signatures in which items $\{1,\ldots,l\}$ are covered and so are l of the other items $\{l+1,\ldots,N\}$. Hence the total number of cache entries is $\sum_{l=0}^{N}\binom{N-l}{l} = F_{N+1}$, a Fibonacci number(!). (See exercise 1.2.8–16.) Moreover, the main loops in steps Z5 and Z6 are executed $(N-1-2l)\binom{N-l}{l}$ times at level l, since steps Z3 and Z4 are executed $\binom{N-l}{l}$ times.

In fact, when N is even, the ZDD that is output for all perfect matchings of K_N turns out to have exactly $\sum_{l=0}^{N}(N-1-2l)\binom{N-l}{l} + 2$ nodes, which is approximately $\frac{N}{5}F_{N+1}$. Exercise 255 shows that the total running time to compute this ZDD is $\Theta(N^2F_N) = \Theta(N^2\phi^N)$; and the same estimate holds also when N is odd and the ZDD has only one node '$\bot$'. This is much smaller than the time needed by Algorithm X, which is $\Theta\bigl((N/e)^{N/2}\bigr)$.

More concretely, Algorithm X computes the 2,027,025 perfect matchings of K_{16} in about 360 megamems, using about 6 kilobytes of memory. Algorithm Z needs only about 2 megamems to characterize those matchings with a 10,228-node ZDD; but it uses 2.5 megabytes of memory. For K_{32} there are 191,898,783,962,510,625 perfect matchings, and the difference is even more dramatic: Algorithm X costs about 34 thousand petamems and 25 kilobytes; Algorithm Z costs about 16 gigamems and 85 megabytes, for a ZDD with 48 meganodes.

This example illustrates several important points: (1) Algorithm Z can greatly reduce the running time of Algorithm X (or Algorithm C), trading time for space. (2) These improvements can also be achieved for problems that have no solutions, like matchings of K_{2q+1}. (3) The number of nodes in the ZDD that is output might greatly exceed the number of memos in Algorithm Z's cache.

Let's take a closer look at the ZDDs. The output for $N=8$ is, schematically,



(123)

where, for example, '(00101101)' represents nodes for the signature 00101101.

invisible
Langford pairs
n queens problem
exact cover problems
XC
color codes

A signature represents a subproblem. If that subproblem has at least one solution, the ZDD for the full problem will have a subZDD that specifies all solutions of the subproblem. And if the signature is in cache location `S[`t`]`, the root of the corresponding subZDD will be stored in `Z[`t`]`, at the end of step Z7.

This subZDD has a very special structure, illustrated in (123). Suppose we branch on item i when working on signature σ; and suppose solutions are found for options o_1, o_2, ..., o_k in the list for item i. Then there will be subZDDs rooted at ζ_1, ζ_2, ..., ζ_k, associated with the subproblems whose signatures are $\sigma \setminus o_1$, $\sigma \setminus o_2$, ..., $\sigma \setminus o_k$. The net effect of steps Z3–Z6 is to construct a subZDD for σ that essentially begins with k conditional instructions:

$$o_k?\ \zeta_k\colon\ \ldots\ o_2?\ \zeta_2\colon o_1?\ \zeta_1\colon \bot\,. \tag{124}$$

(The ZDD is constructed from bottom to top, so it tests o_k first.)

For example, '00101101' in (123) is the root of the subZDD for the subproblem that needs to cover $\{3,5,6,8\}$. We branch on item 3, whose list has the three options '3 5', '3 6', '3 8'. Three branch instructions are output,

$$I_\theta = (\overline{3\ 5}?\ 0\colon \gamma), \quad I_\eta = (\overline{3\ 6}?\ \theta\colon \beta), \quad I_\zeta = (\overline{3\ 8}?\ \eta\colon \alpha);$$

here γ, β, α are the subZDDs for signatures 00000101, 00001001, 00001100, respectively. The subZDD for 00101101 begins at ζ.

Thus (123) illustrates a ZDD with $1\cdot 7 + 7\cdot 5 + 15\cdot 3 + 10\cdot 1 + 2 = 99$ nodes. Notice that the dotted links always go either to $\bot$ or to an "invisible" node, which is one of the $k-1$ subsidiary nodes in a chain of branches such as (124). Every invisible node has a single parent. But there is one visible node for each successful signature, and a visible node may have many parents.

Exercises 256–262 discuss a number of examples where Algorithm Z gives spectacular improvements over Algorithm X (and over Algorithm C when colors are involved). Many additional examples could also be given. But most of the exact cover problems we've been considering in our examples do *not* have an abundance of common subproblems, so they reap little benefit from the memo cache.

For example, consider our old standby, the problem of Langford pairs. Algorithm X needs 15 Gμ to show that there are no solutions when $n = 14$; Algorithm Z reduces this slightly, to 11 Gμ. Algorithm X needs 1153 Gμ to list the 326,721,800 solutions for $n = 16$; Algorithm Z computes that number in 450 Gμ; but it needs 20 *gigabytes* of memory, and produces a ZDD of 500 million nodes!

Similarly, Algorithm Z is not the method of choice for the n queens problem, or for word-packing problems, although it does yield modest speedups more often than one might suspect. Exercise 263 surveys some typical examples.

Summary. We began this section by observing that simple properties of linked lists can enhance the efficiency of backtracking, especially when applied to exact cover problems (XC). Then we noticed that a wide variety of combinatorial tasks, going well beyond matching, turn out to be special cases of exact covering.

The most important "takeaway," however, has been the fact that *color codes* lead to a significant generalization of the classical exact cover problem.

Indeed, the general XCC problem, "exact covering with colors," has a truly extraordinary number of applications. The exercises below exhibit dozens and dozens of instructive problems that are quite naturally describable in terms of "options," which involve "items" that may or may not be colored in certain ways. We've discussed Algorithm X (for XC problems) and Algorithm C (for XCC problems); and the good news is that those algorithms are almost identical.

Furthermore, we've seen how to extend Algorithm C in several directions: Algorithm M handles the general MCC problem, which allows items to be covered with different ranges of multiplicities. Algorithm $C^{\$}$ associates a cost with each option, and finds XCC solutions of minimum total cost. Algorithm Z produces XCC solutions as ZDDs, which can be manipulated and optimized in other ways.

Historical notes. The basic idea of (2) was introduced by H. Hitotumatu and K. Noshita [*Information Processing Letters* **8** (1979), 174–175], who applied it to the n queens problem. Algorithm 7.2.1.2X, which was published by J. S. Rohl in 1983, can be regarded as a simplified version of dancing links, for cases when singly linked lists suffice. (Indeed, as Rohl observed, the n queens problem is such a case.) Its extension to exact cover problems in general, as in Algorithm X above, was the subject of the author's tribute to C. A. R. Hoare in *Millennial Perspectives in Computer Science* (2000), 187–214, where numerous examples were given. [That paper was subsequently reprinted with additions and corrections as Chapter 38 of *FGbook*.] His original implementation, called DLX, used a more complex data structure than (10), involving nodes with four-way links.

Knuth extended Algorithm X to Algorithm C in November 2000, while thinking about two-dimensional de Bruijn sequences. A special case of Algorithm M, in which all multiplicities are fixed, followed in August 2004, when he was thinking about packing various sizes of bricks into boxes. The current form of Algorithm M was developed in January 2017, after he'd studied an independent generalization of Algorithm X that Wei-Hwa Huang had written in 2007.

The first computer programs for exact cover problems were developed independently by J. F. Pierce [*Management Science* **15** (1968), 191–209] and by R. S. Garfinkel and G. L. Nemhauser [*Operations Research* **17** (1969), 848–856]. In both cases the given options each had an associated cost, and the goal was to obtain minimum-cost solutions instead of arbitrary solutions. Both algorithms were similar, although they used different ways to prune nonoptimum choices: Items were chosen for branching according to a fixed, precomputed order, and options were represented as bit vectors. An option was never removed from its item list; it would repeatedly be rejected if its bits intersected with previously chosen items. (*Caution:* Literature from the operations research community traditionally reverses the roles of rows and columns in matrices like (5). For them, items are rows and options are columns, even though bit vectors look like rows.)

The concept of "dancing with ZDDs" was introduced by M. Nishino, N. Yasuda, S. Minato, and M. Nagata, in the *AAAI Conference on Artificial Intelligence* **31** (2017), 868–874, where they presented the special case of Algorithm Z in which all items are primary.

XCC problem
options
items
MCC problem
Historical notes
Hitotumatu
Noshita
n queens problem
Rohl
singly linked lists
author
Hoare
DLX
de Bruijn
Huang
exact cover problems
Pierce
Garfinkel
Nemhauser
cost
minimum-cost solutions
operations research
rows
columns
items
options
dancing with ZDDs
Nishino
Yasuda
Minato
Nagata

The history of XCC solving is clearly still in its infancy, and much more work needs to be done. For example, many applications will benefit from improved ways to choose an item for branching — especially in step M3 of Algorithm M, where only a few strategies have been explored so far. It's important to maintain a good "focus"; furthermore, techniques of "factoring" can dramatically prune away unproductive branches, as shown for example in exercise 343.

Algorithm M deserves to be extended to Algorithm M$^\$$, and perhaps also to produce ZDD output. A further generalization would be to allow each item of each option to have an associated weight. (Thus the associated matrix, analogous to (5), would not consist merely of 0s and 1s.)

Hence we can expect to see many continued advances in XCC solving.

focus
factoring
ZDD
weight
matrix
0s and 1s
doubly linked list
list head
LLINK
RLINK
delete
permutation
binary search tree
undelete
linear equations
incidence matrix
0–1 matrices
families of sets
partition
tiling

EXERCISES — First Set

▶ **1.** [*M25*] A doubly linked list of n elements, with a list head at 0, begins with `LLINK`$(k) = k-1$ and `RLINK`$(k-1) = k$ for $1 \le k \le n$; furthermore `LLINK(0)` $= n$ and `RLINK`$(n) = 0$, as in (3). But after we use operation (1) to delete elements a_1, a_2, $\ldots$, a_n, where $a_1a_2\ldots a_n$ is a permutation of $\{1, 2, \ldots, n\}$, the list will be empty and the links will be entangled as in (4).

a) Show that the final settings of `LLINK` and `RLINK` can be described in terms of the binary search tree that is obtained when the keys a_n, $\ldots$, a_2, a_1 (in reverse order) are inserted by Algorithm 6.2.2T into an initially empty tree.

b) Say that permutations $a_1a_2\ldots a_n$ and $b_1b_2\ldots b_n$ are equivalent if they both yield the same `LLINK` and `RLINK` values after deletion. How many distinct equivalence classes arise, for a given value of n?

c) How many of those equivalence classes contain just one permutation?

2. [*M30*] Continuing exercise 1, we know that the original list will be restored if we use (2) to undelete the elements a_n, $\ldots$, a_2, a_1, reversing the order of deletion.

a) Prove that it's restored *also* if we use the unreversed order a_1, a_2, $\ldots$, a_n (!).

b) Is the original list restored if we undelete the elements in *any* order whatsoever?

c) What if we delete only k of the elements, say $(a_1, \ldots, a_k)$, then undelete them in the same order $(a_1, \ldots, a_k)$. Is the list always restored?

3. [*20*] An $m \times n$ matrix that's supposed to be exactly covered can be regarded as a set of n simultaneous equations in m unknowns. For example, (5) is equivalent to

$$x_2+x_4 = x_3+x_5 = x_1+x_3 = x_2+x_4+x_6 = x_1+x_6 = x_3+x_4 = x_2+x_5+x_6 = 1,$$

where each $x_k = [\text{choose row } k]$ is either 0 or 1.

a) What is the general solution to those seven equations?

b) Why is this approach to exact cover problems almost never useful in practice?

4. [*M20*] Given a graph G, construct a matrix with one row for each vertex v and one column for each edge e, putting the value $[e \text{ touches } v]$ into column e of row v. What do the exact covers of this "incidence matrix" represent?

5. [*18*] Among the many combinatorial problems that can be formulated in terms of 0–1 matrices, some of the most important deal with *families of sets*: The columns of the matrix represent elements of a given universe, and the rows represent subsets of that universe. The exact cover problem is then to partition the universe into such subsets. In geometric contexts, an exact cover is often called a *tiling*.

Equivalently, we can use the terminology of hypergraphs, speaking of hyperedges (rows) that consist of vertices (columns); then the exact cover problem is to find a perfect matching, also called a perfect packing, namely a set of nonoverlapping hyperedges that hit every vertex.

Such problems generally have *duals*, which arise when we transpose the rows and columns of the input matrix. What is the dual of the exact cover problem, in hypergraph terminology?

6. [*15*] If an exact cover problem has N items and M options, and if the total length of all options is L, how many nodes are in the data structures used by Algorithm X?

7. [*16*] Why is `TOP(23)` $= -4$ in Table 1? Why is `DLINK(23)` $= 25$?

8. [*22*] Design an algorithm to set up the initial memory contents of an exact cover problem, as needed by Algorithm X and illustrated in Table 1. The input to your algorithm should consist of a sequence of lines with the following format:

- The very first line lists the names of all items.
- Each remaining line specifies the items of a particular option, one option per line.

9. [*18*] Explain how to branch in step X3 on an item i for which `LEN`(i) is minimum. If several items have that minimum length, i itself should also be minimum. (This choice is often called the "minimum remaining values" (MRV) heuristic.)

10. [*20*] In some applications the MRV heuristic of exercise 9 leads the search astray, because certain primary items have short lists yet convey little information about desirable choices. Modify answer 9 so that an item p whose name does not begin with the character '#' will be chosen only if `LEN`$(p) \leq 1$ or no other choices exist. (This tactic is called the "sharp preference" heuristic.)

▸ **11.** [*19*] Play through Algorithm X by hand, using exercise 9 in step X3 and the input in Table 1, until first reaching step X7. What are the contents of memory at that time?

▸ **12.** [*21*] Design an algorithm that prints the option associated with a given node x, cyclically ordering the option so that `TOP`(x) is its first item. Also print the position of that option in the vertical list for that item. (For example, if $x = 21$ in Table 1, your algorithm should print 'd f a' and state that it's option 2 of 3 in the list for item d.)

13. [*16*] When Algorithm X finds a solution in step X2, how can we use the values of $x_0 x_1 \ldots x_{l-1}$ to figure out what that solution is?

▸ **14.** [*20*] (*Problème des ménages.*) "In how many ways can n male-female couples sit at a circular table, with men and women alternating, and with no couples adjacent?"

a) Suppose the women have already been seated, and let the vacant seats be $(S_0, S_1, \ldots, S_{n-1})$. Let M_j be the spouse of the woman between seats S_j and $S_{(j+1) \bmod n}$. Formulate the ménage problem as an exact cover problem with items S_j and M_j.

b) Apply Algorithm X to find the solutions for $n \leq 10$. Approximately how many mems are needed per solution, with and without the MRV heuristic?

15. [*20*] The options in (16) give us every solution to the Langford pair problem twice, because the left-right reversal of any solution is also a solution. Show that, if a few of those options are removed, we'll get only half as many solutions; the others will be the reversals of the solutions found.

16. [*16*] What are the solutions to the four queens problem, as formulated in (23) and (24)? What branches are taken at the top four levels of Algorithm X's search tree?

17. [*16*] Repeat exercise 16, but consider a_j and b_j to be secondary items and omit the slack options (24). Consider the primary items in order r_3, c_3, r_2, c_2, r_4, c_4, r_1, c_1.

hypergraphs
hyperedges
vertices
perfect matching
matching, perfect
packing
duals
minimum remaining values
MRV
sharp preference
circular table
ménage problem
MRV heuristic
Langford pair
queens
secondary items

18. [*10*] What are the solutions to (6) if items e, f, and g are *secondary*?

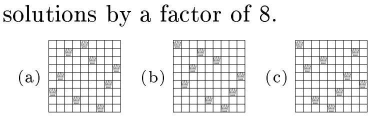
▸ **19.** [*21*] Modify Algorithm X so that it doesn't require the presence of any primary items in the options. A valid solution should not contain any purely secondary options; but it must intersect every such option. (For example, if only items a and b of (6) were primary, the only valid solution would be to choose options '$a\ d\ g$' and '$b\ c\ f$'.)

▸ **20.** [*25*] Generalize (26) to a pairwise ordering of options $(\alpha_0, \ldots, \alpha_{m-1}; \beta_0, \ldots, \beta_{m-1})$ that uses at most $\lceil \lg m \rceil$ of the secondary items $y_1, \ldots, y_{m-1}$ in each option. *Hint:* Think of binary notation, and use y_j at most $2^{\rho j}$ times within each of the α's and β's.

21. [*22*] Extend exercise 20 to k-wise ordering of km options α_j^i, for $1 \le i \le k$ and $0 \le j < m$. The solutions should be $(\alpha_{j_1}^1, \ldots, \alpha_{j_k}^k)$ with $0 \le j_1 \le \cdots \le j_k < m$. Again there should be at most $\lceil \lg m \rceil$ secondary items in each option.

▸ **22.** [*28*] Most of the solutions to the n queens problem are unsymmetrical, hence they lead to seven other solutions when rotated and/or reflected. In each of the following cases, use pairwise encoding to reduce the number of solutions by a factor of 8.

a) No queen is in either diagonal, and n is odd.
b) Only one of the two diagonals contains a queen.
c) There are two queens in the two diagonals.

(a) (b) (c)


23. [*28*] Use pairwise encoding to reduce the number of solutions by *nearly* a factor of 8 in the remaining cases not covered by exercise 22:

a) No queen is in either diagonal, and n is even.
b) A queen is in the center of the board, and n is odd.

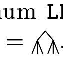
24. [*20*] With Algorithm X, find all solutions to the n queens problem that are unchanged when they're rotated by (a) 180°; (b) 90°.

25. [*20*] By setting up an exact cover problem and solving it with Algorithm X, show that the queen graph Q_8 (exercise 7.1.4–241) cannot be colored with eight colors.

26. [*21*] In how many ways can the queen graph Q_8 be colored in a "balanced" fashion, using eight queens of color 0 and seven each of colors 1 to 8?

27. [*22*] Introduce secondary items cleverly into the options (16), so that only *planar* solutions to Langford's problem are obtained. (See exercise 7–8.)

28. [*M22*] For what integers $c_0, t_0, c_1, t_1, \ldots, c_l, t_l$ with $1 \le c_j \le t_j$ does the text's formula (27) for estimated completion ratio give the value (a) 1/2? (b) 1/3?

▸ **29.** [*26*] Let T be any tree. Construct the 0–1 matrix of an unsolvable exact cover problem for which T is the backtrack tree traversed by Algorithm X with the MRV heuristic. (A *unique* item should have the minimum `LEN` value whenever step X3 is encountered.) Illustrate your construction when $T =$ .

30. [*25*] Continuing exercise 29, let T be a tree in which certain leaves have been distinguished from the others and designated as "solutions." Can all such trees arise as backtrack trees in Algorithm X?

31. [*M21*] The running time of Algorithm X depends on the order of primary items in the active list, as well as on the order of options in the individual item lists. Explain how to *randomize* the algorithm so that (a) every item list is in random order after step X1; (b) step X3 chooses randomly among items with the minimum `LEN`.

32. [*M21*] The solution to an exact cover problem with M options can be regarded as a binary vector $x = x_1 \ldots x_M$, with $x_k =$ [choose option k]. The *distance* between two

secondary
options without primary items
pairwise ordering
binary notation
ruler function ρ
k-wise ordering
unsymmetrical
symmetry breaking
canonical solutions
diagonal
diagonal
central symmetry
90-deg rot symmetry
queen graph
colored
balanced
secondary items
Langford's problem
completion ratio
0–1 matrix
backtrack tree
MRV
order of primary items
randomize
MRV

solutions x and x' can then be defined as the Hamming distance $d(x,x') = \nu(x \oplus x')$, the number of places where x and x' differ. The *diversity* of the problem is the minimum distance between two of its solutions. (If there's at most one solution, the diversity is ∞.)

a) Is it possible to have diversity 1?
b) Is it possible to have diversity 2?
c) Is it possible to have diversity 3?
d) Prove that the distance between solutions of a *uniform* exact cover problem — that is, a problem having the same number of items in each option — is always even.
e) Most of the exact cover problems that arise in applications are at least *quasi-uniform*, in the sense that they have a nonempty subset of primary items such that the problem is uniform when restricted to only those items. (For example, every polyomino or polycube packing problem is quasi-uniform, because every option specifies exactly one piece name.) Can such problems have odd distances?

33. [*M16*] Given an exact cover problem, specified by a 0–1 matrix A, construct an exact cover problem A' that has exactly one more solution than A does. [Consequently it is NP-hard to determine whether an exact cover problem with at least one solution has more than one solution.] Assume that A contains no all-zero rows.

34. [*M25*] Given an exact cover problem A as in exercise 33, construct an exact cover problem A' such that (i) A' has at most three 1s in every column; (ii) A' and A have exactly the same number of solutions.

35. [*M21*] Continuing exercise 34, construct A' having *exactly* three 1s per column.

▸ **36.** [*25*] Let $i_k = \texttt{TOP}(x_k)$ be the item on which branching occurs at level k in Algorithm X. Modify that algorithm so that it finds the solution for which $i_0x_0i_1x_1i_2x_2\ldots$ is *smallest* in lexicographic order. (It's easy to do this by simply setting $i \leftarrow \texttt{RLINK(0)}$ in step X3. But there's a much faster way, by using the MRV heuristic most of the time.)

What is the lexicographically first solution to the 32 queens problem?

37. [*M46*] (N. J. A. Sloane, 2016.) Let $\langle q_n\rangle$ be the lexicographically smallest solution to the ∞ queens problem. (This sequence begins

$$1,3,5,2,4,9,11,13,15,6,8,19,7,22,10,25,27,29,31,12,14,35,37,39,41,16,18,45,\ldots,$$

and it clearly has strange regularities and irregularities.)

a) Prove that every positive integer occurs in the sequence.
b) Prove that q_n is either $n\phi + O(1)$ or $n/\phi + O(1)$.

▸ **38.** [*M25*] Devise an efficient way to compute the sequence $\langle q_n\rangle$ of exercise 37.

▸ **39.** [*M21*] Experiment with exact cover problems that are defined by m *random* options on n items. (Each option is generated independently, with repetitions permitted.)

a) Use a fixed probability p that item i is included in any given option.
b) Let every option be a random sample of r distinct items.

▸ **40.** [*21*] If we merely want to count the number of solutions to an exact cover problem, without actually constructing them, a completely different approach based on bitwise manipulation instead of list processing is sometimes useful.

The following naïve algorithm illustrates the idea: We're given an $m \times n$ matrix of 0s and 1s, represented as n-bit vectors $r_1, \ldots, r_m$. The algorithm works with a (potentially huge) database of pairs (s_j, c_j), where s_j is an n-bit number representing a set of items, and c_j is a positive integer representing the number of ways to cover that set exactly. Let p be the n-bit mask that represents the primary items.

distance
Hamming distance
diversity
uniform
quasi-uniform
polyomino
polycube
0–1
NP-hard
unique solution
lexicographic order
MRV heuristic
queens
Sloane
∞ queens
golden ratio
greedy queens
random exact cover problems
exact cover problem
bitwise manipulation
breadth-first
0s and 1s
primary items

N1. [Initialize.] Set $N \leftarrow 1$, $s_1 \leftarrow 0$, $c_1 \leftarrow 1$, $k \leftarrow 1$.

N2. [Done?] If $k > m$, terminate; the answer is $\sum_{j=1}^{N} c_j\,[s_j \,\&\, p = p]$.

N3. [Append r_k where possible.] Set $t \leftarrow r_k$. For $N \ge j \ge 1$, if $s_j \,\&\, t = 0$, insert $(s_j + t, c_j)$ into the database (see below).

N4. [Loop on k.] Set $k \leftarrow k + 1$ and return to N2. ▮

To insert (s, c) there are two cases: If $s = s_i$ for some (s_i, c_i) already present, we simply set $c_i \leftarrow c_i + c$. Otherwise we set $N \leftarrow N + 1$, $s_N \leftarrow s$, $c_N \leftarrow c$.

Show that this algorithm can be significantly improved by using the following trick: Set $u_k \leftarrow r_k \,\&\, \bar{f}_k$, where $f_k = r_{k+1} \mid \cdots \mid r_m$ is the bitwise OR of all future rows. If $u_k \ne 0$, we can remove any entry from the database for which s_j does not contain $u_k \& p$. We can also exploit the nonprimary items of u_k to compress the database further.

41. [*25*] Implement the improved algorithm of the previous exercise, and compare its running time to that of Algorithm X when applied to the n queens problem.

42. [*M21*] Explain how the method of exercise 40 could be extended to give representations of all solutions, instead of simply counting them.

43. [*M20*] Give formulas for the entries a_{ij}, b_{ij}, c_{ij} of the sudoku squares in (28).

44. [*M04*] Could the clues of a sudoku puzzle be the first 33 digits of π? (See (29a).)

45. [*14*] List the sequence of naked single moves by which Algorithm X cruises to the solution of (29a). (If several such p_{ij} are possible, choose the smallest ij at each step.)

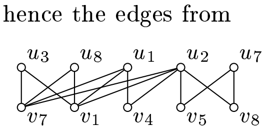
46. [*19*] List all of the *hidden* single sudoku moves that are present in chart (31).

47. [*19*] What hidden singles are present in (32), after '3' is placed in cell $(2, 3)$?

▸ **48.** [*24*] Chart (33) essentially plots rows versus columns. Show that the same data could be plotted as either (a) rows versus values; or (b) values versus columns.

▸ **49.** [*24*] Any solution to an exact cover problem will also solve the "relaxed" subproblems that are obtained by removing some of the items. For example, we might relax a sudoku problem (30) by removing all items c_{jk} and b_{xk}, as well as r_{ik} with $i \ne i_0$. Then we're left with a subproblem in which every option contains just two items, '$p_{i_0 j}$ $r_{i_0 k}$', for certain pairs (j, k). In other words, we're left with a 2D matching problem.

Consider the bipartite graph with u_j — v_k whenever a sudoku option contains '$p_{i_0 j}$ $r_{i_0 k}$'. For example, the graph for $i_0 = 4$ in (33) is illustrated below. A perfect matching of this graph must take u_3 and u_8 to either v_7 or v_1, hence the edges from other u's to those v's can be deleted; that's called a "naked pair" in row i_0. Dually, v_5 and v_8 must be matched to either u_2 or u_7, hence the edges from other v's to those u's can be deleted; that's called a "hidden pair" in row i_0.



In general, q of the u's form a *naked q-tuple* if their neighbors include only q of the v's; and q of the v's form a *hidden q-tuple* if their neighbors include only q of the u's.

a) These definitions have been given for rows. Show that naked and hidden q-tuples can be defined analogously for (i) columns, (ii) boxes.

b) Prove that if the bipartite graph has r vertices in each part, it has a hidden q-tuple if and only if it has a naked $(r - q)$-tuple.

c) Find all the naked and hidden q-tuples of (33). What options do they rule out?

d) Consider deleting items p_{ij} and b_{xk}, as well as all r_{ik} and c_{jk} for $k \ne k_0$. Does this lead to further reductions of (33)?

50. [*20*] How many uniquely solvable 17-clue puzzles contain the 16 clues of (29c)?

bitwise AND
bitwise OR
nonprimary items
n queens problem
sudoku squares
sudoku puzzle
π
naked single
hidden single moves
relaxed
factoring, see also relaxed subproblems
2D matching problem
bipartite graph
naked pair
hidden pair

51. [*22*] In how many ways can (29c) be completed so that every row, every column, and every box contains a permutation of the multiset $\{1, 2, 3, 4, 5, 6, 7, 7, 9\}$?

52. [*40*] Try to find a sudoku puzzle that's as difficult as possible for Algorithm X.

53. [*M26*] Beginners to sudoku might want to cut their teeth on a miniature variant called *shidoku*, which features 4×4 squares divided into four 2×2 boxes.

a) Prove that every uniquely solvable shidoku problem has at least four clues.

b) Two shidoku problems are equivalent if we can get from one to the other by permuting rows and columns in such a way that boxes are preserved, and/or by $90°$ rotation, and/or by permuting the numbers. Show that there are exactly 13 essentially different 4-clue shidoku problems.

▸ **54.** [*35*] (*Minimal clues.*) Puzzle (29a) contains more clues than necessary to make the sudoku solution unique. (For example, the final '95' could be omitted.) Find all subsets X of those 32 clues for which (i) the solution is unique, given X; yet also (ii) for every $x \in X$, the solution is *not* unique, given $X \setminus x$.

55. [*34*] (G. McGuire.) Prove that at least 18 clues are necessary, in any sudoku puzzle whose unique answer is (28a). Also find 18 clues that suffice. *Hint:* At least two of the nine appearances of $\{1, 4, 7\}$ in the top three rows must be among the clues.

Similarly, find a smallest-possible set of clues whose unique answer is (28b).

56. [*47*] What is the largest number of clues in a minimal sudoku puzzle?

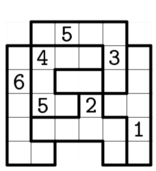
57. [*22*] Every sudoku solution has 27 horizontal trios and 27 vertical trios, namely the 3-digit sets that appear within a single row or column of a box. For example, (28a) has nine horizontal trios $\{1, 2, 3\}$, $\{2, 3, 4\}$, …, $\{9, 1, 2\}$ and three vertical trios $\{1, 4, 7\}$, $\{2, 5, 8\}$, $\{3, 6, 9\}$; (28b) has just three of each. The solution to (29a) has 26 horizontal trios and 23 vertical trios; $\{3, 6, 8\}$ occurs once horizontally and twice vertically.

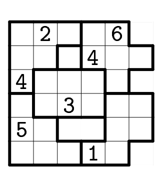
Let T be the 27 trios $\bigl\{\{\mathtt{A}, \mathtt{B}, \mathtt{C}\} \bigm| \mathtt{A} \in \{1, 2, 3\}, \mathtt{B} \in \{4, 5, 6\}, \mathtt{C} \in \{7, 8, 9\}\bigr\}$. Find all sudoku solutions whose horizontal trios and vertical trios are both equal to T.

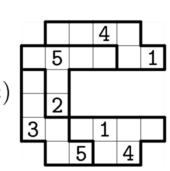
▸ **58.** [*22*] (A. Thoen and A. van de Wetering, 2019.) Find all sudoku solutions for which the 1s, 2s, …, 7s also solve the nine queens problem.

59. [*20*] Solve the jigsaw sudokus in (34). How large is Algorithm X's search tree?

60. [*20*] (*The Puzzlium Sudoku ABC.*) Complete these hexomino-shaped boxes:

(a) 

; (b) 

; (c) 

.

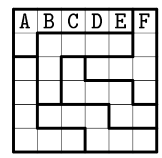
61. [*21*] Turn Behrens's 5×5 gerechte design (35a) into a jigsaw sudoku puzzle, by erasing all but five of its 25 entries.

▸ **62.** [*34*] For $n \le 7$, generate all of the ways in which an $n \times n$ square can be packed with n nonstraight n-ominoes. (These are the possible arrangements of boxes in a square jigsaw sudoku.) How many of them are symmetric? *Hint:* See exercise 7.2.2–76.

63. [*29*] In how many different ways can Behrens's 9×9 array (35c) be regarded as a gerechte latin square? (In other words, how many decompositions of that square into nine boxes of size 9 have a complete "rainbow" $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ in each box? None of the boxes should simply be an entire row or an entire column.)

multiset
sudoku puzzle
hardest puzzle
shidoku
McGuire
trios
minirows of sudoku: see trios
Thoen
van de Wetering
sudoku solutions
9 queens problem
jigsaw sudokus
Puzzlium Sudoku ABC
hexomino
alphabet
Behrens
gerechte design
n-ominoes
boxes
rainbow

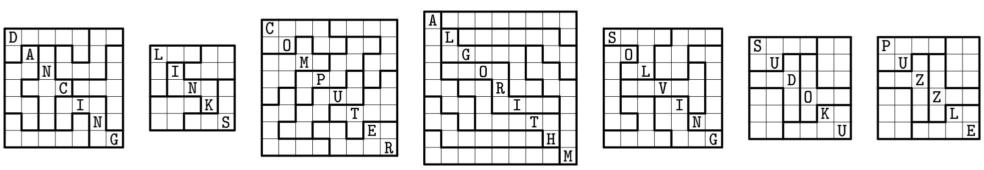
64. [*23*] (*Clueless jigsaw sudoku.*) A jigsaw sudoku puzzle can be called "clueless" if its solution is uniquely determined by the entries in a single row or column, because such clues merely assign names to the n individual symbols that appear. For example, the first such puzzle to be published, discovered in 2000 by Oriel Maxime, is shown here.



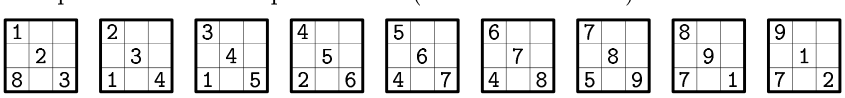
a) Find all clueless sudoku jigsaw puzzles of order $n \le 6$.

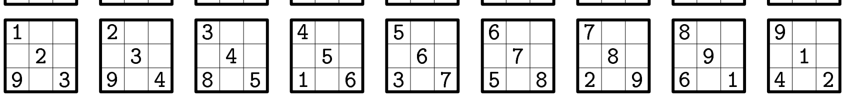
b) Prove that such puzzles exist of all orders $n \ge 4$.

65. [*24*] Find the unique solutions to the following examples of jigsaw sudoku:

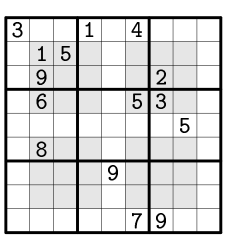
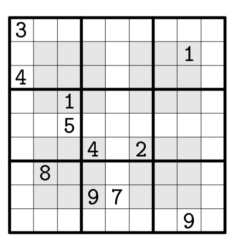


▸ **66.** [*30*] Arrange the following sets of nine cards in a 3×3 array so that they define a sudoku problem with a unique solution. (Don't rotate them.)

i)  ;

ii)  .

▸ **67.** [*22*] *Hypersudoku* extends normal sudoku by adding four more (shaded) boxes in which a complete "rainbow" $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is required to appear:

(i)  ; (ii)  .

(Such puzzles, introduced by P. Ritmeester in 2005, are featured by many newspapers.)

a) Show that a hypersudoku solution actually has 18 rainbow boxes, not only 13.

b) Use that observation to solve hypersudoku puzzles efficiently by extending (30).

c) How much does that observation help when solving (i) and (ii)?

d) True or false: A hypersudoku solution remains a hypersudoku solution if the four 4×4 blocks that touch its four corners are simultaneously rotated 180°, while also flipping the middle half-rows and middle half-columns (keeping the center fixed).

68. [*28*] A polyomino is called *convex* if it contains all of the cells between any two of its cells that lie in the same row or the same column. (This happens if and only if it has the same perimeter as its minimum bounding box does, because each row and column contribute 2.) For example, all of the pentominoes (36) are convex, except for 'U'.

a) Generate all ways to pack n convex n-ominoes into an $n \times n$ box, for $n \le 7$.

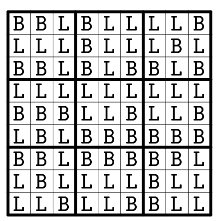
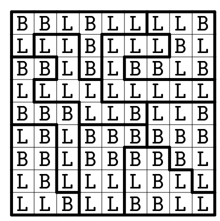
b) In how many ways can nine convex nonominoes be packed into a 9×9 box, when each of them is small enough to fit into a 4×4? (Consider also the symmetries.)

Clueless jigsaw sudoku
Maxime
jigsaw sudoku
sudoku
Hypersudoku
NRC Sudoku, see hypersudoku
rainbow
Ritmeester
pi as random data
polyomino
bounding box
convex polyominoes
pentominoes
enneominoes, see nonominoes

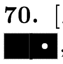
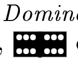
▸ **69.** [*30*] Diagram (i) below shows the 81 communities of Bitland, and their nine electoral districts. The voters in each community are either Big-Endian (B) or Little-Endian (L). Each district has a representative in Bitland's parliament, based on a majority vote.

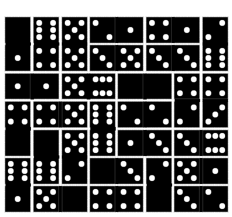
Notice that there are five Ls and four Bs in every district, hence the parliament is 100% Little-Endian. Everybody agrees that this is unfair. So you have been hired as a computer consultant, to engineer the redistricting.

A rich bigwig secretly offers to pay you a truckload of money if you get the best possible deal for his side. You could gerrymander the districts as in diagram (ii), thereby obtaining seven Big-Endian seats. But that would be too blatantly biased.

(i)  ; (ii)  .

Show that seven wins for B are actually obtainable with nine districts that do respect the local neighborhoods of Bitland quite decently, because each of them is a convex nonomino that fits in a 4×4 square (see exercise 68).

70. [*21*] *Dominosa* is a solitaire game in which you "shuffle" the 28 pieces , , . . . ,  of double-six dominoes and place them at random into a 7×8 frame. Then you write down the number of spots in each cell, put the dominoes away, and try to reconstruct their positions based only on that 7×8 array of numbers. For example,

 yields the array $$\begin{pmatrix} 0&6&5&2&1&4&1&2\\ 1&4&5&3&5&3&3&6\\ 1&1&5&6&0&0&4&4\\ 4&4&5&6&2&2&2&3\\ 0&0&5&6&1&3&3&6\\ 6&6&2&0&3&2&5&1\\ 1&5&0&4&4&0&3&2 \end{pmatrix}.$$

a) Show that *another* placement of dominoes also yields the same matrix of numbers.
b) What domino placement yields the array

$$\begin{pmatrix} 3&3&6&5&1&5&1&5\\ 6&5&6&1&2&3&2&4\\ 2&4&3&3&3&6&2&0\\ 4&1&6&1&4&4&6&0\\ 3&0&3&0&1&1&4&4\\ 2&6&2&5&0&5&0&0\\ 2&5&0&5&4&2&1&6 \end{pmatrix}?$$

▸ **71.** [*20*] Show that Dominosa reconstruction is a special case of 3D MATCHING.

72. [*M22*] Generate random instances of Dominosa, and estimate the probability of obtaining a 7×8 matrix with a unique solution. Use two models of randomness: (i) Each matrix whose elements are permutations of the multiset $\{8 \times 0, 8 \times 1, \ldots, 8 \times 6\}$ is equally likely; (ii) each matrix obtained from a random shuffle of the dominoes is equally likely.

73. [*46*] What's the maximum number of solutions to an instance of Dominosa?

Bitland
voters
parliament
redistricting
political districting
gerrymander
Gerry
politics
convex
nonomino
Dominosa
solitaire
game
Pijanowski solitaire, see Dominosa
dominoes
3D MATCHING
permutations of the multiset
random domino placement

Keller
grope
binary operation
multiplication tables
idempotent
commutative
identity element
subgraph
word search puzzle

74. [*22*] (M. Keller, 1987.) Is there a uniquely solvable Dominosa array for which every domino matches two adjacent cells of the array in either three or four places?

▶ **75.** [*M24*] A *grope* is a set G together with a binary operation $\circ$, in which the identity $x \circ (y \circ x) = y$ is satisfied for all $x \in G$ and $y \in G$.

a) Prove that the identity $(x \circ y) \circ x = y$ also holds, in every grope.

b) Which of the following "multiplication tables" define a grope on $\{0, 1, 2, 3\}$?

$$\begin{array}{c}0123\\1032\\2301\\3210\end{array};\quad \begin{array}{c}0321\\3210\\2103\\1032\end{array};\quad \begin{array}{c}0132\\1023\\3210\\2301\end{array};\quad \begin{array}{c}0231\\3102\\1320\\2013\end{array};\quad \begin{array}{c}0312\\2130\\3021\\1203\end{array}.$$

(In the first example, $x \circ y = x \oplus y$; in the second, $x \circ y = (-x - y) \bmod 4$. The last two have $x \circ y = x \oplus f(x \oplus y)$ for certain functions f.)

c) For all n, construct a grope whose elements are $\{0, 1, \ldots, n-1\}$.

d) Consider the exact cover problem that has n^2 items xy for $0 \le x, y < n$ and the following $n + (n^3 - n)/3$ options:

i) 'xx', for $0 \le x < n$;

ii) 'xx xy yx', for $0 \le x < y < n$;

iii) 'xy yz zx', for $0 \le x < y, z < n$.

Show that its solutions are in one-to-one correspondence with the multiplication tables of gropes on the elements $\{0, 1, \ldots, n-1\}$.

e) Element x of a grope is *idempotent* if $x \circ x = x$. If k elements are idempotent and $n - k$ are not, prove that $k \equiv n^2$ (modulo 3).

76. [*21*] Modify the exact cover problem of exercise 75(d) in order to find the multiplication tables of (a) all idempotent gropes — gropes such that $x \circ x = x$ for all x; (b) all commutative gropes — gropes such that $x \circ y = y \circ x$ for all x and y; (c) all gropes with an identity element — gropes such that $x \circ 0 = 0 \circ x = x$ for all x.

77. [*M21*] Given graphs G and H, each with n vertices, use Algorithm X to decide whether or not G is isomorphic to a subgraph of H.

78. [*16*] Show that it's quite easy to pack the 27 mathematicians' names of Fig. 71 into a 12×15 array, with all names reading correctly from left to right. (Of course that would be a *terrible* word search puzzle.)

79. [*M20*] How many options are in (48), when they are completely listed?

80. [*19*] Play through Algorithm C by hand, using exercise 9 in step C3 and the input in Table 2, until first reaching a solution. What are the contents of memory then?

81. [*21*] True or false: An exact cover problem that has no color assignments has exactly the same running time on Algorithms X and C.

82. [*21*] True or false: It's possible to save memory references in Algorithms X and C by not updating the `LEN` fields in the hide/unhide operations when $x > N_1$.

▶ **83.** [*20*] Algorithm C can be extended in the following curious way: Let p be the primary item that is covered first, and suppose that there are k ways to cover it. Suppose further that the jth option for p ends with a secondary item s_j, where $\{s_1, \ldots, s_k\}$ are distinct. Modify the algorithm so that, whenever a solution contains the jth option for p, it leaves items $\{s_1, \ldots, s_{j-1}\}$ uncovered. (In other words, the modified algorithm will emulate the behavior of the unmodified algorithm on a much larger instance, in which the jth option for p contains all of $s_1, s_2, \ldots, s_j$.)

minimax solution
bottleneck optimum, see minimax
modify Algorithm C
Monte Carlo estimate
estimated time
double word square
word square, double
OSPD4
word stair

▶ **84.** [*25*] Number the options of an XCC problem from 1 to M. A *minimax solution* is one whose maximum option number is as small as possible. Explain how to modify Algorithm C so that it determines all of the minimax solutions (omitting any that are known to be worse than a solution already found).

85. [*22*] Sharpen the algorithm of exercise 84 so that it produces *exactly one* minimax solution — unless, of course, there are no solutions at all.

▶ **86.** [*M25*] Modify Algorithm C so that, instead of finding all solutions to a given XCC problem, it gives a *Monte Carlo estimate* of the number of solutions and the time needed to find them, using Theorem 7.2.2E. (Thus the modified algorithm is to Algorithm C as Algorithm 7.2.2E is to Algorithm 7.2.2B.)

87. [*20*] A *double word square* is an $n \times n$ array whose rows and columns contain $2n$ different words. Encode this problem as an XCC problem. Can you save a factor of 2 by not generating the transpose of previous solutions? Does Algorithm C compete with the algorithm of exercise 7.2.2–28 (which was designed explicitly to handle such problems)?

88. [*21*] Instead of finding *all* of the double word squares, we usually are more interested in finding the *best* one, in the sense of using only words that are quite common. For example, it turns out that a double word square can be made from the words of `WORDS`(1720) but not from those of `WORDS`(1719). Show that it's rather easy to find the smallest W such that `WORDS`(W) supports a double word square, via dancing links.

89. [*24*] What are the best double word squares of sizes 2×2, 3×3, …, 7×7, in the sense of exercise 88, with respect to *The Official* SCRABBLE® *Players Dictionary*? [Exercise 7.2.2–32 considered the analogous problem for *symmetric* word squares.]

▶ **90.** [*22*] A *word stair* of period p is a cyclic arrangement of words, offset stepwise, that contains $2p$ distinct words across and down. They exist in two varieties, left and right:

```
          . . .     . . .
        S T A I R   S T A I R
       S H A R P     S L O O P
      S T E M S       S T O O D
     S C R A P         S T E E R
    S T A I R           S T A I R
   S H A R P             S L O O P
  S T E M S    p = 4      S T O O D
 S C R A P                 S T E E R
S T A I R                   S T A I R
. . .                         . . .
```

What are the best five-letter word stairs, in the sense of exercise 88, for $1 \le p \le 10$? *Hint:* You can save a factor of $2p$ by assuming that the first word is the most common.

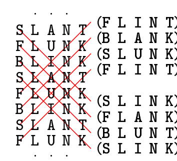
91. [*40*] For given W, find the largest p such that `WORDS`(W) supports a word stair of period p. (There are two questions for each W, examining stairs to the {left, right}.)

92. [*24*] Some p-word cycles define *two-way* word stairs that have $3p$ distinct words:

```
          . . .     . . .
        R A P I D   R A P I D
       R A T E D     R A T E D
      L A C E S       L A C E S
     R O B E S         R O B E S
    R A P I D           R A P I D
   R A T E D             R A T E D
  L A C E S    p = 4      L A C E S
 R O B E S                 R O B E S
R A P I D                   R A P I D
. . .                         . . .
```

What are the best five-letter examples of this variety, for $1 \le p \le 10$?

93. [*22*] Another periodic arrangement of $3p$ words, perhaps even nicer than that of exercise 92 and illustrated here for $p = 3$, lets us read them *diagonally* up or down, as well as across. What are the best five-letter examples of *this* variety, for $1 \le p \le 10$? (Notice that there is $2p$-way symmetry.)



94. [*20*] (É. Lucas.) Find a binary cycle $(x_0x_1\ldots x_{15})$ for which the 16 quadruples $x_kx_{(k+1) \bmod 16}x_{(k+3) \bmod 16}x_{(k+4) \bmod 16}$ for $0 \le k < 16$ are distinct.

▸ **95.** [*20*] Given $0 \le p < q \le n$, explain how to use color controls and Algorithm C to find all cycles $(x_0x_1\ldots x_{m-1})$ of 0s and 1s, where $m = \sum_{k=p}^{q}\binom{n}{k}$, with the property that the m binary vectors $\{x_0x_1\ldots x_{n-1}, x_1x_2\ldots x_n, \ldots, x_{m-1}x_0\ldots x_{n-2}\}$ are distinct and have weight between p and q. (In other words, all n-bit binary vectors $y = y_1\ldots y_n$ with $p \le \nu y \le q$ occur exactly once in the cycle. We studied the special case of *de Bruijn cycles*, for which $p = 0$ and $q = n$, in Section 7.2.1.1.)

For example, when $n = 7$, $p = 0$, and $q = 3$, the cycle

$$(0000000100000110000101000101100010010101001001100100011010000111)$$

exhibits all binary 7-tuples with a majority of 0s. When $n = 7$, $p = 3$, $q = 4$, the cycle

$$(0000111000101100011010010101011010100110011011001011100100111010001111)$$

shows all 7-tuples obtainable by removing the first bit of an 8-tuple with four 0s, four 1s.

Exactly how many cycles exist, when $(n, p, q) = (7, 0, 3)$ or $(7, 3, 4)$? How long does it take for Algorithm C to find them?

96. [*M20*] Find an 8×8 binary torus whose sixty-four 2×3 subrectangles are distinct.

97. [*M21*] Find all 9×9 ternary ourotoruses $D = (d_{i,j})$ that are symmetrical, in the sense that $d_{(i+3) \bmod 9, (j+3) \bmod 9} = (d_{i,j} + 1) \bmod 3$. (See exercise 7.2.1.1–109.)

98. [*25*] Prove that the exact cover problem with color controls is NP-complete, even if every option consists of only two items.

99. [*20*] True or false: Every XCC problem can be reformulated as an *ordinary* exact cover problem with the same solutions and the same number of options.

▸ **100.** [*20*] The general *constraint satisfaction problem* (CSP) is the task of finding all n-tuples $x_1\ldots x_n$ that satisfy a given system of constraints $C_1, \ldots, C_m$, where each constraint is defined by a relation on a nonempty subset of the variables $\{x_1, \ldots, x_n\}$.

For example, a unary constraint is a relation of the form $x_k \in D_k$; a binary constraint is a relation of the form $(x_j, x_k) \in D_{jk}$; a ternary constraint is a relation of the form $(x_i, x_j, x_k) \in D_{ijk}$; and so on.

a) Find all $x_1x_2x_3x_4x_5$ for which $0 \le x_1 \le x_2 \le x_3 \le x_4 \le x_5 \le 2$ and $x_1 + x_3 + x_5 = 3$.
b) Formulate the problem of part (a) as an XCC problem.
c) Explain how to formulate *any* CSP as an XCC problem.

▸ **101.** [*25*] (*The zebra puzzle.*) Formulate the following query as an XCC problem: "Five people, from five different countries, have five different occupations, own five different pets, drink five different beverages, and live in a row of five differently colored houses.

- The Englishman lives in a red house.
- The yellow house hosts a diplomat.
- The Norwegian's house is the leftmost.
- The milk drinker lives in the middle house.
- The painter comes from Japan.
- The coffee-lover's house is green.
- The dog's owner is from Spain.
- The violinist drinks orange juice.

Lucas
color controls
sideways sum ν
de Bruijn cycles
torus
ourotoruses
color controls
NP-complete
2D matching
XCC problem
constraint satisfaction problem
CSP
unary constraint
binary constraint
ternary constraint
XCC problem
zebra puzzle

Japanese arrow puzzle
Music
equal temperament
pitch class
Schoenberg
twelve-tone row
permutation
Berg
n-tone row
all-interval
reversal
perfect

- The white house is just left of the green one.
- The Ukrainian drinks tea.
- The Norwegian lives next to the blue house.
- The sculptor breeds snails.
- The horse lives next to the diplomat.
- The nurse lives next to the fox.

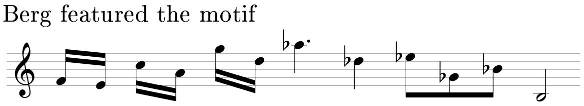
Who trains the zebra, and who prefers to drink just plain water?"

▸ **102.** [*25*] Explain how to find all solutions to a *Japanese arrow puzzle* with Algorithm C. (See exercise 7.2.2–68.)

▸ **103.** [*M28*] Musical pitches in the Western system of "equal temperament" are the notes whose frequency is $440 \cdot 2^{n/12}$ cycles per second, for some integer n. The *pitch class* of such a note is $n \bmod 12$, and seven of the twelve possible pitch classes are conventionally designated by letters:

$$0 = \mathrm{A},\quad 2 = \mathrm{B},\quad 3 = \mathrm{C},\quad 5 = \mathrm{D},\quad 7 = \mathrm{E},\quad 8 = \mathrm{F},\quad 10 = \mathrm{G}.$$

The other classes are named by appending sharp (♯) or flat (♭) signs, to go up or down by 1; thus 1 = A♯ = B♭, 4 = C♯ = D♭, …, 11 = G♯ = A♭.

Arnold Schoenberg popularized a composition technique that he called a twelve-tone row, which is simply a permutation of the twelve pitch classes. For example, his student Alban Berg featured the motif

which is the twelve-tone row 8 7 3 0 10 5 11 4 6 9 1 2, in the first movement of his Lyric Suite (1926), and in another composition he had written in 1925.

In general we can say that an n-tone row $x = x_0x_1\ldots x_{n-1}$ is a permutation of $\{0, 1, \ldots, n-1\}$. Two n-tone rows x and x' are considered to be *equivalent* if they differ only by a transposition — that is, if $x'_k = (x_k + d) \bmod n$ for some d and for $0 \le k < n$. Thus, the number of inequivalent n-tone rows is exactly $(n-1)!$.

a) Berg's 12-tone row above has the additional property that the intervals between adjacent notes, $(x_k - x_{k-1}) \bmod n$, are $\{1, \ldots, n-1\}$. Prove that an n-tone row can have this all-interval property only if n is even and $x_{n-1} = (x_0 + n/2) \bmod n$.

b) Use Algorithm C to find n-tone rows with the all-interval property. How many solutions arise, when $2 \le n \le 12$?

c) Any all-interval n-tone row leads easily to several others. For example, if $x = x_0x_1\ldots x_{n-1}$ is a solution, so is its reversal $x^R = x_{n-1}\ldots x_1x_0$; and so is $cx = (cx_0 \bmod n)(cx_1 \bmod n)\ldots(cx_{n-1} \bmod n)$ whenever $c \perp n$. Prove that the cyclic shift $x^Q = x_k\ldots x_{n-1}x_0\ldots x_{k-1}$ is also a solution, when $x_k - x_{k-1} = \pm n/2$.

d) True or false: In part (c) we always have $x^{RQ} = x^{QR}$.

e) The 12-tone row of Alban Berg shown above is symmetrical, because it is equivalent to x^R. Other kinds of symmetry are also possible; for example, the row $x = 0\,1\,3\,7\,2\,5\,11\,10\,8\,4\,9\,6$ is equivalent to $-x^Q$. How many symmetrical all-interval n-tone rows exist, for $n \le 12$?

104. [*M28*] Assume that $n + 1 = p$ is prime. Given an n-tone row $x = x_0x_1\ldots x_{n-1}$, define $y_k = x_{(k-1) \bmod p}$ whenever k is not a multiple of p, and let $x^{(r)} = y_ry_{2r}\ldots y_{nr}$ be the n-tone row consisting of "every rth element of x" (if x_n is blank). For example, when $n = 12$, every 5th element of x is the sequence $x^{(5)} = x_4x_9x_1x_6x_{11}x_3x_8x_0x_5x_{10}x_2x_7$.

An n-tone row is called *perfect* if it is equivalent to $x^{(r)}$ for $1 \le r \le n$. For example, the amazing 12-tone row $0\,1\,4\,2\,9\,5\,11\,3\,8\,10\,7\,6$ is perfect.

a) Prove that a perfect n-tone row has the all-interval property.

b) Prove that a perfect n-tone row also satisfies $x \equiv x^R$.

105. [*22*] Using the "word search puzzle" conventions of Figs. 71 and 72, show that the words ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN, ELEVEN, and TWELVE can all be packed into a 6×6 square, leaving one cell untouched.

106. [*22*] Also pack *two* copies of ONE, TWO, THREE, FOUR, FIVE into a 5×5 square.

▸ **107.** [*25*] Pack as many of the following words as possible into a 9×9 array, simultaneously satisfying the rules of *both* word search *and* sudoku:

ACRE	COMPARE	CORPORATE	MACRO	MOTET	ROAM
ART	COMPUTER	CROP	META	PARAMETER	TAME

▸ **108.** [*32*] The first 44 presidents of the U.S.A. had 38 distinct surnames: ADAMS, ARTHUR, BUCHANAN, BUSH, CARTER, CLEVELAND, CLINTON, COOLIDGE, EISENHOWER, FILLMORE, FORD, GARFIELD, GRANT, HARDING, HARRISON, HAYES, HOOVER, JACKSON, JEFFERSON, JOHNSON, KENNEDY, LINCOLN, MADISON, MCKINLEY, MONROE, NIXON, OBAMA, PIERCE, POLK, REAGAN, ROOSEVELT, TAFT, TAYLOR, TRUMAN, TYLER, VANBUREN, WASHINGTON, WILSON.

a) What's the smallest square into which all of these names can be packed, using word search conventions, and requiring all words to be *connected* via overlaps?

b) What's the smallest *rectangle*, under the same conditions?

▸ **109.** [*28*] A "wordcross puzzle" is the challenge of packing a given set of words into a rectangle under the following conditions: (i) All words must read either across or down, as in a crossword puzzle. (ii) No letters are adjacent unless they belong to one of the given words. (iii) The words are rookwise connected. (iv) Words overlap only when one is vertical and the other is horizontal. For example, the eleven words ZERO, ONE, ..., TEN can be placed into an 8×8 square under constraints (i) and (ii) as shown; but (iii) is violated, because there are three different components.

```
T H R E E   F
W         S I X
O N E       V
      S E V E N
  Z             I
  E I G H T   N
  R       E   E
F O U R   N
```

Explain how to encode a wordcross puzzle as an XCC problem. Use your encoding to find a correct solution to the problem above. Do those eleven words fit into a *smaller* rectangle, under conditions (i), (ii), and (iii)?

110. [*30*] What's the smallest wordcross square that contains the surnames of the first 44 U.S. presidents? (Use the names in exercise 108, but change VANBUREN to VAN BUREN.)

111. [*21*] Find all 8×8 crossword puzzle diagrams that contain exactly (a) 12 3-letter words, 12 4-letter words, and 4 5-letter words; (b) 12 5-letter words, 8 2-letter words, and 4 8-letter words. They should have no words of other lengths.

▸ **112.** [*28*] A popular word puzzle in Brazil, called 'Torto' ('bent'), asks solvers to find as many words as possible that can be traced by a noncrossing king path in a given 6×3 array of letters. For example, each of the words THE, MATURE, ART, OF, COMPUTER, and PROGRAMMING can be found in the array shown here.

```
O C G
F M N
M I P
A U R
T R O
E H G
```

a) Does that array contain other common words of eight or more letters?

b) Create a 6×3 array that contains TORTO, WORDS, SOLVER, and many other interesting English words of five or more letters. (Let your imagination fly.)

c) Is it possible to pack ONE, TWO, THREE, ..., EIGHT, NINE, TEN into a Torto array?

▸ **113.** [*21*] An 'alphabet block' is a cube whose six faces are marked with letters. Is there a set of five alphabet blocks that are able to spell the 25 words TREES, NODES, STACK, AVAIL, FIRST, RIGHT, ORDER, LISTS, GIVEN, LINKS, QUEUE, GRAPH, TIMES, BLOCK, VALUE, TABLE, FIELD, EMPTY, ABOVE, POINT, THREE, UNTIL, HENCE, QUITE, DEQUE? (Each of these words appears more than 50 times in Chapter 2.)

word search puzzle
word search
sudoku
presidents
I'm not sure
how many of
these names
should go in
the index
connected
wordcross
crisscross puzzle, composing, see wordcross
rookwise connected
components
crossword puzzle diagrams
5-letter words
Torto
noncrossing king path
king path
English words
alphabet block

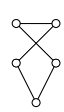
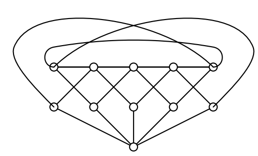
114. [*M25*] Let α be a permutation of the cells of a 9×9 array that takes any sudoku solution into another sudoku solution. We say that α is an *automorphism* of the sudoku solution $S=(s_{ij})$ if there's a permutation π of $\{1,2,\ldots,9\}$ such that $s_{(ij)\alpha}=s_{ij}\pi$ for $0\le i,j<9$. For example, the permutation that takes ij into $(ij)\alpha=ji$, commonly called *transposition*, is an automorphism of (28b), with respect to the permutation $\pi=(24)(37)(68)$; but it is *not* an automorphism of (28a) or (28c).

Show that Algorithm C can be used to find all sudoku solutions that have a given automorphism α, by defining an appropriate XCC problem.

How many sudoku solutions have transposition as an automorphism?

115. [*M25*] Continuing exercise 114, how many *hypersudoku* solutions have automorphisms of the following types? (a) transposition; (b) the transformation of exercise 67(d); (c) 90° rotation; (d) both (b) and (c).

▸ **116.** [*M25*] Given a graph on vertices V, let $\mu(G)$ be the graph obtained by (i) adding new vertices $V'=\{v'\mid v\in V\}$, with $u' \mathrel{-\!\!-} v$ when $u \mathrel{-\!\!-} v$; and also (ii) adding another vertex w, with $w \mathrel{-\!\!-} v'$ for all $v'\in V'$. (If G has m edges and n vertices, $\mu(G)$ has $3m+n$ edges and $2n+1$ vertices.) The *Mycielski graphs* M_c are defined for all $c\ge 2$ by setting $M_2=K_2$ and $M_{c+1}=\mu(M_c)$; they have $\frac{7}{18}3^c-\frac{3}{4}2^c+\frac{1}{2}$ edges and $\frac{3}{4}2^c-1$ vertices:

M_2 = ; M_3 = ; M_4 = ; . . . .

a) Prove that each M_c is triangle-free (contains no subgraph K_3).
b) Prove that the chromatic number $\chi(M_c)=c$.
c) Prove that each M_c is in fact "χ-critical": Removing any edge decreases χ.

▸ **117.** [*24*] (*Graph coloring.*) Suppose we want to find all possible ways to label the vertices of graph G with d colors; adjacent vertices should have different colors.

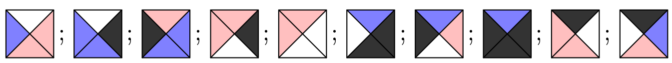
a) Formulate this as an exact cover problem, with one primary item for each vertex and with d secondary items for each edge.
b) Sometimes G's edges are conveniently specified by giving a family $\{C_1,\ldots,C_r\}$ of cliques, where each C_j is a subset of vertices; then $u \mathrel{-\!\!-} v$ if and only if $u\in C_j$ and $v\in C_j$ for some j. (For example, the 728 edges of the queen graph Q_8 can be specified by just $8+8+13+13=42$ cliques — one clique for each row, column, and diagonal.) Modify the construction of (a) so that there are only rd secondary items.
c) In how many ways can Q_8 be 9-colored? (Compare method (a) to method (b).)
d) Each solution to the coloring problem that uses k different colors is obtained $d^{\underline{k}}=d(d-1)\ldots(d-k+1)$ times, because of the symmetry between colors. Modify (a) and (b) so that each essentially different solution is obtained just once, when the symmetry-breaking technique of exercise 122 is used.
e) In how many ways can the Mycielski graph M_c be c-colored, for $2\le c\le 5$?
f) Use Algorithm C to verify that M_c can't be $(c-1)$-colored, for $2\le c\le 5$.
g) Try $(c-1)$-coloring M_c when a random edge is removed, for $2\le c\le 5$.

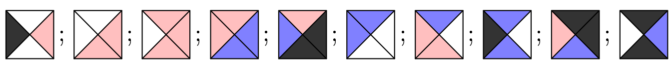
118. [*21*] (*Hypergraph coloring.*) Color the 64 cells of a chessboard with four colors, so that no three cells of the same color lie in a straight line of any slope.

119. [*21*] Show that all solutions to the problem of placing MacMahon's 24 triangles (58) into a hexagon with all-white border can be rotated and reflected so that the all-white triangle has the position that it occupies in (59b). *Hint:* Factorize.

permutation
sudoku solution
automorphism
transposition
automorphism
hypersudoku
Mycielski graphs
triangle-free
chromatic number
χ-critical
Graph coloring
cliques
queen graph
symmetry-breaking
Mycielski graph
Hypergraph coloring
no-three-in-line problem
Factorize

120. [*M29*] Section 2.3.4.3 discussed Hao Wang's "tetrad tiles," which are squares that have specified colors on each side. Find all ways in which the entire plane can be filled with tiles from the following families of tetrad types, always matching colors at the edges where adjacent tiles meet [see *Scientific American* **231**, 5 (Nov. 1965), 103, 106]:

a)  .

b)  .

(The tetrad tiles must *not* be rotated or flipped.) *Hint:* Algorithm C will help.

▸ **121.** [*M29*] Exercise 2.3.4.3–5 discusses 92 types of tetrads that are able to tile the plane, and proves that no such tiling is toroidal (periodic).

a) Show that the tile called βUS in that exercise can't be part of any infinite tiling. In fact, it can appear in only $n+1$ cells of an $m \times n$ array, when $m, n \geq 4$.

b) Show that, for all $k \geq 1$, there's a unique $(2^k-1) \times (2^k-1)$ tiling for which the middle tile is δRD. (Consequently, by the infinity lemma, there's a unique tiling of the entire plane in which δRD is placed at the origin.)

c) Similarly, show that there are exactly (2, 3, 3, 57) tilings of size $(2^k-1) \times (2^k-1)$ whose middle tile is respectively (δRU, δLD, δLU, δSU), for all $k \geq 3$.

d) How many tilings of the infinite plane have (δRU, δLD, δLU, δSU) at the origin?

▸ **122.** [*28*] Extend Algorithm C so that it finds only $1/d!$ of the solutions, in cases where the input options are totally symmetric with respect to d of the color values, and where every solution contains each of those color values at least once. Assume that those values are $\{v, v+1, \ldots, v+d-1\}$, and that all other colors have values $< v$. *Hint:* Modify the algorithm so that the first such color it assigns is always v, then $v+1$, etc.

123. [*M20*] Apply the algorithm of exercise 122 to the following toy problem with parameters m and n: There are n primary items p_k and n secondary items q_k, for $1 \leq k \leq n$; and there are mn options, '$p_k\ q_k{:}j$' for $1 \leq j \leq m$ and $1 \leq k \leq n$. (The solutions to this problem are the mappings of $\{1, \ldots, n\}$ into $\{1, \ldots, m\}$, which may also be regarded as the partitions of $\{1, \ldots, n\}$ into m parts labeled $\{1, \ldots, m\}$.) Algorithm C will obviously find $m^n = {n \brace 1} m^{\underline{1}} + {n \brace 2} m^{\underline{2}} + \cdots {n \brace m} m^{\underline{m}}$ solutions. But the modified algorithm finds only the "unlabeled" partitions, of which there are ${n \brace 1} + {n \brace 2} + \cdots + {n \brace m}$.

▸ **124.** [*M22*] Devise a system of coordinates for representing the positions of equilateral triangles in patterns such as (59). Represent also the edges between them.

125. [*M20*] When a set of s triangles is magnified by an integer k, we obtain sk^2 triangles. Describe the coordinates of those triangles, in term of the coordinates of the originals, using the system of exercise 124.

126. [*23*] Find all solutions of MacMahon's problem (59), by applying Algorithm C to a suitable set of items and options based on the coordinate system in exercise 124. How much time is saved by using the improved algorithm of exercise 122?

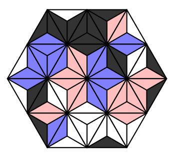
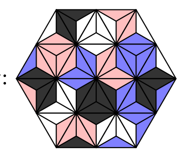
127. [*M28*] There are 4^{12} ways to prescribe the border colors of a hexagon like those in (59). Which of them can be completed to a color-matched placement of all 24 triangles?

▸ **128.** [*25*] Eleven of MacMahon's triangles (58) involve only the first three colors (not black). Arrange them into a pleasant pattern that tiles the entire plane when replicated.

▸ **129.** [*M34*] The most beautiful patterns that can be made with MacMahon's triangles are those with attractive symmetries, which can be of two kinds: *strong symmetry* (a rotation or reflection that doesn't change the pattern, except for permutation of colors)

Wang
tetrad tiles
toroidal
infinity lemma
symmetric
mappings
partitions of a set
Stirling subset numbers
unlabeled
coordinates
triangles, coordinates for
triangles
MacMahon's triangles
symmetries
strong symmetry

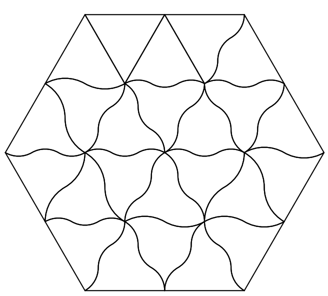
or *weak symmetry* (a rotation or reflection that preserves the “color patches,” the set of boundaries between different colors).

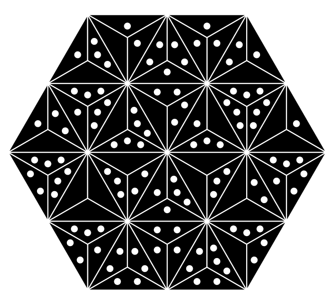
strong symmetry: ; weak symmetry: .

Exactly how many essentially different symmetrical patterns are possible, in a hexagon?

130. [*21*] Partition MacMahon’s triangles (58) into three sets of eight, each of which can be placed on the faces of an octahedron, with matching edge colors. 

131. [*28*] (P. A. MacMahon, 1921.) Instead of using the colored tiles of (58), which yield (59), we can form hexagons from 24 different triangles in two other ways:

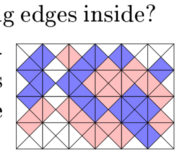




The left diagram shows a “jigsaw puzzle” whose pieces have four kinds of edges. The right diagram shows “triple three triominoes,” which have zero, one, two, or three spots at each edge; adjacent triominoes should have a total of three spots where they meet.

a) In how many ways can that jigsaw puzzle make a hexagon? (All pieces are white.)
b) How many triomino arrangements have that pattern of dots at the edges?

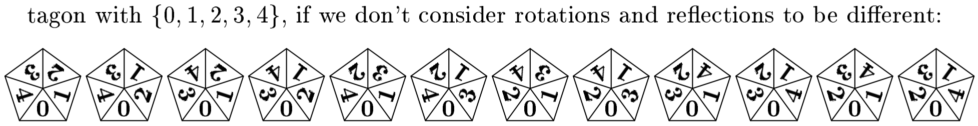
132. [*40*] (W. E. Philpott, 1971.) There are $4624 = 68^2$ tiles in a set that’s like (58), but it uses 24 different colors instead of 4. Can they be assembled into an equilateral triangle of size 68, with constant color on the boundary and with matching edges inside?

133. [*21*] (P. A. MacMahon, 1921.) A set of 24 *square* tiles can be constructed, analogous to the triangular tiles of (58), if we restrict ourselves to just three colors. For example, they can be arranged in a 4×6 rectangle as shown, with all-white border. In how many ways can this be done? 

134. [*23*] The nonwhite areas of the pattern in exercise 133 form polyominoes (rotated 45°); in fact, the lighter color has an S pentomino, while the darker color has both P and V. How often do each of the twelve pentominoes occur, among all of the solutions?

135. [*23*] (H. L. Nelson, 1970.) Show that MacMahon’s squares of exercise 133 can be used to wrap around the faces of a $2 \times 2 \times 2$ cube, matching colors wherever adjacent.

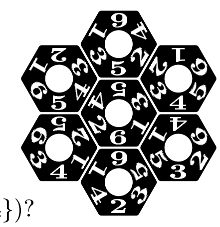
▸ **136.** [*HM28*] (J. H. Conway, 1958.) There are twelve ways to label the edges of a pentagon with $\{0, 1, 2, 3, 4\}$, if we don’t consider rotations and reflections to be different:



Cover a *dodecahedron* with these tiles, matching edge numbers. (Reflections are OK.)

weak symmetry
color patches
octahedron
MacMahon
jigsaw puzzle
triple three triominoes
triominoes
Philpott
equilateral triangle
MacMahon
polyominoes
pentomino
Nelson
wrap around
cube
Conway
pentagon
dodecahedron

137. [*22*] A popular puzzle called Drive Ya Nuts consists of seven "hex nuts" that have been decorated with permutations of the numbers $\{1, 2, 3, 4, 5, 6\}$. The object is to arrange them as shown, with numbers matching at the edges.



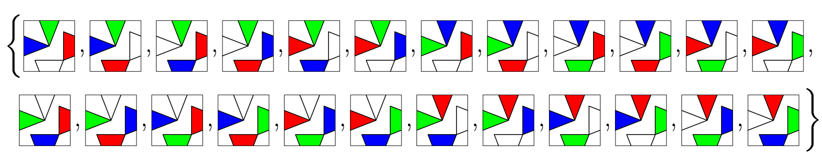
a) Show that this puzzle has a unique solution, with that particular set of seven. (Reflections of the nuts are *not* OK!)

b) Can those seven nuts form the same shape, but with the label numbers summing to 7 where they meet ($\{1, 6\}$, $\{2, 5\}$, or $\{3, 4\}$)?

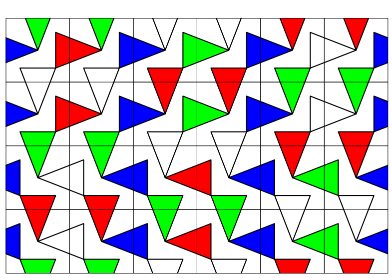
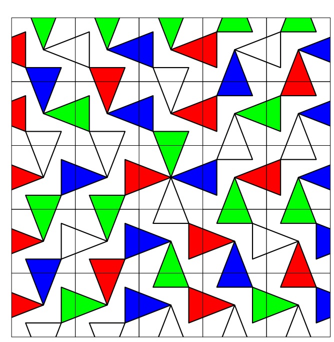
c) Hex nuts can be decorated with $\{1, 2, 3, 4, 5, 6\}$ in $5! = 120$ different ways. If seven of them are chosen at random, what's the approximate probability that they define a puzzle with a unique solution, under matching condition (a)?

d) Find seven hex nuts that have a unique solution under both conditions (a) and (b).

138. [*25*] (*Heads and tails.*) Here's a set of 24 square tiles that MacMahon missed(!):



They each show two "heads" and two "tails" of triangles, in four colors that exhibit all possible permutations, with heads pointing to tails. The tiles can be rotated, but not flipped over. We can match them properly in many ways, such as

 or ,

where the 4×6 arrangement will tile the plane; the 5×5 arrangement has a special "joker" tile in the middle, containing all four heads.

a) How many 4×6 arrangements will tile the plane? (Consider symmetries.)

b) Notice that the half-objects at the top, bottom, left, and right of the 5×5 arrangement match the heads in the middle. How many such arrangements are possible?

c) Devise a 5×5 arrangement that will tile the plane, in conjunction with the 5×5 pattern shown above. *Hint:* Use an "anti-joker" tile, which contains all four tails.

139. [*M25*] Excellent human-scale puzzles have been made by choosing nine of the 24 tiles in exercise 138, redrawing them with whimsical illustrations in place of the triangles, and asking for a 3×3 arrangement in which heads properly match tails.

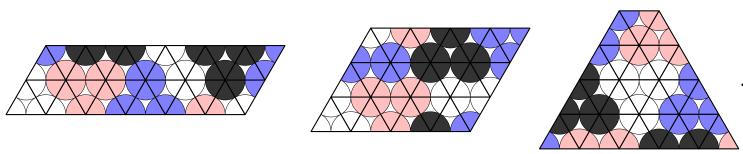
a) How many of the $\binom{24}{9}$ choices of 9 tiles lead to essentially different puzzles?

b) How many of those puzzles have exactly k solutions, for $k = 0, 1, 2, \ldots$?

140. [*29*] (C. D. Langford, 1959.) MacMahon colored the *edges* of his tiles, but we can color the *vertices* instead. For example, we can make two parallelograms, or a

Drive Ya Nuts
cyclic permutations
puzzle design
Heads and tails
MacMahon
permutations
flipped over
tile the plane
torus
symmetries
wallpaper
Langford
MacMahon
vertex matching

truncated triangle, by assembling the 24 vertex-colored analogs of (58):



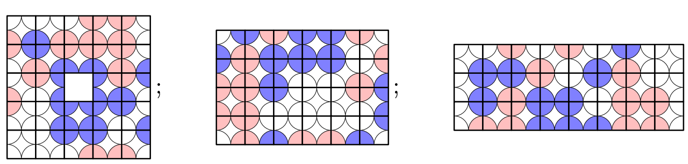
Such arrangements are much rarer than those based on edge matching, because edges are common to only two tiles but vertices might involve up to six.

a) In how many essentially distinct ways can those shapes be formed?

b) The first parallelogram is a scaled-up version of the "straight hexiamond" ▱, with dimensions doubled. How many of the other eleven scaled-up hexiamond shapes can be assembled from Langford's tiles? (See exercises 125 and 309.)

c) Each of the seven *tetrahexes* also yields an interesting shape that consists of 24 triangles. (See exercise 316.) How do Langford's tiles behave in those shapes?

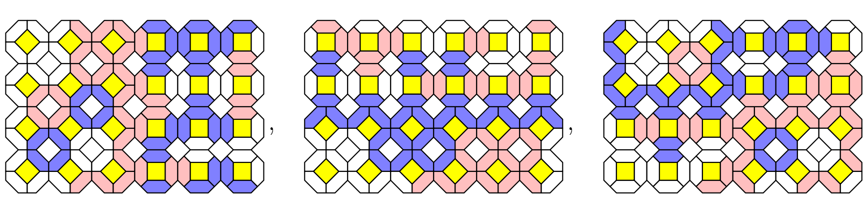
141. [*24*] Combining exercises 133 and 140, we can also adapt MacMahon's 24 tri-colored squares to vertex matching instead of edge matching. Noteworthy solutions are



a) In how many essentially different ways can those 24 tiles be properly packed into rectangles of these sizes, leaving a hole in the middle of the 5×5?

b) Discuss tiling the plane with such solutions.

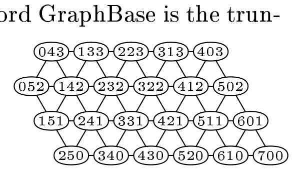
▸ **142.** [*23*] (Zdravko Zivkovic, 2008.) Edge and vertex matching can be combined into a single design if we replace MacMahon's 24 squares by 24 octagons. For example,



(i) (ii) (iii)

illustrate 4×6 arrangements in which there's vertex matching in the (i) left half, (ii) bottom half, or (iii) northwest and southeast quadrants, while edge matching occurs elsewhere. (We get vertex matching when an octagon's center is '◇', edge matching when it's '□'.) How many 4×6 arrangements satisfy (i), (ii), and (iii), respectively?

▸ **143.** [*M25*] The graph *simplex*$(n, a, b, c, 0, 0, 0)$ in the Stanford GraphBase is the truncated triangular grid consisting of all vertices xyz such that $x + y + z = n$, $0 \le x \le a$, $0 \le y \le b$, and $0 \le z \le c$. Two vertices are adjacent if their coordinates all differ by at most 1. The boundary edges always define a convex polygon. For example, *simplex*$(7, 7, 5, 3, 0, 0, 0)$ is illustrated here.



a) What *simplex* graphs correspond to the three shapes in exercise 140?

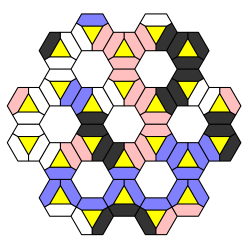
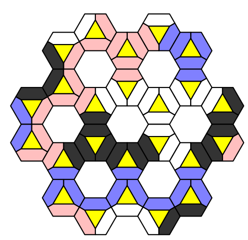
hexiamond
tetrahexes
MacMahon
tiling the plane
Zivkovic
octagons
simplex
Stanford GraphBase
triangular grid
convex polygon

b) The examples in (a) have 24 interior triangles, but *simplex*$(7,7,5,3,0,0,0)$ has 29. Can any other convex polygons be made from 24 triangles, connected edgewise?

c) Design an efficient algorithm that lists all possible convex polygons that can be formed from exactly N triangles, given N. *Hint:* Every convex polygon in a triangular grid can be characterized by the six numbers in its boundary path $x_0x_1x_2x_3x_4x_5$, which moves x_k steps in direction $(60k)^\circ$ for $k = 0, 1, \ldots, 5$. For example, the boundary of *simplex*$(7,7,5,3,0,0,0)$ is 503412.

d) Can every convex polygon in a triangular grid be described by a *simplex* graph?

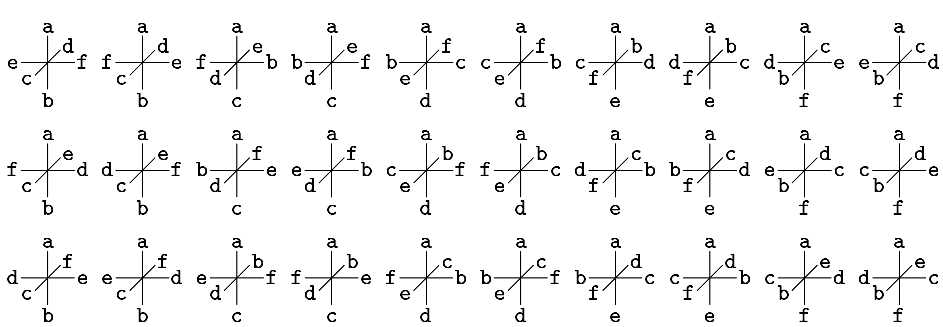
144. [*24*] The idea of exercise 142 applies also to triangles and hexagons, allowing us to do both vertex and edge matching with yet another set of 24 tiles:

(i) , (ii) .

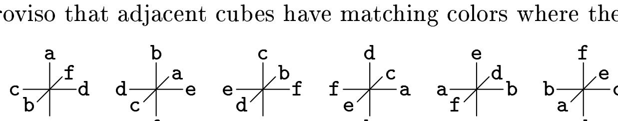
Here there's vertex matching in the bottom five tiles of (i), and in the upper left five and bottom five of (ii), with edge matching elsewhere. In how many ways can the big hexagon be made from these 24 little hexagons, under constraints (i) and (ii)?

▶ **145.** [*M20*] Many problems that involve an $l \times m \times n$ cuboid require a good internal representation of its $(l{+}1)(m{+}1)(n{+}1)$ vertices, its $l(m{+}1)(n{+}1) + (l{+}1)m(n{+}1) + (l{+}1)(m{+}1)n$ edges, and its $lm(n{+}1){+}l(m{+}1)n{+}(l{+}1)mn$ faces, in addition to its lmn individual cells. Show that there's a convenient way to do this with integer coordinates (x,y,z) whose ranges are $0 \le x \le 2l$, $0 \le y \le 2m$, $0 \le z \le 2n$.

▶ **146.** [*M30*] There are 30 ways to paint the colors $\{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f}\}$ on the faces of a cube:



(If a is on top, there are five choices for the bottom color, then six cyclic permutations of the remaining four.) Here's one way to arrange six differently painted cubes in a row, with distinct colors on top, bottom, front, and back (as in "Instant Insanity"), and with the further proviso that adjacent cubes have matching colors where they share a face:

 $(*)$

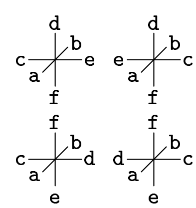
a) Explain why any such arrangement also has the same color at the left and right.

hexagons
cuboid
coordinates
cube, coordinates for
cube
Instant Insanity

bricks
Vertex-colored tetrahedra
simplex
tetrahedron
Endless Chain
Path dominoes
central symmetry
Hamiltonian cycle
loop

b) Invent a way to name each cube, distinguishing it from the other 29.
c) How many essentially different arrangements like (∗) are possible?
d) Can all 30 cubes be used to make five such arrangements simultaneously?

147. [*30*] The 30 cubes of exercise 146 can be used to make "bricks" of various sizes $l \times m \times n$, by assembling $l \cdot m \cdot n$ of them into a cuboid that has solid colors on each exterior face, as well as matching colors on each interior face. For example, each cube naturally joins with its mirror image to form a $1 \times 1 \times 2$ brick. Two such bricks can then join up to make a $1 \times 2 \times 2$; the one illustrated here has `a` in front, `b` in the back, `c` at the left and right, `d` at the top, and `e` at the bottom.



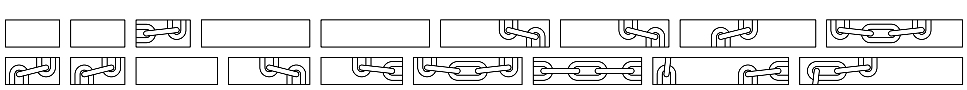
a) Assemble all 30 cubes into a magnificent brick of size $2 \times 3 \times 5$.
b) Compile a catalog of all the essentially different bricks that can be made.

148. [*24*] Find all the distinct cubes whose faces are colored `a`, `b`, or `c`, when opposite faces are required to have different colors. Then arrange them into a symmetric shape (with matching colors wherever they are in contact).

149. [*M22*] (*Vertex-colored tetrahedra.*) The graph *simplex*$(3,3,3,3,3,0,0)$ is a tetrahedron of side 3 with 20 vertices. It has 60 edges, which come from 10 unit tetrahedra.

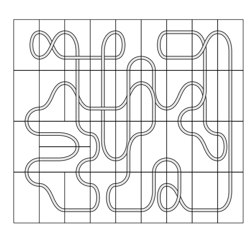
There are ten ways to color the vertices of a unit tetrahedron with four of the five colors $\{$`a`, `b`, `c`, `d`, `e`$\}$, because mirror reflections are distinct. Can those ten colored tetrahedra be packed into *simplex*$(3,3,3,3,3,0,0)$, with matching colors at every vertex?

150. [*23*] Here's a classic 19th century puzzle that was the first of its kind: "Arrange all the pieces to fill the square ... so that all the links of the Chain join together, forming an Endless Chain. The Chain may be any shape, so long as all the links join together, and all the pieces are used. This Puzzle can be done several different ways."



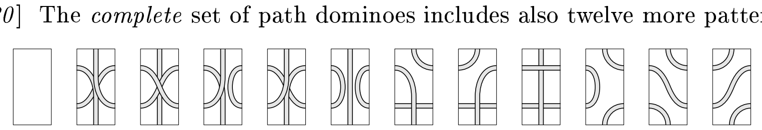
(The desired square is 8×8.) In exactly how many different ways *can* it be solved?

▸ **151.** [*30*] (*Path dominoes.*) A domino has six natural attachment points on its boundary, where we could draw part of a path that connects to neighboring dominoes. Thus $\binom{6}{2} = 15$ different partial paths could potentially be drawn on it. However, only 9 distinct domino patterns with one subpath actually arise, because the 15 possibilities are reduced under 180° rotation to six pairs, plus three patterns that have central symmetry. Similarly, there are 27 distinct domino patterns that contain *two* partial paths (where the paths might cross each other). An 8×9 arrangement, which nicely illustrates all 36 of the possibilities, is shown; notice that its path is a Hamiltonian cycle, consisting of a *single loop*.



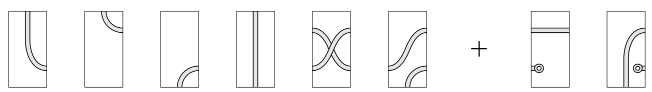
a) Only two of the dominoes in the arrangement above are in horizontal position. Find a single-loop 8×9 arrangement that has 18 horizontals and 18 verticals.
b) Similarly, find an arrangement that has the *maximum* number of horizontals.

152. [*30*] The *complete* set of path dominoes includes also twelve more patterns:



Arrange all 48 of them in an 8×12 array, forming a single loop.

153. [*25*] Here are six of the path dominoes, plus a "start" piece and a "stop" piece:



a) Place them within a 4×5 array so that they define a path from "start" to "stop."
b) How many distinct "start" or "stop" pieces are possible, if they're each supposed to contain a single subpath together with a single terminal point?
c) Design an eight-piece puzzle that's like (a), but it involves *four* of the two-subpath dominoes instead of only two. (Your puzzle should have a unique solution.)

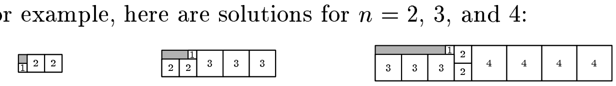
154. [*M30*] (C. R. J. Singleton, 1996.) After twelve days of Christmas, the person who sings a popular carol has received twelve partridges in pear trees, plus eleven pairs of humming birds, ..., plus one set of twelve drummers drumming, from his or her true love. Therefore an "authentic" partridge puzzle should try to pack $(n+1-k)$ squares of size $k \times k$, for $1 \le k \le n$, into a box that contains $P(n) = n \cdot 1^2 + (n-1) \cdot 2^2 + \cdots + 1 \cdot n^2$ cells. For which values of n is $P(n)$ a perfect square?

155. [*20*] That "authentic" partridge puzzle has a square solution when $n = 6$.
a) Exactly how many different solutions does it have in that case?
b) The *affinity score* of a partridge packing is the number of internal edges that lie on the boundary between two squares of the same size. (In (62) the scores are 165 and 67.) What solutions to (a) have the maximum and minimum affinity scores?

▸ **156.** [*30*] Straightforward backtracking will solve the partridge puzzle for $n = 8$, using bitwise techniques to represent a partially filled 36×36 square in just 36 octabytes, instead of by treating it as the huge MCC problem (61) and applying a highly general solver such as Algorithm M. Compare these two approaches, by implementing them both. How many essentially different solutions does that partridge puzzle have?

157. [*22*] Complete the study of small partridges by extending (63) to $n = 6$ and 7.

158. [*23*] Another variation of the partridge puzzle when $2 \le n \le 7$ asks for the *smallest rectangular area* that will contain k nonoverlapping squares of size $k \times k$ for $1 \le k \le n$. For example, here are solutions for $n = 2$, 3, and 4:



(To show optimality for $n = 4$ one must prove that rectangles of sizes 6×17, 8×13, 5×21, and 7×15 are too small.) Solve this puzzle for $n = 5$, 6, and 7.

▸ **159.** [*21*] Suggest a way to speed up the text's solution to the 5-queens problem, by using the symmetries of a square to modify the items and options of (64).

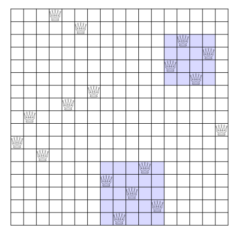
160. [*21*] The 5-queens problem leads to an interesting graph, whose vertices are the 4860 solutions, with $u \mathrel{-\!\!-} v$ when we can get from u to v by moving one queen. How many connected components does this graph have? Is one of them a "giant"?

▸ **161.** [*23*] Three restricted queen-domination problems are prominent in the literature:
i) No two queens of a solution attack each other.
ii) Each queen of a solution is attacked by at least one of the others.
iii) The queens of a solution form a clique.

(The third and fourth examples in (65) are instances of types (ii) and (i).)

Explain how to formulate each of these variants as an MCC problem, analogous to (64). How many solutions of each type are present in the 5-queens problem?

Singleton
partridge puzzle
Diophantine equations
affinity score
backtracking
bitwise techniques
5-queens problem
symmetries of a square
components
giant
queen-domination problems
clique dominators
MCC problem



162. [*24*] Say that a $\mathcal{Q}_n$ is an $n \times n$ array of n nonattacking queens. Sometimes a $\mathcal{Q}_n$ contains a $\mathcal{Q}_m$ for $m < n$; for example, eight of the possible $\mathcal{Q}_5$'s contain a $\mathcal{Q}_4$, and the $\mathcal{Q}_{17}$ illustrated here contains both a $\mathcal{Q}_4$ and a $\mathcal{Q}_5$.

What is the smallest n such that at least one $\mathcal{Q}_n$ contains (a) two $\mathcal{Q}_4$'s? (b) three $\mathcal{Q}_4$'s? (c) four $\mathcal{Q}_4$'s? (d) five $\mathcal{Q}_4$'s? (e) two $\mathcal{Q}_5$'s? (f) three $\mathcal{Q}_5$'s? (g) four $\mathcal{Q}_5$'s? (h) two $\mathcal{Q}_6$'s? (i) three $\mathcal{Q}_6$'s?

163. [*20*] Explain the peculiar rule for setting p in (71).

164. [*17*] When Algorithm M finds a solution $x_0x_1\ldots x_{l-1}$ in step M2, some of the nodes x_j might represent the fact that some primary item will appear in no further options. Explain how to handle this "null" case, by modifying answer 13.

165. [*M30*] Consider an MCC problem in which we must choose 2 of 4 options to cover item 1, and 5 of 7 options to cover item 2; the options don't interact.

a) What's the size of the search tree if we branch first on item 1, then on item 2? Would it better to branch first on item 2, then on item 1?

b) Generalize part (a) to the case when item 1 needs p of $p+d$ options, while item 2 needs q of $q+d$ options, where $q > p$ and $d > 0$.

166. [*21*] Extend answer 9 to the more general situation that arises in Algorithm M:

a) Let θ_p be the number of different choices that will be explored at the current position of the search tree if primary item p is selected for branching. Express θ_p as a function of $\texttt{LEN}(p)$, $\texttt{SLACK}(p)$, and $\texttt{BOUND}(p)$.

b) Suppose $\theta_p = \theta_{p'}$ and $\texttt{SLACK}(p) = \texttt{SLACK}(p') = 0$, but $\texttt{LEN}(p) < \texttt{LEN}(p')$. Should we prefer to branch on p or on p', based on exercise 165?

167. [*24*] Let M_p be the number of options that involve the primary item p in a given MCC problem, and suppose that the upper bound v_p for p's multiplicity is $\geq M_p$. Does the precise value of this upper bound affect the behavior of Algorithm M? (In other words, does $v_p = \infty$ lead to the same running time as $v_p = M_p$?)

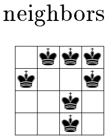
▸ **168.** [*15*] An MCC problem might have two *identical* options α, whose items are allowed to occur more than once. In such cases we might want the second copy of α to be in the solution only if the first copy is also present. How can that be achieved?

▸ **169.** [*22*] Let G be a graph with n vertices. Formulate the problem of finding all of its t-element independent sets as an MCC problem with $1+n$ items and n options.

170. [*22*] Continuing exercise 169, generate all of G's t-element *kernels* — its *maximal* independent sets. (Your formulation will now need additional items and options.)

171. [*25*] Label the vertices of the Petersen graph with ten 5-letter words, in such a way that vertices are adjacent if and only if their labels have a common letter.

▸ **172.** [*29*] A *snake-in-the-box path* in a graph G is a set U of vertices for which the induced graph $G\,|\,U$ is a path. (Thus there are start/stop vertices $s \in U$ and $t \in U$ that each have exactly one neighbor in U; every other vertex of U has exactly two neighbors in U; and $G\,|\,U$ is connected.)



For example, let $G = P_4 \boxtimes P_4$ be the graph of king moves on a 4×4 board. The set of kings illustrated at the right is *not* a snake-in-the-box path in G; but it becomes one if we remove the king in the corner.

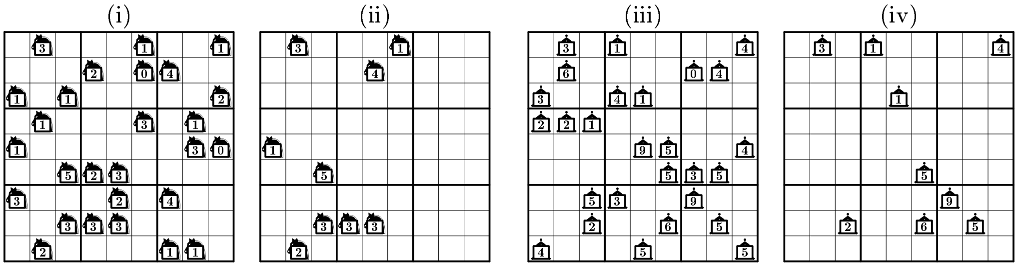
a) Use Algorithm M to discover all of the longest snake-in-the-box paths that are possible on an 8×8 chessboard, when G is the graph of all (i) king moves; (ii) knight moves; (iii) bishop moves; (iv) rook moves; (v) queen moves.

queens
n queens problem
MRV heuristic
LEN
SLACK
BOUND
MCC problem
identical options
independent sets
MCC problem
kernels
Petersen graph
5-letter words
snake-in-the-box path
induced graph
connected
strong product $\boxtimes$
king moves
chessboard
knight moves
bishop moves
rook moves
queen moves

b) Similarly, a *snake-in-the-box cycle* is a set for which $G \mid U$ is a cycle. (In other words, that induced graph is connected and 2-regular.) What are the longest possible snake-in-the-box cycles for those five chess pieces?

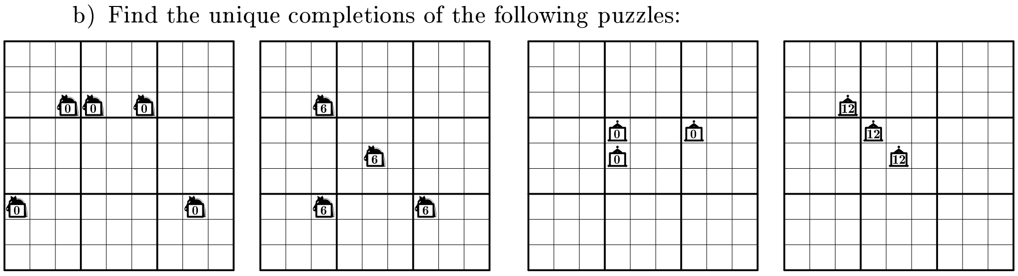
▸ **173.** [*30*] (*Knight and bishop sudoku.*) Diagram (i) shows 27 knights, arranged with three in each row, three in each column, and three in each 3×3 box. Each of them has been labeled with the number of others that are a knight's move away. Diagram (ii) shows 9 of them, from which the positions of the other 18 can be deduced. Diagrams (iii) and (iv) are analogous, but for bishops instead of knights: (iii) solves puzzle (iv).

(i) (ii) (iii) (iv)



a) Explain how to find all completions of such diagrams using Algorithm M.

b) Find the unique completions of the following puzzles:



c) Compose additional puzzles like those of (b), in which all clues have the same numerical labels. Try to use as few clues as possible.

d) Construct a uniquely solvable knight sudoku puzzle that has only three clues.

174. [*35*] (Nikolai Beluhov, 2019.) Find a uniquely solvable sudoku puzzle with nine labeled knights that remains uniquely solvable when the knights are changed to bishops.

▸ **175.** [*M21*] Given an $M \times N$ matrix $A = (a_{ij})$ of 0s and 1s, explain how to find all vectors $x = (x_1 \ldots x_M)$ with $0 \le x_i \le a_i$ for $1 \le i \le M$ such that $xA = (y_1 \ldots y_N)$, where $u_j \le y_j \le v_j$ for $1 \le j \le N$. (This generalizes the MCC problem by allowing the ith option to be repeated up to a_i times.)

▸ **176.** [*M25*] Given an $M \times N$ matrix $A = (a_{ij})$ of 0s, 1s, and 2s, explain how to find all subsets of its rows that sum to exactly (a) 2 (b) 3 (c) 4 (d) 11 in each column, by formulating those tasks as MCC problems.

177. [*M21*] Algorithm 7.2.1.5M generates the $p(n_1, \ldots, n_m)$ partitions of the multiset $\{n_1 \cdot x_1, \ldots, n_m \cdot x_m\}$ into submultisets. Use the previous two exercises to generate these partitions with Algorithm M, in the cases where $n_1 = \cdots = n_s = 1$ and $n_{s+1} = \cdots = n_{s+t} = 2$ and $s + t = m$. Also generate the $q(n_1, \ldots, n_m)$ multipartitions into *distinct* multisets.

178. [*M22*] (*Factorizations of an integer.*) Use Algorithm M to find all representations of 360 as a product $n_1 \cdot n_2 \cdot \ldots \cdot n_t$, where (a) $1 < n_1 < \cdots < n_t$; (b) $2 \le n_1 \le \cdots \le n_t$.

snake-in-the-box cycle
2-regular
Knight and bishop sudoku
pi, as source
Beluhov
01 mtx
MCC problem
repeated options
012 mtx
MCC problems
repeated items
partitions of the multiset
multiset
multipartitions into *distinct* multisets
multipartitions
Factorizations of an integer

Eppstein
strict exact cover problem
Eppstein
strict exact cover problem
extreme problem
exponential generating function
Gould numbers
Peirce's triangle

179. [*15*] By removing duplicate rows and columns, matrix A reduces to A':

$$A=\begin{pmatrix}1&0&0&0&0&0\\0&1&1&1&0&0\\1&1&0&1&1&1\\0&1&1&1&0&0\\0&0&0&0&1&1\\0&0&0&0&1&1\\1&0&1&0&0&0\\0&1&0&1&1&1\\1&1&1&1&0&0\end{pmatrix};\qquad A'=\begin{pmatrix}1&0&0&0\\0&1&1&0\\1&1&0&1\\0&0&0&1\\1&0&1&0\\0&1&0&1\\1&1&1&0\end{pmatrix}.$$

Derive the exact covers of A from the exact covers of A'.

▸ **180.** [*M28*] (D. Eppstein, 2008.) Prove that every strict exact cover problem with parameters $1 \le t' \le t$, as defined in (74), contains t' items $i_1, \ldots, i_{t'}$ and $t{+}t'{-}1$ options

$$o_p = \text{`}i_1 \ldots i_p\text{'},\ \text{for } 1 \le p \le t'; \qquad o_{p+q} = \text{`}\ldots\ i_{t'}\ \ldots\text{'},\ \text{for } 1 \le q < t.$$

Furthermore, $i_r \in o_{p+q}$ if and only if $1 \le q < t - r - t'$, for $1 \le r \le t'$.

181. [*M20*] Find constants c_r such that $D(5n{+}r) = 4^n c_r - \frac{1}{3}$ for $n \ge 3$ and $0 \le r < 5$.

182. [*21*] (D. Eppstein, 2008.) Find a strict exact cover problem with 8 options, whose search tree contains 16 nodes and 7 solutions.

183. [*46*] Let $\widehat{D}(n)$ be the maximum number of nodes in Algorithm X's search tree, taken over all strict exact cover problems with n options. What is $\limsup_{n\to\infty} \widehat{D}(n)^{1/n}$?

▸ **184.** [*M22*] Suppose $0 \le t \le \varpi_n$. Is there a strict exact cover problem with n items that has exactly t solutions? (For example, consider the case $n = 9$, $t = 10000$.)

185. [*M23*] What is the largest number of solutions to a strict exact cover problem that has N_1 primary items and N_2 secondary items?

186. [*M24*] Consider $l = 0$ when Algorithm X is given the extreme problem of order n.

a) How many updates, u_n, does it perform when covering i in step X4?

b) How many does it perform in step X5, when the option containing x_0 has size k?

c) Therefore derive (84).

187. [*HM29*] Let $X(z) = \sum_n x_n z^n/n!$ generate the sequence $\langle x_n\rangle$ of (82).

a) Use (84) to prove that $X(z) = e^{e^z}\int_0^z \bigl((2t-1)e^{4t} - (t-1)e^{3t} + 2te^{2t} + e^t\bigr)e^{-e^t}\,dt$.

b) Let $T_{r,s}(z) = e^{e^z}\int_0^z t^r e^{st} e^{-e^t}\,dt$. Prove that $T_{r,0}(z)/r!$ generates $\langle a_{n,r+1}\rangle$ in (83).

c) Show that $T_{r,0}(z) = (T_{r+1,1}(z) + z^{r+1})/(r+1)$; furthermore, when $s > 0$,

$$T_{r,s}(z) = \left(\sum_{k=0}^{r} \frac{(-1)^k r^{\underline{k}}}{s^{k+1}}\bigl(T_{r-k,s+1}(z) + z^{r-k}e^{sz}\bigr)\right) - \frac{(-1)^r r!}{s^{r+1}}e^{e^z-1}.$$

d) Therefore $X(z) = 22e^{e^z-1} + 12T_{0,0}(z) - (2z-1)e^{3z} - 5ze^{2z} - (12z+5)e^z - 12z - 18$.

▸ **188.** [*M21*] Prove that the Gould numbers $\langle \widehat{\varpi}_n\rangle = \langle 0, 1, 1, 3, 9, 31, 121, 523, 2469, \ldots\rangle$ can be calculated rapidly by forming a triangle of numbers analogous to Peirce's triangle 7.2.1.5–(12):

$$\begin{array}{cccccc}0\\1&1\\3&2&1\\9&6&4&3\\31&22&16&12&9\\121&90&68&52&40&31\end{array}$$

Here the entries $\widehat{\varpi}_{n1}$, $\widehat{\varpi}_{n2}$, ..., $\widehat{\varpi}_{nn}$ of the nth row obey the simple recurrence

$$\widehat{\varpi}_{nk} = \widehat{\varpi}_{(n-1)k} + \widehat{\varpi}_{n(k+1)}, \text{ if } 1 \le k < n; \qquad \widehat{\varpi}_{nn} = \widehat{\varpi}_{(n-1)1}, \text{ if } n > 2;$$

and initially $\widehat{\varpi}_{11} = 0$, $\widehat{\varpi}_{22} = 1$. *Hint:* Give a combinatorial interpretation of $\widehat{\varpi}_{nk}$.

189. [*HM34*] Let $\rho_n = \widehat{\varpi}_n - \hat{g}\varpi_n$ (see (86)). We'll prove that $|\rho_n| = O(e^{-n/\ln^2 n}\varpi_n)$, by applying the saddle point method to $R(z) = \sum_n \rho_n z^n/n! = e^{e^z}\int_z^\infty e^{-e^t}dt$. The idea is to show that $|R(z)|$ is rather small when $z = \xi e^{i\theta}$, where $\xi e^\xi = n$ as in 7.2.1.5–(24).

a) Express $|e^{e^z}|$ and $|e^{-e^z}|$ in terms of x and y when $z = x + iy$.
b) If $0 \le \theta \le \frac{\pi}{2}$, $y = \xi \sin\theta \le \frac{3}{2}$, $0 < c_1 < \cos\frac{3}{2}$, prove $|R(\xi e^{i\theta})| = O(\exp(e^\xi - c_1 e^\xi))$.
c) If $0 \le \theta \le \frac{\pi}{2}$, $y = \xi \sin\theta \ge \frac{3}{2}$, $0 < c_2 < \frac{9}{8}$, prove $|R(\xi e^{i\theta})| = O(\exp(e^\xi - c_2 e^\xi/\xi))$.
d) Consequently $\rho_{n-1}/\varpi_{n-1} = O(e^{-n/\ln^2 n})$, as desired.

190. [*HM46*] Study the *signs* of the residual quantities $\rho_n = \widehat{\varpi}_n - \hat{g}\varpi_n$ in exercise 189.

191. [*HM22*] The length of the tail of a random set permutation is known to have a probability distribution whose generating function is $G(z) = \int_0^\infty e^{-x}(1+x)^z dx - 1 = \sum_{k=1}^\infty \hat{g}_k z^k$. (The first few probabilities in this distribution are

$$(\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_9) \approx (.59635, .26597, .09678, .03009, .00823, .00202, .00045, .00009, .00002);$$

see answer 189.) What is the average length? What is the variance?

192. [*HM29*] What's the asymptotic value of $\hat{g}_n$ when n is large?

193. [*M21*] Why do (87) and (88) count updates when matching in complete graphs?

194. [*HM23*] Consider recurrences of the form $X(t+1) = a_t + tX(t-1)$. For example, $a_t = 1$ yields the total number of nodes in the search tree for matching K_{t+1}.

a) Prove that $1 + 2q + (2q)(2q-2) + \cdots + (2q)(2q-2)\ldots(2) = \lfloor e^{1/2}2^q q!\rfloor$.
b) Find a similar "closed formula" for $1 + (2q-1) + (2q-1)(2q-3) + \cdots + (2q-1)\cdot(2q-3)\ldots(3)(1)$. *Hint:* Use the fact that $e^x \operatorname{erf}(\sqrt{x}) = \sum_{n\ge0} x^{n+1/2}/(n+1/2)!$.
c) Estimate the solution $U(2q+1)$ of (87) to within $O(1)$.
d) Similarly, give a good approximation to the solution $U(2q)$ of (88).

▸ **195.** [*M22*] Approximately how many updates does Algorithm X perform, when it is asked to find all of the perfect matchings of the graph (89)?

▸ **196.** [*M29*] Given a bounded permutation problem defined by $a_1 \ldots a_n$, consider the "dual" problem defined by $b_1 \ldots b_n$, where b_k is the number of j such that $1 \le j \le n$ and $a_j \ge n + 1 - k$. [Equivalently, $b_n \ldots b_1$ is the *conjugate* of the integer partition $a_n \ldots a_1$, in the sense of Section 7.2.1.4.]

a) What is the dual problem when $n = 9$ and $a_1 \ldots a_9 = 246677889$?
b) Prove that the solutions to the dual problem are essentially the *inverses* of the permutations that solve the original problem.
c) If Algorithm X begins with an a_1-way branch on item X_1, how many updates does it perform while preparing for the subproblems at depth 1 of its search tree?
d) How many solutions does a bounded permutation problem have, given $a_1 \ldots a_n$?
e) Give a formula for the total number of updates, assuming that the algorithm always branches on X_j at depth $j - 1$ of the search tree.
f) Evaluate the formula of (e) when $a_j = n$ for $1 \le j \le n$ (that is, *all* permutations).
g) Evaluate the formula of (e) when $a_j = \min(j+1, n)$ for $1 \le j \le n$.
h) Evaluate the formula of (e) when $a_j = \min(2j, n)$ for $1 \le j \le n$.
i) Show, however, that the assumption in (e) is not always correct. How can the total updates be calculated correctly in general?

saddle point method
tail of a random set permutation
probability distribution
generating function
variance
updates
matching
complete graphs
recurrences
bounded permutation problem
dual
conjugate
inverses
permutations

197. [*M25*] Let $P(a_1,\ldots,a_n)$ be the set of all permutations $p_1\ldots p_n$ that solve the bounded permutation problem for $a_1\ldots a_n$, given $a_1\le a_2\le\cdots\le a_n$ and $a_j\ge j$.

a) Prove that $P(a_1,\ldots,a_n)=\{(nt_n)\ldots(2t_2)(1t_1)\mid j\le t_j\le a_n \text{ for } 1\le j\le n\}$.

b) Also prove that $P(a_1,\ldots,a_n)=\{\sigma_{nt_n}\ldots\sigma_{2t_2}\,\sigma_{1t_1}\mid j\le t_j\le a_n \text{ for } 1\le j\le n\}$, where σ_{st} is the $(t+1-s)$-cycle $(t\;t{-}1\;\ldots\,s{+}1\;s)$.

c) Let $C(p)$ be the number of cycles in the permutation p, and let $I(p)$ be the number of inversions. Find the generating functions

$$C(a_1,\ldots,a_n)=\sum_{p\in P(a_1,\ldots,a_n)} z^{C(p)}\qquad\text{and}\qquad I(a_1,\ldots,a_n)=\sum_{p\in P(a_1,\ldots,a_n)} z^{I(p)}.$$

198. [*M25*] Let $\pi_{rs}=\Pr(p_r=s)$, when p is a random element of $P(a_1,\ldots,a_n)$.

a) Compute these probabilities when $n=9$ and $a_1a_2\ldots a_9=255667999$.

b) If $r<r'$ and $s<s'$, show that $\pi_{rs}/\pi_{rs'}=\pi_{r's}/\pi_{r's'}$, when $\pi_{rs'}\pi_{r's'}\ne 0$.

199. [*M25*] Analyze the behavior of Algorithm X on the 3D matching problem whose options are '$a_i\ b_j\ c_k$' for $1\le i,j\le n$ and $1\le k\le (i\le m?\ m-1\colon n)$.

▸ **200.** [*HM25*] (A. Björklund, 2010.) We can use polynomial algebra, instead of backtracking, to decide whether or not a given 3D matching problem is solvable. Let the items be $\{a_1,\ldots,a_n\}$, $\{b_1,\ldots,b_n\}$, $\{c_1,\ldots,c_n\}$, and assign a symbolic variable to each option. If X is any subset of C, let $Q(X)$ be the $n\times n$ matrix whose entry in row i and column j is the sum of the variables for all options '$a_i\ b_j\ c_k$' with $c_k\notin X$.

For example, suppose $n=3$. The seven options t: '$a_1\ b_1\ c_2$', u: '$a_1\ b_2\ c_1$', v: '$a_2\ b_3\ c_2$', w: '$a_2\ b_3\ c_3$', x: '$a_3\ b_1\ c_3$', y: '$a_3\ b_2\ c_1$', z: '$a_3\ b_2\ c_2$' yield the matrices

$$\begin{array}{cccccccc} X= & \emptyset & \{c_3\} & \{c_2\} & \{c_2,c_3\} & \{c_1\} & \{c_1,c_3\} & \{c_1,c_2\}\\ Q(X)= & \begin{pmatrix} t & u & 0\\ 0 & 0 & v{+}w\\ x & y{+}z & 0\end{pmatrix} & \begin{pmatrix} t & u & 0\\ 0 & 0 & v\\ 0 & y{+}z & 0\end{pmatrix} & \begin{pmatrix} 0 & u & 0\\ 0 & 0 & w\\ x & y & 0\end{pmatrix} & \begin{pmatrix} 0 & u & 0\\ 0 & 0 & 0\\ 0 & y & 0\end{pmatrix} & \begin{pmatrix} t & 0 & 0\\ 0 & 0 & v{+}w\\ x & z & 0\end{pmatrix} & \begin{pmatrix} t & 0 & 0\\ 0 & 0 & v\\ 0 & z & 0\end{pmatrix} & \begin{pmatrix} 0 & 0 & 0\\ 0 & 0 & w\\ x & 0 & 0\end{pmatrix}\end{array}$$

(and $Q(C)$ is always zero). The determinant of $Q(\emptyset)$ is $u(v{+}w)x-t(v{+}w)(y{+}z)$.

a) If the given problem has r solutions, prove that the polynomial

$$S\;=\;\sum_{X\subseteq C}(-1)^{|X|}\det Q(X)$$

is the sum of r monomials, each with coefficient ±1. (In the given example it is $uvx-twy$.) *Hint:* Consider the case where all possible options are present.

b) Use this fact to design a randomized algorithm that decides q.s. whether or not a matching exists, in $O(2^n n^4)$ steps.

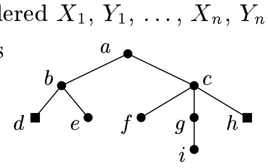
▸ **201.** [*M30*] Consider the bipartite matching problem that has $3n$ options, '$X_j\ Y_k$' for $1\le j,k\le n$ and $(j-k)\bmod n\in\{0,1,n-1\}$. (Assume that $n\ge 3$.)

a) What "natural, intuitively obvious" problem is equivalent to this one?

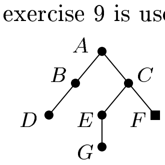
b) How many solutions does this problem have?

c) How many updates does Algorithm X make when finding all solutions, if the items are ordered $X_1, Y_1, \ldots, X_n, Y_n$, and if exercise 9 is used in step X3?

202. [*13*] What is



$\oplus$



?

203. [*M15*] Equation (95) shows that the binary operation $T\oplus T'$ on search trees has an identity element, '▪'. Is that operation (a) associative? (b) commutative?

bounded permutation problem
cycles
inversions
generating functions
Analyze
3D matching
Björklund
polynomial algebra
3D matching problem
randomized algorithm
q.s.
bipartite matching problem
MRV
binary operation
identity element
associative

204. [*M25*] True or false: Node $\alpha\alpha'$ is dominant in $T \oplus T'$ if and only if α is dominant in T and α' is dominant in T'. *Hint:* Express $\deg(\alpha\alpha')$ in terms of $\deg(\alpha)$ and $\deg(\alpha')$.

205. [*M28*] Prove Lemma D, about the structure of $T \oplus T'$.

206. [*20*] If T is minimally dominant and $\deg(\mathrm{root}(T)) \le \deg(\mathrm{root}(T'))$, show that it's easy to describe the tree $T \oplus T'$.

207. [*35*] The principal SAT solver that we shall discuss later, Algorithm 7.2.2.2C, maintains focus by computing "activity scores," which measure recent changes to the data structures. A similar idea can be applied to Algorithm X, by computing the score

$$\alpha_i = \rho^{t_1} + \rho^{t_2} + \cdots, \quad \text{for each item } i,$$

where ρ (typically 0.9) is a user-specified damping factor, and where i's list of active options was modified at times $t - t_1$, $t - t_2$, ...; here t denotes the current "time," as measured by some convenient clock. When step X3 chooses an item for branching, the MRV heuristic of exercise 9 rates i by its degree $\lambda_i = \texttt{LEN}(i)$; the new heuristic replaces this by

$$\lambda_i' = \begin{cases} \lambda_i, & \text{if } \lambda_i \le 1; \\ 1 + \lambda_i/(1 + \mu\alpha_i), & \text{if } \lambda_i \ge 2. \end{cases}$$

Here μ is another user-specified parameter. If $\mu = 0$, decisions are made as before; but larger and larger values of μ cause greater and greater attention to be given to the recently active items, even if they have a somewhat large degree of branching.

a) For example, suppose $\alpha_i = 1$, $\alpha_j = 1/2$, and $\mu = 1$. Which item will be preferable, i or j, if $\texttt{LEN}(i) = \texttt{LEN}(j) + 1$ and $0 \le \texttt{LEN}(j) \le 4$?
b) What modifications to Algorithm X will implement this scheme?
c) What values of ρ and μ will avoid exponential growth, when applied to n independent copies of the toy problems (90) and (92)?
d) Does this method save time in the Y pentomino problem of Fig. 73?

▶ **208.** [*21*] Modify the exact cover problem of Fig. 73 so that none of the Y pentominoes that occur in an 'H' or '⌶' have been flipped over. *Hint:* To prevent the flipped-over Y's marked 8 and b from occurring simultaneously, use the options '1a 2a 3a 4a 2b $V_{\texttt{1b}}$' and '1c 2c 3c 4c 3b $V_{\texttt{1b}}$', where $V_{\texttt{1b}}$ is a secondary item.

209. [*20*] Improve the search tree (93) in the same way that (100) improves on (91), by considering two bipairs of (92).

210. [*21*] A "bitriple" $(\alpha, \beta, \gamma; \alpha', \beta', \gamma')$ is analogous to a bipair, but with (97) replaced by $\alpha + \beta + \gamma = \alpha' + \beta' + \gamma'$. How can we modify an exact cover problem so that it excludes all solutions in which options α', β', and γ' are simultaneously present?

211. [*20*] Do the options of the text's formulation of the Langford pair problem have any bipairs? How about the n queens problem? And sudoku?

▶ **212.** [*M21*] If the primary items of an exact cover problem have been linearly ordered, we can say that the bipair $(\alpha, \beta; \alpha', \beta')$ is canonical if (i) the smallest item in all four options appears in α and α'; and (ii) option α is lexicographically smaller than option α', when their items have been listed in ascending order.

a) Prove that Theorem S applies to exact coverings that are strong according to this definition of canonicity. *Hint:* Show that it's a special case of the text's definition.
b) Does such an ordering justify the choices made in (99)?

213. [*M21*] If π and π' are two partitions of the same set, say that $\pi < \pi'$ if the restricted growth string of π is lexicographically less than the restricted growth string

minimally dominant
SAT solver
focus
activity scores
damping factor
MRV heuristic
Y pentomino
Y pentominoes
secondary item
bipairs
bitriple
bipair
Langford pair problem
n queens problem
sudoku
canonical
set partitions
restricted growth string
lexicographically less

of π'. Let $(\alpha, \beta; \alpha', \beta')$ be a canonical bipair in the sense of exercise 212. Also let π be a partition of the items such that α and β are two of its parts, and let π' be the same partition but with α' and β' substituted. Is $\pi < \pi'$?

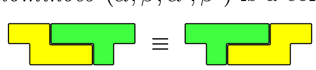
▶ **214.** [*21*] Under the assumptions of Theorem S, how can the set of all solutions to an exact cover problem be found from the set of all strong solutions?

▶ **215.** [*M30*] The perfect matching problem on the complete graph K_{2q+1} is the X2C problem with $2q+1$ primary items $\{0, \ldots, 2q\}$ and $\binom{2q+1}{2}$ options '$i\ j$' for $0 \le i < j \le 2q$.

a) How many bipairs are present in this problem?
b) Say that (i, j, k, l) is *excluded* if there's a canonical bipair $(\alpha, \beta; \alpha', \beta')$ for which $\alpha' =$ '$i\ j$' and $\beta' =$ '$k\ l$'. Prove that, regardless of the ordering of the options, the number of excluded quadruples is 2/3 of the number of bipairs.
c) What quadruples are excluded when the options are ordered lexicographically?
d) We reduce the amount of search by introducing a secondary item (i, j, k, l) for each excluded quadruple, and appending it to the options for '$i\ j$' and '$k\ l$'. Describe the search tree when this has been done for the quadruples of (c).
e) Show that only $\Theta(q^3)$ excluded quadruples suffice to obtain that search tree.
f) Order the options cleverly so that the search tree has only $2q+1$ nodes.
g) How many excluded quadruples suffice to obtain *that* search tree?

216. [*25*] Continuing exercise 215, experiment with the search trees that are obtained by (i) choosing a *random* ordering of the options, and (ii) using only m of the quadruples that are excluded by that ordering (again chosen at random).

217. [*M32*] A *bipair of pentominoes* $(\alpha, \beta; \alpha', \beta')$ is a configuration such as



where two pentominoes occupy a 10-cell region in two different ways. In this example we may write $\alpha = \mathrm{S} + 00 + 01 + 11 + 12 + 13$, $\beta = \mathrm{Y} + 02 + 03 + 04 + 05 + 14$, $\alpha' = \mathrm{S}+04+05+12+13+14$, $\beta' = \mathrm{Y}+00+01+02+03+11$; hence $\alpha+\beta = \alpha'+\beta'$ as in (97).

Compile a complete catalog of all bipairs that are possible with distinct pentominoes. In particular, show that each of the twelve pentominoes participates in at least one such bipair. (It's difficult to do this by hand without missing anything. One good approach is to exploit the equation $\alpha - \alpha' = -(\beta - \beta')$: First find all the delta values $\pm(\alpha - \alpha')$ for each of the twelve pentominoes individually; then study all deltas that are shared by two or more of them. For example, the S and Y pentominoes both have $00 + 01 - 04 - 05 + 11 - 14$ among their deltas.)

▶ **218.** [*20*] Why must i be uncolored, in the definition of "forcing" for Algorithm P?

219. [*20*] Suppose p and q are primary items of an XCC problem, and that every option containing p or q includes an uncolored instance of either i or j (or both), where i and j are other items; yet p and q never occur in the same option. Prove that every option that contains i or j, but neither p nor q, can be removed without changing the problem.

220. [*28*] Step P5 of Algorithm P needs to emulate step C5 of Algorithm C, to see if some primary item will lose all of its options. Describe in detail what needs to be done.

221. [*23*] After all options that begin with item i have been examined in step P5, those that were found to be blocked appear on a stack, starting at S. Explain how to delete them. *Caution:* The problem might become unsolvable when an option goes away.

222. [*22*] Before item i is deleted in step P7, it should be removed from every option that contains S, by changing the corresponding node into a spacer. All options that involve i but not S should also be deleted. Spell out the low-level details of this process.

strong solutions
perfect matching
complete graph
X2C problem
bipairs
canonical bipair
secondary item
pentominoes
forcing
stack
spacer

223. [*20*] Implement the output phase of Algorithm P (step P10).

▸ **224.** [*M21*] Construct an exact cover problem with $O(n)$ options that causes Algorithm P to perform n rounds of reduction (that is, it executes step P2 n times).

225. [*21*] Why does Algorithm P remove 235 options in the 6×10 pentomino problem, but only 151 options in the 6×15 case?

226. [*M20*] Assume that $a_1 \ldots a_{2n}$ is a Langford pairing, and let $a'_k = a_{2n+1-k}$ so that $a'_1 \ldots a'_{2n}$ is the reverse of $a_1 \ldots a_{2n}$. Are there any obvious relations between the sums

$$\Sigma_1 = \textstyle\sum_{k=1}^{2n} ka_k, \quad \Sigma'_1 = \sum_{k=1}^{2n} ka'_k, \quad \Sigma_2 = \sum_{k=1}^{2n} k^2 a_k, \quad \Sigma'_2 = \sum_{k=1}^{2n} k^2 a'_k?$$

What about the analogous sums $S = \sum_{k=1}^{2n} ka_k^2$ and $S' = \sum_{k=1}^{2n} k(a'_k)^2$?

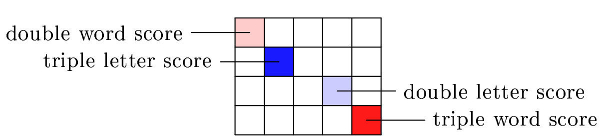
227. [*10*] What cost should be assigned to option (16), to minimize (a) Σ_2? (b) S?

228. [*M30*] The Langford pairings for $n=16$ that minimize Σ_2 turn out to be precisely the 12,016 pairings that minimize Σ_1; and their reversals turn out to be precisely the 12,016 pairings that *maximize* both Σ_2 and Σ_1. Is this surprising, or what?

▸ **229.** [*25*] What Langford pairings for $n=16$ are lexicographically smallest and largest?

230. [*20*] Explain how Algorithm X$^\$$, which *minimizes* the sum of option costs, can also be used to *maximize* that sum, in problems like that of Fig. 74.

231. [*21*] What's the maximum SCRABBLE®-like score you can achieve by filling the grid below with 4-letter and 5-letter words that all are among the (a) 1000 (b) 2000 (c) 3000 most common words of English having that many letters?



$\mathtt{A}_1$ $\mathtt{B}_3$ $\mathtt{C}_3$ $\mathtt{D}_2$ $\mathtt{E}_1$ $\mathtt{F}_4$ $\mathtt{G}_2$
$\mathtt{H}_4$ $\mathtt{I}_1$ $\mathtt{J}_8$ $\mathtt{K}_5$ $\mathtt{L}_1$ $\mathtt{M}_3$ $\mathtt{N}_1$
$\mathtt{O}_1$ $\mathtt{P}_3$ $\mathtt{Q}_{10}$ $\mathtt{R}_1$ $\mathtt{S}_1$ $\mathtt{T}_1$
$\mathtt{U}_1$ $\mathtt{V}_4$ $\mathtt{W}_4$ $\mathtt{X}_8$ $\mathtt{Y}_4$ $\mathtt{Z}_{10}$

For example, `WATCH|AGILE|RADAR|TREND` scores 26+10+7+18+14+9+5+7+24 points.

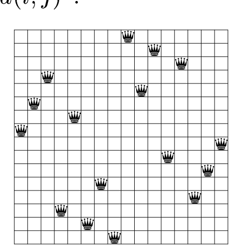
232. [*20*] The costs supplied to Algorithm X$^\$$ must be nonnegative integers; but $d(i,j)$ in the 16 queens problem of Fig. 74 is never an integer. Is it OK to use $\$\lfloor d(i,j)\rfloor$ instead of $\$d(i,j)$ for the cost of placing a queen in cell (i,j)?

233. [*20*] Minimize and maximize the *product* of the 16 queen distances, not the sum.

234. [*M20*] What is the minimum-cost placement of n nonattacking queens when the cost of a queen in cell (i,j) is $\$d(i,j)^2$, the *square* of its distance from the center?

▸ **235.** [*21*] Solve the problem of Fig. 74 using the (integer) costs $\$d(i,j)^4$.

▸ **236.** [*M41*] When the cost of a queen in cell (i,j) is $\$d(i,j)^N$, for larger and larger values of N, the minimum-cost solutions to the n queens problem eventually converge to a fixed pattern. And those "ultimate" solutions turn out to be quite attractive—indeed, this family of solutions is arguably the most beautiful of all! For example, the case $n=16$, illustrated here, can actually be discovered by hand, with a few moments of concentrated thought. Notice that it is doubly symmetric and nicely "rounded."



Discover such optimum placements for as many n as you can (*not* by hand).

▸ **237.** [*M21*] True or false: Two solutions to the text's prime square problem cannot have the same product unless they are transposes of each other.

rounds
pentomino problem
Langford pairing
reverse
reflected string
lexicographically
SCRABBLE®
4-letter
5-letter
word rectangles
English
min-cost exact cover
16 queens problem
doubly symmetric
prime square problem

238. [*24*] Find $3\times n$ arrays filled with distinct 3-digit and n-digit primes, for $3 \le n \le 7$, having the minimum and maximum possible product.

▸ **239.** [*M27*] Given a family $\{S_1, \ldots, S_m\}$ of subsets of $\{1, \ldots, n\}$, together with positive weights $(w_1, \ldots, w_m)$, the *optimum set cover* problem asks for a minimum-weight way to cover $\{1, \ldots, n\}$ with a union of S_j's. Formulate this problem as an *optimum exact cover* problem, suitable for solution by Algorithm X$^\$$. *Hint:* Maximize the weight of all sets that do *not* participate in the cover.

240. [*16*] What usable 6-state options include `MT` and `TX` in the USA-partition problem?

241. [*21*] Does preprocessing by Algorithm P remove the useless option (114)?

▸ **242.** [*M23*] Extend the algorithm of exercise 7.2.2–78 so that it visits only subgraphs that don't cut off connected regions whose size isn't a sum of integers in $[L \,..\, U)$.

243. [*M20*] Assume that every item i of an XCC problem has been given a *weight* w_i, and that every solution to the problem involves exactly d options. If the cost of every option is $\$(x^2)$, where x is the sum of the option's weights, prove that every minimum-cost solution also minimizes $\sum_{k=1}^{d}(x_k - r)^2$, for any given real number r.

244. [*M21*] The induced subgraphs $G \mid U$ of a graph or digraph G have an *interior cost*, defined to be the number of ordered pairs of vertices in U that are *not* adjacent. For example, the interior cost of option (114) is 20, which is the maximum possible for six connected vertices of an undirected graph.

Consider any exact cover problem whose items are the vertices of G, and whose options all contain exactly t items. True or false: A solution that minimizes the sum of the interior costs also minimizes the sum of the exterior costs, as defined in the text.

245. [*23*] Augment the USA graph by adding a 49th vertex, `DC`, adjacent to `MD` and `VA`. Partition this graph into seven connected components, (a) all of size 7, removing as few edges as possible; (b) of any size, equalizing their populations as much as possible.

246. [*22*] The left-hand graph partition in (116) has a bizarre component that connects `AZ` with `ND` and `OK`, without going through `NM`, `CO`, or `UT`. Would we obtain more reasonable-looking solutions if we kept the same options, but minimized the exterior costs instead of the squared populations? (That is, on the left we'd consider the 34,111 options with population in $[37 \,..\, 39]$ million, plus two options that include New York, New England, and possibly New Jersey. The options of the right-hand example would again be the connected subsets with population in $[50.5 \,..\, 51.5]$ million.)

Consider also minimizing the *interior* costs, as defined in exercise 244.

247. [*23*] Specify step C1$^\$$, which takes the place of step C1 when Algorithm C is extended to Algorithm C$^\$$. Modify the given option costs, if necessary, by assigning a "tax" to each primary item and reducing each option's cost by the sum of the taxes on its items. These new costs should be nonnegative; and every primary item should belong to at least one option whose cost is now zero. Be sure to obey condition (118).

248. [*22*] Let $\vartheta = T - C_l$ in step C3$^\$$, where T is the current cutoff threshold and C_l is the cost of the current partial solution on levels less than l. Explain how to choose an active item i that probably belongs to the fewest options of cost $< \vartheta$. Instead of taking the time to make a complete search, assume conservatively that there are `LEN(`i`)` such items, after verifying that item i has at least L of them, where L is a parameter.

249. [*21*] A set of dk costs, with $0 \le c_1 \le c_2 \le \cdots \le c_{dk}$, is said to be bad if $c_k + c_{2k} + \cdots + c_{dk} \ge \theta$. Design an "online algorithm" that identifies a bad set as quickly as possible, when the costs are learned one by one in arbitrary order.

word rectangles
set cover
exact cover
preprocessing
connected regions
weight
induced subgraphs
interior cost
exterior costs
USA graph
New York
New England
tax
cutoff threshold
online algorithm
every kth cost

For example, suppose $d = 6$, $k = 2$, and $\theta = 16$. If costs appear in the order $(3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8)$, your algorithm should stop after seeing the 2.

250. [*21*] Users of Algorithm C$^\$$ are allowed to supply hints that speed up the computation, by specifying (i) a set Z of characters, such that every element of Z is the first character of exactly one primary item in every option; also (ii) a number $z > 0$, meaning that every option contains exactly z primary items whose names *don't* begin with a character in Z. (For example, $Z = \{p, r, c, b\}$ in the sudoku options (30); $z = 1$ in options (110). In the options (16) for Langford pairs, we could change the name of each numeric item i to '!i', then let $Z = \{!\}$ and $z = 2$.) Explain how to use these hints to supply an early-cutoff test at the beginning of step C3$^\$$, as explained in the text.

251. [*18*] If a given problem is solvable, when does Algorithm Z first discover that fact?

▸ **252.** [*20*] Algorithm Z produces the ZDD (120) from the options (121) if step Z3 simply chooses the leftmost item i_1 = `ROOT(0)` instead of using the MRV heuristic. What ZDD would have been obtained if the method of exercise 9 had been used instead?

▸ **253.** [*21*] Extend Algorithm Z so that it reports the total number of solutions.

▸ **254.** [*28*] The signature σ computed by Algorithm Z in step Z2 is supposed to characterize the current subproblem completely. It contains one bit for each primary item, indicating whether or not that item still needs to be covered.

a) Explain why one bit isn't sufficient for secondary items with colors.

b) Suggest a good way to implement the computation of σ.

c) Algorithm C uses the operations hide$'$ and unhide$'$ in (50)–(57), in order to avoid unnecessary accesses to memory in nodes for secondary items. Explain why Algorithm Z does not want to use those optimizations. *Hint:* Algorithm Z needs to know whether the option list for a secondary item is empty.

d) When the list for item i is purified, its options of the wrong color are removed from other lists. But they *remain* on list i, in order to be unpurified later. How then can Algorithm Z know when list i is no longer relevant to the current subproblem?

255. [*HM29*] Express the exact number of updates made by Algorithm Z when it finds the perfect matchings of K_N, as well as the exact number of ZDD nodes produced, in terms of Fibonacci numbers. *Hint:* See exercise 193.

▸ **256.** [*M23*] What is the behavior of Algorithm Z when it is asked to find all perfect matchings of the "bizarre" graph (89)?

▸ **257.** [*21*] How does Algorithm Z do on the "extreme" exact cover problem, with n items and $2^n - 1$ options? (See the discussion preceding (82).)

a) What signatures are formed in step Z2?

b) Draw the schematic ZDD, analogous to (123), when $n = 4$.

258. [*HM21*] How many updates does Algorithm Z perform, in that extreme problem?

259. [*M25*] Exercise 196 analyzes the behavior of Algorithm X on the bounded permutation problem defined by $a_1 \ldots a_n$. Show that Algorithm Z is considerably faster, by determining the number of memos, ZDD nodes, and updates when $a_1 a_2 \ldots a_{n-1} a_n$ is (a) $n\,n \ldots n\,n$ [with $n!$ solutions]; (b) $2\,3 \ldots n\,n$ [with 2^{n-1} solutions]. Assume that the items are $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n$, in that order.

260. [*M21*] Exercises 14 and 201 are bipartite matching problems related to choosing seats at a circular table. Test Algorithm Z empirically on those problems, and show that it solves the latter in linear time (despite exponentially many solutions).

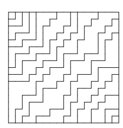
pi as source
sudoku
Langford pairs
ZDD
MRV heuristic
hide$'$
unhide$'$
purified
analysis of algorithms
perfect matchings
matchings
Fibonacci numbers
extreme
bounded permutation problem
bipartite matching
bipartite matching problems
seats at a circular table
circular table

▸ **261.** [*23*] Let G be a directed acyclic graph, with source vertices S and sink vertices T.

a) Use Algorithm C (or Z) to find all sets of m vertex-disjoint paths from S to T.

b) Also find all such sets of paths from s_k to t_k for $1 \le k \le m$, given s_k and t_k.

c) Apply (a) to find all sets of $n-1$ disjoint paths that enter an $n \times n$ square at the north or east edge, proceed by south and/or west steps, and exit at the south or west edge, avoiding the corners. (A random 16×16 example is shown.)

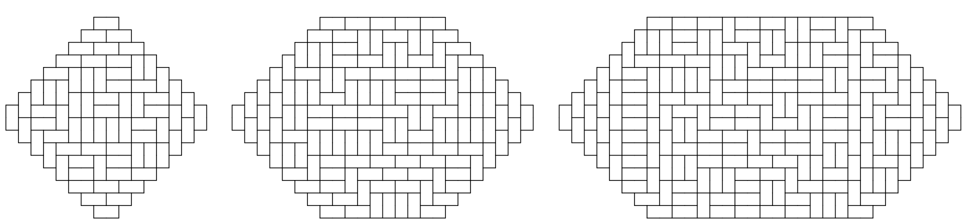


1	2	3	4	5	6	7	8
3	4	1	2	0	8	5	6
1	2	3	8	5	6	0	7
4	8	5	2	3	7	0	0
5	1	4	8	6	2	3	7
0	0	0	1	4	7	6	2
0	5	8	7	6	1	4	3
8	7	6	5	4	3	2	1

d) Apply (b) to find all vertex-disjoint, downward paths of eight knights that start on the top row of a chessboard and end on the bottom row in reverse order.

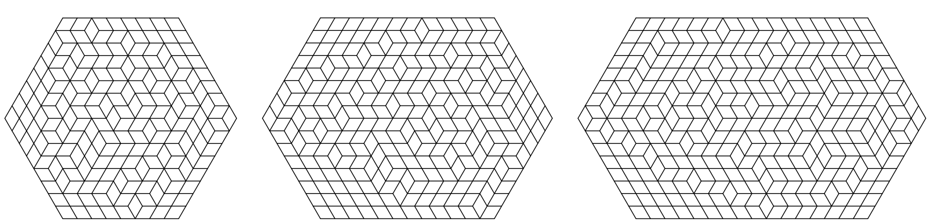
▸ **262.** [*M23*] One of the advantages of Algorithm Z is that a ZDD allows us to generate *uniformly random solutions.* (See the remarks following 7.1.4–(13).)

a) Determine the number of ZDD nodes output by Algorithm Z for the set of all domino tilings of S_n, where S_n is the shape obtained after right triangles of side 7 have been removed from each corner of a $16 \times n$ rectangle:



How many tilings are possible for S_{16} (the Aztec diamond of order 8)? For S_{32}?

b) Similarly, determine the ZDD size for the family of all diamond tilings of T_n — the grid *simplex*$(n+16, n+8, 16, n+8, 0, 0, 0)$, a hexagon of sides $(8, 8, n, 8, 8, n)$:



263. [*24*] Compare the time and space requirements of Algorithms C and Z when they are applied to (a) the 16 queens problem; (b) pentominoes, as in exercises 271 and 274; (c) MacMahon's triangle problem, as in exercise 126; (d) the generalized de Bruijn sequences of exercise 95; (e) the "right word stair" problem of exercise 90; (f) the 6×6 "word search" problem of exercise 105; (g) the kakuro problem in exercise 431.

264. [*M21*] Suppose step Z3 always chooses the first active item $i = \texttt{RLINK(0)}$, instead of using the MRV heuristic, unless some other active item has $\texttt{LEN}(i) = 0$. Prove that Algorithm Z will then output an *ordered* ZDD.

▸ **265.** [*22*] Prove that Algorithm Z will never produce identical ZDD nodes $(\bar{o}_i ?\, l_i : h_i) = (\bar{o}_j ?\, l_j : h_j)$ for $i \ne j$, if all items are primary. But secondary items *can* cause duplicates.

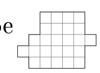
vertex-disjoint
parallominoes
knights
random solutions of XCC problems
domino tilings
Aztec diamond
bipartite matching problems
diamond tilings
simplex
16 queens problem
pentominoes
MacMahon's triangle problem
de Bruijn sequences
word stair
word search
kakuro
MRV heuristic
ordered ZDD
reduced ZDD

EXERCISES — Second Set

Thousands of fascinating recreational problems have been based on polyominoes and their polyform cousins (the polycubes, polyiamonds, polyhexes, polysticks, . . .). The following exercises explore "the cream of the crop" of such classic puzzles, as well as a few gems that were not discovered until recently.

In most cases the point of the exercise is to find a good way to discover all solutions, usually by setting up an appropriate exact cover problem that can be solved without taking an enormous amount of time.

▶ **266.** [*25*] Sketch the design of a utility program that will create sets of options by which an exact cover solver will fill a given shape with a given set of polyominoes.

267. [*18*] Using Conway's piece names, pack five pentominoes into the shape  so that they spell a common English word when read from left to right.

▶ **268.** [*21*] There are 1010 ways to pack the twelve pentominoes into a 5×12 box, not counting reflections. What's a good way to find them all, using Algorithm X?

269. [*21*] How many of those 1010 packings decompose into $5 \times k$ and $5 \times (12-k)$?

270. [*21*] In how many ways can the eleven nonstraight pentominoes be packed into a 5×11 box, not counting reflections as different? (Reduce symmetry cleverly.)

271. [*20*] There are 2339 ways to pack the twelve pentominoes into a 6×10 box, not counting reflections. What's a good way to find them all, using Algorithm X?

272. [*23*] Continuing exercise 271, explain how to find special kinds of packings:

a) Those that decompose into $6 \times k$ and $6 \times (10-k)$.
b) Those that have all twelve pentominoes touching the outer boundary.
c) Those with all pentominoes touching that boundary *except* for V, which doesn't.
d) Same as (c), with each of the other eleven pentominoes in place of V.
e) Those with the *minimum* number of pentominoes touching the outer boundary.
f) Those that are characterized by Arthur C. Clarke's description, as quoted below.

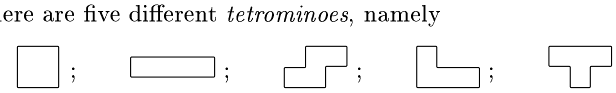
That is, the X pentomino should touch only the F (aka R), the N (aka S), the U, and the V — no others.

Very gently, he replaced the titanite cross
in its setting between the F, N, U, and V pentominoes.
— ARTHUR C. CLARKE, *Imperial Earth* (1976)

273. [*25*] All twelve pentominoes fit into a 3×20 box only in two ways, shown in (36).

a) How many ways are there to fit *eleven* of them into that box?
b) In how many solutions to (a) are the five holes *nonadjacent*, kingwise?
c) In how many ways can eleven pentominoes be packed into a 3×19 box?

274. [*21*] There are five different *tetrominoes*, namely



square straight skew ell tee

In how many essentially different ways can each of them be packed into an 8×8 square together with the twelve pentominoes?

275. [*21*] If an 8×8 checkerboard is cut up into thirteen pieces, representing the twelve pentominoes together with one of the tetrominoes, some of the pentominoes will have more black cells than white. Is it possible to do this in such a way that U, V, W, X, Y, Z have a black majority while the others do not?

276. [*18*] Design a nice, simple tiling pattern that's based on the five tetrominoes.

polyform
Conway
five-letter words
pentominoes
nonstraight
symmetry
pentominoes
pentominoes, names of
CLARKE
tetrominoes
tetrominoes, names of
pattern
tetrominoes
beauty contest

277. [*25*] How many of the 6×10 pentomino packings are *strongly three-colorable*, in the sense that each individual piece could be colored red, white, or blue in such a way that no pentominoes of the same color touch each other — not even at corner points?

▸ **278.** [*32*] Use the catalog of bipairs in exercise 217 to reduce the number of 6×10 pentomino packings, listing strong solutions only (see Theorem S). How much time is saved?

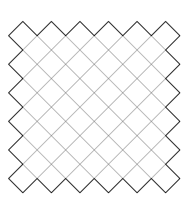
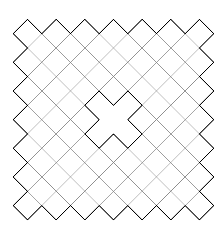
279. [*40*] (H. D. Benjamin, 1948.) Show that the twelve pentominoes can be wrapped around a cube of size $\sqrt{10} \times \sqrt{10} \times \sqrt{10}$. For example, here are front and back views of such a cube, made from twelve colorful fabrics by the author's wife in 1993:

(Photos by
Héctor García)

What is the best way to do this, minimizing undesirable distortions at the corners?

▸ **280.** [*M26*] Arrange the twelve pentominoes into a Möbius strip of width 4. The pattern should be "faultfree": Every straight line must intersect some piece.

▸ **281.** [*20*] The white cells of a $(2n+1 \times (2n+1)$ checkerboard, with black corners, form an interesting graph called the *Aztec diamond* of order n; and the black cells form the Aztec diamond of order $n+1/2$. For example, the diamonds of orders $11/2$ and $13/2$ are

(i)  and (ii) ,

where (ii) has a "hole" of order $3/2$. Thus (i) has 61 cells, and (ii) has 80.

a) Find all ways to pack (i) with the twelve pentominoes and one monomino.

b) Find all ways to pack (ii) with the 12 pentominoes and 5 tetrominoes.

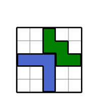
Speed up the process by not producing solutions that are symmetric to each other.

▸ **282.** [*22*] (Craig S. Kaplan.) A polyomino can sometimes be surrounded by non-overlapping copies of itself that form a *fence*: Every cell that touches the polyomino — even at a corner — is part of the fence; conversely, every piece of the fence touches the inner polyomino. Furthermore, the pieces must not enclose any unoccupied "holes."

Find the (a) smallest and (b) largest fences for each of the twelve pentominoes. (Some of these patterns are unique, and quite pretty.)

283. [*22*] Solve exercise 282 for fences that satisfy the *tatami* condition of exercise 7.1.4–215: No four edges of the tiles should come together at any "crossroads."

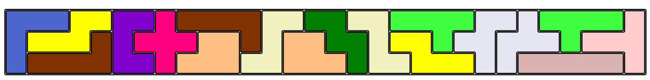
▸ **284.** [*27*] Solomon Golomb discovered in 1965 that there's only one placement of two pentominoes in a 5×5 square that blocks the placement of all the others.



Place (a) $\{I, P, U, V\}$ and (b) $\{F, P, T, U\}$ into a 7×7 square in such a way that none of the other eight will fit in the remaining spaces.

three-colorable
graph coloring
bipairs
strong solutions
Benjamin
cube, wrapped
Knuth, Jill
Garcia, Hector
Möbius strip
faultfree
checkerboard
Aztec diamond
symmetric
Kaplan
fence
holes
tatami
crossroads
Golomb

285. [*21*] (T. H. O'Beirne, 1961.) The *one-sided pentominoes* are the eighteen distinct 5-cell pieces that can arise if we aren't allowed to flip pieces over:



Notice that there now are two versions of P, Q, R, S, Y, and Z.

In how many ways can all eighteen of them be packed into rectangles?

286. [*21*] If you want to pack the twelve pentominoes into a 6×10 box *without* turning any pieces over, 2^6 different problems arise, depending on the orientations of the one-sided pieces. Which of those 64 problems has (a) the fewest (b) the most solutions?

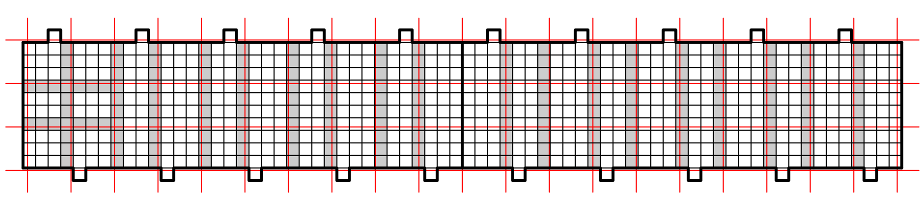
▸ **287.** [*23*] A princess asks you to pack an $m\times n$ box with pentominoes, rewarding you with $\$c\cdot(ni+j)$ if you've covered cell (i,j) with piece c, where $c=(1,2,\ldots,12)$ for pieces $(\mathrm{O},\mathrm{P},\ldots,\mathrm{Z})$. (The most valuable packing will be "closest to alphabetic order.")

Use Algorithm $\mathrm{X}^\$$ to maximize your bounty when packing boxes of sizes 4×15, 5×12, 6×10, 10×6, 12×5, and 15×4. Consider also the princess's circle-shaped subset of a 9×9 box, where you are to cover only the 60 cells whose distances from the center are between 1 and $\sqrt{18}$. How do the running times of Algorithm $\mathrm{X}^\$$ compare to the amounts of time that Algorithm X would take to find *all* solutions?

288. [*21*] Similarly, pack the one-sided pentominoes optimally into 9×10 and 10×9.

▸ **289.** [*29*] (*Pentominoes of pentominoes.*) Magnify the 3×20 pentomino packing (36) by replacing each of its unit cells by (a) 3×4 rectangles; (b) 4×3 rectangles. In how many ways can the resulting 720-cell shape be packed with twelve complete sets of twelve pentominoes, using one set for each of the original pentomino regions?

(c) Also partition the 720-cell shape below into 3×20 approximately *square* 12-cell regions, by assigning each gray cell to an adjacent region. (This shape has been superimposed on a grid whose $\sqrt{12}\times\sqrt{12}$ regions are *perfectly* square.) Minimize the total perimeter of the 60 resulting regions, and try for a pleasantly symmetrical solution.



Use your partition to present a scaled-up version of (36), again with 12 complete sets.

290. [*21*] When tetrominoes are both checkered and one-sided (see exercises 275 and 285), ten possible pieces arise. In how many ways can all ten of them fill a rectangle?

291. [*24*] (*A puzzle a day.*) Using the two trominoes, the five tetrominoes, and three of the pentominoes, one can cover up 11 of the 12 "months" and 30 of the 31 "days" in the following pair of diagrams, thereby revealing the current month and day:

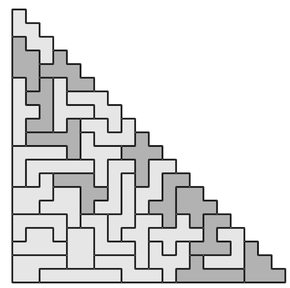
I	II	III	IV
V	VI	VII	VIII
IX	X	XI	XII

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Which of the $\binom{12}{3}$ sets of three pentominoes always allow this to be done?

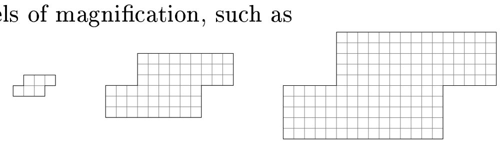
O'Beirne
one-sided pentominoes
flip pieces over
princess
max-cost solution
circle, discrete
one-sided pentominoes
polyforms of polyforms
tetrominoes
checkered
one-sided
trominoes
daily puzzle
tetrominoes

292. [*20*] There are 35 *hexominoes*, first enumerated in 1934 by the master puzzlist H. D. Benjamin. At Christmastime that year, he offered ten shillings to the first person who could pack them into a 14×15 rectangle — although he wasn't sure whether or not it could be done. The prize was won by F. Kadner, but not as expected: Kadner proved that the hexominoes actually *can't* be packed into *any* rectangle! Nevertheless, Benjamin continued to play with them, eventually discovering that they fit nicely into the triangle shown here.



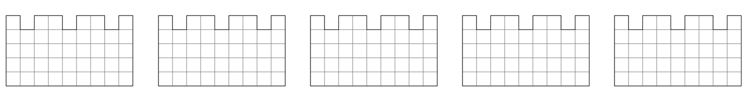
Prove Kadner's theorem. *Hint:* See exercise 275.

293. [*24*] (Frans Hansson, 1947.) The fact that $35 = 1^2 + 3^2 + 5^2$ suggests that we might be able to pack the hexominoes into three boxes that represent a *single* hexomino shape at three levels of magnification, such as



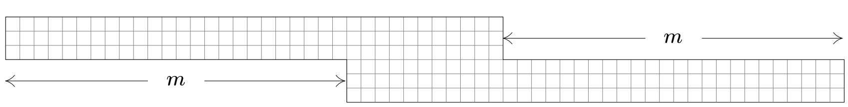
For which hexominoes can this be done?

▸ **294.** [*30*] Show that the 35 hexominoes can be packed into five "castles":

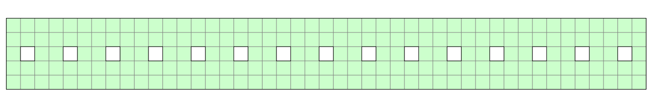


In how many ways can this be done?

295. [*41*] For which values of m can the hexominoes be packed into a box like this?



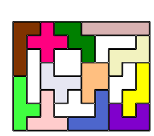
296. [*41*] Perhaps the nicest hexomino packing uses a 5×45 rectangle with 15 holes



proposed by W. Stead in 1954. In how many ways can the 35 hexominoes fill it?

297. [*24*] (P. Torbijn, 1989.) Can the 35 hexominoes be packed into six 6×6 squares?

▸ **298.** [*22*] In how many ways can the twelve pentominoes be placed into an 8×10 rectangle, leaving holes in the shapes of the five tetrominoes? (The holes should not touch the boundary, nor should they touch each other, even at corners; one example is shown at the right.) Explain how to encode this puzzle as an XCC problem.



299. [*39*] If possible, solve the analog of exercise 298 for the case of 35 *hexominoes* in a 5×54 rectangle, leaving holes in the shapes of the twelve *pentominoes*.

▸ **300.** [*23*] In how many ways can the twelve pentominoes be arranged in a 10×10 square, filling exactly six of the cells in every row and exactly six of the cells in every column, if we also require that (a) the cells on both diagonals are completely empty? (b) the cells on both diagonals are completely filled? (c) the design is really interesting?

hexominoes
Benjamin
Kadner
Hansson
magnification
triplication
castles
Stead
Torbijn
666
pentominoes
tetrominoes
color controls
hexominoes

301. [*25*] Here's one way to place the twelve pentominoes into a 5×5 square, covering the cells of rows (1, 2, 3, 4, 5) exactly (2, 3, 2, 3, 2) times:

QY	SX	ST	ST	RT
QXY	XYZ	RXZ	RST	RSV
QY	XZ	UW	RT	UV
QYZ	QWZ	UVW	PUV	PUV
OW	OW	OP	OP	OP

a) How many such placements are possible?

b) Suppose we've placed `O` first, `P` next, `Q` next, ..., `Z` last, when making the arrangement above. Then `Z` is above `W` is above `V` is above `U` is above `P` is above `O`; hence the pentominoes have been stacked up on six levels. Show that a different order of placement would require only four levels.

c) Find a solution to (a) that needs only three levels.

d) Find a solution to (a) that can't be achieved with only four levels.

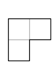
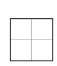
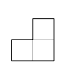
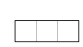
302. [*26*] Say that an n-omino is "small" if it fits in a $(\sqrt{n}+1) \times (\sqrt{n}+1)$ box, and "slim" if it contains no 2×2 tetromino. Thus, for example, pentominoes O, Q, S, Y aren't small; P isn't slim.

a) How many nonominoes are both small and slim?

b) Fit nine different small-and-slim nonominoes into a 9×9 box.

c) Use a solution to (b) as the basis of a jigsaw sudoku puzzle with a unique solution. The clues of your puzzle should be the initial digits of π.

▸ **303.** [*HM35*] A *parallelogram polyomino*, or "parallomino" for short, is a polyomino whose boundary consists of two paths that each travel only north and/or east. (Equivalently, it is a "staircase polygon," "skew Young tableau," or a "skew Ferrers board," the difference between the diagrams of two tableaux or partitions; see Sections 5.1.4 and 7.2.1.4.) For example, there are five parallominoes whose boundary paths have length 4:

NNNE ENNN ; NNEE ENEN ; NNEE EENN ; NENE EENN ; NEEE EEEN .

a) Find a one-to-one correspondence that maps the set of ordered trees with m leaves and n nodes into the set of parallominoes with width m and height $n-m$. The area of each parallomino should be the path length of its corresponding tree.

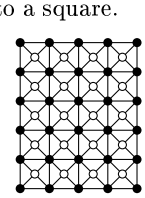
b) Study the generating function $G(w,x,y) = \sum_{\text{parallominoes}} w^{\text{area}} x^{\text{width}} y^{\text{height}}$.

c) Prove that the parallominoes whose width-plus-height is n have total area 4^{n-2}.

d) Part (c) suggests that we might be able to pack all of those parallominoes into a $2^{n-2} \times 2^{n-2}$ square, *without* rotating them or flipping them over. Such a packing is clearly impossible when $n = 3$ or $n = 4$; but is it possible when $n = 5$ or $n = 6$?

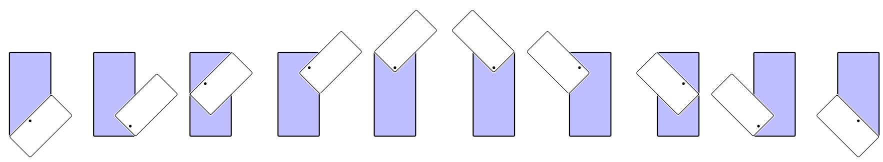
304. [*M25*] Prove that it's NP-complete to decide whether or not n given polyominoes, each of which fits in a $\Theta(\log n) \times \Theta(\log n)$ square, can be exactly packed into a square.

305. [*25*] When a square grid is scaled by $1/\sqrt{2}$ and rotated 45°, we can place half of its vertices on top of the original ones; the other "odd-parity" vertices then correspond to the centers of the original square cells.



Using this idea we can glue a small domino of area 1 over portions of an ordinary domino of area 2, thereby obtaining ten distinct two-layer pieces called the *windmill dominoes*:

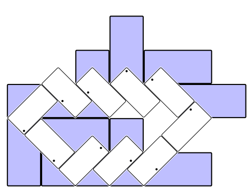
small polyominoes
slim polyominoes
nonominoes
jigsaw sudoku
pi as src
parallelogram polyomino
parallomino
staircase polygon
skew Young tableau
Young tableaux
skew Ferrers board
Ferrers diagrams
tableaux
partitions
trees
path length
generating function
NP-complete
polyominoes
grid, rotated
parity
two-layer pieces
windmill dominoes



a) Arrange four windmill dominoes so that the upper layer resembles a windmill.
b) Place all ten windmill dominoes inside a 4×5 box, without overlapping.
c) Similarly, pack them all into a 2×10 box.
d) Place them so that the upper layer fills a $(4/\sqrt{2}) \times (5/\sqrt{2})$ rectangle.
e) Similarly, fit the upper layer into a $(2/\sqrt{2}) \times (10/\sqrt{2})$ rectangle.

In each case (a)–(e), use Algorithm X to count the total number of possible placements. Also look at the output and choose arrangements that are especially pleasing.

▸ **306.** [*30*] (S. Grabarchuk, 1996.) In how many ways can the ten windmill dominoes be arranged so that the 20 large squares define a snake-in-the-box cycle, in the sense of exercise 172(b), and so do the 20 small squares? (For example, arrangements like



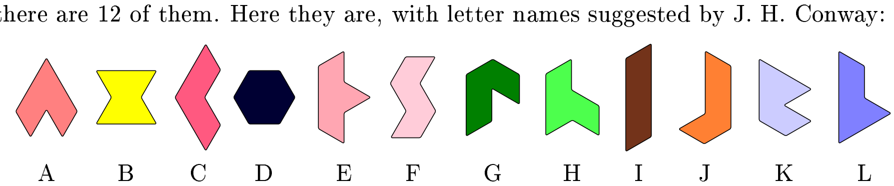
satisfy one snake-in-the-box condition but not the other.)

307. [*M21*] If a $(3m+1)\times(3n+2)$ box is packed with $3mn+2m+n$ straight trominoes and one domino, where must the domino be placed?

308. [*22*] A *polyiamond* is a connected set of triangles in a triangular grid, inspired by the diamond —just as a polyomino is a connected set of squares in a square grid, inspired by the domino. Thus we can speak of moniamonds, diamonds, triamonds, etc.

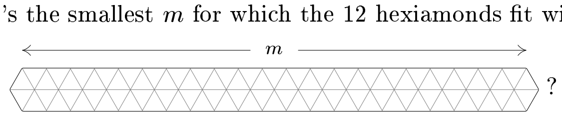
a) Extend exercise 266 to the triangular grid, using the coordinate system of exercise 124. How many base placements do each of the tetriamonds have?
b) Find all ways to pack the pentiamonds into a convex polygon (see exercise 143).
c) Similarly, find all such ways to pack the *one-sided* pentiamonds.

309. [*24*] The *hexiamonds* are particularly appealing, because—like pentominoes—there are 12 of them. Here they are, with letter names suggested by J. H. Conway:



a) How many base placements do *they* have?
b) In how many ways can *they* be packed into convex polygons, as in exercise 308?

310. [*23*] What's the smallest m for which the 12 hexiamonds fit without overlap in

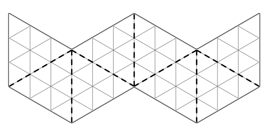


Find a pleasant way to place them inside of that smallest box.

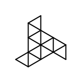
Grabarchuk
snake-in-the-box cycle
coil-in-the-box
straight trominoes
domino
polyiamond
moniamonds
diamonds
triamonds
lozenges, see diamonds
tetriamonds
pentiamonds
one-sided
hexiamonds
Conway

▶ **311.** [*30*] (*Hexiamond wallpaper.*) Place the twelve hexiamonds into a region of N triangles, so that (i) shifted copies of the region fill the plane; (ii) the hexiamonds of the resulting infinite pattern do not touch each other, even at vertices; (iii) N is minimum.

312. [*22*] The following shape can be folded, to cover the faces of an octahedron:



Fill it with hexiamonds so that they cross the folded edges as little as possible.

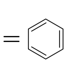
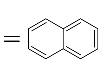
▶ **313.** [*29*] (*Hexiamonds of hexiamonds.*) A "whirl," shown here, is an interesting dodeciamond that tiles the plane in a remarkably beautiful way. 

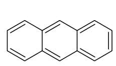
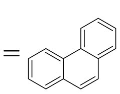
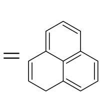
If each triangle '△' of a hexiamond is replaced by a whirl, in how many ways can the resulting 72-triangle shape be packed with the full set of hexiamonds? (Exercise 289 discusses the analogous problem for pentominoes.)

Consider also using "flipped whirls," the left-right reflections of each whirl.

▶ **314.** [*28*] (G. Sicherman, 2008.) Can the four pentiamonds be used to make two 10-iamonds of the same shape? Formulate this question as an exact cover problem.

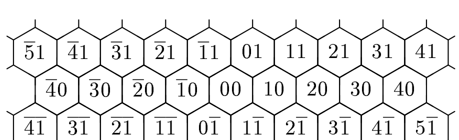
315. [*20*] A *polyhex* is a connected shape formed by pasting hexagons together at their edges, just as polyominoes are made from squares and polyiamonds are made from triangles. For example, there's one monohex and one dihex, but there are three trihexes. Chemists have studied polyhexes since the 19th century, and named the small ones:

benzene = ; naphthalene = ;

anthracene = , phenanthrene = , phenalene = ; etc.

(Groups of six carbon atoms can bond together in a nearly planar fashion, forming long chains of hexagons, with hydrogen atoms attached. But the correspondence between polyhexes and polycyclic aromatic hydrocarbons is not exact.)

Represent the individual hexagons of an infinite grid by Cartesian-like coordinates



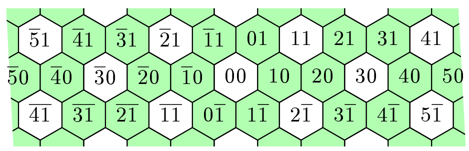
where $\bar{1} = -1$, $\bar{2} = -2$, etc. Extending exercises 266 and 308(a), explain how to find the base placements of a polyhex, given the coordinates of its cells when placed on this grid.

316. [*20*] Show that the complete set of trihexes and tetrahexes can be packed nicely into a rosette that consists of 37 concentric hexagons. In how many ways can it be done?

317. [*22*] (*Tetrahexes of tetrahexes.*) If we replace each hexagon of a tetrahex by a rosette of seven hexagons, we get a 28-hex. In how many ways can that scaled-up shape be packed with the seven distinct tetrahexes? (See exercises 289 and 313.)

wallpaper
octahedron
wrapping a polyhedron
polyforms of polyforms
whirl
dodeciamond
Sicherman
pentiamonds
congruent pairs
polyhex
hexagons
Chemists
anthracene
phenanthrene
phenalene
aromatic hydrocarbons
grid, hexagonal
hexagons, coordinates for
coordinates
base placements
rosette
tetrahexes
polyforms of polyforms
rosette

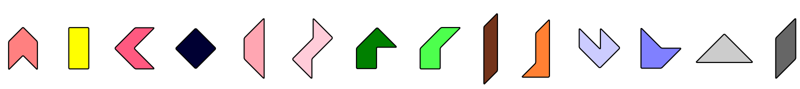
▸ **318.** [*20*] Let's say that the *T-grid* is the set of all hexagons xy with $x \not\equiv y$ (modulo 3):



Show that there's a one-to-one correspondence between the hexagons of the T-grid and the triangles of the infinite triangular grid, in which every polyiamond corresponds to a polyhex. (Therefore the study of polyiamonds is a special case of the study of polyhexes!) *Hint:* Exercise 124 discusses a coordinate system for representing triangles.

319. [*21*] After polyominoes, polyiamonds, and polyhexes, the next most popular polyforms are the *polyaboloes*, originally proposed by S. J. Collins in 1961. These are the shapes obtainable by attaching isosceles right triangles at their edges; for example, there are three diaboloes: $\{TT, TT, Q\}$. Notice that a new idea arises here, because the square diabolo arises from two monoboloes in two ways, TT and TT, yet it counts as only one shape. Also, any n-abolo corresponds to a $2n$-abolo, when scaled up by $\sqrt{2}$.

The 14 tetraboloes can be named by using rough resemblances to hexiamonds:

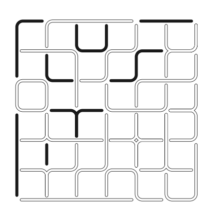


Show that the study of polyaboloes can be reduced to the study of (slightly generalized) polyominoes, just as exercise 318 reduces polyiamonds to polyhexes.

▸ **320.** [*M28*] Explain how to enumerate all of the N-aboloes that are *convex*. How many of the convex 56-aboloes can be packed by the fourteen tetraboloes?

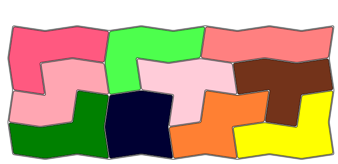
321. [*24*] (T. H. O'Beirne, 1962.) In how many ways can a square be formed from the eight *one-sided tetraboloes* and their mirror images?

322. [*23*] The *polysticks* provide us with another intriguing family of shapes that can be combined in interesting ways. An "n-stick" is formed by joining n horizontal and/or vertical unit line segments together at grid points. For example, there are two disticks, and five tristicks; they're shown here in black, accompanied by the monostick, and surrounded by the sixteen tetrasticks in white.



Polysticks introduce yet another twist into polyform puzzles, because we must not allow different pieces to cross each other when we pack them into a container. Extend exercise 266 to polysticks, so that Algorithm X can deal with them conveniently.

323. [*M25*] We've now seen polyominoes, polyiamonds, polyhexes, ..., polysticks, each of which have contributed new insights; and many other families of "polyforms" have in fact been studied. Let's close our survey with *polyskews*, a relatively new family that seems worthy of further exploration. Polyskews are the shapes that arise when we join squares alternately with rhombuses, in checkerboard fashion. For example, here are the ten tetraskews:



There are two monoskews, one diskew, and five triskews.

a) Explain how to draw such skewed pixel diagrams.

b) Show that polyskews, like polyaboloes, can be reduced to polyominoes.

c) In how many ways do the tetraskews make a skewed rectangle?

▸ **324.** [*20*] Extend exercise 266 to three dimensions. How many base placements do each of the seven Soma pieces have?

325. [*27*] The *Somap* is the graph whose vertices are the 240 distinct solutions to the Soma cube problem, with $u — v$ if and only if u can be obtained from an equivalent of v by changing the positions of at most three pieces. The *strong* Somap is similar, but it has $u — v$ only when a change of just *two* pieces gets from one to the other.

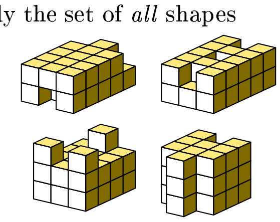
a) What are the degree sequences of the Somap graphs?

b) How many connected components do they have? How many bicomponents?

▸ **326.** [*M25*] Use factorization to prove that Fig. 75's W-wall cannot be built.

327. [*24*] Figure 75(a) shows some of the many "low-rise" (2-level) shapes that can be built from the seven Soma pieces. Which of them is hardest (has the fewest solutions)? Which is easiest? Answer those questions also for the 3-level prism shapes in Fig. 75(b).

▸ **328.** [*M23*] Generalizing the first four examples of Fig. 75, study the set of *all* shapes obtainable by deleting three cubies from a $3 \times 5 \times 2$ box. (Two examples are shown here.) How many essentially different shapes are possible? Which shape is easiest? Which shape is hardest?

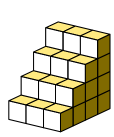


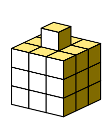
329. [*22*] Similarly, consider (a) all shapes that consist of a $3 \times 4 \times 3$ box with just three cubies in the top level; (b) all 3-level prisms that fit into a $3 \times 4 \times 3$ box.

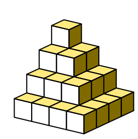
330. [*25*] How many of the 1285 *nonominoes* define a prism that can be realized by the Soma pieces? Do any of those packing problems have a unique solution?

331. [*M40*] Make empirical tests of Piet Hein's belief that the number of shapes achievable with seven Soma pieces is approximately the number of 27-cubie polycubes.

332. [*20*] (B. L. Schwartz, 1969.) Show that the Soma pieces can make shapes that appear to have more than 27 cubies, because of holes hidden inside or at the bottom:

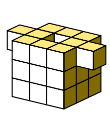

staircase

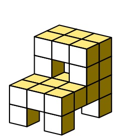

penthouse

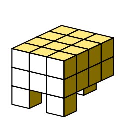

pyramid

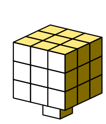
In how many ways can these three shapes be constructed?

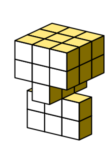
333. [*22*] Show that the seven Soma pieces can also make structures such as


casserole


cot


vulture


mushroom


cantilever

,

which are "self-supporting" via gravity. (You may need to place a small book on top.)

skewed rectangle
base placements
Somap
Soma cube
degree sequences
connected components
bicomponents
factorization
W-wall
Soma pieces
nonominoes
Hein
Schwartz
self-supporting
gravity

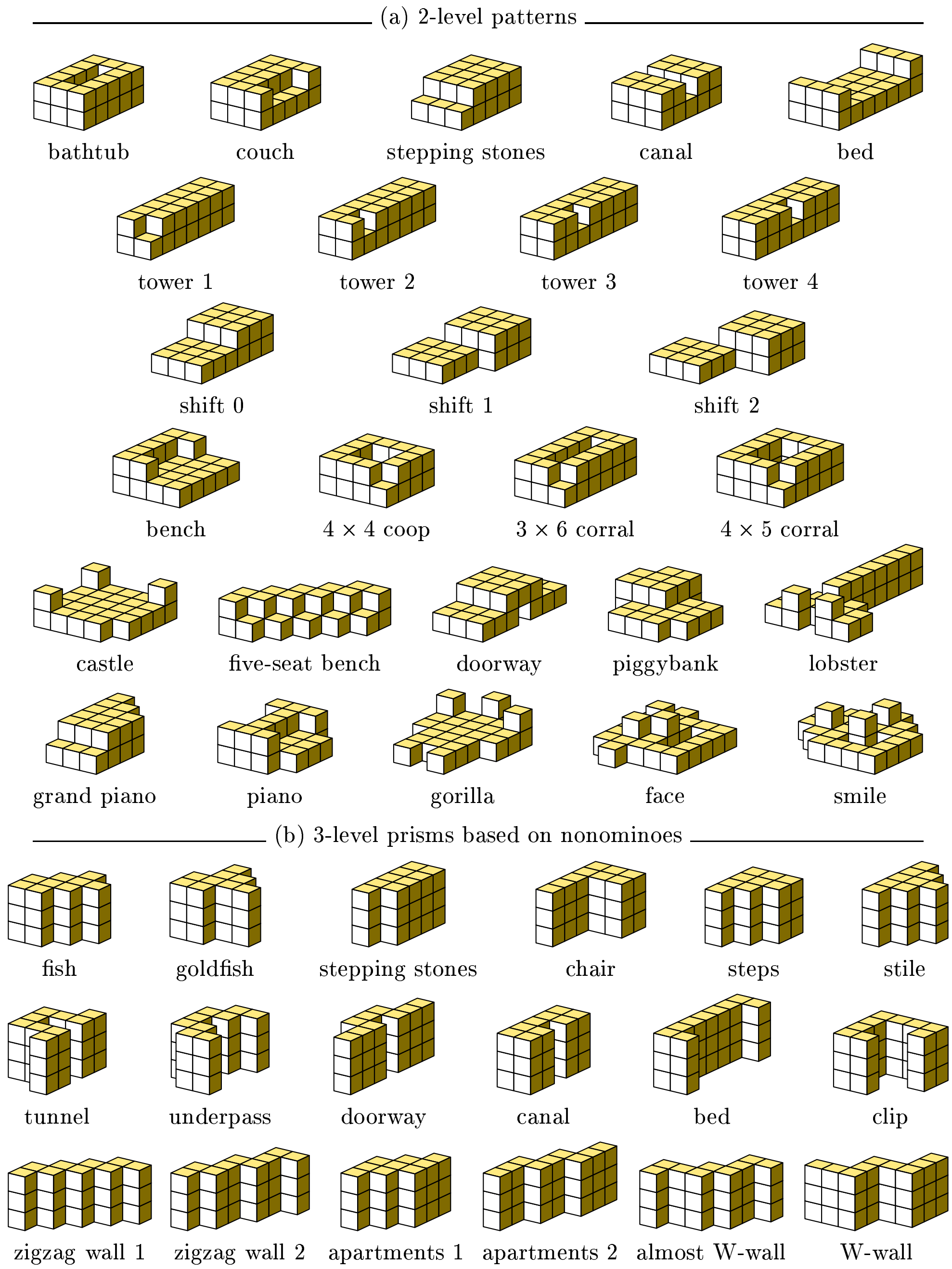
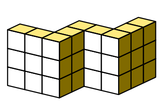
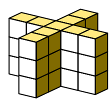
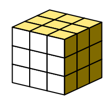


Fig. 75. Gallery of noteworthy polycubes that contain 27 cubies. All of them can be built from the seven Soma pieces, except for the W-wall. Many constructions are also stable when tipped on edge and/or when turned upside down. (See exercises 326–334.)

▶ **334.** [*M32*] Impossible structures *can* be built, if we insist only that they look genuine when viewed from the front (like façades in Hollywood movies)! Find all solutions to

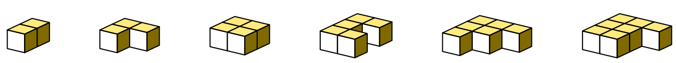

W-wall


X-wall


cube

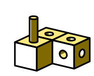
that are visually correct. (To solve this exercise, you need to know that the illustrations here use the non-isometric projection $(x, y, z) \mapsto (30x - 42y, 14x + 10y + 45z)u$ from three dimensions to two, where u is a scale factor.) All seven Soma pieces must be used.

335. [*30*] The earliest known example of a polycube puzzle is the "Cube Diabolique," manufactured in late nineteenth century France by Charles Watilliaux; it contains six flat pieces of sizes 2, 3, . . . , 7:

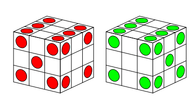


a) In how many ways do these pieces make a $3 \times 3 \times 3$ cube?
b) Are there six polycubes, of sizes 2, 3, . . . , 7, that make a cube in just *one* way?

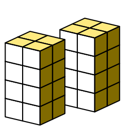
336. [*21*] (*The L-bert Hall.*) Take two cubies and drill three holes through each of them; then glue them together and attach a solid cubie and dowel, as shown. Prove that there's only one way to pack nine such pieces into a $3 \times 3 \times 3$ box.

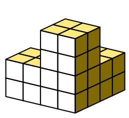


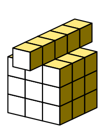
337. [*29*] (Angus Lavery, 1989.) Design a puzzle that consists of nine bent tricubes, whose face squares are either blank or colored with a red or green spot. The goal is to assemble the pieces into a $3 \times 3 \times 3$ cube in two ways:
(i) No green spots are visible, and the red spots match a right-handed die.
(ii) No red spots are visible, and the green spots match a left-handed die.

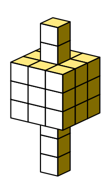


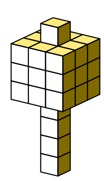
338. [*22*] Show that there are exactly eight different *tetracubes* — polycubes of size 4. Which of the following shapes can they make, respecting gravity? How many solutions are possible?

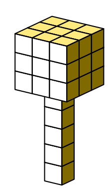

twin towers


double claw


cannon

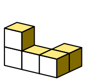
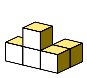
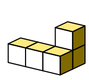
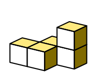
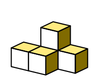
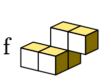

up 3

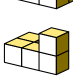
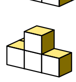
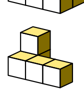
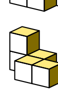
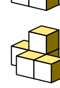
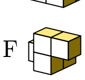

up 4

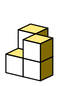
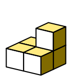
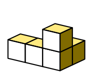
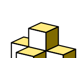
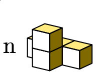

up 5

339. [*25*] How many of the 369 *octominoes* define a 4-level prism that can be realized by the tetracubes? Do any of those packing problems have a unique solution?

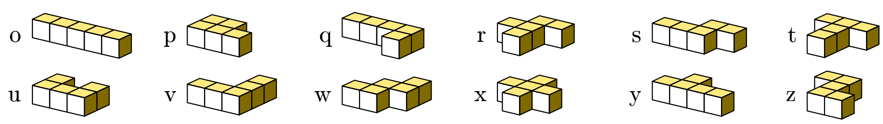
340. [*30*] There are 29 *pentacubes*, conveniently identified with one-letter codes:

a  b  c  d  e  f 

A  B  C  D  E  F 

j  k  l  m  n 

façades
movies
isometric
projection
three dimensions
Cube Diabolique
Diabolical Cube
Watilliaux
L-bert Hall
holes
dowel
Lavery
bent tricubes
tricubes
dice
tetracubes
gravity
octominoes
pentacubes



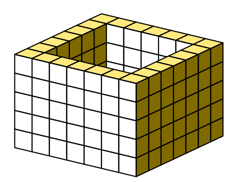
Pieces o through z are called, not surprisingly, the *solid pentominoes* or *flat pentacubes.*

a) What are the mirror images of a, b, c, d, e, f, A, B, C, D, E, F, j, k, l, . . . , z?

b) In how many ways can the solid pentominoes be packed into an $a \times b \times c$ cuboid?

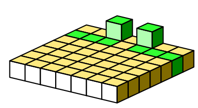
c) What "natural" set of 25 pentacubes is able to fill the $5 \times 5 \times 5$ cube?

▸ **341.** [*25*] The full set of 29 pentacubes can build an enormous variety of elegant structures, including a particularly stunning example called "Dowler's Box." This $7 \times 7 \times 5$ container, first considered by R. W. M. Dowler in 1979, is constructed from five flat slabs. Yet only 12 of the pentacubes lie flat; the other 17 must somehow be worked into the edges and corners.

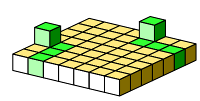


Despite these difficulties, Dowler's Box has so many solutions that we can actually impose many further conditions on its construction:

a) Build Dowler's Box in such a way that the chiral pieces a, b, c, d, e, f and their images A, B, C, D, E, F all appear in horizontally mirror-symmetric positions.



horizontally symmetric c and C

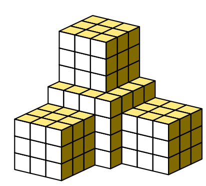


diagonally symmetric c and C

b) Alternatively, build it so that those pairs are *diagonally* mirror-symmetric.

c) Alternatively, place piece x in the center, and build the remaining structure from four congruent pieces that have seven pentacubes each.

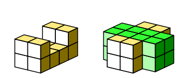
342. [*25*] The 29 pentacubes can also be used to make the shape shown here, exploiting the curious fact that $3^4 + 4^3 = 29 \cdot 5$. But Algorithm X will take a long, long time before telling us how to construct it, unless we're lucky, because the space of possibilities is huge. How can we find a solution quickly?



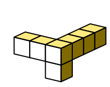
343. [*40*] (T. Sillke, 1995.) For each of the twelve pentomino shapes, build the tallest possible tower whose walls are vertical and whose floors all have the given shape, using distinct pentacubes. *Hint:* Judicious factorization will give tremendous speedup.

344. [*20*] In how many distinct ways can a $5 \times 5 \times 5$ cube by packed with 25 solid Y pentominoes? (See Fig. 73.) Discuss how to remove the 48 symmetries of this problem.

345. [*20*] Pack twelve U-shaped dodecacubes into a $4 \times 6 \times 6$ box without letting any two of them form a "cross."



346. [*M30*] An (l, m, n)-*tripod* is a cluster of $l + m + n + 1$ cubies in which three "legs" of lengths l, m, and n are attached to a corner cubie, as in the (1,2,3)-tripod shown here. A "pod" is the special case where the tripod is



$$(l, m, n) \cup \{(l', m, n) \mid 0 \le l' < l\} \cup \{(l, m', n) \mid 0 \le m' < m\} \cup \{(l, m, n') \mid 0 \le n' < n\}.$$

a) Prove that, for all $m, n \ge 0$, shifted copies of nonoverlapping $(1, m, n)$-tripods are able to fill all of 3-dimensional space, without rotation or reflection. *Hint:* Pack N^2 of them into an $N \times N \times N$ torus, where $N = m + n + 2$.

solid pentominoes
flat pentacubes
mirror images
pentominoes
$5 \times 5 \times 5$ cube
Dowler's Box
chiral
mirror
Sillke
pentomino
tower
factorization
Y pentominoes
symmetries
cross
tripod
pod
torus, 3D

b) Show that 7/9 of 3-dimensional space can be packed with shifted $(2,2,2)$-tripods.
c) Similarly, at least 65/108 of 3D space can be packed with shifted $(3,3,3)$-tripods.
d) Let $r(l,m,n)$ be the maximum number of pods that can be packed in an $l\times m\times n$ cuboid. Prove that at least $(1{+}l{+}m{+}n)r(l,m,n)/(4lmn)$ of 3-dimensional space can be packed with shifted (l,m,n)-tripods.
e) Use Algorithm M to evaluate $r(l,m,n)$ for $4\le l\le m\le n\le 6$.

▸ **347.** [*M21*] (N. G. de Bruijn, 1961.) Prove that an $l\times m\times n$ box can be completely filled with $1\times 1\times k$ bricks only if k is a divisor of l, m, or n. (Consequently, it can be completely filled with $a\times b\times c$ bricks only if a, b, and c all satisfy this condition.)

348. [*M41*] Find the maximum number of "canonical bricks" $(1\times 2\times 4)$ that can be packed into an $l\times m\times n$ box, leaving as few empty cells as possible.

▸ **349.** [*M27*] (D. Hoffman.) Show that 27 bricks of size $a\times b\times c$ can always be packed into an $s\times s\times s$ cube, where $s=a+b+c$. But if $s/4<a<b<c$, 28 bricks won't fit.

350. [*22*] Can 28 bricks of size $3\times 4\times 5$ be packed into a $12\times 12\times 12$ cube?

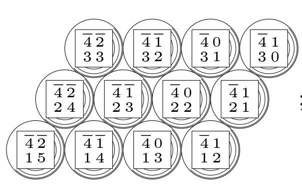
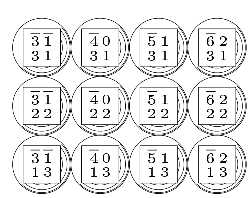
351. [*M46*] Can 5^5 hypercuboids of size $a\times b\times c\times d\times e$ always be packed into a 5-dimensional hypercube of size $(a+b+c+d+e)\times\cdots\times(a+b+c+d+e)$?

352. [*21*] In how many ways can the 12 pentominoes be packed into a $2\times 2\times 3\times 5$ box?

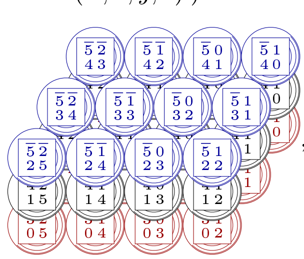
353. [*20*] A *weak polycube* is a set of cubies that are loosely connected via common edges, not necessarily via common faces. In other words, we consider cubies to be adjacent when their centers are at most $\sqrt{2}$ units apart; up to 18 neighbors are possible. Find all the weak polycubes of size 3, and pack them into a symmetrical container.

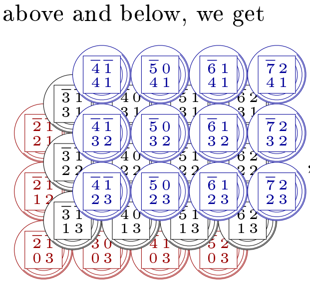
▸ **354.** [*M30*] A *polysphere* is a connected set of spherical cells that belong to the "face-centered cubic lattice," which is one of the two principal ways to pack cannonballs (or oranges) with maximum efficiency. That lattice is conveniently regarded as the set S of all quadruples (w,x,y,z) of integers for which $w+x+y+z=0$. Each cell of S has 12 neighbors, obtained by adding 1 to one coordinate and subtracting 1 from another.

It's instructive to view S in two different ways, by slicing it into plane layers that either have constant $x{+}y{+}z$ (hence constant w) or constant $y{+}z$ (hence constant $w{+}x$):

$x+y+z=4$  ; $y+z=4$  .

(Here $\boxed{\begin{smallmatrix}w\,x\\ y\,z\end{smallmatrix}}$ stands for (w,x,y,z).) If we include the layers above and below, we get

$x+y+z=5$
$x+y+z=4$
$x+y+z=3$  ,

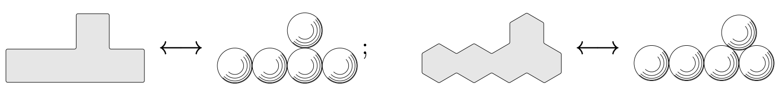
$y+z=5$
$y+z=4$
$y+z=3$  ,

with each sphere nestling in the gap between the three or four spheres below it. In the "hex layers" on the left, (w,x,y,z) lies directly above $(w+3,x-1,y-1,z-1)$,

de Bruijn
bricks
canonical bricks
Hoffman
bricks
polysphere
face-centered cubic lattice
cannonballs
oranges
ball-piling
barycentric coordinates
close packing of spheres

but doesn't touch it; in the "quad layers" on the right, (w, x, y, z) lies directly above $(w+1, x+1, y-1, z-1)$, but doesn't touch it.

a) Show that every polyomino, and every polyhex, may be regarded as a polysphere:



b) Conversely, every *planar* polysphere looks like either a polyomino or a polyhex.

c) Every polysphere $\{(w_1, x_1, y_1, z_1), \ldots, (w_n, x_n, y_n, z_n)\}$ has a unique *base placement* $\{(w'_1, x'_1, y'_1, z'_1), \ldots, (w'_n, x'_n, y'_n, z'_n)\}$ obtained by subtracting (w', x', y', z') from each (w_k, x_k, y_k, z_k), where $x' = \min\{x_1, \ldots, x_n\}$, $y' = \min\{y_1, \ldots, y_n\}$, $z' = \min\{z_1, \ldots, z_n\}$, and $w' + x' + y' + z' = 0$. Prove that $x'_k + y'_k + z'_k < n$.

d) As with polycubes, we say that polyspheres v and v' are *equivalent* if the base placement of v is also a base placement of some rotation of v' in three dimensions. (Reflections of "chiral" polyspheres are not considered to be equivalent.) Formally speaking, a rotation of S about a line through the origin is an orthogonal 4×4 matrix that has determinant 1 and preserves $w + x + y + z$. Find such matrices for (i) rotation of the hex layers by 120°; (ii) rotation of the quad layers by 90°.

e) A planar polysphere is equivalent to its reflection, because we can rotate by 180° around a line in its plane. Find suitable 4×4 matrices by which we can legally reflect polyspheres that are equivalent to (i) polyominoes; (ii) polyhexes.

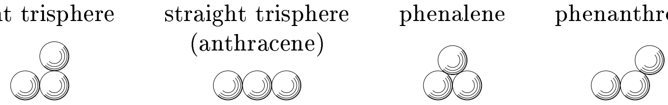
f) Prove that every rotation that takes a polysphere into another polysphere is obtainable as a product of the matrices exhibited in (d) and (e).

355. [*25*] The theory in exercise 354 allows us to represent polysphere cells with three integer coordinates xyz, because x, y, and z are nonnegative in base placements. The other variable, w, is redundant (but worth keeping in mind); it always equals $-x-y-z$.

a) What's a good way to find all the base placements of a given polysphere $\{x_1y_1z_1, x_2y_2z_2, \ldots, x_ny_nz_n\}$? *Hint:* Use exercise 354 to tweak the method of exercise 324.

b) Any three points of three-dimensional space lie in a plane. So exercise 354(b) tells us that there are just four trispheres: a tromino, two trihexes, and one that's both:

bent trisphere | straight trisphere (anthracene) | phenalene | phenanthrene

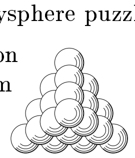


What are their base placements?

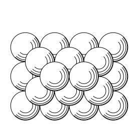
c) According to exercise 354(c), every base placement of a tetrasphere occurs in the SGB graph *simplex*$(3, 3, 3, 3, 3, 0, 0)$. Use exercise 7.2.2–75 to find all of the four-element connected subsets of that graph, and identify all of the distinct tetraspheres. How many times does each tetrasphere occur in the graph?

356. [*23*] Polysphere puzzles often involve the construction of three kinds of shapes:

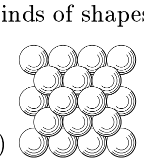
n-tetrahedron (as seen from the top, for $n = 4$)



$m \times n$ roof (shown for $m = 3$, $n = 4$)



stretched $m \times n$ roof (shown for $m{=}3, n{=}4$)



(An $n \times n$ roof or stretched roof is called an "n-pyramid" or a "stretched n-pyramid.")

a) Define each of these configurations by specifying a suitable base placement.

b) Each of the shapes illustrated is made from 20 spheres, and so is the stretched 4×3 roof. Find all multisets of five *tetraspheres* that suffice to make these shapes.

planar polysphere
base placement
rotation
chiral
orthogonal 4×4 matrix
base placements
trispheres
tromino
trihexes
anthracene
phenanthrene
phenalene
SGB graph
simplex
connected subsets
tetraspheres
pyramid
tetrahedron
roof
tetraspheres

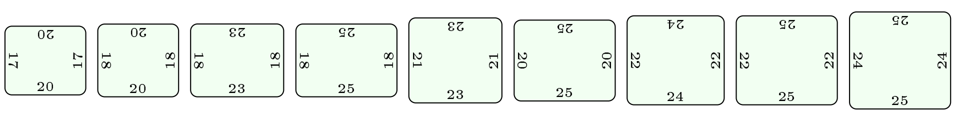
c) The 4-pyramid and the stretched 4-pyramid involve 30 spheres. What multisets of ten *trispheres* are able to make them?

d) The *truncated octahedron* that represents all permutations of $\{1, 2, 3, 4\}$ is a noteworthy 24-cell subset of S (see exercise 5.1.1– 10). What multisets of six tetraspheres can build it?

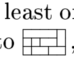
357. [*M40*] Investigate "polysplatts," which are the sets of truncated octahedra that can be built by pasting adjacent faces together (either square or hexagonal).

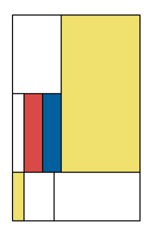
358. [*HM41*] Investigate "polyhexaspheres," which are the connected sets of spheres in the *hexagonal close packing*. (This packing differs from that of exercise 354 because each sphere of a hexagonal layer is directly above a sphere that's 2, not 3, layers below it.)

359. [*29*] Nick Baxter devised an innocuous-looking but maddeningly difficult "Square Dissection" puzzle for the International Puzzle Party in 2014, asking that the nine pieces



be placed flat into a 65×65 square. One quickly checks that $17\times 20+18\times 20+\cdots+24\times 25 = 65^2$; yet nothing seems to work! Solve his puzzle with the help of Algorithm X.

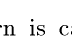
▸ **360.** [*20*] The next group of exercises is devoted to the decomposition of rectangles into rectangles, as in the Mondrianesque pattern shown here. The *reduction* of such a pattern is obtained by distorting it, if necessary, so that it fits into an $m \times n$ grid, with each of the vertical coordinates $\{0, 1, \ldots, m\}$ used in at least one horizontal boundary and each of the horizontal coordinates $\{0, 1, \ldots, n\}$ used in at least one vertical boundary. For example, the illustrated pattern reduces to , where $m = 3$ and $n = 5$. (Notice that the original rectangles needn't have rational width or height.)



A pattern is called *reduced* if it is equal to its own reduction. Design an exact cover problem by which Algorithm M will discover all of the reduced decompositions of an $m\times n$ rectangle, given m and n. How many of them are possible when $(m, n) = (3, 5)$?

361. [*M25*] The maximum number of subrectangles in a reduced $m \times n$ pattern is obviously mn. What is the *minimum* number?

362. [*10*] A reduced pattern is called *strictly reduced* if each of its subrectangles $[a \mathinner{..} b)\times[c \mathinner{..} d)$ has $(a, b) \neq (0, m)$ and $(c, d) \neq (0, n)$ — in other words, if no subrectangle "cuts all the way across." Modify the construction of exercise 360 so that it produces only strictly reduced solutions. How many 3×5 patterns are strictly reduced?

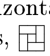
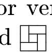
363. [*20*] A rectangle decomposition is called *faultfree* if it cannot be split into two or more rectangles. For example,  is *not* faultfree, because it has a fault line between rows 2 and 3. (It's easy to see that every reduced faultfree pattern is *strictly* reduced, unless $m = n = 1$.) Modify the construction of exercise 360 so that it produces only faultfree solutions. How many reduced 3×5 patterns are faultfree?

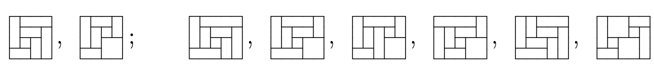
364. [*23*] True or false: Every faultfree packing of an $m\times n$ rectangle by 1×3 trominoes is reduced, except in the trivial cases $(m, n) = (1, 3)$ or $(3, 1)$.

365. [*22*] (*Motley dissections.*) Many of the most interesting decompositions of an $m\times n$ rectangle involve strictly reduced patterns whose subrectangles $[a_i \mathinner{..} b_i)\times[c_i \mathinner{..} d_i)$ satisfy the extra condition

$$(a_i, b_i) \neq (a_j, b_j) \quad \text{and} \quad (c_i, d_i) \neq (c_j, d_j) \quad \text{when } i \neq j.$$

trispheres
truncated octahedron
permutations
polysplatts
polyhexaspheres
hexagonal close packing
Baxter
Square Dissection
rectangles into rectangles
Mondrian
reduction
strictly reduced
faultfree
trominoes
straight trominoes: 1×3
Motley dissections
dissection: decomposition of one structure into substructures

Thus no two subrectangles are cut off by the same pair of horizontal or vertical lines. The smallest such "motley dissections" are the 3×3 pinwheels,  and , which are considered to be essentially the same because they are mirror images of each other. There are eight essentially distinct motley rectangles of size $4 \times n$, namely



The two 4×4s can each be drawn in 8 different ways, under rotations and reflections. Similarly, most of the 4×5s can be drawn in 4 different ways. But the last two have only two forms, because they're symmetric under 180° rotation. (And the last two are actually equivalent, if we swap the two x coordinates in the middle.)

Design an exact cover problem by which Algorithm M will discover all of the motley dissections of an $m \times n$ rectangle, given m and n. (When $m = n = 4$ the algorithm should find $8 + 8$ solutions; when $m = 4$ and $n = 5$ it should find $4+4+4+4+4+2+2$.)

▸ **366.** [*25*] Improve the construction of the previous exercise by taking advantage of symmetry to cut the number of solutions in half. (When $m = 4$ there will now be $4+4$ solutions when $n = 4$, and $2+2+2+2+2+1+1$ when $n = 5$.) *Hint:* A motley dissection is never identical to its left-right reflection, so we needn't visit both.

367. [*20*] The *order* of a motley dissection is the number of subrectangles it has. There are no motley dissections of order six. Show, however, that there are $m \times m$ motley dissections of order $2m-1$ and $m \times (m+1)$ motley dissections of order $2m$, for all $m > 3$.

368. [*M21*] (H. Postl, 2017.) Show that an $m \times n$ motley dissection of order t can exist only if $n < 2t/3$. *Hint:* Consider adjacent subrectangles.

369. [*21*] An $m \times n$ motley dissection must have order less than $\binom{m+1}{2}$, because only $\binom{m+1}{2} - 1$ intervals $[a_i \,.\,.\, b_i)$ are permitted. What is the maximum order that's actually achievable by an $m \times n$ motley dissection, for $m = 5$, 6, and 7?

▸ **370.** [*23*] Explain how to generate all of the $m \times n$ motley dissections that have 180°-rotational symmetry, as in the last two examples of exercise 365, by modifying the construction of exercise 366. (In other words, if $[a \,.\,.\, b) \times [c \,.\,.\, d)$ is a subrectangle of the dissection, its complement $[m-b \,.\,.\, m-a) \times [n-d \,.\,.\, n-c)$ must also be one of the subrectangles, possibly the same one.) How many such dissections have size 8×16?

371. [*24*] Further symmetry is possible when $m = n$ (as in exercise 365's pinwheel).

a) Explain how to generate all of the $n \times n$ motley dissections that have 90°-rotational symmetry. This means that $[a \,.\,.\, b) \times [c \,.\,.\, d)$ implies $[c \,.\,.\, d) \times [n-b \,.\,.\, n-a)$.

b) Explain how to generate all of the $n \times n$ motley dissections that are symmetric under reflection about both diagonals. This means that $[a \,.\,.\, b) \times [c \,.\,.\, d)$ implies $[c \,.\,.\, d) \times [a \,.\,.\, b)$ and $[n-d \,.\,.\, n-c) \times [n-b \,.\,.\, n-a)$, hence $[n-b \,.\,.\, n-a) \times [n-d \,.\,.\, n-c)$.

c) What's the smallest n for which symmetric solutions of type (b) exist?

372. [*M30*] (*Orthogonal digraphs.*) Let G be a digraph on the vertices $\{0, 1, \ldots, m\}$. Its arcs can be defined by an $(m+1) \times r$ incidence matrix $A = (a_{ik})$, where

$$a_{ik} \;=\; \begin{cases} +1, & \text{if } i \text{ is the initial vertex of arc } k; \\ -1, & \text{if } i \text{ is the final vertex of arc } k; \\ \;\;\,0, & \text{otherwise;} \end{cases} \qquad \text{for } 0 \le i \le m,\ 0 \le k < r.$$

Similarly, let H be a digraph on the vertices $\{0, 1, \ldots, n\}$, with arcs defined by the $(n+1) \times r$ incidence matrix $B = (b_{jk})$. We say that G and H are *orthogonal* if (i) $a_i \cdot b_j = 0$ for $0 \le i \le m$ and $0 \le j \le n$; (ii) G and H are connected; and (iii) $r = m + n$.

pinwheels
rotations and reflections
symmetric under 180° rotation
symmetry
order
Postl
180°-rotational symmetry
complement
pinwheel
90°-rotational symmetry
reflection about both diagonals
bidiagonal symmetry
Orthogonal digraphs
incidence matrix

transitive tournament
tatami condition
sphere
stereographic projection
pinwheels
perfectly decomposed rectangle
incomparable dissection
motley
reduction
linear inequalities

For example, when $m = n = 3$ and $r = 6$, the digraphs defined by

$$A = \begin{pmatrix} +1 & 0 & 0 & +1 & 0 & +1 \\ -1 & +1 & 0 & 0 & +1 & 0 \\ 0 & -1 & +1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} +1 & 0 & 0 & 0 & +1 & -1 \\ 0 & +1 & +1 & 0 & -1 & 0 \\ -1 & -1 & 0 & +1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & +1 \end{pmatrix}$$

are orthogonal; here G is the transitive tournament $\vec{K_4}$.

a) Find all H orthogonal to G when $m = 1$ and $A = \left(\begin{smallmatrix} +1 & +1 & \ldots & +1 \\ -1 & -1 & \ldots & -1 \end{smallmatrix}\right)$.

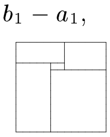
b) Find all H orthogonal to the digraph $0 \longrightarrow 1$, $0 \longrightarrow 2$, $1 \longrightarrow 2$, $1 \longrightarrow 3$, $2 \longrightarrow 3$.

c) Explain how to find all H orthogonal to a given digraph, using Algorithm M.

d) Does the transitive tournament $\vec{K_5}$ have an orthogonal mate?

e) Show that any rectangular dissection of order t leads to a pair of orthogonal digraphs with $r = t + 1$, by the following construction: Assume for convenience that no four subrectangles meet at a common point, and let the dissection consist of $m + 1$ horizontal line segments plus $n + 1$ vertical line segments, where the segments are as long as possible. If the jth subrectangle is bounded by horizontal segments β_j and τ_j at the bottom and top, and by vertical segments λ_j and ρ_j at the left and right, then the jth arcs of G and H are $\beta_j \longrightarrow \tau_j$ and $\lambda_j \longrightarrow \rho_j$. Include also arcs for the *outer* region — everything outside the main rectangle — since we can regard the infinite plane as a sphere, using stereographic projection. This outer region has $(\beta_t, \tau_t, \lambda_t, \rho_t) =$ (bottom, top, right, left) of the main rectangle (which we observe "from behind"). For example, when this construction is applied to the pinwheels of exercise 365, it yields matrices A and B above.

373. [*26*] A "perfectly decomposed rectangle" of order t is a faultfree dissection of a rectangle into t subrectangles $[a_i \,..\, b_i) \times [c_i \,..\, d_i)$ such that the $2t$ dimensions $b_1 - a_1$, $d_1 - c_1$, …, $b_t - a_t$, $d_t - c_t$ are distinct. For example, five rectangles of sizes 1×2, 3×7, 4×6, 5×10, and 8×9 can be assembled to make the perfectly decomposed 13×13 square shown here. What are the *smallest possible* perfectly decomposed squares of orders 5, 6, 7, 8, 9, and 10, having integer dimensions?



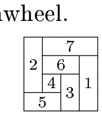
374. [*M28*] An "incomparable dissection" of order t is a decomposition of a rectangle into t subrectangles, none of which will fit inside another. In other words, if the heights and widths of the subrectangles are respectively $h_1 \times w_1$, …, $h_t \times w_t$, we have neither ($h_i \le h_j$ and $w_i \le w_j$) nor ($h_i \le w_j$ and $w_i \le h_j$) when $i \ne j$.

a) True or false: An incomparable dissection is perfectly decomposed.

b) True or false: The reduction of an incomparable dissection is motley.

c) True or false: The reduction of an incomparable dissection can't be a pinwheel.

d) Prove that every incomparable dissection of order ≤ 7 reduces to the first 4×4 motley dissection in exercise 365; and its seven regions can be labeled as shown, with $h_7 < h_6 < \cdots < h_2 < h_1$ and $w_1 < w_2 < \cdots < w_6 < w_7$.



e) Suppose the reduction of an incomparable dissection is $m \times n$, and suppose its regions have been labeled $\{1, \ldots, t\}$. Then there are numbers $x_1, \ldots, x_n, y_1, \ldots, y_m$ such that the widths are sums of the x's and the heights are sums of the y's. (For example, in (d) we have $w_2 = x_1$, $h_2 = y_1 + y_2 + y_3$, $w_7 = x_2 + x_3 + x_4$, $h_7 = y_1$, etc.) Prove that such a dissection exists with $w_1 < w_2 < \cdots < w_t$ if and only if the linear inequalities $w_1 < w_2 < \cdots < w_t$ have a positive solution $(x_1, \ldots, x_n)$ and the linear inequalities $h_1 > h_2 > \cdots > h_t$ have a positive solution $(y_1, \ldots, y_m)$.

375. [*M29*] Among all the incomparable dissections of order (a) seven and (b) eight, restricted to integer sizes, find the rectangles with smallest possible perimeter. Also

concatenated
finite basis theorem
Q pentomino
Y pentomino
disconnected
subcuboids
Kim
Postl
nesting
primitive

find the smallest possible *squares* that have incomparable dissections in integers. *Hint:* Show that there are 2^t potential ways to mix the h's with the w's, preserving their order; and find the smallest perimeter for each of those cases.

▸ **376.** [*M25*] Find seven *different* rectangles of area $1/7$ that can be assembled into a square of area 1, and prove that the answer is unique.

377. [*M28*] Two rectangles of shapes $h \times w$ and $h' \times w'$ can be *concatenated* to form a larger rectangle of size $(h+h') \times w$ if $w = w'$, or of size $h \times (w+w')$ if $h = h'$.

a) Given a set S of rectangle shapes, let $\Lambda(S)$ be the set of all shapes that can be made from the elements of S by repeated concatenation. Describe $\Lambda(\{1{\times}2, 3{\times}1\})$.
b) Find the smallest $S \subseteq T$ such that $T \subseteq \Lambda(S)$, where $T = \{\, h{\times}w \mid 1 < h < w\}$.
c) What's the smallest S with $\Lambda(S) = \{\, h{\times}w \mid h, w > 1 \text{ and } hw \bmod 8 = 0\}$?
d) Given m and n, solve (c) with $\Lambda(S) = \{\, h{\times}w \mid h, w > m \text{ and } hw \bmod n = 0\}$.

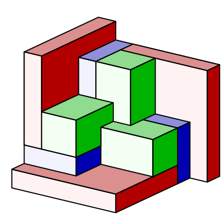
▸ **378.** [*M30*] (*A finite basis theorem.*) Continuing exercise 377, prove that *any* set T of rectangular shapes contains a finite subset S such that $T \subseteq \Lambda(S)$.

▸ **379.** [*23*] What $h \times w$ rectangles can be packed with copies of the Q pentomino? *Hint:* It suffices to find a finite basis for all such rectangles, using the previous exercise.

380. [*35*] Solve exercise 379 for the Y pentomino.

381. [*20*] Show that $3n$ copies of the *disconnected* shape '□ □□ □' can pack a $12 \times n$ rectangle for all sufficiently large values of n.

▸ **382.** [*18*] There's a natural way to extend the idea of motley dissection to three dimensions, by subdividing an $l \times m \times n$ cuboid into subcuboids $[a_i \,.\,.\, b_i) \times [c_i \,.\,.\, d_i) \times [e_i \,.\,.\, f_i)$ that have no repeated intervals $[a_i \,.\,.\, b_i)$ or $[c_i \,.\,.\, d_i)$ or $[e_i \,.\,.\, f_i)$.



For example, Scott Kim has discovered a remarkable motley $7 \times 7 \times 7$ cube consisting of 23 individual blocks, 11 of which are illustrated here. (Two of them are hidden behind the others.) The full cube is obtained by suitably placing a mirror image of these pieces in front, together with a $1 \times 1 \times 1$ cubie in the center.

By studying this picture, show that Kim's construction can be defined by coordinate intervals $[a_i \,.\,.\, b_i) \times [c_i \,.\,.\, d_i) \times [e_i \,.\,.\, f_i)$, with $0 \le a_i, b_i, c_i, d_i, e_i, f_i \le 7$ for $1 \le i \le 23$, in such a way that the pattern is symmetrical under the transformation $xyz \mapsto \bar{y}\bar{z}\bar{x}$. In other words, if $[a \,.\,.\, b) \times [c \,.\,.\, d) \times [e \,.\,.\, f)$ is one of the subcuboids, so is $[7-d \,.\,.\, 7-c) \times [7-f \,.\,.\, 7-e) \times [7-b \,.\,.\, 7-a)$.

383. [*29*] Use exercise 382 to construct a perfectly decomposed $92 \times 92 \times 92$ cube, consisting of 23 subcuboids that have 69 distinct integer dimensions. (See exercise 373.)

384. [*24*] By generalizing exercises 365 and 366, explain how to find *every* motley dissection of an $l \times m \times n$ cuboid, using Algorithm M. *Note:* In three dimensions, the strictness condition '$(a_i, b_i) \ne (0, m)$ and $(c_i, d_i) \ne (0, n)$' of exercise 362 should become

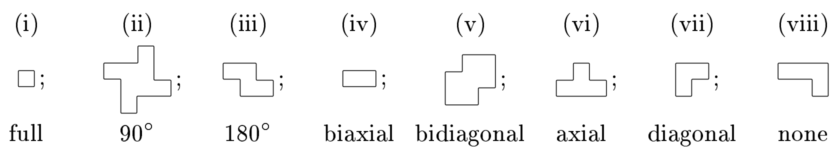
$$\bigl[(a_i, b_i) = (0, l)\bigr] + \bigl[(c_i, d_i) = (0, m)\bigr] + \bigl[(e_i, f_i) = (0, n)\bigr] \;\le\; 1.$$

What are the results when $l = m = n = 7$?

385. [*M36*] (H. Postl, 2017.) Arbitrarily large motley cuboids can be constructed by repeatedly nesting one motley cuboid within another (see answer 367). Say that a motley cuboid is *primitive* if it doesn't contain a nested motley subcuboid.

Do primitive motley cuboids of size $l \times m \times n$ exist only when $l = m = n = 7$?

▶ **386.** [*M34*] A polyomino can have eight different types of symmetry:



(Case (i) is often called 8-fold symmetry; case (iii) is often called central symmetry; case (vi) is often called left-right symmetry. Cases (ii), (iv), (v) are 4-fold symmetries; cases (ii) and (iii) are rotation symmetries; cases (iv)–(vii) are reflection symmetries.) In each case an n-omino of that symmetry type has been shown, where n is minimum.

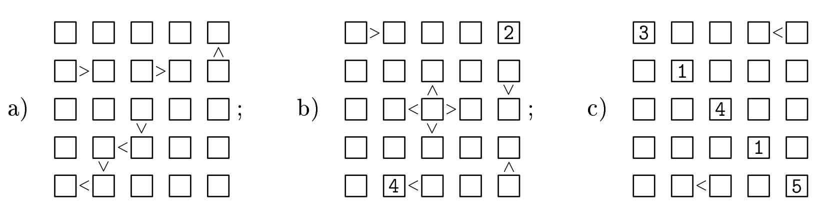
How many symmetry types can a polyiamond or polyhex have? Give example n-iamonds and n-hexes of each type, where n is minimum.

▶ **387.** [*M36*] Continuing exercise 386, how many symmetry types can a polycube have? Give an example of each type, using the minimum number of cubies.

EXERCISES — Third Set

The following exercises are based on several intriguing logic puzzles that have recently become popular: futoshiki, kenken, masyu, slitherlink, kakuro, etc. Like sudoku, these puzzles typically involve a hidden pattern, for which only partial information has been revealed. The point of each exercise is usually to set up an appropriate exact cover problem, and to use it either to solve such a puzzle or to create new ones.

▶ **388.** [*21*] The goal of a *futoshiki* puzzle is to deduce the entries of a secret latin square, given only two kinds of hints: A "strong clue" is an explicit entry; a "weak clue" is a greater-than relation between neighboring entries. The entries are the numbers 1 to n, where n is usually 5 as in the following examples:



Solve these puzzles by hand, using sudoku-like principles.

389. [*20*] Sketch a simple algorithm that finds simple lower and upper bounds for each entry that is part of a weak clue in a futoshiki puzzle, by repeatedly using the rule that $a \le x < y \le b$ implies $x \le b-1$ and $y \ge a+1$. (Your algorithm shouldn't attempt to give the best possible bounds; that would solve the puzzle! But it should deduce the values of five entries in puzzle (a) of exercise 388, as well as entry $(4,2)$ of puzzle (b).)

▶ **390.** [*21*] Show that every futoshiki puzzle is a special case of an exact cover problem. In fact, show that every such puzzle can be formulated in at least two different ways:

a) Use a *pairwise ordering trick* analogous to (25) or (26), to encode the weak clues.
b) Use *color controls* to formulate an XCC problem suitable for Algorithm C.

391. [*20*] A futoshiki puzzle is said to be *valid* if it has exactly one solution. Use Algorithm X to generate all possible 5×5 latin squares. Explain why many of them can't be the solution to a valid futoshiki puzzle unless it has at least one strong clue.

polyomino
8-fold symmetry
central symmetry
left-right symmetry
biaxial
bidiagonal
axial
90°
180°
bilateral symmetry, see biaxial
symmetry types
polyiamond
polyhex
polycube
logic puzzles
futoshiki
kenken
masyu
slitherlink
kakuro
sudoku
futoshiki
latin square
strong clue
weak clue
pi as source
sudoku
lower and upper bounds
bounds
exact cover problem
pairwise ordering trick
color controls
XCC problem
valid

▶ **392.** [*25*] There are $2^6\binom{40}{6} = 245656320$ ways to construct a 5×5 futoshiki puzzle that has six weak clues and no strong ones. How many of them (a) are valid? (b) have no solutions? (c) have more than one solution? Also refine those counts, by considering how many such puzzles of types (a), (b), and (c) have at least one "long path" $p < q < r < s < t$ (like the path that's present in exercise 388(a)). Give an example of each case.

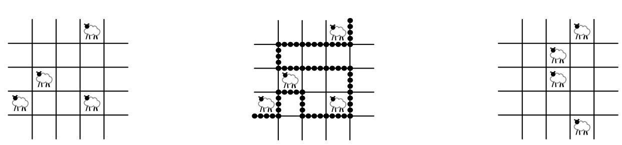
393. [*25*] There are $5^6\binom{25}{6} = 2767187500$ ways to construct a 5×5 futoshiki puzzle that has six strong clues and no weak ones. How many of them (a) are valid? (b) have no solutions? (c) have more than one solution? Give an example of each case.

394. [*29*] Show that every 5×5 futoshiki puzzle that has only five clues — strong, weak, or a mixture of both — has at least four solutions. Which puzzles attain this minimum?

395. [*25*] Continuing exercise 391, find a 5×5 latin square that cannot be the solution to a valid futoshiki puzzle unless at least *three* strong clues have been given.

▶ **396.** [*25*] Inspired by exercise 388(c), construct a valid 9×9 futoshiki puzzle whose diagonal contains the strong clues $(3, 1, 4, 1, 5, 9, 2, 6, 5)$ in that order. Every *other* clue should be a weak '$<$' — not a '$>$', not a '$\wedge$', not a '$\vee$'.

▶ **397.** [*30*] (*Save the sheep.*) Given a grid in which some of the cells are occupied by sheep, the object of this puzzle is to construct a fence that keeps all the sheep on one side. The fence must begin and end at the edge of the grid, and it must follow the grid lines without visiting any point twice. Furthermore, *exactly two edges of each sheep's square should be part of the fence.* For example, consider the following 5×5 grids:

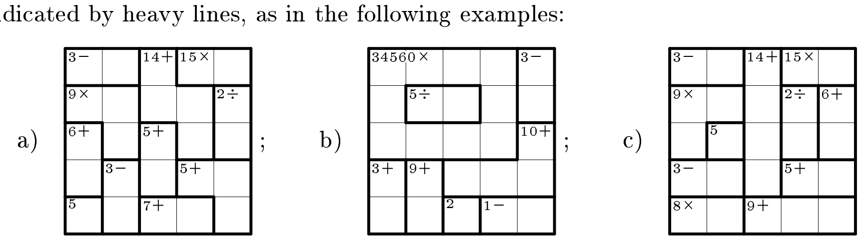


The four sheep on the left can be "saved" only with the fence shown in the middle. Once you understand why, you'll be ready to save the four sheep on the right.

a) Explain how Algorithm C can help to solve puzzles like this, by showing that every solution satisfies a certain XCC problem. *Hint:* Imagine "coloring" each square with 0 or 1, with 1 indicating the cells on the sheep's side of the fence.

b) Devise an interesting 8×8 puzzle that has a unique solution and at most 10 sheep.

398. [*23*] (*KenKen®.*) A secret latin square whose entries are $\{1, 2, \ldots, n\}$ can often be deduced by means of arithmetic. A kenken puzzle specifies the sum, difference, product, or quotient of the entries in each of its "cages," which are groups of cells indicated by heavy lines, as in the following examples:



(When the operation is '−' or '÷', the cage must have just two cells. A one-cell cage simply states its contents, without any operation; hence its solution is a no-brainer.)

Cages look rather like the boxes of jigsaw sudoku (see (34)); but in fact the rules are quite different: Two entries of the same cage can be equal, if they belong to

pi
Save the sheep
sheep
fence
XCC
KenKen®
cages
pi, as source

different rows and different columns. For example, the '9×' in (a) can be achieved only by multiplying the three entries $\{1, 3, 3\}$; hence there's exactly one way to fill that cage.

Solve (a), (b), (c) by hand. Show that one of them is actually *not* a valid puzzle.

▸ **399.** [*22*] How can all solutions to a kenken puzzle be obtained with Algorithm C?

400. [*21*] Many clues of a kenken puzzle often turn out to be redundant, in the sense that the contents of one cage might be fully determined by the clues from other cages. For example, it turns out that any one of the clues in puzzle 398(a) could actually be omitted, without permitting a new solution.

Find all subsets of those 11 clues that suffice to determine a unique latin square.

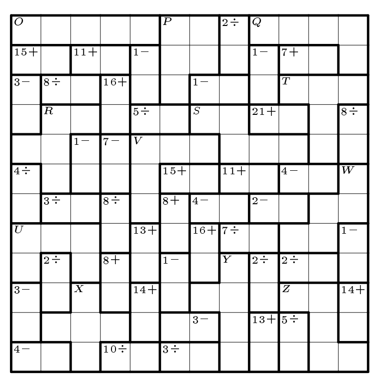
401. [*22*] Find all 4×4 kenken puzzles whose unique solution is the latin square shown at the right, and whose cages all have two cells. Furthermore, there should be exactly two cages for each of the four operations $+$, $-$, $\times$, $\div$.

1234
2143
4312
3421

402. [*24*] Solve this 12×12 kenken puzzle, using hexadecimal digits from `1` to `C`:

The five-cell cages of this puzzle have multiplicative clues, associated with the names of the twelve pentominoes:

O, 9240×
P, 5184×
Q, 3168×
R, 720×
S, 15840×
T, 19800×
U, 10560×
V, 4032×
W, 1620×
X, 5040×
Y, 576×
Z, 17248×



▸ **403.** [*31*] Inspired by exercises 398(a) and 398(c), construct a valid 9×9 kenken puzzle whose clues exactly match the decimal digits of π, for as many places as you can.

▸ **404.** [*25*] (*Hidato*®.) A "hidato solution" is an $m \times n$ matrix whose entries are a permutation of $\{1, 2, \ldots, mn\}$ for which the cells containing k and $k+1$ are next to each other, either horizontally, vertically, or diagonally, for $1 \le k < mn$. (In other words, it specifies a Hamiltonian path of king moves on an $m \times n$ board.) A "hidato puzzle" is a subset of those numbers, which uniquely determines the others; the solver is supposed to recreate the entire path from the given clues.

	3	14	1
5	9		
		8	

(i)

	3	14	1
	4	2	
5	9		
		8	

(ii)

	3	14	1
4		2	
5	9		
6		8	

(iii)

16	3	14	1
4	15	2	13
5	9	10	12
6	7	8	11

(iv)

For example, consider the 4×4 puzzle (i). There's only one place to put '2'. Then there are two choices for '4'; but one of them blocks the upper left corner (see (ii)), so we must choose the other. Similarly, '6' must not block any corner. Therefore (iii) is forced; and it's easy to fill in all of the remaining blanks, thereby obtaining solution (iv).

Explain how to encode such puzzles for solution by Algorithm C.

valid
hexadecimal digits
pentominoes
names of pentominoes
π
Hidato®
Hamiltonian path
king moves
8-neighbors, see King neighbors
pi, as "random"

simple paths
NP-hard
Beluhov
π
Slitherlink
closed paths
loops
cycles
rectangular grid
dead end

405. [*21*] The preceding exercise needs a subroutine to determine the endpoints of all simple paths of lengths $1, 2, \ldots, L$ from a given vertex v in a given graph. That problem is NP-hard; but sketch an algorithm that works well for small L in small graphs.

406. [*16*] Show that the following hidato puzzle isn't as hard as it might look at first:

19	52	53	54	4	62	63	64
20							1
21							60
41							59
31							58
32							9
33							10
35	34	37	28	27	26	11	12

▸ **407.** [*20*] Here's a curious 4×8 array that is consistent with 52 hidato solutions:

22							12
	29		26	16	8	3	

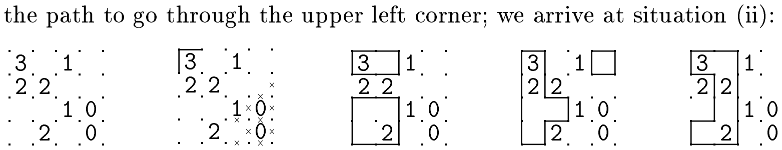
Change it to a valid hidato puzzle, by adding one more clue.

408. [*28*] (N. Beluhov.) Construct 6×6 hidato puzzles that have (a) only five clues; (b) at least eighteen clues, all of which are necessary.

▸ **409.** [*30*] Can the first 10 clues of a 10×10 hidato puzzle be the first 20 digits of π?

410. [*22*] (*Slitherlink.*) Another addictive class of puzzles is based on finding closed paths or "loops" in a given graph, when the allowable cycles must satisfy certain constraints. For instance, a slitherlink puzzle prescribes the *number of loop edges* that surround particular cells of a rectangular grid, as in diagram (i) below.

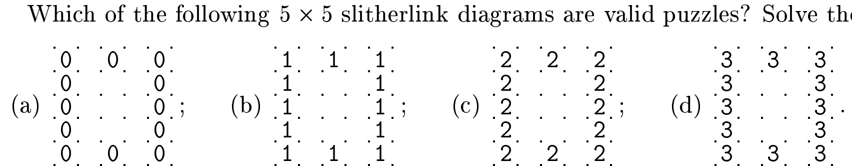
The first step in solving puzzle (i) is to note where the secret edges are definitely absent or definitely present. The 0s prohibit not only the edges immediately next to them but also a few more, because the path can't enter a dead end. Conversely, the 3 forces the path to go through the upper left corner; we arrive at situation (ii):



(i) (ii) (iii) (iv) (v)

Some experimentation now tells us which edge must go with the lower 1. We must not form two loops, as in (iii) or (iv). And hurrah: There's a unique solution, (v).

Which of the following 5×5 slitherlink diagrams are valid puzzles? Solve them.



411. [*20*] True or false: A slitherlink diagram with a numeric clue given in every cell always has at most one solution. *Hint:* Consider the 2×2 case.

▶ **412.** [*22*] A "weak solution" to a slitherlink diagram is a set of edges that obeys the numeric constraints, and touches every vertex of the grid either twice or not at all; but it may form arbitrarily many loops. For example, the diagram of exercise 410(i) has six weak solutions, three of which are shown in 410(iii), (iv), and (v).

Show that there's a nice way to obtain all the weak solutions of a given diagram, by formulating a suitable XCC problem. *Hint:* Think of the edges as constructed from tiles centered at the vertices, and use even/odd coordinates as in answer 133.

▶ **413.** [*30*] Explain how to modify Algorithm C so that the construction of exercise 412 will produce only the true "single-loop" solutions. Your modified algorithm shouldn't be specific to slitherlink; it should apply also to masyu and other loop-discovery puzzles.

414. [*25*] The "strongest possible" answer to exercise 413 would cause the modified Algorithm C to backtrack as soon as the current choice of edge colors is incompatible with any single loop. Show that the algorithm in that answer is *not* as strong as possible, by examining its behavior on the puzzle at the right.

1	0			
	2		3	
			3	0

▶ **415.** [*M33*] Exactly $5\cdot(2^{25}-1)$ nonempty slitherlink diagrams of size 5×5 are "homogeneous," in the sense that all of their clues involve the same digit $d\in\{0,1,2,3,4\}$. (See exercise 410(a)–(d).) How many of them are valid puzzles? What are the minimum and maximum number of clues, for each d, in puzzles that contain no redundant clues?

416. [*M30*] For each $d\in\{0,1,2,3,4\}$, construct valid $n\times n$ slitherlink diagrams whose nonblank clues are all equal to d, for infinitely many n.

417. [*M46*] (N. Beluhov, 2018.) Exercise 410(a, b, d) illustrates three homogeneous slitherlink puzzles that are valid for exactly the same pattern of nonblank clues. Do infinitely many such square puzzles exist?

418. [*M29*] An $m\times n$ slitherlink diagram is said to be *symmetrical* if cells (i,j) and $(m-1-i,\ n-1-j)$ are both blank or both nonblank, for $0\le i<m$ and $0\le j<n$. (Many grid-based puzzles obey this oft-unwritten rule.)

a) There are exactly $6^{25}\approx 2.8\times 10^{19}$ slitherlink diagrams of size 5×5, since each of the 25 cells can contain either '0', '1', '2', '3', '4', or ' '. How many of those diagrams are symmetrical?

b) How many of the symmetric diagrams in (a) are valid puzzles?

c) How many of those valid puzzles are *minimal*, in the sense that the deletion of nonblank clues in (i,j) and $(4-i,4-j)$ would make the solution nonunique?

d) What is the minimum number of clues in a valid 5×5 symmetrical puzzle?

e) What is the maximum number of clues in a minimal 5×5 symmetrical puzzle?

419. [*30*] What surprise is concealed in the following symmetrical slitherlink puzzle?

		2		1	1	1				1		1		1			
		2					0	1				0	1			1	2
			2	1				2		1		2		1	1		1
2		2				2	2	1		1		2					
3		0		0			2			1	2		0	0	0		0
				1			1		0			1		0			
						2	0		1		1	1	0				

				2	1	1		0		1	1						
			1		1			0		1			0				
0		0	2	1		1	0			1			0		0		0
					2		0		1	1	1				0		0
0		2	1		1		0		1				1	0			
0	1			1	1				1	1					0		
			1		1		0				1	2	1		0		

weak solution
two-factor, induced
XCC problem
tiles
even/odd coordinates
masyu
homogeneous
redundant clues
Beluhov
symmetrical puzzles
surprise

Masyu
XCC problem
minimal

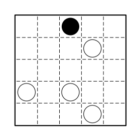
420. [*M22*] Consider an $m \times n$ slitherlink with m and n odd, having 2s in the pattern

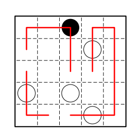
2 2 2 2 2 2 2
2 2 2 2 2 2 2
2 2 2 2 2 2 2

(and possibly other clues). Show that there's no solution if $m \bmod 4 = n \bmod 4 = 1$.

▸ **421.** [*20*] (*Masyu.*) A masyu ("evil influence") puzzle, like slitherlink, conceals a hidden loop of straight segments. But there are two important differences. First, the loop passes through the *centers* of grid cells, instead of following the edges. Second, no numerical quantities are involved; the clues are entirely visual and geometrical.

Clues appear in *circles* through which the loop must pass: (i) The path must *turn* 90° at every black circle; but it must travel *straight* through the two neighboring cells just before and after turning. (ii) The path must *not* turn 90° when it goes through a white circle; and it must *not* travel straight through the two neighboring cells just before and after not turning. (Thus it must actually turn, at one or both of those cells. We get at least one turn per clue, and at least one straight segment.)





Consider, for example, a 5×5 puzzle with a black clue in cell 02, and with white clues in cells 13, 30, 32, and 43 as shown. The loop clearly will have to include the subpaths 20 — 30 — 40 — 41 and 42 — 43 — 44 — 34 in some order. It also must include either 00 — 01 — 02 — 12 — 22 or 04 — 03 — 02 — 12 — 22, because of the black clue. But the latter alternative is impossible, because it leaves no way to go straight through the white clue in 13. Thus 10 — 00 — 01 — 02 — 12 — 22 is forced; and also 23 — 13 — 03 — 04 — 14 — 24 — 34. (We couldn't go 24 — 23, because that would close the loop prematurely.) The rest of the path now sort of falls into place.

Show that one of the clues in this example puzzle is actually redundant. But if any of the other four clues are absent, show that alternative solutions are possible.

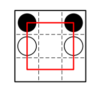
422. [*21*] Show that the "weak solutions" to any given masyu puzzle are the solutions to an easily constructed XCC problem, by adapting the solution of exercise 412.

▸ **423.** [*M25*] For each of the $(m-1)n + m(n-1)$ potential edges e in the solution of an $m \times n$ masyu puzzle, let x_e be the boolean variable '[e is present]'. The XCC problem constructed in exercise 422 is essentially a set of constraints on those variables.

Explain how to improve that construction dramatically, by exploiting the following special property that is enjoyed by masyu puzzles: Let N, S, E, and W be the edges leading out of a cell that holds a clue. If the clue is black, we have $N = {\sim}S$ and $E = {\sim}W$; if the clue is white, we have $N = S$, $E = W$, and $E = {\sim}N$. (Thus every clue reduces the number of independent variables by at least 2.)

▸ **424.** [*36*] Make an exhaustive study of 6×6 masyu, and gather whatever statistics you think are particularly interesting. For example, how many of the $3^{36} \approx 1.5 \times 10^{17}$ ways to place white or black clues lead to a valid puzzle? Which of the valid puzzles have the fewest clues? the most clues? the shortest loops? the longest loops? only white clues? only black clues? How many of those puzzles are *minimal*, in the sense that none of their clues can be removed without allowing a new solution?

How many of the $2^{36} \approx 6.9 \times 10^{10}$ ways to occupy cells occur as the pattern of white clues in a valid puzzle? How many of them occur as the pattern of black clues? How many puzzles remain valid when white and black are interchanged? Which 6×6 masyu puzzle do you think is most difficult to solve?

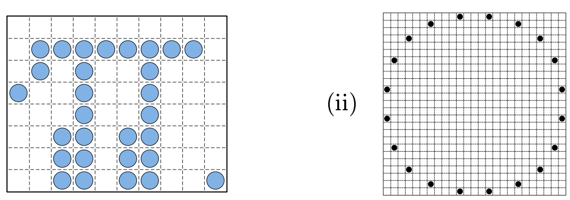


425. [*28*] The solution to a masyu puzzle is composed of five kinds of “tiles”: ‘ ’, ‘ ’, ‘ ’, ‘ ’, and blank. For example, the 3×3 solution shown here contains two tiles of each nonblank type.

Find 4×4, 5×5, and 6×6 puzzles whose unique solutions have exactly k tiles of each nonblank type, for every possible k.

▸ **426.** [*31*] Obtain a valid masyu puzzle from diagram (i) below by changing each ‘ ’ clue into either ‘ ’ or ‘ ’.

(i) (ii)

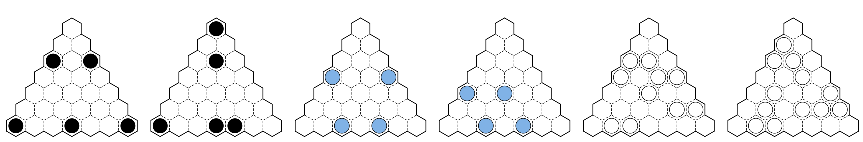


▸ **427.** [*25*] Design a 25×25 masyu puzzle by adding white clues (only) to diagram (ii) above. All of your clues should preserve the 8-fold symmetry of this pattern.

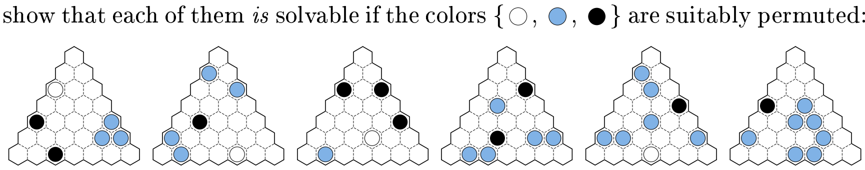
428. [*M28*] For infinitely many n, construct a valid $n \times n$ masyu puzzle with $O(n)$ clues whose loop goes through all four corner cells, where all clues are (a) black; (b) white.

429. [*21*] A closed path on a triangular grid may have “sharp turns,” which change the direction by $120°$, or “slack turns,” which change the direction by $60°$, or both. Therefore *triangular masyu* has three flavors of clues: ‘ ’ for the sharp turns, ‘ ’ for the slack turns, and of course ‘ ’ for the non-turns.

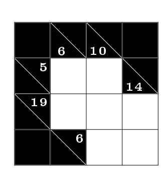
a) Solve the following homogeneous triangular masyu puzzles:



b) The following patterns for triangular masyu are clearly impossible to solve. But show that each of them *is* solvable if the colors { , , } are suitably permuted:



▸ **430.** [*26*] (*Kakuro.*) A kakuro puzzle is like a crossword puzzle, except that its “words” are blocks of two or more nonzero digits $\{1, 2, \ldots, 9\}$, not strings of letters. The digits of each block must be distinct, and their *sum* is given as a clue. Every cell to be filled belongs to exactly one horizontal block and one vertical block.



For example, the mini-kakuro shown here has just three horizontal blocks and three vertical blocks. Notice that the desired sums are indicated to the immediate left or above each block; thus the first horizontal block is supposed to be filled with two digits that sum to 5, so there are four possibilities: 14, 23, 32, 41. The first vertical block should sum to 6; again there are four possibilities, this time 15, 24, 42, 51 (because 33 is forbidden). The second horizontal block has three digits that should sum to 19; it is considerably less constrained. Indeed, there are thirty ways to obtain 19-in-three, namely the permutations of $\{2, 8, 9\}$ or $\{3, 7, 9\}$ or $\{4, 6, 9\}$ or $\{4, 7, 8\}$ or $\{5, 6, 8\}$.

pi, as random
8-fold symmetry
triangular grid
sharp turns
slack turns
triangular masyu
pi, as random
homogeneous
Kakuro
crossword puzzle
blocks
ordered partitions into distinct parts

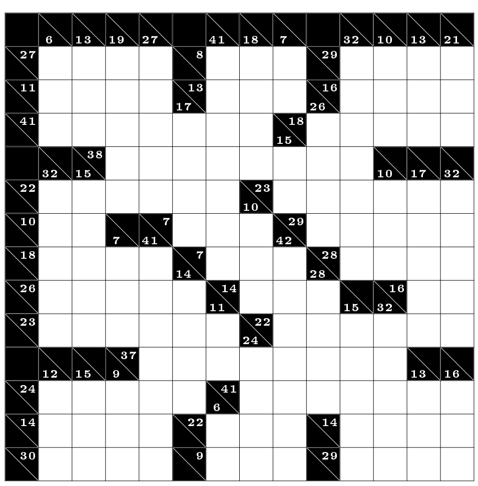
a) Solve the puzzle. *Hint:* There's only one possibility for the lower right corner.

b) Sketch a simple way to build a table of all suitable combinations of n-in-k, for $2 \le k \le 9$ and $2 \le n \le 45$. Which n and k have the most? *Hint:* Use bitmaps.

c) *Generalized kakuro* is a related puzzle, for which each block of length k has a specified set of combinations, chosen from among the $\binom{9}{k}$ possibilities (regardless of their sum). For example, suppose the three horizontal blocks of mini-kakuro must be filled respectively with permutations of $\{1,3\}$, $\{3,5\}$, or $\{5,7\}$; $\{1,3,5\}$, $\{1,7,9\}$, $\{2,4,6\}$, $\{6,8,9\}$, or $\{7,8,9\}$; $\{2,4\}$, $\{4,6\}$, or $\{6,8\}$; and require the same for the three vertical blocks. Find the unique solution to that puzzle.

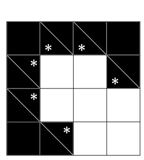
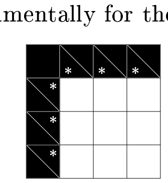
d) It would be easy to formulate kakuro as an XCC problem, as we did word squares in exercise 87, by simply giving one option for each possible placement of a block. But the resulting problem might be gigantic: For example, long blocks are not uncommon in kakuro, and each 9-digit block would have $9! = 362{,}880$ options(!). Show that generalized kakuro *can* be formulated efficiently as an XCC problem.

▸ **431.** [*30*] The inventor of kakuro, Jacob E. Funk of Manitoba (who always called his puzzles "Cross Sums"), published the following challenge on pages 50 and 66 of the August/September 1950 issue of *Dell Official Crossword Puzzles*:



Many ingenious constructions are present here; but unfortunately, he failed to realize that there is more than one solution. Find all solutions, and obtain a valid puzzle by repairing some of his original clues.

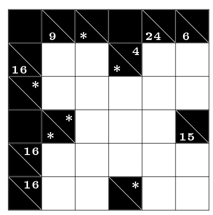
▸ **432.** [*M25*] We can't simply design new kakuro puzzles by randomly filling the blanks and using the resulting sums as the constraints, because the vast majority of feasible sums yield nonunique solutions. Verify this experimentally for the generic diagrams

a)  ; b)  .

In each case determine the exact number of ways to fill the blanks, without repeated digits in any row or column, as well as exactly how many of those filled-in diagrams are uniquely reconstructible from their block sums. Consider also symmetry.

combinations
bitmaps
Generalized kakuro
word squares
efficient: reasonably fast
Funk
Cross Sums

433. [*26*] Six of the sum-clues in this little kakuro diagram are unspecified:



In how many ways can you obtain valid puzzles by specifying them?

434. [*30*] Exactly how many kakuro diagrams are possible in a 9×9 grid? (Every row and every column should contain at least one block of empty cells, except that the topmost row and leftmost column are completely black. All blocks must have length ≥ 2. Empty cells needn't be rookwise connected.) What is the maximum number of blocks?

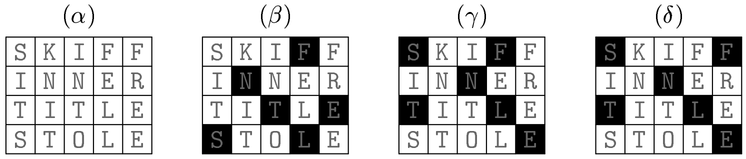
435. [*31*] Design a rectangular kakuro puzzle for which the blocks at the top of the solution are 31, 41, 59, 26, 53, 58, 97 (the first fourteen digits of π).

▸ **436.** [*20*] (*Hitori.*) Let's wind up this potpourri of examples by considering a completely different combinatorial challenge. A hitori puzzle ("alone") is an $m \times n$ array in which we're supposed to cross elements out until three conditions are achieved:

i) No row or column contains repeated elements.
ii) Adjacent elements cannot be crossed out.
iii) The remaining elements are rookwise connected.

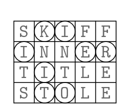
For example, consider the 4×5 word rectangle (α). Conditions (i) and (ii) can be satisfied in sixteen ways, such as (β) and (γ). But only (δ) satisfies also (iii).

(α) (β) (γ) (δ)



A crossed-out cell is said to be black; the other cells are white. While solving a hitori, it's helpful to *circle* an entry that is certain to become white. We can initially circle all the "seeds" — the entries that don't match any others in their row or column.

For example, puzzle (α) has eight seeds. If we decide to blacken a cell, we immediately circle its neighbors (because they cannot also be black). Thus, for instance, we shouldn't cross out the `E` in cell $(2,4)$: That would circle the `L` in $(2,3)$, forcing the other `L` to be black and cutting off the corner `E` as in (β).



The precise value of a seed is immaterial to the puzzle; it can be replaced by any other symbol that differs from everything else in its row or column.

We say as usual that a hitori puzzle is *valid* if it has exactly one solution. Explain why (a) a valid hitori puzzle has exactly one solution with all seeds white; (b) a hitori puzzle that has a unique solution with all seeds white is valid if and only if all the seed cells *not* adjacent to black in that solution are "articulation points" for the set of white cells — that is, their removal would disconnect the whites. (See $(3,1)$ and $(3,2)$ in (δ).)

▸ **437.** [*21*] A *weak solution* to a hitori puzzle is a solution for which all seeds are white, and for which properties (i) and (ii) of exercise 436 hold. Given a hitori puzzle, define an XCC problem whose solutions are precisely its weak solutions.

π
Hitori
word rectangle
seeds
articulation points
weak solution

438. [*30*] Explain how to modify Algorithm C so that, when given an XCC problem from the construction in answer 437, it will produce only solutions that satisfy also the connectivity condition (iii). *Hint:* See exercise 413; also consider reachability.

439. [*M20*] Let G be a graph on the vertices V. A *hitori cover* of G is a set $U \subseteq V$ such that (i) $G\,|\,U$ is connected; (ii) if $v \notin U$ and $u \mathrel{-\!\!-} v$ then $u \in U$; (iii) if $u \in U$ and if $v \in U$ for all $u \mathrel{-\!\!-} v$, then $G\,|\,(U \setminus u)$ is not connected.

a) Describe a hitori cover in terms of standard graph theory terminology.

b) Show that the solution of a valid hitori puzzle is a hitori cover of $P_m \mathbin{\square} P_n$.

440. [*21*] True or false: If the letter `A` occurs exactly twice in the top row of a valid hitori puzzle, exactly one of those occurrences will survive in the solution.

441. [*18*] Describe every valid hitori puzzle of size $1 \times n$ on a d-letter alphabet.

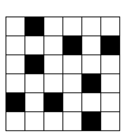
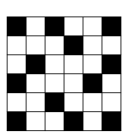
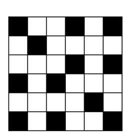
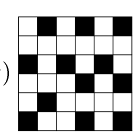
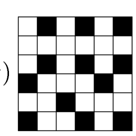
▸ **442.** [*M33*] Enumerate all hitori covers of $P_m \mathbin{\square} P_n$, for $1 \le m \le n \le 9$.

▸ **443.** [*M30*] Prove that an $m \times n$ hitori cover has at most $(mn + 2)/3$ black cells.

444. [*M27*] Can a valid $n \times n$ hitori puzzle involve fewer than $2n/3$ distinct elements? Construct a valid puzzle of size $3k \times 3k$, using only the elements $\{0, 1, \ldots, 2k\}$.

▸ **445.** [*M22*] It's surprisingly difficult to construct a valid hitori puzzle that has no seeds. In fact, there are no $n \times n$ examples for $n \le 9$ except when $n = 6$. But it turns out that quite a few seedless 6×6 hitori puzzles do exist.

Consider the five hitori covers below. Determine, for each of them, the exact number of valid hitori puzzles with no seeds, having that pattern of white and black cells as the solution. *Hint:* In some cases the answer is zero.

(i)  (ii)  (iii)  (iv)  (v) 

▸ **446.** [*24*] The digits of e, 2.718281828459045..., are well known to have a curious repeating pattern. In fact, the first 25 digits actually define a valid 5×5 hitori puzzle! What is the probability that a random 5×5 array of decimal digits will have that property? And what about octal digits? Hexadecimal digits?

447. [*22*] (Johan de Ruiter.) Are there any values of $m > 1$ and $n > 1$ for which the first mn digits of π define a valid $m \times n$ hitori puzzle?

448. [*22*] Do any of the 31344 double word squares formed from `WORDS`(3000) make valid hitori puzzles? (See exercise 87.)

449. [*40*] (*Hidden nuggets.*) Johan de Ruiter noticed in 2017 that George Orwell had included a valid hitori puzzle in his novel *Nineteen Eighty-Four* (part 2, chapter 9):

B	E	I	N	G	I	N	A	M	I
N	O	R	I	T	Y	E	V	E	N
A	M	I	N	O	R	I	T	Y	O
F	O	N	E	D	I	D	N	O	T
M	A	K	E	Y	O	U	M	A	D

Did Homer, Shakespeare, Tolstoy, and others also create hitori puzzles accidentally?

450. [*22*] Use Algorithm X to solve the "tot tibi sunt dotes" problem of Section 7.2.1.7.

modify Algorithm C
reachability
hitori cover
seedless hitori puzzles
e
de Ruiter
π
double word squares
`WORDS`(3000)
de Ruiter
Orwell
Homer
Shakespeare
Tolstoy
tot tibi sunt dotes

Dr Pell was wont to say, that in the Resolution of Questiones,
the main matter is the well stating them:
which requires a good mother-witt & Logick: as well as Algebra:
for let the Question be but well-stated, and it will worke of it selfe:
. . . By this way, an man cannot intangle his notions, & make a false Steppe.
— JOHN AUBREY, *An Idea of Education of Young Gentlemen* (c. 1684)

Pell
AUBREY
Catalan number
topological sortings
partial order
degenerate
height
bipartite
hitting set problem
vertex cover problem
Secondary

SECTION 7.2.2.1

1. (a) Note first that Algorithm 6.2.2T has its own `LLINK` and `RLINK` fields, for left and right children; they shouldn't be confused with the links of the doubly linked list. After all deletions are done, `LLINK`(k) will be the largest search-tree ancestor of k that's less than k; `RLINK`(k) will be the smallest ancestor of k that's greater than k; but if there's no such ancestor, the link will be 0. (For example, in Fig. 10 of Section 6.2.2, `RLINK(LEO)` would be `PISCES` and `LLINK(AQUARIUS)` would be the list head.)

(b) There are $C_n = \binom{2n}{n}\frac{1}{n+1}$ classes (the Catalan number), one for each binary tree.

(c) The size of each class is the number of topological sortings of the partial order generated by the relations $k \prec$ `LLINK`(k), $k \prec$ `RLINK`(k). And this number equals 1 only in the 2^{n-1} "degenerate" trees of height n (see exercise 6.2.2–5).

2. (a) Let `L` and `R` denote the binary tree links. Operation (2) changes `RLINK`(k) only if we are undeleting j with `LLINK`$(j) = k$. If no such elements exist, we have `R`$(k) = \Lambda$, and `RLINK`(k) was never changed by any deletion. Otherwise those elements are $\{j_1, \ldots, j_t\}$, where `R`$(k) = j_1$, `L`$(j_i) = j_{i+1}$, and `L`$(j_t) = \Lambda$. Hence $j_t = k + 1$. Undeleting j_i will set `RLINK`$(j_{i-1}) \leftarrow j_i$, for $i = t, t-1, \ldots, 2$; this leaves `RLINK`$(k) = k + 1$. A similar argument works for `LLINK`, and for links involving the list head.

(Programmers are advised to use this amazing fact only with *great* care, because the lists are malformed during the process and fully reconstructed only at the end.)

(b) No. For example, delete 1, 2, 3; then undelete 1, 3, 2.

(c) Yes. The argument of (a) applies to each maximal interval of affected elements.

3. (a) $(x_1, \ldots, x_6) = (1, 0, 0, 1, 1, 0)$. (In general the solutions to linear equations won't always be 0 or 1. For example, the equations $x_1+x_2 = x_2+x_3 = x_1+x_3 = 1$ imply that $x_1 = x_2 = x_3 = \frac{1}{2}$; hence the corresponding exact cover problem is unsolvable.)

(b) In practice, m is much larger than n. Example (5) is just a "toy problem"! The best we can hope to achieve from n simultaneous equations is to express n of the variables in terms of the other $m - n$; that leaves 2^{m-n} cases to try.

4. If G is bipartite, the exact covers are the ways to choose the vertices of one part. (Hence there are 2^k solutions, if G has k components.) Otherwise there are no solutions. (Algorithm X will discover that fact quickly, although Algorithm 7B is faster.)

5. Given a hypergraph, find a set of vertices that hits each hyperedge exactly once. (In an ordinary graph this is the scenario of exercise 4.)

Similarly, the so-called hitting set problem is dual to the vertex cover problem.

6. The header nodes, numbered 1 through N, are followed by L ordinary nodes and $M+1$ spacers; hence the final node Z is number $L+M+N+1$. (There also are $N+1$ records for the horizontal list of items; those "records" aren't true "nodes.")

7. Node 23 is a spacer; '-4' indicates that it follows the 4th option. (Any nonpositive number would work, but this convention aids debugging.) Option 5 ends at node 25.

8. (*Secondary* items, which are introduced in the text after (24), are also handled by the steps below. Such items should occur *after* all of the primary items have been listed on the first line, and separated from them by some distinguishing mark.)

secondary items, list of active
spacer
nonsharp preference

I1. [Read the first line.] Set $N_1 \leftarrow -1$, $i \leftarrow 0$. Then, for each item name α on the first line, set $i \leftarrow i+1$, `NAME`$(i) \leftarrow \alpha$, `LLINK`$(i) \leftarrow i-1$, `RLINK`$(i-1) \leftarrow i$. If α names the first secondary item, also set $N_1 \leftarrow i-1$. (In practice α is limited to at most 8 characters, say. One should report an error if $\alpha =$ `NAME`(j) for some $j < i$.)

I2. [Finish the horizontal list.] Set $N \leftarrow i$. If $N_1 < 0$ (there were no secondary items), set $N_1 \leftarrow N$. Then set `LLINK`$(N+1) \leftarrow N$, `RLINK`$(N) \leftarrow N+1$, `LLINK`$(N_1+1) \leftarrow N+1$, `RLINK`$(N+1) \leftarrow N_1+1$, `LLINK`$(0) \leftarrow N_1$, `RLINK`$(N_1) \leftarrow 0$. (The active secondary items, if any, are accessible from record $N+1$.)

I3. [Prepare for options.] Set `LEN`$(i) \leftarrow 0$ and `ULINK`$(i) \leftarrow$ `DLINK`$(i) \leftarrow i$ for $1 \le i \le N$. (These are the header nodes for the N item lists, which are initially empty.) Then set $M \leftarrow 0$, $p \leftarrow N+1$, `TOP`$(p) \leftarrow 0$. (Node p is the first spacer.)

I4. [Read an option.] Terminate with $Z \leftarrow p$ if no input remains. Otherwise let the next line of input contain the item names $\alpha_1 \ldots \alpha_k$, and do the following for $1 \le j \le k$: Use an algorithm from Chapter 6 to find the index i_j for which `NAME`$(i_j) = \alpha_j$. (Report an error if unsuccessful. Complain also if an item name appears more than once in the same option, because a duplicate might make Algorithm X fail spectacularly.) Set `LEN`$(i_j) \leftarrow$ `LEN`$(i_j)+1$, $q \leftarrow$ `ULINK`(i_j), `ULINK`$(p+j) \leftarrow q$, `DLINK`$(q) \leftarrow p+j$, `DLINK`$(p+j) \leftarrow i_j$, `ULINK`$(i_j) \leftarrow p+j$, `TOP`$(p+j) \leftarrow i_j$.

I5. [Finish an option.] Set $M \leftarrow M+1$, `DLINK`$(p) \leftarrow p+k$, $p \leftarrow p+k+1$, `TOP`$(p) \leftarrow -M$, `ULINK`$(p) \leftarrow p-k$, and return to step I4. (Node p is the next spacer.) ▮

9. Set $\theta \leftarrow \infty$, $p \leftarrow$ `RLINK`(0). While $p \ne 0$, do the following: Set $\lambda \leftarrow$ `LEN`(p); if $\lambda<\theta$ set $\theta \leftarrow \lambda$, $i \leftarrow p$; and set $p \leftarrow$ `RLINK`(p). (We could exit the loop immediately if $\theta = 0$.)

10. If `LEN`$(p) > 1$ and `NAME`(p) doesn't begin with '#', set $\lambda \leftarrow M +$ `LEN`(p) instead of `LEN`(p). (Similarly, the "nonsharp preference" heuristic favors nonsharp items.)

11. Item a is selected at level 0, trying option $x_0 = 12$, '$a\ d\ g$', and leading to (7). Then item b is selected at level 1, trying $x_1 = 16$, '$b\ c\ f$'. Hence, when the remaining item e is selected at level 2, it has no options in its list, and backtracking becomes necessary. Here are the current memory contents — substantially changed from Table 1:

i:	0	1	2	3	4	5	6	7
`NAME`(i):	—	a	b	c	d	e	f	g
`LLINK`(i):	0	0	0	0	3	0	5	6
`RLINK`(i):	0	2	3	5	5	0	0	0
x:	0	1	2	3	4	5	6	7
`LEN`(x):	—	2	1	1	1	0	0	1
`ULINK`(x):	—	20	16	9	27	5	6	25
`DLINK`(x):	—	12	16	9	27	5	6	25
x:	8	9	10	11	12	13	14	15
`TOP`(x):	0	3	5	-1	1	4	7	-2
`ULINK`(x):	—	3	5	9	1	4	7	12
`DLINK`(x):	10	3	5	14	20	21	25	18
x:	16	17	18	19	20	21	22	23
`TOP`(x):	2	3	6	-3	1	4	6	-4
`ULINK`(x):	2	9	6	16	12	4	18	20
`DLINK`(x):	2	3	6	22	1	27	6	25
x:	24	25	26	27	28	29	30	
`TOP`(x):	2	7	-5	4	5	7	-6	
`ULINK`(x):	16	7	24	4	10	25	27	
`DLINK`(x):	2	7	29	4	5	7	—	

12. Report that x is out of range if $x \le N$ or $x > Z$ or $\mathtt{TOP}(x) \le 0$. Otherwise set $q \leftarrow x$ and do "print $\mathtt{NAME(TOP}(q))$ and set $q \leftarrow q+1$; if $\mathtt{TOP}(q) \le 0$ set $q \leftarrow \mathtt{ULINK}(q)$" until $q = x$. Then set $i \leftarrow \mathtt{TOP}(x)$, $q \leftarrow \mathtt{DLINK}(i)$, and $k \leftarrow 1$. While $q \ne x$ and $q \ne i$, set $q \leftarrow \mathtt{DLINK}(q)$ and $k \leftarrow k+1$. If $q \ne i$, report that the option containing x is 'k of $\mathtt{LEN}(i)$' in item i's list; otherwise report that it's not in that list.

13. For $0 \le j < l$, node x_j is part of an option in the solution. By setting $r \leftarrow x_j$ and then $r \leftarrow r+1$ until $\mathtt{TOP}(r) < 0$, we'll know exactly what that option is: It's option number $-\mathtt{TOP}(r)$, which begins at node $\mathtt{ULINK}(r)$. (Many applications of Algorithm X have a custom-made output routine, to convert $x_0 \ldots x_{l-1}$ into an appropriate format — presenting it directly as a sudoku solution or a box packing, etc.)

Exercise 12 explains how to provide further information, not only identifying the option of x_j but also showing its position in the search tree.

14. (a) The options are '$S_j\, M_k$', for all $0 \le j, k < n$ except $j = k$ or $j = (k+1) \bmod n$.

(b) There are $(u_3, \ldots, u_{10}) = (1, 2, 13, 80, 579, 4738, 43387, 439792)$ solutions. The running time for $n = 10$ is about 180 (or 275) mems per solution with (or without) MRV.

[This problem has a rich history: E. Lucas presented and named it in his *Théorie des Nombres* (1891), 215, 491–495. An equivalent problem had, however, already been posed by P. G. Tait, and solved by A. Cayley and T. Muir; see *Trans. Royal Soc. Edinburgh* **28** (1877), 159, and *Proc. Royal Soc. Edinburgh* **9** (1878), 338–342, 382–391, **11** (1880), 187–190. In particular, Muir found the recurrence relation

$$(n-1)u_{n+1} = (n^2-1)u_n + (n+1)u_{n-1} + (-1)^n \cdot 4, \quad \text{for } n > 1.$$

Clearly $u_2 = 0$; a careful consideration of initial values shows that the choices $u_0 = 1$ and $u_1 = -1$ give mathematically clean expressions, such as the explicit formula

$$u_n = \sum_{k=0}^{n} (-1)^k \frac{2n}{2n-k} \binom{2n-k}{k} (n-k)!.$$

(See J. Touchard, *Comptes Rendus Acad. Sci.* **198** (Paris, 1934), 631–633; I. Kaplansky, *Bull. Amer. Math. Soc.* **49** (1943), 784–785.) The kth term of this formula can also be written $n! \sum_j (-1)^{j+k} 2^{k-2j} / ((k-2j)!\, j!\, (n-1)^{\underline{j}})$; hence we have the curious identity

$$\frac{u_n}{n!} = \sum_{j=0}^{n/2} \frac{(-1)^j}{j!} \frac{T_{n-2j}}{(n-1)^{\underline{j}}} = T_n - \frac{T_{n-1}}{n-1} + \frac{T_{n-2}/2!}{(n-1)(n-2)} - \frac{T_{n-3}/3!}{(n-1)(n-2)(n-3)} + \cdots,$$

where $T_n = \sum_{k=0}^{n} (-2)^k/k!$ is the sum of the first $n+1$ terms of the power series for e^{-2}. The ménage numbers therefore satisfy the interesting asymptotic formula

$$u_n = \frac{n!}{e^2}\Big(1 - \frac{1}{n-1} + \frac{1/2!}{(n-1)(n-2)} + \cdots + \frac{(-1)^k/k!}{(n-1)\ldots(n-k)} + O(n^{-k-1})\Big)$$

for all fixed $k \ge 0$, discovered by I. Kaplansky and J. Riordan (*Scripta Mathematica* **12** (1946), 113–124). In fact, M. Wyman and L. Moser proved that the sum of this series for $0 \le k < n$ differs from u_n by less than 1/2 (*Canadian J. Math.* **10** (1958), 468–480). Among many other things, they also found a (complicated) expression for the exponential generating function $\sum_n u_n z^n/n!$. The *ordinary* generating function $\sum_n u_n z^n$ has the surprisingly nice form $((1-z)/(1+z))F(z/(1+z)^2)$, where $F(z) = \sum_{n\ge 0} n!\, z^n$; see P. Flajolet and R. Sedgewick, *Analytic Combinatorics* (2009), 368–372.]

Z
N
sudoku
Historical remarks
Lucas
Tait
Cayley
Muir
recurrence relation
Touchard
Kaplansky
asymptotic formula
Riordan
Wyman
Moser
exponential generating function
generating function
factorial generating function
Flajolet
Sedgewick

MRV heuristic
secondary items, list of active
second death
bitwise AND
author
combination generation

15. Omit the options with $i = n - [n \text{ even}]$ and $j > n/2$.

(Other solutions are possible. For example, we could omit the options with $i = 1$ and $j \ge n$; that would omit $n-1$ options instead of only $\lfloor n/2 \rfloor$. However, the suggested rule turns out to make Algorithm X run about 10% faster.)

16. The two solutions are 'r_1 c_2 a_3 b_{-1}' 'r_2 c_4 a_6 b_{-2}' 'r_3 c_1 a_4 b_2' 'r_4 c_3 a_7 b_1' 'a_2' 'a_5' 'a_8' 'b_{-3}' 'b_0' 'b_3'; 'r_1 c_3 a_4 b_{-2}' 'r_2 c_1 a_3 b_1' 'r_3 c_4 a_7 b_{-1}' 'r_4 c_2 a_6 b_2' 'a_2' 'a_5' 'a_8' 'b_{-3}' 'b_0' 'b_3'. At the top levels, the MRV heuristic causes Algorithm X to branch first on the slack variables a_2, a_8, b_{-3}, and b_3, which each have at most two possibilities. (And that's actually a pretty strange way to tackle the four queens problem!)

17. Branch first on r_3, which has four options. If 'r_3 c_1 a_4 b_2', there's just one option for c_2, then c_3, then r_2, so we get the first solution: 'r_3 c_1 a_4 b_2' 'r_1 c_2 a_3 b_{-1}' 'r_4 c_3 a_7 b_1' 'r_2 c_4 a_6 b_{-2}'. If 'r_3 c_2 a_5 b_1', c_3 is forced, then r_2 can't be covered. If 'r_3 c_3 a_6 b_0', r_2 is forced, then c_2 can't be covered. If 'r_3 c_4 a_7 b_{-1}', we cruise to the second solution: 'r_3 c_4 a_7 b_{-1}' 'r_1 c_3 a_4 b_{-2}' 'r_2 c_1 a_3 b_1' 'r_4 c_2 a_6 b_2'. (And that's a good way.)

18. 'c e' 'a d f' 'b g' (as before) and 'b c f' 'a d g' (new).

19. When all primary items have been covered in step X2, accept a solution only if `LEN`$(i) = 0$ for all of the active secondary items, namely the items accessible from `RLINK`$(N+1)$. [This algorithm is called the "second death" method, because it checks that all of the purely secondary options have been killed off by primary covering.]

20. For $1 \le k < m$, set $t \leftarrow k \,\&\, (-k)$; include secondary item y_k in option α_j for $k \le j < \min(m, k+t)$ and in option β_j for $k - t \le j < k$.

Equivalently, to set up option α_j, include a and set $t \leftarrow j$; while $t > 0$, include y_t and set $t \leftarrow t \,\&\, (t-1)$. To set up option β_j, include b and set $t \leftarrow -1-j$; while $t > -m$, include y_{-t} and set $t \leftarrow t \,\&\, (t-1)$.

If $j > k$, options α_j and β_k both contain $y_{j \& -2^{\lfloor \lg(j-k) \rfloor}}$.

21. The options α_j^i will contain the primary item a_i. Simply do $k-1$ pairwise orderings, with secondary items y_k^i to ensure that $j_k \le j_{k+1}$. If m is a power of 2, it turns out that the options for $1 < i < k$ each have exactly $\lg m$ secondary items. For example, if $m = 4$ and $k > 2$, the options α_j^2 are 'a_2 y_1^1 y_2^1', 'a_2 y_2^1 y_1^2', 'a_2 y_3^1 y_2^2', 'a_2 y_3^2 y_2^2'.

(The author attempted to knock out options for $\alpha^{i'}$ with $i' < i-1$ or $i' > i+1$, by adding additional secondary items, but that turned out to be a bad idea.)

Of course, this method doesn't compete with the lightning-quick methods for combination generation in Section 7.2.1.3; for instance, when $m = 20$ and $k = 8$ it needs 1.1 Gμ to crank out the $\binom{27}{8} = 2220075$ coverings, about 500 mems per solution.

22. (a) Let $n' = \lfloor n/2 \rfloor + 1$. By rotation/reflection we can assume that the queen in column n' (the middle column) is in row i and the queen in row n' is in column j, where $1 \le i < j < n'$. We obtain a suitable exact cover problem by leaving out the options $o(i,j) =$ 'r_i c_j a_{i+j} b_{i-j}' for $i = j$ or $i + j = n + 1$; also omit $o(i,j)$ for $i > j$ when $j = n'$; $j > i$ when $i = n'$; and $(i,j) = (n'-1, n')$ or $(n', 1)$. Then include secondary items to force the pairwise ordering of $\alpha_k = o(k+1, n')$ and $\beta_k = o(n', k+2)$, for $0 \le k < m = n' - 2$.

(b) Now we assume a queen in (j,j), where $1 \le j < n'$, and that the queen in row n is closer to the bottom right corner than the queen in column n. So we omit options $o(i,j)$ for $i+j = n+1$ or $i = j \ge n'$ or $(i,j) = (n,2)$ or $(i,j) = (n-1,n)$; we make item b_0 primary; and we let $\alpha_k = o(n, n-k-1)$, $\beta_k = o(n-k-2, n)$ for $0 \le k < m = n-3$.

(c) This time we want queens in (i,i) and $(j, n+1-j)$ where $1 \le i < j < n'$. We promote a_{n+1} and b_0 to primary; omit $o(i,j)$ when $i = j \ge n'-1$ or $i = n+1-j \ge n'$ or $(i,j) = (1,n)$; and let $\alpha_k = o(k+1, k+1)$, $\beta_k = o(k+2, n-k-1)$ for $0 \le k < m = n'-2$.

In case (a) there are (0, 0, 1, 8, 260, 9709, 371590) solutions for $n = (5, 7, \ldots, 17)$; Algorithm X handles $n = 17$ in 3.4 Gμ. [In case (b) there are (0, 0, 1, 4, 14, 21, 109, 500, 2453, 14498, 89639, 568849) for $n = (5, 6, \ldots, 16)$; and $n = 16$ costs 6.0 Gμ. In case (c), similarly, there are (1, 0, 3, 6, 24, 68, 191, 1180, 5944, 29761, 171778, 1220908) solutions; $n = 16$ costs 5.5 Gμ.]

pairwise comparisons
author
asymmetric
singly symmetric
doubly symmetric

23. (a) Consider the queens in column a of row 1, row b of column n, column $\bar{c}$ of row n, and row $\bar{d}$ of column 1, where $\bar{x} = n + 1 - x$. (These four queens are distinct, because no queen is in a corner. Notice also that neither $\bar{a}$ nor $\bar{b}$ nor $\bar{c}$ nor $\bar{d}$ can equal a.) Repeated rotations and/or reflections will change these numbers from (a, b, c, d) to

$$(b,c,d,a),\ (c,d,a,b),\ (d,a,b,c),\ (\bar{d},\bar{c},\bar{b},\bar{a}),\ (\bar{c},\bar{b},\bar{a},\bar{d}),\ (\bar{b},\bar{a},\bar{d},\bar{c}),\ (\bar{a},\bar{d},\bar{c},\bar{b}).$$

Those eight 4-tuples are usually distinct, and in such cases we can save a factor of 8 by eliminating all but one of them. There always is a solution with $a \le b, c, d < \bar{a}$; and those inequalities can be enforced by doing three simultaneous pairwise comparisons, between the options for row 1 and the respective options for column n, row n, and column 1. For example, the options that correspond to $a = 1$ when $n = 16$ are 'r_1 c_2 a_3 b_{-1}'; 'r_2 c_{16} a_{18} b_{-14} x_1 x_2 x_4'; 'r_{15} c_{16} a_{31} b_{-1} x_1 x_2 x_4'; 'r_{16} c_2 a_{18} b_{14} y_1 y_2 y_4'; 'r_{16} c_{14} a_{30} b_2 y_1 y_2 y_4'; 'r_2 c_1 a_3 b_1 z_1 z_2 z_4'; 'r_{15} c_1 a_{16} b_{14} z_1 z_2 z_4'. (Here $m = n/2 - 1 = 7$.)

With this change, the number of solutions for $n = 16$ drops from 454376 to 64374 (ratio ≈ 7.06), and the running time drops from 4.3 Gμ to 1.2 Gμ (ratio ≈ 3.68).

[The author experimented with further restrictions, so that solutions were allowed only if (i) $a < b, c, d$; (ii) $a = b < c, d$; (iii) $a = b = c < d$; (iv) $a = b = c = d$; (v) $a = c < b, d$. Five options were given for each value of $a < n/2 - 1$, and m was 6 instead of 7. The number of solutions decreased to 59648; but the running time increased to 1.9 Gμ. Thus a point of diminishing returns had been reached. (A completely canonical reduction would have produced 57188 solutions, with considerable difficulty.)]

(b) This case is almost identical to (a), because the queen in the center vacates all other diagonal cells. Requiring $a \le b, c, d < \bar{a}$ reduces the number of solutions for $n = 17$ from 4067152 to 577732 (ratio ≈ 7.04), and run time to 3.2 Gμ (ratio ≈ 4.50).

24. We simply combine compatible options into (a) pairs, (b) quadruplets, and force a queen in the center when n is odd. For example, when $n = 4$ we replace (23) by (a) 'r_1 c_2 a_3 b_{-1} r_4 c_3 a_7 b_1'; 'r_1 c_3 a_4 b_{-2} r_4 c_2 a_6 b_2'; 'r_2 c_1 a_3 b_1 r_3 c_4 a_7 b_{-1}'; 'r_2 c_4 a_6 b_{-2} r_3 c_1 a_4 b_2'; (b) 'r_1 c_2 a_3 b_{-1} r_2 c_4 a_6 b_{-2} r_4 c_3 a_7 b_1 r_3 c_1 a_4 b_2'; 'r_2 c_1 a_3 b_1 r_3 c_4 a_7 b_{-1} r_1 c_3 a_4 b_{-2} r_4 c_2 a_6 b_2'. The options when $n = 5$ are (a) 'r_1 c_2 a_3 b_{-1} r_5 c_4 a_9 b_1'; 'r_1 c_4 a_5 b_{-3} r_5 c_2 a_7 b_3'; 'r_2 c_1 a_3 b_1 r_4 c_5 a_9 b_{-1}'; 'r_2 c_5 a_7 b_{-3} r_4 c_1 a_5 b_3'; 'r_3 c_3 a_6 b_0'; (b) 'r_1 c_2 a_3 b_{-1} r_2 c_5 a_7 b_{-3} r_5 c_4 a_9 b_1 r_4 c_1 a_5 b_3'; 'r_2 c_1 a_3 b_1 r_1 c_4 a_5 b_{-3} r_4 c_5 a_9 b_{-1} r_5 c_2 a_7 b_3'; 'r_3 c_3 a_6 b_0'.

An n-queen solution is either *asymmetric* (changed by 180° rotation) or *singly symmetric* (changed by 90° rotation but not 180°) or *doubly symmetric* (unchanged by 90° rotation). Let $Q_{\rm a}(n)$, $Q_{\rm s}(n)$, $Q_{\rm d}(n)$ be the number of such solutions that are essentially different; then $Q(n) = 8Q_{\rm a}(n) + 4Q_{\rm s}(n) + 2Q_{\rm d}(n)$ when $n > 1$. Furthermore there are $4Q_{\rm s}(n) + 2Q_{\rm d}(n)$ solutions to (a) and $2Q_{\rm d}(n)$ solutions to (b). Hence we can determine the individual values just by counting solutions, and we obtain these results for small n:

$n =$	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$Q_{\rm a}(n) =$	0	1	0	4	11	42	89	329	1765	9197	45647	284743	1846189	11975869
$Q_{\rm s}(n) =$	0	0	1	2	1	4	3	12	18	32	105	310	734	2006
$Q_{\rm d}(n) =$	1	1	0	0	0	0	0	0	4	4	0	0	32	64

We can reduce the solutions to (a) by a factor of 2, by simply eliminating the options that contain $\{r_1, c_k\}$ for $k \ge \lceil n/2 \rceil$. We can reduce the solutions to (b) by a factor of $2^{\lfloor n/4\rfloor}$, by simply eliminating the options that contain $\{r_j, c_k\}$ for $j < \lceil n/2\rceil$ and $k \geq \lceil n/2\rceil$. With these simplifications, the computation of $Q_{\rm d}(16)$ needs only 70 Kμ; and then the computation of $Q_{\rm s}(16)$ needs only 5 Mμ. Only 20 Mμ are needed to determine that $Q_{\rm d}(32) = 2^7 \cdot 1589$.

eight queens problem
Gosset
Dudeney
gadget
RainBones puzzle
duplicate options

25. With 64 items, one for each cell of the chessboard, let there be 92 options, one for each of the 92 solutions to the eight queens problem (see Fig. 68). Every option names eight of the 64 items; so an 8-coloring is equivalent to solving this exact cover problem. Algorithm X needs only 25 kilomems and a 7-node search tree to show that such a mission is impossible. [In fact no *seven* solutions can be disjoint, because each solution touches at least three of the twenty cells 13, 14, 15, 16, 22, 27, 31, 38, 41, 48, 51, 58, 61, 68, 72, 77, 83, 84, 85, 86. See Thorold Gosset, *Messenger of Mathematics* **44** (1914), 48. However, Henry E. Dudeney found the illustrated way to occupy all but two cells, in *Tit-Bits* **32** (11 September 1897), 439; **33** (2 October 1897), 3.]

```
12345678
78563412
46718235
23854167
84236751
51672384
67481523
 512784
```

26. This is an exact cover problem with $92 + 312 + 396 + \cdots + 312 = 3284$ options (see exercise 7.2.2–6). Algorithm X needs about 32 megamems to find the solution shown, and about 1.3 Tμ to find all 11,092 of them.

```
07348652
18650437
75421860
26835071
34072186
52183704
80564213
61207345
```

27. Let u_{jh} and d_{jh} be secondary items for $1 \leq j \leq 2n$ and $1 \leq h \leq \lceil n/2\rceil$. Insert the gadget

$$u_{j1}\ u_{j2}\ \ldots\ u_{j\lceil i/2\rceil}\ u_{(j+1)\lceil i/2\rceil}\ \ldots\ u_{k\lceil i/2\rceil}\ \ldots\ u_{k2}\ u_{k1}$$

into each option (16); also append similar options, but with 'u' changed to 'd', except when $i = n$. [Solutions whose planar graph "splits" will be obtained more than once. One such example is 12 10 8 6 4 11 9 7 5 4 6 10 12 5 7 9 11 3 1 2 1 3 2.]

28. (a) Denoting that formula by $\rho(c_0, t_0; \ldots; c_l, t_l)$, notice that if $c'_j = t_j + 1 - c_j$ we have $\rho(c_0, t_0; \ldots; c_l, t_l) + \rho(c'_0, t_0; \ldots; c'_l, t_l) = 1$. Consequently the completion ratio is $1/2$ if and only if $c'_j = c_j$ for all j, namely when $t_j = 2c_j - 1$.

(b) The ratio $\rho(c_0, t_0; \ldots; c_l, t_l)$ never has an odd denominator, because $p/q + p'/q'$ has an even denominator whenever q and p' are odd and q' is even. But we can get arbitrarily close to $1/3$, since $\rho(2, 4; \ldots; 2, 4) = 1/3 + 1/(24 \cdot 4^l)$.

29. If T has only a root node, let there be one column, no rows. Otherwise let T have $d \geq 1$ subtrees $T_1, \ldots, T_d$, and assume that we've constructed matrices with rows R_j and columns C_j for each T_j. Let $C = C_1 \cup \cdots \cup C_d$. The matrix for T is obtained by appending three new columns $\{0, 1, 2\}$ and the following new rows: (i) '0 1 2 and all columns of $C \backslash C_j$', for $1 \leq j \leq d$; (ii) 'j and all columns of C', for $j \in \{1, 2\}$. The matrix for the example tree has 15 columns and 14 rows.

```
011111000000000
101111000000000
110111000000000
111010000000000
111100000000000
000000011111000
000000101111000
000000110111000
000000111010000
000000111100000
000000111111111
111111000000111
111111111111010
111111111111100
```

30. Yes, assuming that duplicate options are permitted. Use the previous construction, but change '$C \backslash C_j$' to 'C' if T_j is a solution node. (Without duplicate options, no two solution nodes can be siblings.)

31. (a) In step I4 of answer 8, insert $p + j$ into the rth position of the list for i_j, instead of at the bottom, where r is uniform between 1 and LEN(i_j).

(b) In answer 9, when $\lambda < \theta$ also set $r \leftarrow 1$; when $\lambda = \theta$, set $r \leftarrow r + 1$, and change $i \leftarrow p$ with probability $1/r$.

32. (a) No. Otherwise there would be an option with no primary items.

(b) Yes, but only if there are two options with the same primary items.

(c) Yes, but only if there are two options whose union is also an option, when restricted to primary items.

(d) The number of places, j, where $x = 1$ and $x' = 0$ must be the same as the number where $x = 0$ and $x' = 1$. For if A has exactly k primary items in every option, exactly jk primary items are being covered in different ways.

(e) Again distances must be even, because every solution also solves the restricted problem, which is uniform. (Consequently it makes sense to speak of the *semidistance* $d(x, x')/2$ between solutions of a quasi-uniform exact covering problem. The semidistance in a polyform packing problem is the number of pieces that are packed differently.)

33. (Solution by T. Matsui.) Add one new column at the left of A, all 0s. Then add two rows of length $n + 1$ at the bottom: $10\ldots0$ and $11\ldots1$. This $(m + 2) \times (n + 1)$ matrix A' has one solution that chooses only the last row. All other solutions choose the second-to-last row, together with rows that solve A.

34. (Solution by T. Matsui.) Assume that all 1s in column 1 appear in the first t rows, where $t > 3$. Add two new columns at the left, and two new rows $1100\ldots0$, $1010\ldots0$ of length $n + 2$ at the bottom. For $1 \le k \le t$, if row k was $1\alpha_k$, replace it by $010\alpha_k$ if $k \le t/2$, $011\alpha_k$ if $k > t/2$. Insert 00 at the left of the remaining rows $t + 1$ through m.

This construction can be repeated (with suitable row and column permutations) until no column sum exceeds 3. If the original column sums were $(c_1, \ldots, c_n)$, the new A' has $2T$ more rows and $2T$ more columns than A did, where $T = \sum_{j=1}^{n}(c_j \dot{-} 3)$.

One consequence is that the exact cover problem is NP-complete even when restricted to cases where all row and column sums are at most 3.

Notice, however, that this construction is *not* useful in practice, because it disguises the structure of A: It essentially *destroys* the minimum remaining values heuristic, because all columns whose sum is 2 look equally good to the solver!

35. Take a matrix with column sums $(c_1, \ldots, c_n)$, all ≤ 3, and extend it with three columns of 0s at the right. Then add the following four rows: $(x_1, \ldots, x_n, 0, 1, 1)$, $(y_1, \ldots, y_n, 1, 0, 1)$, $(z_1, \ldots, z_n, 1, 1, 0)$, and $(0, \ldots, 0, 1, 1, 1)$, where $x_j = [c_j < 3]$, $y_j = [c_j < 2]$, $z_j = [c_j < 1]$. The bottom row must be chosen in any solution.

36. The following modifications (which work also with Algorithm C) will find *all* solutions in lexicographic order; we can terminate early if we want only the first one.

Set $\mathtt{LL} \leftarrow 0$ in step X1. (We will use the MRV heuristic, but only on levels $> \mathtt{LL}$.)

If $\mathtt{RLINK(0)} = 0$ and $l = \mathtt{LL} + 1$ in step X2, visit the current solution as usual. Otherwise, however, set $\mathtt{LL} \leftarrow \mathtt{LL} + 1$ and do the following while $l > \mathtt{LL}$ (because the current solution was not found lexicographically): Set $l \leftarrow l - 1$, $i \leftarrow \mathtt{TOP}(x_l)$; uncover the items $\ne i$ in the option that contains x_l (as in X6); uncover i (as in X7).

In step X3, if $l = \mathtt{LL}$ simply set $i \leftarrow \mathtt{RLINK(0)}$. Otherwise use exercise 9, say.

If $l < \mathtt{LL}$ after setting $l \leftarrow l - 1$ in step X8, set $\mathtt{LL} \leftarrow l$.

To get the lexicographically smallest solution to the n queens problem, make sure that the first n items are $r_1, r_2, \ldots, r_n$. (The other primary items, c_j, can follow in *any* order.) The first solution for $n = 32$, found after 4.2 Gμ, has queens in columns 1, 3, 5, 2, 4, 9, 11, 13, 15, 6, 18, 24, 26, 30, 25, 31, 28, 32, 27, 29, 16, 19, 10, 8, 17, 12, 21, 7, 14, 23, 20, 22. (Without MRV the computation would have taken 35.6 Gμ.)

[The analogous problem for $n = 48$ is already quite difficult; that case was first solved by Wolfram Schubert. The best results currently known for large n have been obtained via sophisticated methods of integer programming: In November 2017, Matteo Fischetti and Domenico Salvagnin were the first to solve the case $n = 56$ and many larger cases, although $n = 62$ was still unsolved; see arXiv:1907.08246 [cs.DS] (2019), 14 pages. See also OEIS A141843 for the latest developments.]

semidistance
Matsui
Matsui
NP-complete
minimum remaining values heuristic
Schubert
integer programming
Fischetti
Salvagnin

37. (a) Let $a_{i,j}=0$ if $i\le 0$ or $j\le 0$; otherwise

$$a_{i,j}=\operatorname{mex}(\{a_{i,j-k}\mid k>0\}\cup\{a_{i-k,j}\mid k>0\}\cup\{a_{i-k,j-k}\mid k>0\}\cup\{a_{i+k,j-k}\mid k>0\})$$

where 'mex' is defined in exercise 7.1.3–8. It is not difficult to verify that $a_{i,q_i}=1$ and that each of the sequences $\langle a_{i,n}\rangle$, $\langle a_{n,j}\rangle$ for $n\ge 1$ is a permutation of the positive integers. (See OEIS A065188 and Alec Jones's A269526.)

(b) The following exercise gives strong empirical evidence for this conjecture. And in the *full* plane, the analogous spiral sequence *can* be analyzed: See F. M. Dekking, J. Shallit, and N. J. A. Sloane, *Electronic J. Combinatorics*, to appear.

38. The following method, inspired by Eq. 7.2.2–(6) and the previous exercise, uses binary vectors a, b, c, where c has both positive and negative subscripts.

G1. [Initialize.] Set $r\leftarrow 0$, $s\leftarrow 1$, $t\leftarrow 0$, $n\leftarrow 0$. (We've computed q_k for $1\le k\le n$.)

G2. [Try for $q_n\le n$.] (At this point $a_k=1$ for $1\le k<s$ and $a_s=0$; also $c_k=1$ for $-r<k\le t$ and $c_{-r}=c_{t+1}=0$; each vector contains n 1s.) Set $n\leftarrow n+1$, $k\leftarrow s$.

G3. [Found?] If $a_k=b_{k+n}=c_{k-n}=0$, go to G5. Otherwise set $k\leftarrow k+1$, and repeat this step if $k\le n-r$.

G4. [Make $q_n>n$.] Set $t\leftarrow t+1$, $q_n\leftarrow n+t$, $a_{n+t}\leftarrow b_{2n+t}\leftarrow c_t\leftarrow 1$, and return to G2.

G5. [Make $q_n\le n$.] Set $q_n\leftarrow k$, $a_k\leftarrow b_{k+n}\leftarrow c_{k-n}\leftarrow 1$. If $k=s$, set $s\leftarrow s+1$ repeatedly until $a_s=0$. If $k=n-r$, set $r\leftarrow r+1$ repeatedly until $c_{-r}=0$. Return to G2. ▌

In step G2 we have $s\approx n-r\approx t\approx n/\phi$; hence the running time is extremely short. Empirically, in fact, the calculation of q_n requires at most 19 accesses to the bit vectors (averaging about 5.726 accesses), for each n. Agreement with exercise 37 is very close:

$$(q_{999999997},\ldots,q_{1000000004})=(618033989, 1618033985, 618033988,\\ 1618033988, 1618033990, 1618033992, 1618033994, 618033991).$$

Moreover, it's likely that $q_n\in[n/\phi-3\,.\,.\,n/\phi+5]\cup[n\phi-2\,.\,.\,n\phi+1]$ for all n.

39. (a) With probability $(1-p)^n$, no items will be selected; in such cases we must restart the clause generator, because options can't be empty. Ten random trials with $m=500$, $n=100$, and $p=.05$ gave respectively (444, 51, 138, 29, 0, 227, 26, 108, 2, 84) solutions, costing about 100 megamems per solution.

Although the exercise did not call for a mathematical analysis, we can derive a formula for the expected number of solutions by computing the probability that a given subset of the options is an exact cover, then summing over all subsets. If the subset has k items, and if each item in each option were present with probability p, this probability would be $(kp(1-p)^{k-1})^n$. However, we've excluded empty options; the true probability $f(n,p,k)$ turns out to be $k!{n\brace k}(p(1-p)^{k-1})^n/(1-(1-p)^n)^k$. The sum $\sum_k\binom{m}{k}f(n,p,k)$, when $(m,n,p)=(500,100,.05)$, is approximately 3736.96 with the incorrect formula and 297.041 with the correct one.

[In unpublished notes, Robin Pemantle and Boris Pittel have independently derived asymptotic results for $m=\alpha n$ and $p=r/n$, for fixed α and r as $n\to\infty$. The behavior of Algorithm X with this random model is not easy to analyze, but an analysis may be within reach because of the recursive structure.]

(b) This case has completely different behavior. In the first place, n must obviously be a multiple of r. In the second place, we'll need more options to get even one solution when $n=100$ and $r=5$, because conveniently small options don't exist.

minimal excludant
Jones
OEIS: The Online Encyc...
Dekking
Shallit
Sloane
Stirling subset numbers
Pemantle
Pittel

Proof: The total number of set partitions into twenty subsets of size 5 is $P = 100!/(20! \cdot 5!^{20}) \approx 10^{98}$; the total number of possible options is $N = \binom{100}{5} = 75287520$. The probability that any particular set partition occurs as a solution is the probability that twenty given options occur in a random sample of m, with replacement, namely $g(N, m, 20) = \sum_k \binom{20}{k}(-1)^k(N-k)^m/N^m = \sum_t {m \brace t} t! \binom{N-20}{t-20}/N^m$. If m isn't extremely large, this is almost the same as the probability without replacement, namely $\binom{N-20}{m-20}/\binom{N}{m} \approx (m/N)^{20}$. The expected number of solutions when $m = (500, 1000, 1500)$, respectively, is $P\,g(N, m, 20) \approx (.000002, 2.41, 8500)$.

40. Set $f_m \leftarrow 0$ and $f_{k-1} \leftarrow f_k \mid r_k$ for $m \ge k > 1$. The bits of u_k represent items that are being changed for the last time.

Let $u_k = u' + u''$, where $u' = u_k \,\&\, p$. If $u_k \ne 0$ at the beginning of step N4, we compress the database as follows: For $N \ge j \ge 1$, if $s_j \,\&\, u' \ne u'$, delete (s_j, c_j); otherwise if $s_j \,\&\, u'' \ne 0$, delete (s_j, c_j) and insert $((s_j \,\&\, \bar{u}_k) \mid u', c_j)$.

To delete (s_j, c_j), set $(s_j, c_j) \leftarrow (s_N, c_N)$ and $N \leftarrow N - 1$.

When this improved algorithm terminates in step N2, we always have $N \le 1$. Furthermore, if we let $p_k = r_1 \mid \cdots \mid r_{k-1}$, the size of N never exceeds 2^{ν_k}, where $\nu_k = \nu\langle p_k r_k f_k\rangle$ is the size of the “frontier” (see exercise 7.1.4–55).

[In the special case of n queens, represented as an exact cover problem as in (23), this algorithm is due to I. Rivin, R. Zabih, and J. Lamping, *Inf. Proc. Letters* **41** (1992), 253–256. They proved that the frontier for n queens never has more than $3n$ items.]

41. The author has had reasonably good results using a triply linked binary search tree for the database, with randomized search keys. (Beware: The swapping algorithm used for deletion was difficult to get right.) This implementation was, however, limited to exact cover problems whose matrix has at most 64 columns; hence it could do n queens via (23) only when $n < 12$. When $n = 11$ its database reached a maximum size of 75,009, and its running time was about 25 megamems. But Algorithm X was noticeably better: It needed only about 12.5 Mμ to find all $Q(11) = 2680$ solutions.

In theory, this method will need only about 2^{3n} steps as $n \to \infty$, times a small polynomial function of n. A backtracking algorithm such as Algorithm X, which enumerates each solution explicitly, will probably run asymptotically slower (see exercise 7.2.2–15). But in practice, a breadth-first approach needs too much space.

On the other hand, this method did beat Algorithm X on the n queen bees problem of exercise 7.2.2–16: When $n = 11$ its database grew to 364,864 entries; it computed $H(11) = 596{,}483$ in just 30 Mμ, while Algorithm X needed 440 Mμ.

42. The set of solutions for s_j can be represented as a regular expression α_j instead of by its size, c_j. Instead of inserting $(s_j + t, c_j)$ in step N3, insert $\alpha_j k$. If inserting (s, α), when (s_i, α_i) is already present with $s_i = s$, change $\alpha_i \leftarrow \alpha_i \cup \alpha$. [Alternatively, if only one solution is desired, we could attach a single solution to each s_j in the database.]

43. Let $i = (i_1 i_0)_3$ and $j = (j_1 j_0)_3$; then cell (i, j) belongs to box $(i_1 j_1)_3$. Mathematically, it's cleaner to consider the matrices $a'_{ij} = a_{ij} - 1$, $b'_{ij} = b_{ij} - 1$, $c'_{ij} = c_{ij} - 1$, which are the “multiplication tables” of interesting binary operators on $\{0, \ldots, 8\}$. We have $a'_{ij} = ((i_0 i_1)_3 + j) \bmod 9$; $b'_{ij} = ((i_0 + j_1) \bmod 3, (i_1 + j_0) \bmod 3)_3$; and $c'_{ij} = ((i_0 + i_1 + j_1) \bmod 3, (i_0 - i_1 + j_0) \bmod 3)_3$. (Furthermore the latter two operators are “isotopic”: $c'_{ij} = b'_{(i\pi)(j\pi^-)}\pi$, when $(i_1, i_0)_3\pi = (i_1, (i_0 + i_1) \bmod 3)_3$.)

[A pattern like (28c) appeared in a Paris newspaper of 1895, in connection with magic squares. But no properties of its 3×3 subsquares were mentioned; it was a sudoku solution purely by coincidence. See C. Boyer, *Math. Intelligencer* **29**, 2 (2007), 63.]

sample
frontier
n queens
Rivin
Zabih
Lamping
author
triply linked
binary search tree
backtracking algorithm
asymptotically
theory vs practice
practice vs theory
n queen bees
regular expression
0-origin indexing
multiplication tables
binary operators
isotopic
magic squares
sudoku solution
Boyer

44. No. The 33rd digit is 0. [A sudoku whose clues are π's first 32 digits was first constructed by Johan de Ruiter in 2007; see `www.puzzlepicnic.com/puzzle?346`. Furthermore, π's first 22 digits can actually be arranged *in a circle* to give a uniquely solvable sudoku, if we also require the elements of both main diagonals to be distinct! See Aad Thoen and Aad van de Wetering, *Exotische Sudoku's* (2016), 144.]

45. Step X3 chooses p_{44}, p_{84}, p_{74}, p_{24}, p_{54}, p_{14}, p_{82}, p_{42}, p_{31}, p_{32}, p_{40}, p_{45}, p_{46}, p_{50}, p_{72}, p_{60}, p_{00}, p_{62}, p_{61}, p_{65}, p_{35}, p_{67}, p_{70}, p_{71}, p_{75}, p_{83}, p_{13}, p_{03}, p_{18}, p_{16}, p_{07}, p_{01}, p_{05}, p_{15}, p_{21}, p_{25}, p_{76}, p_{36}, p_{33}, p_{37}, p_{27}, p_{28}, p_{53}, p_{56}, p_{06}, p_{08}, p_{58}, p_{77}, p_{88}, in that order.

46. The lists for items p_{44}, p_{84}, r_{33}, r_{44}, r_{48}, r_{52}, r_{59}, r_{86}, r_{88}, c_{22}, c_{43}, b_{07}, b_{32}, b_{39}, b_{43}, b_{54}, and b_{58} have length 1 when Algorithm X begins to tackle puzzle (29a). Step X3 will branch on whichever item was placed first in step X1. (The author's sudoku setup program puts p before r before c before b in that step.)

47. r_{13}, c_{03}, b_{03}, b_{24}, b_{49}, b_{69}. The latter three were hidden already in (32).

48. In case (a) we list the available columns; in case (b) we list the available rows:

(a)

	1	2	3	4	5	6	7	8	9
0	45	**3**	**6**	**8**	**0**	12457	1245	2457	157
1	**0**	278	**1**	**3**	**5**	2478	2468	2467	78
2	**6**	0127	**3**	012	**8**	012457	01245	02457	157
3	1234578	1278	78	12456	27	12345	1234568	24567	**0**
4	12378	**5**	**0**	12	27	**6**	1238	27	**4**
5	12378	01278	**5**	0126	**6**	0123	012368	0267	78
6	4578	**6**	78	045	**1**	04578	045	**3**	**2**
7	125	012	**4**	0125	**6**	0125	**7**	**8**	**3**
8	58	**4**	**2**	**7**	**3**	058	05	**1**	**6**

(b)

	0	1	2	3	4	5	6	7	8
1	**1**	3457	3457	345	036	03678	**2**	3456	34568
2	257	2357	12357	**0**	**8**	**4**	**6**	1235	135
3	**4**	**1**	**8**	**2**	**7**	**5**	**0**	36	36
4	2567	23457	23457	**1**	36	367	35	**8**	**0**
5	**0**	**6**	34	**8**	**5**	**1**	**7**	34	**2**
6	25678	02357	012357	35	01236	023678	**4**	0126	168
7	2568	02345	012345	345	01236	02368	135	**7**	1345
8	25	**8**	012345	**6**	0123	023	135	012345	**7**
9	**3**	02	**6**	**7**	**4**	02	**8**	0125	15

(Notice that "hidden" singles and pairs, etc., become "naked" in this representation. Similar plots, which relate boxes to values, are also possible; but they're trickier, because boxes aren't orthogonal to rows or columns.)

49. (a) For columns, remove all items r_{ik} and b_{xk}, as well as c_{jk} with $j \neq j_0$; let u_j — v_k when an option contains '$p_{ij_0}\ c_{j_0k}$'. For boxes, remove all r_{ik}, c_{jk}, and b_{xk} with $x \neq x_0$; let u_j — v_k when an option contains '$p_{(3\lfloor x_0/3\rfloor+\lfloor j/3\rfloor)(3(x \bmod 3)+(j \bmod 3))}\ b_{x_0k}$'.

(b) The $n-q$ *non*-neighbors of a hidden q-tuple (e.g., $\{u_3, u_8, u_1\}$) are "naked."

(c) By (b) it suffices to list the naked ones (and only those for which $q<r$). Let's denote the option in (30) by ijk. In row 4 we find the naked pair $\{u_3, u_8\}$, hence we can delete options 411, 417, 421, 427, 471; also the naked triple $\{u_1, u_3, u_8\}$, so we can also delete option 424. There's no nakedness in the columns. The naked triple $\{u_0, u_3, u_6\}$ in box 4 allows deletion of options 341, 346, 347, 351, 356, 357.

(d) Let u_i — v_j if there's an option that contains '$r_{ik_0}\ c_{jk_0}$'. When $k_0 = 9$ there's a naked pair $\{u_1, u_5\}$, so we can delete options 079 and 279.

[Many other reductions have been proposed. For example, (33) has a "pointing pair" in box 4: Since '4' and '8' must occupy that box in row 3, we can remove options 314, 324, 328, 364, 368, 378. Classic references are the early tutorials by W. Gould, *The Times Su Doku Book 1* (2005); M. Mepham, *Solving Sudoku* (2005). A comprehensive theory, applicable also to many other problems, has been developed by D. Berthier, *Pattern-Based Constraint Satisfaction and Logic Puzzles* (2012).]

de Ruiter
Thoen
van de Wetering
author
sudoku setup program
hidden
naked
pointing pair
Gould
Mepham
Berthier

50. Such a puzzle must add a 7 or 8 in one of 18 places, because (29c) has just 2 solutions. So there are 36 of them (18 isomorphic pairs).

9	3	4	5	1	7		6	
7	6	2	4	9	3	1	7	5
7	5	1	7			4	9	3
2	7	5	9	7	1	6	3	4
6	4	9		3	5		1	
1	7	3				5		9
4	1	7	6	5	9	3		
3	2	7	1			9	5	6
5	9	6	3			7	4	1

51. We can solve this problem with Algorithm M, using options (30) with $k \neq 8$ and giving multiplicity 2 to each of the items r_{i7}, c_{j7}, b_{x7}. There are six solutions, all of which extend the partial solution shown. Only one yields a sudoku square when we change half of the 7s to 8s.

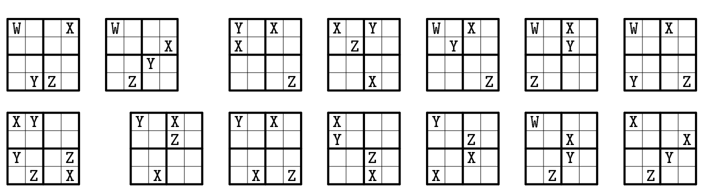
52. Puzzles claiming to be "the world's hardest" keep appearing in online forums. From the standpoint of search tree size via Algorithm X, the toughest of these extreme puzzles that has been seen so far in the author's tests is shown here in a canonical form. (It's the third in a list available from `sites.google.com/site/sudoeleven/` (2011).) Although its randomized search tree sizes are 4666 ± 550 — astonishingly high for sudoku — the running time is less than 3 Mμ.)

1	2		3			4		
4			1					
		5		6				
3							1	
	7					2	3	
						6		8
	4		2				7	
		9		8				
					5			6

53. (a) Every shidoku solution is equivalent to one of the two special solutions A or B below (which incidentally have respectively 32 and 16 automorphisms, in the sense of exercise 114). We can't uniquely specify either solution unless we have at least one clue in each of the regions $\{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$ of C.

$$A = \begin{array}{|cc|cc|}\hline 1&2&3&4\\ 3&4&1&2\\ \hline 2&1&4&3\\ 4&3&2&1\\ \hline\end{array}\,, \qquad B = \begin{array}{|cc|cc|}\hline 1&2&3&4\\ 3&4&2&1\\ \hline 2&1&4&3\\ 4&3&1&2\\ \hline\end{array}\,, \qquad C = \begin{array}{|cc|cc|}\hline \mathtt{A}&\mathtt{A}&\mathtt{B}&\mathtt{B}\\ \mathtt{C}&\mathtt{C}&\mathtt{D}&\mathtt{D}\\ \hline \mathtt{A}&\mathtt{A}&\mathtt{B}&\mathtt{B}\\ \mathtt{C}&\mathtt{C}&\mathtt{D}&\mathtt{D}\\ \hline\end{array}\,.$$

(b) Only $4^4 = 256$ sets of four clues meet the conditions of (a), for each of A and B; we can test them all. Reducing by the automorphisms leaves two for A and eleven for B:



(There also are 22 essentially different shidoku puzzles with five irredundant clues, and a unique puzzle with six. The latter, which is solved by A, is shown above at the bottom left; it cannot omit a clue without having an empty region in either C or C^T. These results were discovered by Ed Russell in 2006.)

54. For example, removing clues one at a time shows that only 10 of the 32 givens are actually essential. The best strategy for finding all minimal X is probably to examine candidate sets in order of decreasing cardinality: Suppose $W \subseteq X$, and suppose that previous tests have shown that the solution is unique, given X, but not given $X \setminus w$ for any $w \in W$. Thus X is minimal if $W = X$. Otherwise let $X \setminus W = \{x_1, \ldots, x_t\}$, and test $X \setminus x_i$ for each i. Suppose the solution turns out to be unique if and only if $i > p$. Then we schedule the $t-p$ candidate pairs $(W \cup \{x_1, \ldots, x_p\}, X \setminus x_i)$, $p < i \leq t$, for processing in the next round. With suitable caching of previous results, we can avoid testing the same subset of clues more than once. Furthermore we can readily modify Algorithm X so that it backtracks immediately after discovering a single unwanted solution.

All 777 minimal subsets were found in this manner, involving 15441 invocations of Algorithm X, but needing a total of only about 1.5 gigamems of computation. Altogether (1, 22, 200, 978, 2780, 4609, 4249, 1950, 373, 22) candidate pairs were examined in rounds (32, 31, ..., 23); and exactly (8, 154, 387, 206, 22) solutions were

author
canonical form
Eleven (anonymous?)
automorphisms
Russell

found of sizes (27, 26, 25, 24, 23). The lexicographically last 23-clue subset, which is illustrated below, turns out to be a fairly tough puzzle, with 220 nodes in its search tree.

(Let $f(x_1, \ldots, x_{32})$ be the monotone Boolean function '[the solution is unique, given the clues with $x_j = 1$]'. This problem essentially asks for f's prime implicants.)

(29a)

				1				
4	1						9	
		6	5					
5				8				9
	7						3	2
	3	8			4			
			2	6		4		
			3					8
3					7		5	

; (28a)

1	2	3						
			7	8	9			
2	3	4						
						5	6	7
		5	6	7				
9	1							8

; (28b)

1					6		8	
	5					1		
		9				4		
	3		5					7
				9				
8					1			
							7	
6	4				8			2
			3					5

.

55. If only one of those nine appearances has been specified, the other eight can always be permuted into another solution. And the entire diagram can be partitioned into nine disjoint sets of nine, all with the same property, thus requiring at least $2 \cdot 9$ clues.

This argument proves that all 18-clue characterizations must have a very special form. The interesting solution above makes a particularly satisfying puzzle. (The author found it with the help of a SAT solver; see Section 7.2.2.2.)

The same argument shows that (28b) needs at least 18 clues. But this time the corresponding SAT instance is unsatisfiable. Moreover, any 19-clue solution must have three clues in just one critical group of nine; the associated SAT instance, which insists on having at least one clue in each of the 2043 subsets of at most 18 cells that can be rearranged into new solutions, also is unsatisfiable. (Proved in 177 Mμ.) But hurrah, the special structure does lead to 20-clue examples, like the one above.

(The constructions for (28b) apply also to (28c), via the isotopism in answer 43.)

56. (We assume that a decent sudoku problem has only one solution.) An example with 40 irredundant clues, shown here, was first discovered by Mladen Dobrichev in 2014, after examining a huge number of cases. (Incidentally, the solution to this problem has no automorphisms.) An example with 41 irredundant clues would be a big surprise.

	1	2		3	4	5	6	7
	3	4	5		6	1	8	2
		1		5	8	2		6
		8	6					1
	2				7		5	
		3	7		5		2	8
	8			6		7		
2		7		8	3	6	1	5

57. There are only $2 \cdot 3! \cdot 3! \cdot 3! \cdot 3! = 2592$ possibilities for each box. So we can set up an exact cover problem with $9 \cdot 2592$ options, each of which names a box, nine row-column pairs, three horizontal trios, and three vertical trios. We can assume by symmetry that there's only one option for box 0, namely 'b_0 r_{01} c_{01} r_{04} c_{14} r_{07} c_{27} r_{18} c_{08} r_{12} c_{12} r_{15} c_{25} r_{26} c_{06} r_{29} c_{19} r_{23} c_{23} h_{147} h_{258} h_{369} v_{168} v_{249} v_{357}'. Furthermore row 0 can be restricted to `1472AB3CD`, where $\{$`A`, `C`$\} = \{$`5`, `6`$\}$ and $\{$`B`, `D`$\} = \{$`8`, `9`$\}$. That reduces the number of options to 16417; and Algorithm X quickly ((58+54)Mμ) finds 864 solutions.

Such solutions were first discovered by A. Thoen and A. van de Wetering; see Thoen's book *Sudoku Patterns* (2019), §2.7. All 864 are isomorphic under sudoku-solution-preserving permutations of rows and columns. One of the nicest is

1	4	7	2	5	9	3	6	8
8	2	5	7	3	6	9	1	4
6	9	3	4	8	1	5	7	2
2	6	9	3	4	8	1	5	7
7	3	4	9	1	5	8	2	6
5	8	1	6	7	2	4	9	3
3	5	8	1	6	7	2	4	9
9	1	6	8	2	4	7	3	5
4	7	2	5	9	3	6	8	1

, which has a remarkable inner symmetry between diagonally adjacent boxes:

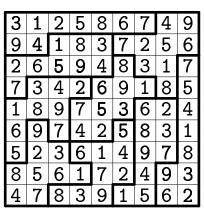
A	B	C			
D	E	F			
G	H	I			
			I	B	D
			H	A	F
			G	C	E

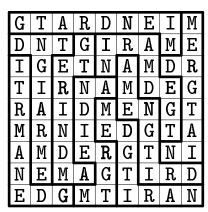
.

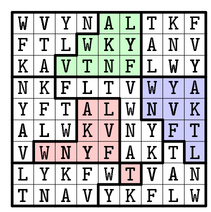
monotone Boolean function
order ideals
prime implicants
author
SAT solver
isotopism
Dobrichev
automorphisms
Thoen
van de Wetering

58. Use the standard 729 sudoku options (30); but also include queen items '$a'_{(i+j)k}$ $b'_{(i-j)k}$' in option (i,j,k) when $k \le 7$. Furthermore, in order to avoid getting each solution $7!\,2! = 10080$ times, force row 0 by adding a new primary item '$*$' and new secondary items '$*_j$' for $0 \le j < 9$, together with 20 options '$*$ $*_0{:}f(0,p,q)$ $\ldots$ $*_8{:}f(8,p,q)$' for $0 \le p < q < 9$, $p+q < 9$, where $f(j,p,q) = (j = p?\ 8{:}\ j = q?\ 9{:}\ 1 + j - [j > p] - [j > q])$. There are only two solutions, found in 3 Gμ, both centrally symmetric. (See Appendix E, and Thoen's book *Sudoku Patterns* (2019), §3.4.)

59. When ps precede rs precede cs precede bs in X1, the tree sizes are 1105, 910, 122.

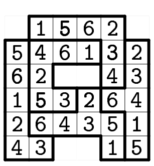
(34a) 

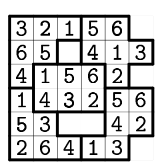
; (34b) 

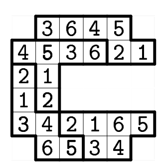
; (34c) 

.

60. Using the options (30), items r_{ik} and c_{jk} should be *secondary* when row i or column j contains fewer than 6 cells. The puzzles are fun to solve by hand; but in a pinch, Algorithm X will traverse search trees of sizes 23, 26, and 16 to find the answers:

(a) 

; (b) 

; (c) 

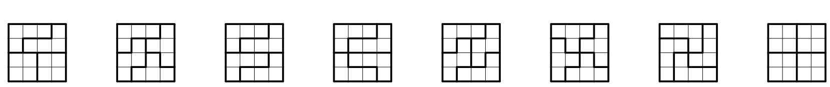
.

[These are the first of 26 elegant puzzles announced by Serhiy and Peter Grabarchuk on Martin Gardner's 100th birthday (21 October 2014) and posted at `puzzlium.com`.]

61. Exactly 1315 of the $\binom{25}{5} = 53130$ ways to retain five clues result in a unique solution, and 175 of them involve all five digits. The lexicographically first is Fig. A–2(a).

62. Follow the hint; the undesired *straight* n-ominoes can be rejected easily in step R2 by examining v_{n-1} and v_0. This quickly produces (16, 105, 561, 2804, 13602) box options, for $n = (3, 4, 5, 6, 7)$, which can be fed to Algorithm X to get jigsaw patterns.

There are no patterns for $n = 3$. But $n = 4$ has 33 patterns, which divide into eight equivalence classes under rotation and/or reflection:



1 1 2 2 2 4 4 8

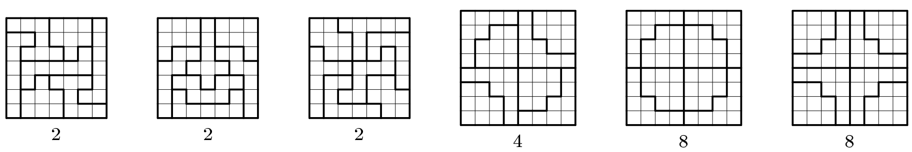
(The number of symmetries is shown below each arrangement; notice that $8/1 + 8/1 + 8/2 + 8/2 + 8/2 + 8/4 + 8/4 + 8/8 = 33$.) Similarly, $n = 5$ has 266 equivalence classes, representing $256 \cdot (8/1) + 7 \cdot (8/2) + 3 \cdot (8/4) = 2082$ total patterns; $n = 6$ has 40237 classes, representing $39791 \cdot (8/1) + 439 \cdot (8/2) + 7 \cdot (8/4) = 320098$ patterns in all.

The computation gets more serious in the case $n = 7$, when Algorithm X needs about 1.9 Tμ to generate the 132,418,528 jigsaw patterns. These patterns include 16,550,986 classes with no symmetry, and 2660 with one nontrivial symmetry. The latter break down into 2265 that are symmetric under 180° rotation, 354 that are symmetric under horizontal reflection, and 41 that are symmetric under diagonal

secondary
S Grabarchuk
P Grabarchuk
Gardner
straight n-ominoes
rotation
reflection
symmetries
180° rotation
central symmetry
reflection
reflection symmetry

8-fold symmetry
windmill sudoku
Harris
faultfree
Harris
diagonal
author
Henle

reflection. Here are some typical symmetric examples:



(It's not difficult to generate all of the *symmetric* solutions for slightly higher values of n; three of the classes for $n = 8$, shown above, have more than 2 symmetries. And the case $n = 9$ contains two patterns with 8-fold symmetry *besides* the standard sudoku boxes: See Fig. A–2(b) and (c), where the latter might be called windmill sudoku! For complete counts for $n = 8$ and $n = 9$, with straight n-ominoes allowed, see Bob Harris's preprint "Counting nonomino tilings," presented at G4G9 in 2010.)

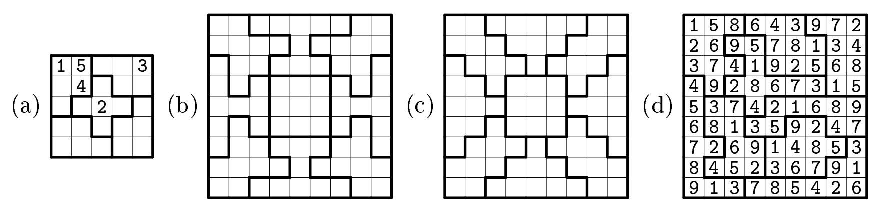


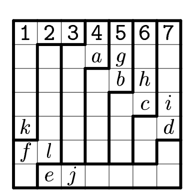
Fig. A–2. Jigsaw sudoku patterns.

63. A simple modification of exercise 7.2.2–76 will generate the 3173 boxes that have the desired rainbow property. An exact cover problem, given those 3173 options, shows (after 1.2 Gμ of computation) that the boxes can be packed in 98556 ways. If we restrict the options to the 3164 that aren't sudoku boxes, the number of packings goes down to 42669, of which 24533 are faultfree. Figure A–2(d) is a faultfree example.

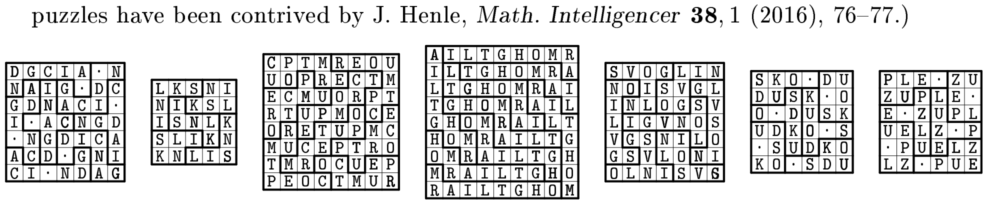
64. (a) When $n = 4$, one of the eight classes in answer 62 (the 2nd) has *no* solutions; another (the 5th) is clueless. When $n = 5$, eight of the 266 classes have no solution; six are clueless. When $n = 6$, 1966 of 40237 are vacuous and 28 are clueless.

(Maxime's original puzzle appeared in the newsletter of Chicago Area Mensa [*ChiMe* **MM**, 3 (March 2000), 15]. Algorithm X solves it with a 40-node search tree. But the tree size would have been 215 if he'd put `ABCDEF` in the next row down!)

(b) (Solution by Bob Harris, `www.bumblebeagle.org/dusumoh/proof/`, 2006.) The clueless jigsaw for $n = 4$ generalizes to all larger n, as illustrated here for $n = 7$: First $a = 3$; hence $b = 3$; $\ldots$; hence $f = 3$. Then $g = 4$; hence $h = 4$; $\ldots$; hence $l = 4$. And so on. Finally we know where to place the `2`'s and the `1`'s. (This proof shows that, for odd $n > 3$, there's always an $n \times n$ jigsaw sudoku whose clues lie entirely on the main diagonal. Is there also a general construction that works for *even* values of n? An 8×8 example appears in exercise 65.)



65. (The author designed these puzzles with the aid of exercises 62 and 64. Similar puzzles have been contrived by J. Henle, *Math. Intelligencer* **38**, 1 (2016), 76–77.)



66. (Puzzles like this might be too difficult for humans, but not for Algorithm C.) Extend the 729 options (30) by adding '$ij{:}k$', where ij is a new secondary item for $0 \le i, j < 9$. Also add eighteen new primary items k for $1 \le k \le 9$ and s_j for $0 \le j < 9$, where k represents card k and s_j represents a slot in the 3×3 array. Each item k has nine options, for the nine slots in which it might be placed; for example, the options for item 2 are '2 s_0 00:2 11:3 22:4 20:1', '2 s_1 03:2 14:3 25:4 23:1', ..., '2 s_8 66:2 77:3 88:4 86:1'.

There are 9! ways to place the cards in slots; but only $9!/(3!\,3!) = 10080$ are actually different, because the rows and columns can be permuted independently without changing the number of sudoku solutions. Suppose card c_j goes into slot s_j; then we can assume without loss of generality that $c_0 = 1$ and that $c_4 = \min(c_4, c_5, c_7, c_8)$. (To incorporate these constraints, give only one option for card 1 and only eight options for cards 2–9; use ordering tricks like (26) to ensure that $c_4 < c_5$, $c_4 < c_7$, $c_4 < c_8$.)

With this understanding, puzzle (i) has only one solution, and only when $c_0 \ldots c_8 =$ 192435768. (That solution has six automorphisms, in the sense of exercise 114.) Puzzle (ii) has a unique solution when $c_0 \ldots c_8 = 149523786$. It also has ten sudoku solutions when the slot permutation is 149325687; so we can't use *that* placement.

67. (a) (Solution by A. E. Brouwer, `homepages.cwi.nl/~aeb/games/sudoku/nrc.html`, 2006.) The four new boxes force also `aaaaaaaaa`, ..., `eeeeeeeee` to be rainbows.

e	a	a	a	e	b	b	b	e
c				c				c
c				c				c
c				c				c
e	a	a	a	e	b	b	b	e
d				d				d
d				d				d
d				d				d
e	a	a	a	e	b	b	b	e

(i)

3	2	7	1	5	4	8	6	9
6	1	5	8	2	9	4	7	3
8	9	4	3	7	6	2	1	5
9	6	2	7	1	5	3	8	4
7	4	3	9	8	2	1	5	6
5	8	1	4	6	3	7	9	2
1	3	6	2	9	8	5	4	7
4	7	9	5	3	1	6	2	8
2	5	8	6	4	7	9	3	1

(ii)

3	1	6	8	9	4	7	2	5
5	7	8	3	2	6	4	1	9
4	2	9	5	1	7	3	8	6
7	4	1	6	3	9	2	5	8
2	9	5	7	8	1	6	3	4
8	6	3	4	5	2	9	7	1
9	8	7	1	4	3	5	6	2
6	5	2	9	7	8	1	4	3
1	3	4	2	6	5	8	9	7

(b) Introduce new primary items b'_{yk} for $0 \le y < 9$ and $1 \le k \le 9$. Add b'_{yk} to option (30) with $y = 3\lfloor i\tau/3\rfloor + \lfloor j\tau/3\rfloor$, where τ is the permutation (03)(12)(58)(67).

(c) With items b'_{yk} only considered for $y \in \{0, 2, 6, 8\}$, Algorithm X's search tree grows from 77 nodes to 231 for (i), and from 151 nodes to 708 for (ii).

[Puzzle (ii) is a variant of an 11-clue example constructed by Brouwer. The minimum number of clues necessary for hypersudoku is unknown.]

(d) True. (That's the permutation τ in (b), applied to both rows and columns.)

68. (a) A simple backtrack program generates all convex n-ominoes whose top cell(s) are in row 0 and whose leftmost cell(s) are in column 0. [This problem has respectively (1, 2, 6, 19, 59, 176, 502) solutions for $1 \le n \le 7$; see M. Bousquet-Mélou and J.-M. Fédou, *Discrete Math.* **137** (1995), 53–75, for the generating function.] The resulting (1, 4, 22, 113, 523, 2196, 8438) placements into an $n \times n$ box yield exact cover problems as in answer 62. Considering symmetries, we find $1 \cdot (8/4) = 2$ patterns when $n = 2$; $1\cdot(8/1)+1\cdot(8/4) = 10$ patterns when $n = 3$; $10\cdot(8/1)+7\cdot(8/2)+4\cdot(8/4)+1\cdot(8/8) = 117$ when $n = 4$; $355\cdot(8/1)+15\cdot(8/2)+4\cdot(8/4) = 2908$ when $n = 5$; $20154\cdot(8/1)+342\cdot(8/2)+8\cdot(8/4) = 162616$ when $n = 6$; $2272821\cdot(8/1)+1181\cdot(8/2)+5\cdot(8/4) = 18187302$ when $n = 7$. (Exercise 62 had different results because it disallowed straight n-ominoes.)

(b) There are 325 such nonominoes touching row 0 and column 0, leading to 12097 placements and $1014148 \cdot (8/1) + 119 \cdot (8/2) + 24 \cdot (8/4) + 1 \cdot (8/8) = 8113709$ patterns. If we exclude the 3×3 nonomino, and its 49 placements, the number of patterns goes down to $675797 \cdot (8/1) = 5406376$.

[Convex polyominoes were introduced by Klarner and Rivest; see answer 303.]

XCC problem
color controls
automorphisms
Brouwer
backtrack program
Bousquet-Mélou
Fédou
generating function
Klarner
Rivest

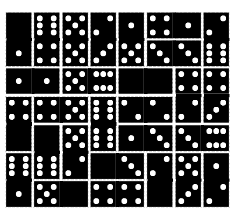
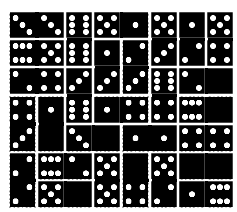
69. Say that an "N_k" is a suitable nonomino placement that has k `B`s and $9 - k$ `L`s. Only two cases give seven wins for `B`: 1 N_6, 6 N_5, 2 N_0; 7 N_5, 1 N_1, 1 N_0. With the given voting pattern there are respectively (1467, 2362, 163, 2) options for N_6, N_5, N_1, N_0. Algorithm M provides the desired multiplicities. After 12 Mμ of computation we find that there are no solutions in case 1 but 60 solutions in case 2, one of which is shown.

B	B	L	B	L	L	L	L	B
L	L	L	B	L	L	L	B	L
B	B	L	B	L	B	B	L	B
L	L	L	L	L	L	L	L	L
B	B	B	L	L	B	L	L	B
L	B	L	B	B	B	B	B	B
B	B	L	B	B	B	B	B	L
L	B	L	L	L	L	B	L	L
L	L	B	L	L	B	B	L	L

(Of course the author does not recommend secret deals such as this! The point is that unfair gerrymandering is easy to do and hard to detect. Indeed, a trial of 1000 random voter patterns, each with 5/4 split in the nine standard 3×3 districts, included 696 cases that could be gerrymandered to seven Big-Endian districts using only convex nonominoes that fit in a 5×5. Eight of those cases could also achieve a 4×4 fit.)

[Similar studies, using realistic data, go back to R. S. Garfinkel's Ph.D. thesis *Optimal Political Districting* (Baltimore: Johns Hopkins University, 1968).]

70. In (a), four pieces change; in (b) the solution is unique:

(a) ; (b) .

Notice that the spot patterns , , and  are rotated when a domino is placed vertically; these visual clues, which would disambiguate (a), don't show up in the matrix.

(Dominosa was invented by O. S. Adler [Reichs Patent #71539 (1893); see his booklet *Sperr-Domino und Dominosa* (1912), 23–64, written with F. Jahn]. Similar "quadrille" problems had been studied earlier by E. Lucas and H. Delannoy; see Lucas's *Récréations Mathématiques* **2** (1883), 52–63]; W. E. Philpott, *JRM* **4** (1971), 229–243.)

71. Define 28 vertices Dxy for $0 \le x \le y \le 6$; 28 vertices ij for $0 \le i < 7$, $0 \le j < 8$, and $i + j$ even; and 28 similar vertices ij with $i + j$ odd. The matching problem has 49 triples of the form $\{\mathrm{D}xy, ij, i(j{+}1)\}$ for $0 \le i, j < 7$, as well as 48 of the form $\{\mathrm{D}xy, ij, (i{+}1)j\}$ for $0 \le i < 6$ and $0 \le j < 8$, corresponding to potential horizontal or vertical placements. For example, the triples for exercise 70(a) are $\{\mathrm{D}06, 00, 01\}$, $\{\mathrm{D}56, 01, 02\}$, ..., $\{\mathrm{D}23, 66, 67\}$; $\{\mathrm{D}01, 00, 10\}$, $\{\mathrm{D}46, 01, 11\}$, ..., $\{\mathrm{D}12, 57, 67\}$.

72. Model (i) has $M = 56!/8!^7 \approx 4.10 \times 10^{42}$ equally likely possibilities; model (ii) has $N = 1292697 \cdot 28! \cdot 2^{21} \approx 8.27 \times 10^{41}$, because there are 1292697 ways to pack 28 dominoes in a 7×8 frame. (Algorithm X will quickly list them all.) The expected number of solutions per trial in model (i) is therefore $N/M \approx 0.201$.

Ten thousand random trials with model (i) gave 216 cases with at least one solution, including 26 where the solution was unique. The total number $\sum x$ of solutions was 2256; and $\sum x^2 = 95918$ indicated a heavy-tailed distribution whose empirical standard deviation is ≈ 3.1. The total running time was about 250 Mμ.

Ten thousand random trials with model (ii), using random choices from a precomputed list of 1292687 packings, gave 106 cases with a unique solution; one case had 2652 of them! Here $\sum x = 508506$ and $\sum x^2 = 144119964$ indicated an empirical mean of ≈ 51 solutions per trial, with standard deviation ≈ 109. Total time was about 650 Mμ.

73. From 66110144/26611514/52132140/55322200/53242006/36430565/33643054 we get 730,924 solutions, which is the current record. This array, found by Michael Keller

multiplicities
author
Garfinkel
Adler
Patent
Jahn
quadrille
Lucas
Delannoy
Philpott
dimer tilings
heavy-tailed distribution
empirical standard deviation
Keller

in 2004, has the surprising property that *every* candidate placement, except for the '21' in '521', occurs in at least one solution. (In fact, in at least 31,370 solutions!)

74. One way to obtain candidate arrays is to formulate an MCC problem: Given one of the 1292697 matchings of answer 72, let there be options 'P_{uv} xy $t_u{:}x$ $t_v{:}y$', 'P_{uv} xy $t_u{:}y$ $t_v{:}x$' for uv in the matching, and 'Q_{uv} D_{xy} $t_u{:}x$ $t_v{:}y$', 'Q_{uv} D_{xy} $t_u{:}y$ $t_v{:}x$' for uv not in the matching; here $0 \le x \le y \le 6$, and duplicate options are omitted when $x = y$. Give each D_{xy} multiplicity 3. Also add 28 further options '# D_{xy}', where # has multiplicity 15 (because 15 pairs xy should have only two spurious appearances).

For fun, the author chose a *tatami tiling* for the matching (see exercise 7.1.4–215), and obtained one candidate every 70 Mμ or so when the nonsharp variant of Algorithm M was applied with randomization as in exercise 31. Surprisingly, the first 10000 candidates yielded 2731 solutions, of which the hardest (with a 572-node search tree) was 15133034/21446115/22056105/65460423/22465553/61102332/63600044.

[See `www.puzzlelaboratory.com/DominoGG.html`.]

75. (a) $(x \circ y) \circ x = (x \circ y) \circ (y \circ (x \circ y)) = y$.

(b) All five are legitimate. (The last two are gropes because $f(t + f(t)) = t$ for $0 \le t < 4$ in each case; they are isomorphic if we interchange any two elements. The third is isomorphic to the second if we interchange $1 \leftrightarrow 2$. There are 18 grope tables of order 4, of which (4, 12, 2) are isomorphic to the first, third, and last tables shown here.)

(c) For example, let $x \circ y = (-x - y) \bmod n$. (More generally, if G is any group and if $\alpha \in G$ satisfies $\alpha^2 = 1$, we can let $x \circ y = \alpha x^- \alpha y^- \alpha$. If G is commutative and $\alpha \in G$ is arbitrary, we can let $x \circ y = x^- y^- \alpha$.)

(d) For each option of type (i) in an exact covering, define $x \circ x = x$; for each of type (ii), define $x \circ x = y$, $x \circ y = y \circ x = x$; for each of type (iii), define $x \circ y = z$, $y \circ z = x$, $z \circ x = y$. Conversely, every grope table yields an exact covering in this way.

(e) Such a grope covers n^2 items with k options of size 1, all other options of size 3. [F. E. Bennett proved, in *Discrete Mathematics* **24** (1978), 139–146, that such gropes exist for *all* k with $0 \le k \le n$ and $k \equiv n^2$ (modulo 3), except when $k = n = 6$.]

Notes: The identity $x \circ (y \circ x) = y$ seems to have first been considered by E. Schröder in *Math. Annalen* **10** (1876), 289–317 [see '(C_0)' on page 306], but he didn't do much with it. In a class for sophomore mathematics majors at Caltech in 1968, the author defined gropes and asked the students to discover and prove as many theorems about them as they could, by analogy with the theory of groups. The idea was to "grope for results." The official modern term for a grope is a real jawbreaker: *semisymmetric quasigroup*.

76. (a) Eliminate the n items for xx; use only the $2\binom{n}{3}$ options of type (iii) for which $y \ne z$. (Idempotent gropes are equivalent to "Mendelsohn triples," which are families of $n(n-1)/3$ three-cycles (xyz) that include every ordered pair of distinct elements. N. S. Mendelsohn proved [*Computers in Number Theory* (New York: Academic Press, 1971), 323–338] that such systems exist for all $n \not\equiv 2$ (modulo 3), except when $n = 6$.

(b) Use only the $\binom{n+1}{2}$ items xy for $0 \le x \le y < n$; replace options of type (ii) by 'xx xy' and 'xy yy' for $0 \le x < y < n$; replace those of type (iii) by 'xy xz yz' for $0 \le x < y < z < n$. (Such systems, Schröder's '(C_1) and (C_2)', are called totally symmetric quasigroups; see S. K. Stein, *Trans. Amer. Math. Soc.* **85** (1957), 228–256, §8. If idempotent, they're equivalent to Steiner triple systems.)

(c) Omit items for which $x = 0$ or $y = 0$. Use only the $2\binom{n-1}{3}$ options of type (iii) for $1 \le x < y, z < n$ and $y \ne z$. (Indeed, such systems are equivalent to idempotent gropes on the elements $\{1, \ldots, n-1\}$.)

MCC problem
author
tatami tiling
nonsharp
randomization
isomorphic
isomorphic
Bennett
Schröder
Caltech
author
groups
quasigroup
semisymmetric quasigroup
Mendelsohn triples
Schröder
totally symmetric quasigroups
Stein
Steiner triple systems

word search
CUTOFF

77. Use primary items v and v' for each vertex of G and H; also secondary items ee' for each pair of vertices e and e' in G and the *complement* of H. There are n^2 options, namely '$v\ v'\ \bigcup_{e(v),e'(v')} e(v)e'(v')$', where $e(v)$ ranges over all edges $v \mathbin{—} u$ in G and $e'(v')$ ranges over all nonedges $v' \not\mathbin{—} u'$ in H. (The solutions to this problem are the one-to-one matchings $v \longleftrightarrow v'$ of the vertices such that $u \mathbin{—} v$ implies $u' \mathbin{—} v'$.)

78. For example, `CATALANDAUBOREL`, `GRAMARKOFFKNOPP`, `ABELWEIERSTRASS`, `BERTRAND-HERMITE`, `CANTORFROBENIUS`, `GLAISHERHURWITZ`, `HADAMARDHILBERT`, `HENSELKIRCHHOFF`, `JENSENSYLVESTER`, `MELLINSTIELTJES`, `NETTORUNGESTERN`, `MINKOWSKIPERRON`.

79. In an $n \times n$ array for word search, every k-letter word generates $(n+1-k)\cdot n\cdot 4$ horizontal/vertical options and $(n+1-k)^2\cdot 4$ diagonal options. So the desired answer is $(2, 5, 6, 5, 3, 5, 0, 1)\cdot(1296, 1144, 1000, 864, 736, 616, 504, 400) = 24320$.

80. Item `q` is selected at level 0, trying option $x_0 = 8$, '`q x y:A p`'. We cover `q`, then cover `x`, then purify `y` to color `A`, and cover `p`; but at level 1 we find that item `r`'s list is empty. So we backtrack: Uncover `p`, unpurify `y`, uncover `x` — and try option $x_0 = 20$, '`q x:A`', hence purifying `x` to color `A`. This time at level 1 we try $x_1 = 12$, '`p r x:A y`'. That causes us to cover `p`, then cover `r`, and then (since `x` is already purified) to cover `y`. At level 2 we discover that we've found a solution! Here's what's in memory:

i:	0	1	2	3	4	5	6
`NAME`(i):	—	`p`	`q`	`r`	`x`	`y`	—
`LLINK`(i):	0	0	1	0	6	4	4
`RLINK`(i):	0	3	3	0	6	6	4
x:	0	1	2	3	4	5	6
`LEN`(x):	—	1	2	1	2	0	0
`ULINK`(x):	—	12	20	23	18	5	—
`DLINK`(x):	—	12	8	23	14	5	10
x:	7	8	9	10	11	12	13
`TOP`(x):	1	2	4	5	−1	1	3
`ULINK`(x):	1	2	4	5	7	1	3
`DLINK`(x):	12	20	14	15	15	1	23
`COLOR`(x):	0	0	0	`A`	—	0	0
x:	14	15	16	17	18	19	20
`TOP`(x):	4	5	−2	1	4	−3	2
`ULINK`(x):	4	5	12	12	14	17	8
`DLINK`(x):	18	24	18	1	4	21	2
`COLOR`(x):	−1	0	—	0	`B`	—	0
x:	21	22	23	24	25		
`TOP`(x):	4	−4	3	5	−5		
`ULINK`(x):	18	20	3	5	23		
`DLINK`(x):	4	24	3	5	—		
`COLOR`(x):	`A`	—	0	`B`	—		

81. Almost true, if `TOP` and `COLOR` are stored in the same octabyte (so that only one is charged to read both). The only difference is when processing the input, because Algorithm X has no `COLOR` fields to initialize but Algorithm C zeroes them out.

82. True; the `LEN` field of secondary items doesn't affect the computation.

83. Before setting $i \leftarrow$ `TOP`(x_0) in step C6 when $l = 0$, let node x be the spacer at the right of x_0's option, and set $j \leftarrow$ `TOP`$(x-1)$. If $j > N$ (that is, if that option ends with the secondary item j), and if `COLOR`$(x-1) = 0$, cover(j).

84. Let `CUTOFF` (initially ∞) point to the spacer at the end of the best solution found so far. We'll essentially remove all nodes > `CUTOFF` from further consideration.

Whenever a solution is found, let node PP be the spacer at the end of the option for which $x_k = \max(x_0, \ldots, x_{l-1})$. If PP $\neq$ CUTOFF, set CUTOFF $\leftarrow$ PP, and for $0 \le k < l$ remove all nodes $>$ CUTOFF from the list for TOP(x_k). (It's easy to do this because the list is sorted.) Minimax solutions follow the last change to CUTOFF.

Begin the subroutine 'uncover$'(i)$' by removing all nodes $>$ CUTOFF from item i's list. After setting $d \leftarrow$ DLINK(q) in unhide$'(p)$, set DLINK(q) $\leftarrow d \leftarrow x$ if $d >$ CUTOFF. Make the same modifications also to the subroutine 'unpurify(p)'.

Subtle point: Suppose we're uncovering item i and encounter an option '$i\ j\ \ldots$' that should be restored to the list of item j; and suppose that the original successor '$j\ a\ \ldots$' of that option for item j lies below the cutoff. We know that '$j\ a\ \ldots$' contains at least one primary item, and that every primary item was covered before we changed the cutoff. Hence '$j\ a\ \ldots$' was *not* restored, and we needn't worry about removing it. We merely need to correct the DLINK, as stated above.

85. Now let CUTOFF be the spacer just *before* the best solution known. When resetting CUTOFF, backtrack to level $k-1$, where x_k maximizes $\{x_0, \ldots, x_{l-1}\}$.

86. The steps below also estimate the profile of the search tree. Running time is estimated in terms of "updates" and "cleansings." The user specifies a random seed and a desired number of trials; the final estimates are the averages of the (unbiased) estimates from each trial. Here we specify only how to make a single trial.

In step C1, also set $D \leftarrow 1$.

In step C2, estimate that the search tree has D nodes at level l. If RLINK(0) $= 0$, also estimate that there are D solutions.

In step C3, let θ be the number of options in the list of the chosen item i. If $\theta = 0$, estimate that there are 0 solutions, and go to C7.

At the end of step C4, let k be uniformly random in $[0\,..\,\theta-1]$; then set $x_l \leftarrow$ DLINK(x_l), k times.

Just before setting $l \leftarrow l+1$ at the end of step C5, suppose you've just done U updates and C cleansings. (An "update" occurs when 'cover' sets LLINK(r) or 'hide' sets ULINK(d). A "cleansing" occurs when 'commit' calls 'purify' or 'purify' sets COLOR(q) $\leftarrow -1$.) Estimate that level l does $D(U' + \theta \cdot U)$ updates and DC cleansings, where U' is the number of updates just done in step C4. Then set $D \leftarrow \theta \cdot D$.

Step C6 now should do absolutely nothing. Steps C7 and C8 don't change.

Upon termination, all data structures will have been returned to their original state, ready for another random trial. These steps will have estimated the number of nodes, updates, and cleansings at each level. Sum these estimates to get the *total* estimated number of nodes, updates, and cleansings.

87. Use $2n$ primary items a_i, d_j for the "across" and "down" words, together with n^2 secondary items ij for the individual cells. Also use W secondary items w, one for each legal word. The XCC problem has $2Wn$ options, namely 'a_i $i1{:}c_1\ \ldots\ in{:}c_n$ $c_1 \ldots c_n$' and 'd_j $1j{:}c_1\ \ldots\ nj{:}c_n$ $c_1 \ldots c_n$' for $1 \le i, j \le n$ and each legal word $c_1 \ldots c_n$. (See (110).)

We can avoid having both a solution and its transpose by introducing W further secondary items w' and appending $c_1 \ldots c_n{}'$ at the right of each option for a_1 and d_1. Then exercise 83's variant of Algorithm C will never choose a word for d_1 that it has already tried for a_1. (Think about it.)

But this construction is *not* a win for "dancing links," because it causes massive amounts of data to go in and out of the active structure. For example, with the five-letter words of WORDS(5757), it correctly finds all 323,264 of the double word squares, but its running time is 15 *tera*mems! Much faster is to use the algorithm of exercise

sorted
profile
search tree
backtrack tree
updates
cleansings

7.2.2–28, which needs only 46 gigamems to discover all of the 1,787,056 unrestricted word squares; the double word squares are easily identified among those solutions.

88. One could do a binary search, trying varying values of W. But the best way is to use the construction of exercise 87 together with the minimax variant of Algorithm C in exercise 84. This works perfectly, when the options for most common words come first.

Indeed, this method finds the double square '`BLAST|EARTH|ANGER|SCOPE|TENSE`' and proves it best in just 64 Gμ, almost as fast as the specialized method of exercise 7.2.2–28. (That square contains `ARGON`, the 1720th most common five-letter word, in its third column; the next-best squares use `PEERS`, which has rank 1800.)

89. The "minimax" method of exercise 88 finds the first five squares of

```
I S      M A Y      S H O W      S T A R T      C H E S T S      H E R T Z E S
T O      A G E      N O N E      T H R E E      L U S T R E      O P E R A T E
         N O T      O P E N      R O O F S      O B T A I N      M I M I C A L
                    W E S T      A S S E T      A R E N A S      A C E R A T E
                                 P E E R S      C I R C L E      G E N E T I C
                                                A S S E S S      E N D M O S T
                                                                 R E S E N T S
```

in respectively 200 Kμ, 15 Mμ, 450 Mμ, 25 Gμ, 25.6 Tμ. It struggles to find the best 6×6, because too few words are cut off from the search; and it thrashes miserably with the 24 thousand 7-letter words, because those words yield only seven extremely esoteric solutions. For those lengths it's best to cull the 2038753 and 14513 *unrestricted* word squares, which the method of exercise 7.2.2–28 finds in respectively 4.6 Tμ and 8.7 Tμ.

90. An XCC problem works nicely, as in answer 88: There are $2p$ primary items a_i and d_i for the final words, and $pn + W$ secondary items ij and w for the cells and potential words, where $0 \le i < p$ and $1 \le j \le n$. The Wp options going across are 'a_i $i1{:}c_1$ $i2{:}c_2$ $\ldots$ $in{:}c_n$ $c_1 \ldots c_n$'. The Wp options going down are 'd_i $i1{:}c_1$ $((i{+}1) \bmod p)2{:}c_2$ $\ldots$ $((i{+}n{-}1) \bmod p)n{:}c_n$ $c_1 \ldots c_n$' for left-leaning stairs; 'd_i $i1{:}c_n$ $((i{+}1) \bmod p)2{:}c_{n-1}$ $\ldots$ $((i{+}n{-}1) \bmod p)n{:}c_1$ $c_1 \ldots c_n$' for right-leaning stairs. The modification to Algorithm C in exercise 83 saves a factor of $2p$; and the minimax modification in exercise 84 hones in quickly on optimum solutions.

There are no left word stairs for $p = 1$, since we need two distinct words. The left winners for $2 \le p \le 10$ are: '`WRITE|WHOLE`'; '`MAKES|LIVED|WAXES`'; '`THERE|SHARE|WHOLE|WHOSE`'; '`STOOD|THANK|SHARE|SHIPS|STORE`'; '`WHERE|SHEEP|SMALL|STILL|WHOLE|SHARE`'; '`MAKES|BASED|TIRED|WORKS|LANDS|LIVES|GIVES`'; '`WATER|MAKES|LOVED|GIVES|LAKES|BASED|NOTES|BONES`'; '`WHERE|SHEET|STILL|SHALL|WHITE|SHAPE|STARS|WHILE|SHORE`'; '`THERE|SHOES|SHIRT|STONE|SHOOK|START|WHILE|SHELL|STEEL|SHARP`'. They all belong to `WORDS(500)`, except that $p = 8$ needs `WORDS(504)` for `NOTED`.

The right winners have a bit more variety: '`SPOTS`'; '`STALL|SPIES`'; '`STOOD|HOLES|LEAPS`'; '`MIXED|TEARS|SLEPT|SALAD`'; '`YEARS|STEAM|SALES|MARKS|DRIED`'; '`STEPS|SEALS|DRAWS|KNOTS|TRAPS|DROPS`'; '`TRIED|FEARS|SLIPS|SEAMS|DRAWS|ERECT|TEARS`'; '`YEARS|STOPS|HOOKS|FRIED|TEARS|SLANT|SWORD|SWEEP`'; '`START|SPEAR|SALES|TESTS|STEER|SPEAK|SKIES|SLEPT|SPORT`'; '`YEARS|STOCK|HORNS|FUELS|BEETS|SPEED|TEARS|PLANT|SWORD|SWEEP`'. They belong to `WORDS(1300)` except when p is 2 or 3.

[Arrangements equivalent to left word stairs were introduced in America under the name "Flower Power" by Will Shortz in *Classic Crossword Puzzles* (Penny Press, February 1976), based on Italian puzzles called "Incroci Concentrici" in *La Settimana Enigmistica*. Shortly thereafter, in *GAMES* magazine and with $p = 16$, he called them "Petal Pushers," usually based on six-letter words but occasionally going to seven. Left

binary search
minimax
minimax
minimax
Flower Power
Shortz
Incroci Concentrici
Petal Pushers

word stairs are much more common than the right-leaning variety, because the latter mix end-of-word with beginning-of-word letter statistics.]

91. Consider all "kernels" $c_1 \ldots c_{14}$ that can appear as illustrated, within a right word stair of 5-letter words. Such kernels arise for a given set of words only if there are letters $x_1 \ldots x_{12}$ such that $x_3x_4x_5c_2c_3$, $c_4c_5c_6c_7c_8$, $c_9c_{10}c_{11}c_{12}x_6$, $c_{13}c_{14}x_7x_8x_9$, $x_1x_2x_5c_5c_9$, $c_1c_2c_6c_{10}c_{13}$, $c_3c_7c_{11}c_{14}x_{10}$, and $c_8c_{12}x_7x_{11}x_{12}$ are all in the set. Thus it's an easy matter to set up an XCC problem that will find the multiset of kernels, after which we can extract the set of *distinct* kernels.

$$\begin{array}{cccccccc}
 & & x_1 & & & & & \\
 & & x_2 & c_1 & & & & \\
x_3 & x_4 & x_5 & c_2 & c_3 & & & \\
 & c_4 & c_5 & c_6 & c_7 & c_8 & & \\
 & & c_9 & c_{10} & c_{11} & c_{12} & x_6 & \\
 & & & c_{13} & c_{14} & x_7 & x_8 & x_9 \\
 & & & & x_{10} & x_{11} & & \\
 & & & & & x_{12} & &
\end{array}$$

Construct the digraph whose arcs are the kernels, and whose vertices are the 9-tuples that arise when kernel $c_1 \ldots c_{14}$ is regarded as the transition

$$c_1c_2c_3c_4c_5c_6c_7c_9c_{10} \;\rightarrow\; c_3c_7c_8c_9c_{10}c_{11}c_{12}c_{13}c_{14}.$$

This transition contributes two words, $c_4c_5c_6c_7c_8$ and $c_1c_2c_6c_{10}c_{13}$, to the word stair. Indeed, *right word stairs of period p are precisely the p-cycles in this digraph for which the $2p$ contributed words are distinct.*

Now we can solve the problem, if the graph isn't too big. For example, `WORDS`(1000) leads to a digraph with 180524 arcs and 96677 vertices. We're interested only in the oriented cycles of this (very sparse) digraph; so we can reduce it drastically by looking only at the largest induced subgraph for which each vertex has positive in-degree and positive out-degree. (See exercise 7.1.4–234, where a similar reduction was made.) And wow: That subgraph has only 30 vertices and 34 arcs! So it is totally understandable, and we deduce quickly that the longest right word stair belonging to `WORDS`(1000) has $p = 5$. That word stair, which we found directly in answer 90, corresponds to the cycle

$$\texttt{SEDYEARST} \rightarrow \texttt{DRSSTEASA} \rightarrow \texttt{SAMSALEMA} \rightarrow \texttt{MESMARKDR} \rightarrow \texttt{SKSDRIEYE} \rightarrow \texttt{SEDYEARST}.$$

A similar approach applies to left word stairs, but the kernel configurations are reflected left-to-right; transitions then contribute the words $c_8c_7c_6c_5c_4$ and $c_1c_2c_6c_{10}c_{13}$. The digraph from `WORDS`(500) turns out to have 136771 arcs and 74568 vertices; but this time 6280 vertices and 13677 arcs remain after reduction. Decomposition into strong components makes the task simpler, because every cycle belongs to a strong component. Still, we're stuck with a giant component that has 6150 vertices and 12050 arcs.

The solution is to reduce the current subgraph repeatedly as follows: Find a vertex v of out-degree 1. Backtrack to discover a simple path, from v, that contributes only distinct words. If there is no such path (and there usually isn't, and the search usually terminates quickly), remove v from the graph and reduce it again.

With this method one can rapidly show that the longest left word stair from `WORDS`(500) has period length 36: '`SHARE|SPENT|SPEED|WHEAT|THANK|CHILD|SHELL|SHORE|STORE|STOOD|CHART|GLORY|FLOWS|CLASS|NOISE|GAMES|TIMES|MOVES|BONES|WAVES|GASES|FIXED|TIRED|FEELS|FALLS|WORLD|ROOMS|WORDS|DOORS|PARTY|WANTS|WHICH|WHERE|SHOES|STILL|STATE`', with 36 other words that go down. Incidentally, `GLORY` and `FLOWS` have ranks 496 and 498, so they just barely made it into `WORDS`(500).

Larger values of W are likely to lead to quite long cycles from `WORDS`(W). Their discovery won't be easy, but the search will no doubt be instructive.

92. Use $3p$ primary items a_i, b_i, d_i for the final words; $pn + 2W$ secondary items ij, w, w' for the cells and potential words, with $0 \le i < p$ and $1 \le j \le n$ (somewhat as in answer 90). The Wp options going across are 'a_i $i1{:}c_1$ $i2{:}c_2$ $\ldots$ $in{:}c_n$ $c_1 \ldots c_n$

kernels
induced subgraph
in-degree
out-degree
strong components

Mathews
Brotchie
Oulipo
Flye Sainte-Marie
MRV heuristic
focus
universal cycles
ourotoruses

$c_1 \ldots c'_n$'. The $2Wp$ options going down in each way are 'b_i $i1{:}c_1$ $((i{+}1) \bmod p)2{:}c_2$ $\ldots$ $((i{+}n{-}1) \bmod p)n{:}c_n$ $c_1 \ldots c_n$' and 'd_i $i1{:}c_n$ $((i{+}1) \bmod p)2{:}c_{n-1}$ $\ldots$ $((i{+}n{-}1) \bmod p)n{:}c_1$ $c_1 \ldots c_n$'. The items w' at the right of the a_i options save us a factor of p.

Use Algorithm C (modified). We can't have $p = 1$. Then comes 'SPEND|SPIES'; 'WAVES|LINED|LEPER'; 'LOOPS|POUTS|TROTS|TOONS'; 'SPOOL|STROP|STAID|SNORT|SNOOT';'DIMES|MULES|RIPER|SIRED|AIDED|FINED';'MILES|LINTS|CARES|LAMED|PIPED|SANER|LIVER';'SUPER|ROVED|TILED|LICIT|CODED|ROPED|TIMED|DOMED';'FORTH|LURES|MIRES|POLLS|SLATS|SPOTS|SOAPS|PLOTS|LOOTS'; 'TIMES|FUROR|RUNES|MIMED|CAPED|PACED|LAVER|FINES|LIMED|MIRES'. (Lengthy computations were needed for $p \geq 8$.)

93. Now $p \leq 2$ is impossible. A construction like the previous one allows us again to save a factor of p. (There's also top/bottom symmetry, but it is somewhat harder to exploit.) Examples are relatively easy to find, and the winners are 'MILES|GALLS|BULLS'; 'FIRES|PONDS|WALKS|LOCKS'; 'LIVES|FIRED|DIKES|WAVED|TIRES'; 'BIRDS|MARKS|POLES|WAVES|WINES|FONTS'; 'LIKED|WARES|MINES|WINDS|MALES|LOVES|FIVES'; 'WAXES|SITES|MINED|BOXES|CAVES|TALES|WIRED|MALES';'CENTS|HOLDS|BOILS|BALLS|MALES|WINES|FINDS|LORDS|CARES'; 'LOOKS|ROADS|BEATS|BEADS|HOLDS|COOLS|FOLKS|WINES|GASES|BOLTS'. [Such patterns were introduced by Harry Mathews in 1975, who gave the four-letter example 'TINE|SALE|MALE|VINE'. See H. Mathews and A. Brotchie, *Oulipo Compendium* (London: Atlas, 1998), 180–181.]

94. Set up an XCC problem with primary items k, p_k, and secondary items x_k, for $0 \leq k < 16$, and with options 'j p_k $x_k{:}a$ $x_{(k+1) \bmod 16}{:}b$ $x_{(k+3) \bmod 16}{:}c$ $x_{(k+4) \bmod 16}{:}d$' for $0 \leq j, k < 16$, where $j = (abcd)_2$. The solution (0000011010111011) is essentially unique (except for cyclic permutation, reflection, and complementation). [See C. Flye Sainte-Marie, L'*Intermédiaire des Mathématiciens* **3** (1896), 155–161.]

95. Use $2m$ primary items a_k, b_k, and m secondary items x_k, for $0 \leq k < m$. Define m^2 options of size $2 + n$, namely 'a_j b_k $x_j{:}t_1$ $x_{(j+1) \bmod m}{:}t_2$ $\ldots$ $x_{(j+n-1) \bmod m}{:}t_n$', where $t_1 t_2 \ldots t_n$ is the kth binary vector of interest. However, save a factor of m by omitting the options with $j = 0$ and $k > 0$, and the options with $j > 0$ and $k = 0$.

The case $(7, 0, 3)$ has 137216 solutions, found in 8.5 gigamems; the case $(7, 3, 4)$ has 41280 solutions, found in 3.2 gigamems. (We can make the items b_k secondary instead of primary. This makes the search tree a bit larger. But it actually *saves* a little time, because the MRV heuristic causes branching on a_j and maintains a good focus; less time is spent computing that heuristic when b_k isn't primary. Alternatively we could make the items a_k secondary (or even omit them entirely, which would have the same effect). But that would be a disaster! For example, the running time for case $(7, 0, 3)$ would then increase to nearly 50 *tera*mems, because focus is lost.)

Section 7.2.1 discusses other "universal cycles," which can be handled similarly.

96. In fact, there are 80 solutions for which the bottom four rows are the complements of the top four. (This problem extends the idea of "ourotoruses" in exercise 7.2.1.1–109. One can also consider windows that aren't rectangles. For example, the thirty-two ways to fill a cross of five cells can be identified with 32 positions of the generalized torus whose offsets are $(4, \pm 4)$; see exercise 7–137.)

00000110
00010111
11001010
10001110
11111001
11101000
00110101
01110001

97. Use primary items jk, p_{jk}, and secondary items $d_{j,k}$, for $0 \leq j < 3$ and $0 \leq k < 9$, with the following three options for each $0 \leq i, j < 3$ and $0 \leq k, k' < 9$: 'jk $p_{j'k'}$ $d_{j',k'}{:}i$ $d_{j',k'+1}{:}(i+a)$ $d_{j'+1,k'}{:}(i+b)$ $d_{j'+1,k'+1}{:}(i+c)$', for $0 \leq j' \leq 1$, and 'jk $p_{2k'}$ $d_{2,k'}{:}i$ $d_{2,k'+1}{:}(i{+}a)$ $d_{0,k'-3}{:}(i{+}b{-}1)$ $d_{0,k'-2}{:}(i{+}c{-}1)$', where $9j + k = (abc)_3$; sums involving i are mod 3, while sums involving k' are mod 9. We can assume that 00 is paired with p_{00}. Then there are $2 \cdot 2898 = 5796$ solutions D; all have $D \neq D^T$.

3SAT
all-different
stable extensions
Bittencourt

98. Given a 3SAT problem with clauses $(l_{i1} \vee l_{i2} \vee l_{i3})$ for $1 \le i \le m$, with each $l_{ij} \in \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$, construct an XCC problem with $3m$ primary items ij $(1 \le i \le m, 1 \le j \le 3)$ and n secondary items x_k $(1 \le k \le n)$, having the following options: (i) '$l_{i1}\ l_{i2}$', '$l_{i2}\ l_{i3}$', '$l_{i3}\ l_{i1}$'; (ii) '$l_{ij}\ x_k$:1' if $l_{ij} = x_k$, '$l_{ij}\ x_k$:0' if $l_{ij} = \bar{x}_k$. That problem has a solution if and only if the given clauses are satisfiable.

99. True — but perhaps with many more secondary items and much longer options: Let x be a secondary item to which a color has been assigned, in some XCC problem A; and let O be the options in which x appears. Replace A by a new problem A', by deleting item x and adding new secondary items $x_{\{o,p\}}$ for each $o, p \in O$ for which x gets different colors in A. And for each $o \in O$, replace item x in o by the set of all $x_{\{o,p\}}$ that apply. If A' still involves colors, replace it by A'' in a similar way, until all colors disappear.

100. (a) There are five solutions: 00112, 00122, 01112, 01122, 11111.

(b) Let there be five primary items, $\{\#1, \#2, \#3, \#4, \#5\}$, and five secondary items, $\{x_1, x_2, x_3, x_4, x_5\}$. Item #1 enforces the binary constraint $x_1 \le x_2$, and has the options '#1 x_1:0 x_2:0'; '#1 x_1:0 x_2:1'; '#1 x_1:0 x_2:2'; '#1 x_1:1 x_2:1'; '#1 x_1:1 x_2:2'; '#1 x_1:2 x_2:2'. Similar options for #2, #3, and #4 will enforce the constraints $x_2 \le x_3$, $x_3 \le x_4$, and $x_4 \le x_5$. Finally, the options '#5 x_1:0 x_3:1 x_5:2'; '#5 x_1:0 x_3:2 x_5:1'; '#5 x_1:1 x_3:0 x_5:2'; '#5 x_1:1 x_3:1 x_5:1'; '#5 x_1:1 x_3:2 x_5:0'; '#5 x_1:2 x_3:0 x_5:1'; '#5 x_1:2 x_3:1 x_5:0' will enforce the ternary constraint $x_1 + x_3 + x_5 = 3$.

(c) Use primary items #j for $1 \le j \le m$, one for each constraint, and secondary items x_k for $1 \le k \le n$, one for each variable. If constraint C_j involves the d variables $x_{i_1}, \ldots, x_{i_d}$, include options '#j x_{i_1}:a_1 ... x_{i_d}:a_d' for each legal d-tuple $(a_1, \ldots, a_d)$.

(Of course this construction isn't efficient for all instances of CSP; furthermore, we can often find substantially better ways to encode a particular CSP as an XCC instance, because this method uses only one primary item in each option. But the idea that underlies this construction is a useful mental tool when formulating particular problems.)

101. Notice that the final sentence implies two further clues:

- Somebody trains a zebra.
- Somebody prefers to drink just plain water.

Let there be primary items #k for $1 \le k \le 16$, one for each clue. And let the $5 \cdot 5$ secondary items N_j, J_j, P_j, D_j, C_j represent the nationality, job, pet, drink, and color associated with house j, for $0 \le j < 5$. There are respectively (5, 5, 5, 5, 1, 5, 1, 5, 4, 5, 8, 5, 8, 8, 5, 5) options for clues (1, ..., 16), typified by '#1 N_j:England C_j:red', for $0 \le j < 5$; '#5 N_0:Norway'; '#9 C_i:white C_{i+1}:green', for $0 \le i < 4$; '#14 J_i:nurse P_{i+1}:fox', '#14 P_i:fox J_{i+1}:nurse', for $0 \le i < 4$; '#15 P_j:zebra', for $0 \le j < 5$.

A more complex formulation enforces the redundant "all-different" constraint by introducing $5 \cdot 5$ additional secondary items to represent the *inverses* of N_j, J_j, P_j, D_j, C_j. For example, the options for #1 then become '#1 N_j:England N^-_{England}:j C_j:red C^-_{red}:j'. (With those additional items, Algorithm C will infer C_1:blue immediately from #5 and #11; but without them, #5 doesn't immediately make N_1:Norway illegal. They reduce the search tree size from 112 to 32 nodes. However, the time they save during the search just barely compensates for the extra time that they consume in step C1.)

The inverses alone are *not* sufficient; they don't forbid, say, $N^-_{\text{England}} = N^-_{\text{Japan}}$.

[The author of this now-famous puzzle is unknown. Its first known publication, in *Life International* **35** (17 December 1962), 95, used cigarettes instead of occupations.]

102. As in answer 7.2.2–68, let's find all stable extensions of a given partially labeled digraph. And let's allow sinks too; we can assume that every vertex with out-degree $d \le 1$ is labeled d. The following XCC formulation is based on ideas of R. Bittencourt.

Let Δ be the maximum out-degree. Introduce primary items H_v, I_v, E_{vd}, and secondary items v, h_{vd}, i_{vd}, for $0 \le d \le \Delta$ and all vertices v. The color of v will be $\lambda(v)$, the label of v; the color of h_{vd} will denote the Boolean quantity '$[v$ sees $d]$', meaning that $\lambda(w) = d$ for some w with $v \longrightarrow w$; and the color of i_{vd} will denote '$[\lambda(v) = d]$'. The options for H_v are 'H_v $v{:}d$ $\bigcup_{k=0}^{\Delta}\{h_{vk}{:}e_k\}$' where $e_0 + \cdots + e_\Delta = d$. The options for I_v are 'I_v $v{:}d$ $\bigcup_{k=0}^{\Delta}\{i_{vk}{:}[k = d]\}$ $\bigcup_{u\longrightarrow v}\{h_{ud}{:}1\}$'. And the options for E_{vd} are 'E_{vd} $h_{vd}{:}1$ $i_{w_k d}{:}1$ $\bigcup_{j=1}^{k-1}\{i_{w_j d}{:}0\}$' for $1 \le k \le d^+(v)$ and 'E_{vd} $h_{vd}{:}0$ $\bigcup_{j=1}^{d^+(v)}\{i_{w_j d}{:}0\}$', when $v \longrightarrow w_1$, …, $v \longrightarrow w_{d^+(v)}$. For example, if the vertices of the puzzle in exercise 7.2.2–68 are named 00, …, 99, some of the options of its unique solution are 'H_{00} 00:3 h_{000}:0 h_{001}:0 h_{002}:0 h_{003}:0 h_{004}:1 h_{005}:0 h_{006}:0 h_{007}:1 h_{008}:0 h_{009}:1'; 'I_{00} 00:3 i_{000}:0 … i_{002}:0 i_{003}:1 i_{004}:0 … i_{009}:0 h_{013}:1 h_{033}:1 h_{053}:1 h_{703}:1 h_{803}:1'; 'E_{004} h_{004}:1 i_{104}:0 … i_{404}:0 i_{504}:1'.

$d^+(v)$: out-degree
pi day puzzle
Bittencourt
Lehmer
Gilbert
totient function φ

Of course many of those options can be greatly simplified, because many of the quantities are known from the given labels. We know the color of i_{vd} when $\lambda(v)$ is given; we know the color of h_{vd} when v sees d in the given puzzle. We don't need I_v when v is labeled; we don't even need E_{vd}, when v is known to see d. If v has out-degree d and already sees some label twice, we know that i_{vd} is 0. And so on. In the pi day puzzle such simplifications reduce 60 thousand options on $1200+1831$ items to 11351 options on $880+1216$ items. That's still a lot, and Algorithm C needs 135 Mμ to input them; but then it finds the solution and proves it unique after 25 more Mμ. (The highly tuned method of answer 7.2.2–68 needed only 7 Mμ to prove uniqueness. But that method solves only a small class of problems that happen to reduce nicely.)

Bittencourt notes that further speedup is possible when two arrows point in the same direction. (This happens 123 times in the pi day puzzle.) In general if $v \longrightarrow w$ implies $u \longrightarrow w$, we must have $\lambda(u) \ge \lambda(v)$; and this condition can be enforced by introducing a new primary variable whose options allow u and v to have only appropriate combinations of colors.

103. (a) An all-interval row always has $x_{n-1} = \bigl(x_0 + 1 + \cdots + (n-1)\bigr) \bmod n = \bigl(x_0 + n(n-1)/2\bigr) \bmod n = \bigl(x_0 + [n \text{ even}]\, n/2\bigr) \bmod n$.

(b) Let j, p_j, d_k, q_k be primary items and let x_j be a secondary item, for $0 \le j < n$ and $1 \le k < n$. There's an option 'j p_t $x_j{:}t$' for $0 \le j,t < n$, omitted when ($j = 0$ and $t \ne 0$) or ($j = n-1$ and $t \ne n/2$). And there's an option 'd_k q_t $x_{t-1}{:}i$ $x_t{:}(i+k) \bmod n$' for $1 \le k,t < n$ and $0 \le i < n$. Then the tone row and its intervals are permutations.

There are (1, 2, 4, 24, 288, 3856) solutions for $n = (2, 4, 6, 8, 10, 12)$. [These values were first computed by D. H. Lehmer, *Proc. Canadian Math. Congress* **4** (1959), 171–173, for $n = 12$ and E. N. Gilbert *SIAM Review* **7** (1965), 189–198, for $n < 12$.]

For larger n, Algorithm C is *not* at all competitive with a straightforward backtrack algorithm, which uses Algorithm 7.2.1.2X to find all suitable permutations of the $n-1$ intervals: That method needs only 100 Mμ to find all 89328 solutions when $n = 14$, compared to 107 Gμ by Algorithm C! With backtracking we can generate all 2755968 solutions for $n = 16$ in 4.7 Gμ, and all 103653120 solutions for $n = 18$ in 281 Gμ.

(c) The intervals between adjacent classes in x^Q are the same as those of x, except that $x_k - x_{k-1}$ is replaced by $x_0 - x_{n-1}$. And we know that $x_0 - x_{n-1} = \pm n/2$.

(d) True; both are $x_{k-1} \ldots x_0 x_{n-1} \ldots x_k$.

(e) The solution for $n = 2$ has every possible symmetry; and the solution $x = 0132$ for $n = 4$ is equivalent to x^R, $-x^Q$, and $-x^{QR}$. For $n = (6, 8, 10, 12)$ we can argue that x can be equivalent to at most one of the $4\varphi(n)$ rows cx, cx^R, cx^Q, cx^{QR} besides itself: We can't have $x \equiv cx$ when $c \ne 1$. We can have $x \equiv cx^R$ or cx^Q only if $c^2 \bmod n = 1$. The fact that $ct \bmod n = t$ has an odd number of solutions $0 < t < n$ shows that

$x \equiv x^R$ or $x \equiv -x^Q$ are the only possibilities. A more complicated case analysis is necessary when analyzing solutions to $x \equiv cx^{QR}$; Gilbert stated incorrectly [page 196] that no such solutions exist. (He overlooked 12-tone rows such as 0 3 9 1 2 4 11 8 7 5 10 6, 0 1 4 9 3 11 10 8 5 7 2 6, 0 1 8 11 10 3 9 5 7 4 2 6, for which $x \equiv 5x^{QR}$. Similarly, the 20-tone row 0 1 3 11 2 19 13 9 12 7 14 18 4 17 16 8 6 15 5 10 satisfies $x \equiv 9x^{QR}$.)

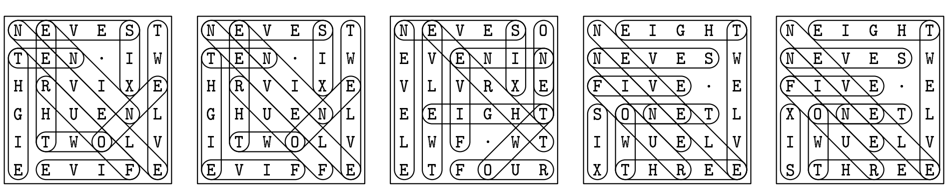
At any rate, the transformations of (c) partition the solutions into clusters of size $2\varphi(n)$ when there's symmetry, $4\varphi(n)$ when there's not. Gilbert enumerated the cases of symmetry when $n < 12$; R. Morris and D. Starr did it when $n = 12$ [*J. Music Theory* **18** (1974), 364–389]. For $n = (6, 8, 10, 12, 14, 16, 18)$ the number of clusters with $x \equiv x^R$ turns out to be respectively (1, 1, 6, 22, 48, 232, 1872); the number of clusters with $x \equiv -x^Q$ turns out to be (0, 0, 2, 15, 0, 0, 1346); also $n = 12$ has 15 cases with $x \equiv 5x^{QR}$.

104. We may assume that $x_0 = 0$. There's a constant c_r such that $y_{kr} \equiv x_{k-1} + c_r$ (modulo n) for $1 \le k \le n$. Thus $y_r = x_{r-1} \equiv c_r$; $y_{r^2} = x_{(r^2-1) \bmod p} \equiv x_{r-1} + c_r \equiv 2c_r$; $y_{r^3} = x_{(r^3-1) \bmod p} \equiv x_{(r^2-1) \bmod p} + c_r \equiv 3c_r$; etc. Let r be primitive modulo p, so that $\{r \bmod p, \ldots, r^n \bmod p\} = \{1, \ldots, p-1\}$, and let $R = r^d$ where $c_r d \bmod n = 1$. Then we've proved $R^{x_{(r^k-1) \bmod p}} \equiv (r^k \bmod p)$ (modulo p) for $1 \le k \le n$; that is, $R^{x_{k-1}} \equiv k$.

Now suppose $x_k - x_{k-1} \equiv x_l - x_{l-1}$ (modulo n). Then $R^{x_k} R^{x_{l-1}} \equiv R^{x_{k-1}} R^{x_l}$ (modulo p); consequently $(k+1)l \equiv k(l+1)$ (modulo p), hence $k = l$.

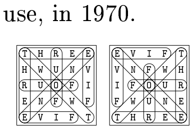
(b) $x^{(n)} = x^R$. [See the papers by Lehmer and Gilbert in answer 103.]

105. There are just five solutions; the latter two are flawed by being disconnected:



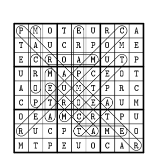
Historical notes: The earliest known word search puzzle was "Viajando" by Henrique Ramos of Brazil, published in *Almanaque de Seleções Recreativas* (1966), page 43. Such puzzles were independently invented in America by Norman E. Gibat (1968). Jo Ouellet of Canada developed "Wonderword," which puts the unused letters to use, in 1970.

106. When Algorithm C is generalized to allow non-unit item sums as in Algorithm M, it needs just 24 megamems to prove that there are exactly eight solutions — which all are rotations of the two shown here.



107. To pack w given words, use primary items $\{\mathrm{P}ij, \mathrm{R}ic, \mathrm{C}ic, \mathrm{B}ic, \#k \mid 1 \le i,j \le 9,\ 1 \le k \le w,\ c \in \{\mathtt{A},\mathtt{C},\mathtt{E},\mathtt{M},\mathtt{O},\mathtt{P},\mathtt{R},\mathtt{T},\mathtt{U}\}\}$ and secondary items $\{ij \mid 1 \le i,j \le 9\}$. There are 729 options '$\mathrm{P}ij$ $\mathrm{R}ic$ $\mathrm{C}jc$ $\mathrm{B}bc$ $ij{:}c$', where $b = 3\lfloor (i-1)/3 \rfloor + \lceil j/3 \rceil$, together with an option '$\#k$ $i_1j_1{:}c_1$ $\ldots$ $i_lj_l{:}c_l$' for each placement of an l-letter word $c_1 \ldots c_l$ into cells $(i_1, j_1), \ldots, (i_l, j_l)$. Furthermore, it's important to use the sharp preference heuristic (exercise 10) in step C3 of the algorithm.

A brief run then establishes that `COMPUTER` and `CORPORATE` cannot both be packed. But all of the words *except* `CORPORATE` do fit together; the (unique) solution shown is found after only 7.3 megamems, most of which are needed simply to input the problem. [This exercise was inspired by a puzzle in *Sudoku Masterpieces* (2010) by Huang and Snyder.]

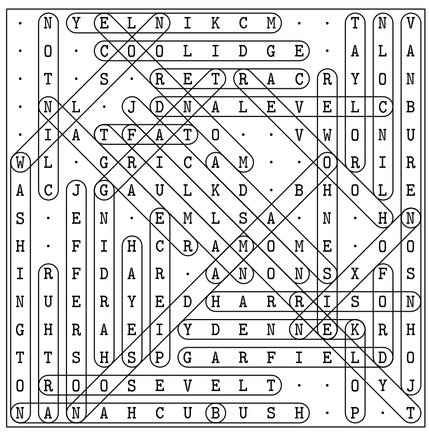


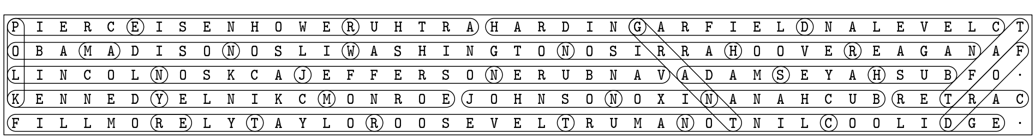
108. (a, b) The author's best solutions, thought to be minimal (but there is no proof), are below. In both cases, and in Fig. 71, an interactive method was used: After the

Morris
Starr
primitive
Lehmer
Gilbert
disconnected
Historical notes
Ramos
Gibat
Ouellet
Wonderword
sharp preference
branch, choice of
choice of item to cover
Huang
Snyder
author
interactive method

longest words were placed strategically by hand, Algorithm C packed the others nicely.

Gordon
Eckler





[Solution (b) applies an idea by which Leonard Gordon was able to pack the names of presidents 1–42 with one less column. See A. Ross Eckler, *Word Ways* **27** (1994), 147; see also page 252, where `OBAMA` miraculously fits into Gordon's 15×15 solution!]

109. To pack w given words, use $w + m(n-1) + (m-1)n$ primary items $\{\#k \mid 1 \le k \le w\}$ and $\{\mathrm{H}ij, \mathrm{V}ij \mid 1 \le i \le m,\ 1 \le j \le n\}$, but with $\mathrm{H}in$ and $\mathrm{V}mj$ omitted; $\mathrm{H}ij$ represents the edge between cells (i,j) and $(i,j+1)$, and $\mathrm{V}ij$ is similar. There also are $2mn$ secondary items $\{ij, ij' \mid 1 \le i \le m,\ 1 \le j \le n\}$. Each horizontal placement of the kth word $c_1 \ldots c_l$ into cells $(i,j+1), \ldots, (i,j+l)$ generates the option

$$\#k\ ij{:}\texttt{.}\ \ ij'{:}\texttt{0}\ \ i(j{+}1){:}c_1\ \ i(j{+}1)'{:}\texttt{1}\ \ \mathrm{H}i(j{+}1)\ \ i(j{+}2){:}c_2\ \ i(j{+}2)'{:}\texttt{1}\ \ \mathrm{H}i(j{+}2)\ \ \ldots$$
$$\mathrm{H}i(j{+}l{-}1)\ \ i(j{+}l){:}c_l\ \ i(j{+}l)'{:}\texttt{1}\ \ i(j{+}l{+}1){:}\texttt{.}\ \ i(j{+}l{+}1)'{:}\texttt{0}$$

with $3l+4$ items, except that '$ij{:}\texttt{.}\ ij'{:}\texttt{0}$' is omitted when $j = 0$ and '$i(j{+}l{+}1){:}\texttt{.}\ i(j{+}l{+}1)'{:}\texttt{0}$' is omitted when $j{+}l = n$. Each vertical placement is similar. For example,

$$\#1\ \ 11{:}\texttt{Z}\ \ 11'{:}\texttt{1}\ \ \mathrm{V}11\ \ 21{:}\texttt{E}\ \ 21'{:}\texttt{1}\ \ \mathrm{V}21\ \ 31{:}\texttt{R}\ \ 31'{:}\texttt{1}\ \ \mathrm{V}31\ \ 41{:}\texttt{O}\ \ 41'{:}\texttt{1}\ \ 51{:}\texttt{.}\ \ 51'{:}\texttt{0} \qquad (*)$$

is the first vertical placement option for `ZERO`, if `ZERO` is word #1. When $m = n$, however, we save a factor of 2 by omitting all of the vertical placements of word #1.

To enforce the tricky condition (ii), we also introduce $3m(n-1) + 3(m-1)n$ options:

$$\begin{array}{ll}
\mathrm{H}ij\ \ ij'{:}\texttt{0}\ \ i(j{+}1)'{:}\texttt{1}\ \ ij{:}\texttt{.} & \mathrm{V}ij\ \ ij'{:}\texttt{0}\ \ (i{+}1)j'{:}\texttt{1}\ \ ij{:}\texttt{.}\\
\mathrm{H}ij\ \ ij'{:}\texttt{1}\ \ i(j{+}1)'{:}\texttt{0}\ \ i(j{+}1){:}\texttt{.} & \mathrm{V}ij\ \ ij'{:}\texttt{1}\ \ (i{+}1)j'{:}\texttt{0}\ \ (i{+}1)j{:}\texttt{.}\\
\mathrm{H}ij\ \ ij'{:}\texttt{0}\ \ i(j{+}1)'{:}\texttt{0}\ \ ij{:}\texttt{.}\ \ i(j{+}1){:}\texttt{.} & \mathrm{V}ij\ \ ij'{:}\texttt{0}\ \ (i{+}1)j'{:}\texttt{0}\ \ ij{:}\texttt{.}\ \ (i{+}1)j{:}\texttt{.}
\end{array}$$

This construction works nicely because each edge must encounter either a word that crosses it or a space that touches it. (Beware of a slight glitch: A valid solution to the puzzle might have several compatible choices for $\mathrm{H}ij$ and $\mathrm{V}ij$ in "blank" regions.)

Important: As in answer 107, the sharp preference heuristic should be used here, because it gives an enormous speedup.

The XCC problem for our 11-word example has 1192 options, $123 + 128$ items, and 9127 solutions, found in 29 Gμ. But only 20 of those solutions are connected; and they yield only the three distinct word placements below. A slightly smaller rectangle, 7×9, also has three valid placements. The smallest rectangle that admits a solution to (i) and (ii) is 5×11; that placement is *unique*, but it has two components:

```
   F  F
ZERO SIX
 I U  V
 G R TEN
 H   H I
 TWO R N
   N E E
  SEVEN
```

```
  F TWO
  ONE  S
  U NINE
ZERO   V
 I  F  E
 G SIX N
 H  V
 THREE
```

```
  FIVE
TWO  I
  U  G
ZERO H
   N TEN
SEVEN  I
I      N
X  THREE
```

```
 E T
FIVE  SIX
 G NINE
 H    V F
 TWO ZERO
   N  N U
THREE   R
```

```
F SIX F T
ONE  EIGHT
U V T V R T
R E E E E W
  NINE ZERO
```

Instead of generating all solutions to (i) and (ii) and discarding the disconnected ones, there's a much faster way to guarantee connectedness throughout the search; but it requires major modifications to Algorithm C. Whenever no H or V is forced, we can list all active options that are connected to word #1 and not smaller than choices that could have been made earlier. Then we branch on them, instead of branching on an item. For example, if (∗) above is used to place `ZERO`, it will force H00 and H20 and V30. The next decision will be to place either `EIGHT` or `ONE`, in the places where they overlap `ZERO`. (However, we'll be better off if we order the words by decreasing length, so that for instance #1 is `EIGHT` and #11 is `ONE`.) Interested readers are encouraged to work out the instructive details. This method needs only 630 Mμ to solve the example problem, as it homes right in on the three connected solutions.

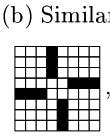
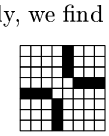
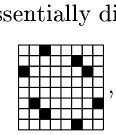
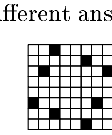
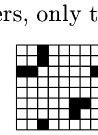
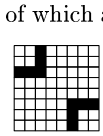
110. Gary McDonald found this remarkable 20×20 solution in 2017:

```
WILSON TAFT       P
     I A O C JOHNSON
     X Y R L E    L
T H COOLIDGE F A  K
REAGAN O   V F R
U R R  ROOSEVELT
M R T B  B L R HAYES
A I E U  ADAMS U  I
N S R C  M N O R  S
  O   H HARDING   E
  N   A O      GRANT
F   MONROE J      H
I     A V WASHINGTON
L MCKINLEY C    A W
L  L    R  K PIERCE
MADISON  BUSH   F R
O  N       O    I
R  TYLER VAN BUREN
E  O            L
 LINCOLN   KENNEDY
```

A 19×19 is surely impossible, although no proof is known. L. Gordon had previously fit the names of presidents 1–42 into an 18×22 rectangle [*Word Ways* **27** (1994), 63].

111. (a) Set up an XCC problem as in answer 109, but with just three words `AAA`, `AAAA`, `AAAAA`; then adjust the multiplicities and apply Algorithm M. The two essentially distinct answers are shown below; one of them is disconnected, hence disqualified.

(b) Similarly, we find four essentially different answers, only two of which are OK:

, ; , , , .

sharp preference heuristic
McDonald
Gordon

Algorithm M handles case (b) with ease (5 Gμ). But it does not explore the space of possibilities for case (a) intelligently, and costs 591 Gμ.

112. (a) Yes: IMMATURE, MATURING, COMMUTER, GROUPING, TROUPING, AUTHORING, and THRUMMING. A straightforward backtracking program will quickly determine the presence or absence of any given string of letters.

(b) Let's put DANCING and LINKS in there too. Then we obtain an array with 24 words from WORDS(5757) (like LOVER, ROSIN, SALVO, TOADS, TROVE); also ASKING, DOSING, LOSING, ORDAIN, SAILOR, SIGNAL, SILVER; also LANCING, LOANING, SOAKING, and even ORTOLAN. (Notice that TORTO occurs in two ways.)

C	G	N
N	I	K
L	A	S
V	O	D
E	R	T
T	O	W

To find such arrays, as suggested by R. Bittencourt, we can let word k be $c_0 \ldots c_{t-1}$, and introduce primary items W_{kl} for $1 \le l < t$ to represent the placement of $c_{l-1}c_l$. Let X_u be a secondary item, for each cell u of the array, to be colored with some letter. Represent the king path for word k by giving color u to P_{kl} and color l to Q_{ku} when c_l is in cell u, where P_{kl} and Q_{ku} are additional secondary items. There also are secondary items D_{kv}, for each internal vertex v. For example, if the cells and vertices are numbered rowwise, two of the options chosen for DANCING and LINKS in the example above are 'W_{03} X_3:N X_0:C P_{02}:3 P_{03}:0 Q_{03}:2 Q_{00}:3' and 'W_{12} X_4:I X_2:N P_{11}:4 P_{12}:2 Q_{14}:1 Q_{12}:2 D_{11}'. The 'D_{11}' in the latter will prevent another step of word 1 between cells 2 and 4.

We can save a factor of nearly 4 by restricting the placement of, say, $c_{l-1}c_l$ in word 0 when $l = \lfloor t_0/2 \rfloor$, so that c_{l-1} lies in the upper left quadrant and c_l isn't in the rightmost column. Then W_{0l} has only 26 options instead of the usual 94.

It turns out that exactly 10 essentially different Torto arrays contain DANCING, LINKS, TORTO, WORDS, and SOLVER; exactly 1444 contain THE, ART, OF, COMPUTER, and PROGRAMMING. They're found by Algorithm C in 713 Gμ and 126 Gμ, respectively.

(c) Yes, in 140 ways (but we can't add ELEVEN). Similarly, we can pack ZERO, ONE, ..., up to EIGHT, in 553 ways. And FIRST, ..., SIXTH can be packed in 72853 ways, sometimes without using more than 16 of the 18 cells. (These computations took (16, 5, 1.5) Tμ. Interesting words lurk in these arrays — can you spot them? See Appendix E.)

F	X	N		F	S	X		E	T	V
S	I	N		Z	I	E		F	H	X
G	E	V		G	E	V		F	I	T
H	R	E		H	R	N		O	R	S
U	T	N		T	O	U		U	D	E
F	O	W		W	N	F		N	O	C

[The name 'Torto' was trademarked by Coquetel/Ediouro of Rio de Janeiro in 1977, and an example appeared in issue #1 of *Coquetel Total* magazine that year. Monthly puzzles still appear regularly in Coquetel's magazine *Desafio Cérebro*. Bittencourt posed the problem of constructing Torto arrays from a given list of required words in 2011; see blog.ricbit.com/2011/05/torto-reverso.html.]

113. First, we can find all sets of six or fewer letters that could be on such a block, by solving an MCC problem with 25 primary items TREES, ..., DEQUE of multiplicity [1 . . 26] and one primary item # of multiplicity [0 . . 6]. There are 22 options, '# ABOVE AVAIL GRAPH STACK TABLE VALUE' through '# EMPTY', one for each potential letter (listing all words that include that letter). This covering problem has 3223 solutions, found in 4 Mμ and ranging alphabetically from {A, B, C, D, E, I} to {E, L, R, T} to {L, N, R, T, U, V}.

Then we set up an XCC problem with 25 primary items TREES, ..., DEQUE and five primary items 1, ..., 5, together with $5 \cdot 22$ secondary items $\mathtt{A}_j$, ..., $\mathtt{Y}_j$ for $1 \le j \le 5$. Each word has an option for each permutation of its letters (see Algorithm 7.2.1.2L), showing which letters it needs for that permutation of the blocks. (For example, QUEUE will have 30 options, beginning with 'QUEUE $\mathtt{E}_1$:1 $\mathtt{E}_2$:1 $\mathtt{Q}_3$:1 $\mathtt{U}_4$:1 $\mathtt{U}_5$:1', which means that block 1 should have an E, ..., block 5 should have a U.) Break symmetry by giving only one of the 120 options for one of the words (FIRST, for example). Each of the 3223

backtracking program
WORDS(5757)
Bittencourt
Bittencourt
MCC problem
covering problem
XCC problem
Break symmetry

Riekstiņš
Castawords
Russell
Michel
Alon
Mycielski
Sachs

potential sets of letters has five options of size 23, showing exactly which letters are present if block j uses that set; for example, the five options for {A, B, C, D, E, I} are 'j $\mathtt{A}_j$:1 … $\mathtt{E}_j$:1 $\mathtt{F}_j$:0 … $\mathtt{I}_j$:1 … $\mathtt{Y}_j$:0' for $1 \le j \le 5$. There are 18486 options altogether, of total length 403357; Algorithm C solves them in 225 Gμ.

For these words the five blocks must be {E, F, G, L, O, S}, {C, E, T, R, U, Y}, {A, L, M, N, Q, R}, {A, B, E, P, S, T}, {D, H, K, T, U, V}. (The XCC problem actually has 8 solutions, because TIMES, TREES, and VALUE can each be formed in two ways from those blocks.)

[This exercise is based on an idea by E. Riekstiņš, who realized that a classic puzzle called Castawords could be extended to words of length 5.]

114. Besides the primary items p_{ij}, r_{ik}, c_{jk}, b_{xk} of (30), introduce R_{ik}, C_{jk}, and B_{xk} for the permuted array, as well as u_k and v_l to define a permutation. Also introduce secondary items π_k to record the permutation and ij to record the value at cell (i, j). The permutation is defined by 81 options 'u_k v_l π_k:l' for $1 \le k, l \le 9$. And there are $9^4 = 6561$ other options, one for each cell (i, j) of the board and each pair (k, l) of values before and after α is applied. If $(ij)\alpha = i'j'$, let $x' = 3\lfloor i'/3\rfloor + \lfloor j'/3\rfloor$. Then option (i, j, k, l) is normally 'p_{ij} r_{ik} c_{jk} b_{xk} $R_{i'l}$ $C_{j'l}$ $B_{x'l}$ ij:k $i'j'$:l π_k:l'. However, if $i' = i$ and $j' = j$, that option is shortened to 'p_{ij} r_{ik} c_{jk} b_{xk} R_{il} C_{jl} B_{xl} ij:k π_k:l'; and it is omitted when $i = i'$, $j = j'$, $k \ne l$. The options $(0, j, k, l)$ are also omitted when $k \ne j + 1$, in order to force '123456789' on the top row.

With that top row and with α = transposition, Algorithm C produces 30,258,432 solutions in 2.2 teramems. (These solutions were first enumerated in 2005 by E. Russell; see www.afjarvis.staff.shef.ac.uk/sudoku/sudgroup.html.)

115. A similar method applies, but with additional items b'_{yk} and $B'_{y'l}$ as in answer 67(b). The number of solutions is (a) 7784; (b) 16384; (c) 372; (d) 32. Here are examples of (a) and (d); the latter is shown with labels {0, …, 7, *}, to clarify its structure. [Enumerations (a), (b), (c) were first carried out by Bastian Michel in 2007.]

(a)

1	2	3	4	5	6	7	8	9
9	7	4	3	1	8	5	6	2
8	5	6	9	7	2	1	3	4
5	8	2	1	3	9	4	7	6
4	1	7	8	6	5	2	9	3
6	3	9	2	4	7	8	5	1
7	4	1	5	9	3	6	2	8
3	6	8	7	2	4	9	1	5
2	9	5	6	8	1	3	4	7

;

(d)

7	0	2	5	1	3	*	4	6
5	3	6	*	7	4	2	0	1
*	1	4	2	0	6	7	5	3
2	5	7	0	6	1	3	*	4
0	4	3	7	*	5	1	6	2
6	*	1	3	4	2	5	7	0
1	7	5	4	2	0	6	3	*
3	2	0	6	5	*	4	1	7
4	6	*	1	3	7	0	2	5

.

116. (a) Any triangle in $\mu(G)$ must be in G, because $u' \not\!\!-\!\!\!- v'$.

(b) Suppose $\mu(G)$ can be c-colored with some coloring function α, where $\alpha(w) = c$. If $\alpha(v) = c$ for any $v \in V$, change it to $\alpha(v')$. This gives a $(c-1)$-coloring of G. [Hence a triangle-free graph on n vertices can have chromatic number $\Omega(\log n)$. One can show nonconstructively that the triangle-free chromatic number can actually be $\Omega(n/\log n)^{1/2}$; but currently known methods of explicit construction for large n achieve only $\Omega(n^{1/3})$. See N. Alon, *Electronic J. Combinatorics* **1** (1994), #R12, 1–8.]

(c) If G is χ-critical, so is $\mu(G)$: Let $e \in E$ and suppose α is a $(c-1)$-coloring of $G \setminus e$. Then we get c-colorings of all but one edge of $\mu(G)$ in several ways: (i) Set $\alpha(v') \leftarrow c$ for all $v \in V$, and $\alpha(w) \leftarrow 1$. (ii) Let $u \in e$, and set $\alpha(u) \leftarrow \alpha(w) \leftarrow c$; also set $\alpha(v') = \alpha(v)$ for all $v \in V$, either before or after changing $\alpha(u)$. If you want to remove an edge of $\mu(G)$ that's in G, use (i); otherwise use (ii).

[See J. Mycielski, *Colloquium Mathematicum* **3** (1955), 161–162; H. Sachs, *Einführung in die Theorie der endlichen Graphen* (1970), §V.5.]

Monte Carlo
modified Algorithm C
symmetry
intersection
torus
supertile

117. (a) Use the answer to (b), with each clique consisting of a single edge.

(b) Each vertex v has d options '$v\ c_{1j}\ \ldots\ c_{kj}$' for $1 \le j \le d$, where the cliques containing v are $\{c_1, \ldots, c_k\}$.

(c) We save a factor of $9! = 362880$ by fixing the colors of the queens in the top row. Then there are 262164 solutions, found by Algorithm X in 8.3 Tμ with method (a) but in only 0.6 Tμ with method (b).

(d) Insert 'v':j' into the jth option for v, where v' is secondary. (This reduces the running time for method (a) in part (c) to 5.0 Tμ, *without* fixing any colors.)

(e) Using (d) to save a factor of $c!$, we get $(2!\cdot 1,\ 3!\cdot 5,\ 4!\cdot 520,\ 5!\cdot 23713820)$ solutions, in approximately (600, 4000, 130000, 4100000000) mems. [Monte Carlo estimates can be made for larger cases by combining exercises 86 and 122; the true branching factor at each level can be determined by rejecting options that involve illegal purification. It appears that M_6 can be 6-colored in approximately $6! \cdot 2.0 \times 10^{17}$ ways.]

(f) Now (d) saves a factor of $(c-1)!$, despite having no solutions; the running times are roughly (100, 600, 5000, 300000) mems. (But then for 5-coloring M_6 it's 45 Tμ!)

(g) There are $(1!\cdot 1,\ 2!\cdot 1,\ 3!\cdot(5 \text{ or } 7),\ 4!\cdot(1432, 1544, 1600, 2200, 2492, 2680, 3744, 4602, \text{ or } 6640))$ such colorings, depending on which edge is deleted.

118. In general, colorings of a hypergraph can be found with the construction of the previous exercise, but using Algorithm M and giving multiplicity $[0\,.\,.\,(r-1)]$ to each hyperedge of size r. In this case, however, there are 380 independent sets of size 16 (see exercise 7.1.4–242); we can simply use them as options to an exact cover problem with 64 items. There are four solutions, having a curious symmetry so that only two are "essentially different": One is shown, and the other is obtained by keeping A and C fixed but *transposing* the B's and D's.

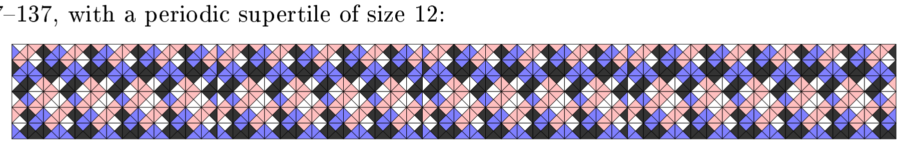
```
A B C D C D A B
B D B D C A C A
C B A C D B A D
D D A B A B C C
A A D C D C B B
B C D B A C D A
C A C A B D B D
D C B A B A D C
```

119. Exactly three interior edges are white in every solution. Any other placement of the all-white piece defines those three edges. That leaves no way to place all three of the two-white pieces.

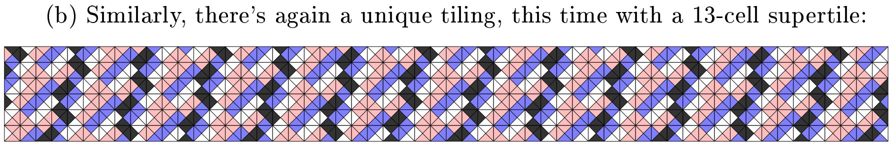
120. (a) Call the types 0, 1, ..., 9, and use Algorithm C to find all ways to place a given type at the center of a 5×5 array. There are respectively (16, 8, 19, 8, 8, 8, 10, 8, 16, 24) ways to do this; and the *intersection* of all solutions for a given type shows that

```
?????   0490?   ?????   68568   21721   32032   2032?   49049   0320?   17217
?????   2032?   ??0??   0490?   ?6856   17217   ?217?   0320?   21721   68568
??0??,  ?217?,  ??2??,  2032?,  9049?,  68568,  8568?,  21721,  ?6856,  049??
??2??   8568?   ?????   ?217?   320??   049??   490??   ?6856   9049?   20???
?????   490??   ?????   8568?   172??   20???   032??   9049?   320??   ?2???
```

are the respective neighborhoods that are forced near a given type in any infinite tiling. Consequently every such tiling contains at least one 5; and if we place 5 at the origin everything in the entire plane is forced. The result is a torus in the sense of exercise 7–137, with a periodic supertile of size 12:



(b) Similarly, there's again a unique tiling, this time with a 13-cell supertile:



121. (a) Marek Tyburec noticed in 2017 that there are no 2×2 solutions with βUS at lower right; similarly, there are no 3×4 solutions with βUS at lower left. Hence βUS can appear only in the top row, or at the left of the next-to-top row.

(b) Let (A_k, B_k, C_k, D_k) be the $(2^k-1)\times(2^k-1)$ tilings defined by $(\alpha a, \alpha b, \alpha c, \alpha d)$ when $k = 1$, otherwise by placing $(\delta Na, \delta Nb, \delta Nc, \delta Nd)$ in the middle and placing A_{k-1}, B_{k-1}, C_{k-1}, D_{k-1} at the corners as in answer 2.3.4.3–5. The unique tiling requested here has δRD in the middle and D_{k-1}, C_{k-1}, B_{k-1}, A_{k-1} at the corners.

(c) With δRU or δLD in the middle, another solution has C_{k-1}, D_{k-1}, A_{k-1}, B_{k-1} at the corners. With δLU or δSU, there's a third solution with B_{k-1}, A_{k-1}, D_{k-1}, C_{k-1} at the corners. And δSU also has 54 additional solutions with C_{k-2} in the upper left corner; they use $\{DL, DP, DS, DT, UL, UP, UR, US, UT\}$ in the upper half when choices need to be made, and independently $\{R, YR, L, P, S, T\}$ in the lower half.

(d) Only one of each survives. As in (b), its four quadrants are D_∞, C_∞, B_∞, A_∞.

[Each of the other 86 types occurs in A_6, hence in every sufficiently large tiling. Incidentally, the "dragon sequence" (see answer 4.5.3–41) arises in the colors at the edges of A_∞, B_∞, C_∞, D_∞.]

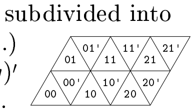
122. A new global variable Θ, initially v, is the current "color threshold." Every item has a new field `CTH` in addition to `NAME`, `LLINK`, and `RLINK`. That field is normally zero in primary items, although it has a special use in step C3 as described below. In secondary items, `CTH` will be used to undo changes to Θ.

Insert '$\texttt{CTH}(i) \leftarrow \Theta$; if $c = \Theta$, set $\Theta \leftarrow \Theta + 1$' just after '$i \leftarrow \texttt{TOP}(p)$' in the purify routine (55). Insert '$\Theta \leftarrow \texttt{CTH}(i)$' just after '$i \leftarrow \texttt{TOP}(p)$' in the unpurify routine (57). Modify the commit routine (54) so that it jumps to the end of the uncommit routine (56), if $\texttt{COLOR}(p) > \Theta$, without changing j or p. (The effect is to avoid committing to any option that would have set a color value greater than Θ, by jumping from step C5 into the appropriate place within step C6.*)

Finally, change step C3 so that it never chooses an item i for which $\texttt{CTH}(i) > \Theta$. That step should then go to C8 if no item is choosable. (This mechanism prohibits branching on primary items for which the assumption of total symmetry between all colors $\geq \Theta$ isn't yet valid. Exercise 126 has an example.)

123. When, say, $m = 4$ and $n = 10$, Algorithm C takes 49 megamems to produce 1048576 solutions. The modified algorithm (where we set $v = 1$) takes 2 megamems to produce 43947 solutions. (Notice that the value vectors $q_1 \ldots q_n$ are equivalent to the *restricted growth strings* $a_1 \ldots a_n$ of 7.2.1.5–4, with $q_k = a_k + 1$.)

124. Let (x, y) denote a Δ triangle, and let $(x, y)'$ denote the ∇ triangle that lies immediately to its right. (Think of a square cell (x, y) that has been subdivided into right triangles by its main diagonal, then slanted and yscaled by $\sqrt{3}/2$.) For example, an $m \times n$ parallelogram has $2mn$ triangles (x, y) and $(x, y)'$ for $0 \leq x < m$ and $0 \leq y < n$, Cartesianwise; the 3×2 case is illustrated.



The boundary edges of triangle (x, y) are conveniently denoted by $/xy$, $\backslash xy$, and $-xy$. Then the boundary edges of $(x, y)'$ are $/(x{+}1)y$, $\backslash xy$, and $-x(y{+}1)$.

[A "barycentric" alternative with three coordinates is also of interest, because it's more symmetrical: Each triangle corresponds to an ordered triple of integers (x, y, z) such that $x + y + z = 1$ or 2, under the correspondence $(x, y) \leftrightarrow (x, y, 2 - x - y)$ and

* Backtrack programs often run into such cases where it is permissible, even desirable, to *jump into the middle of a loop.* See Examples 6c and 7a in the author's paper "Structured programming with **go to** statements," *Computing Surveys* **6** (1974), 261–301.

Tyburec
dragon sequence
threshold
CTH
purify
unpurify
commit
uncommit
restricted growth strings
$m \times n$ parallelogram
Cartesianwise
parallelogram
barycentric
coordinates

$(x,y)' \leftrightarrow (x,y,1-x-y)$. The twelve symmetries are then the six permutations of $\{x,y,z\}$ with an optional flip between (x,y,z) and $(\bar{x},\bar{y},\bar{z}) = (1-x,1-y,1-z)$.]

[One can also use "barycentric even/odd coordinates," inspired by exercise 145, which are ordered triples (x,y,z) with $|x+y+z| \le 1$. Cases with x, y, z odd represent triangles, with $(x,y) \leftrightarrow (2x-1,2y-1,3-2x-2y)$, $(x,y)' \leftrightarrow (2x-1,2y-1,1-2x-2y)$. Cases with x, y, z even represent vertices. Cases with just one even coordinate represent edges (the average of two adjacent triangles). Cases with two even coordinates could represent directed edges.]

dihedral group
even/odd coordinates
Gottfried
MacMahon
symmetries
Monte Carlo

125. Every original triangle (x,y) or $(x,y)'$ expands to k^2 triangles of the forms $(kx+p,kx+q)$ or $(kx+p',kx+q')'$ for $0 \le p,q,p',q' < k$. Those obtained from (x,y) have $p+q<k$ and $p'+q'<k-1$ (of which there are $\binom{k+1}{2}$ and $\binom{k}{2}$, respectively). The others are obtained from $(x,y)'$.

126. Let there be 24 primary items `01'`, `02`, `02'`, ..., `32` for the triangles, and 24 primary items `aaa`, `aab`, ..., `ddd` for the tiles, together with 42 secondary items `\01`, `-02`, `/02`, ..., `/41` for the edges. There are $24 \cdot 64$ options '`01' aaa -02:a /11:a \01:a`', '`01' aab -02:a /11:a \01:b`', '`01' aab -02:a /11:b \01:a`', ..., '`32 ddd -32:d /32:d \32:d`' — one for each way to place a tile. Finally, to force the boundary conditions, add another primary item '`*`', and another option '`* -20:a -30:a /40:a ... \10:a`'.

Algorithm C finds 11,853,792 solutions, after 340 Gμ of computation; this total includes 72 different versions of every distinct solution, hence there really are just 164,636 of them (a number that was unknown until Toby Gottfried computed it in 2001).

Using exercise 119 we can remove all options for `aaa` except '`20 aaa -20:a /20:a \20:a`'. Algorithm C then finds $11853792/12 = 987{,}816$ solutions, in 25 Gμ.

Furthermore, using exercise 122 (with $v = $ `b`), and not allowing step C3 to branch on a tile name until $\Theta = $ `e` (because there's total symmetry with respect to triangle locations but not tile names), finds every distinct solution just once, in 6.9 Gμ.

Finally, we can allow branching on `aab` whenever $\Theta \ge$ `c`, and in general on a piece name whenever Θ exceeds all colors in its name. This reduces the runtime to 4.5 Gμ.

[MacMahon specifically designed pattern (59b) to include all three of the nonwhite solid-color triangles in the center. If we fix them in those positions, an unmodified Algorithm C quickly finds 2138 solutions. There also are 2670 solutions with those three fixed in positions $\{$`11'`, `21'`, `12'`$\}$ instead of $\{$`12`, `21`, `22`$\}$.]

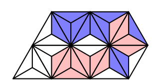
127. Every color appears in $(3 \cdot 24)/4 = 18$ places among the triangles, hence $18-2k$ times on the border when it occurs k times in the interior of a solution. Consequently no color occurs an odd number of times on the border. That leaves 2099200 possibilities.

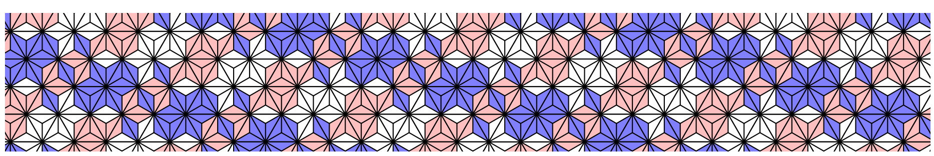
All of those 2099200 are actually completable. (MacMahon would have been very happy to have known this!) The number of cases can be reduced to only 4054, using the methods of Section 7.2.3, because there are 576 symmetries: cyclic shifting and/or reflection and/or permutation of colors. The Monte Carlo procedure of Algorithm 7.2.2E not only finds solutions in each of those cases, it finds oodles of them. In fact, we can be confident that *every all-even-but-not-constant border specification has more than four times as many solutions as the pure-white border does.*

(More precisely, the pure-white border 000000000000 has 11853792 solutions, without reducing by symmetry; the next-smallest border, 000000000011, has 48620416; the next-smallest, 000000000101, has 49941040; and so on. There are more than 100 million solutions in the vast majority of cases, but probably never more than 500 million. Incidentally, 001022021121 is the only valid color pattern that has exactly three automorphisms.)

128. We can pack them into the 11-triangle region obtained by deleting triangle $(2,1)'$ from the 2×3 parallelogram in answer 124, in such a way that the edge colors satisfy `-00` = `-20`, `/01` = `/30`, `-02` = `-12`. There are 1032 ways to do this, one of which is shown. This yields a "supertile" that nicely tiles the plane, in combination with its 180° rotation:

supertile tiles the plane





129. First consider rotation symmetry. Only 180° rotation applies, because of the four single-color tiles. To generate all of the strong solutions, assume that rotation changes `a` ↔ `d`, `b` ↔ `c`, and combine the options of answer 126 into pairs such as '`02 abc -02:a /02:b \02:c 31' bdc -32:d /41:c \31:b`'. The resulting 768 options have 68,024,064 solutions (found in about 0.5 Tμ); but many of those solutions are essentially the same (that is, obtainable from each other by rotation, reflection and/or color permutation).

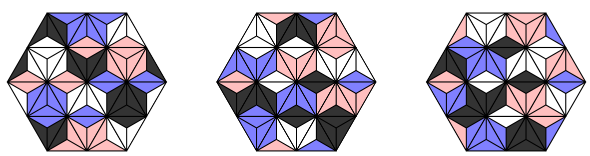
It's somewhat tricky to count the essentially distinct patterns; canonical representations can be obtained by distinguishing six types of solutions: (1) `02 aaa` (hence `31' ddd`) and `03 bbb` (hence `30' ccc`), and `/12:a` or `/12:c`. [The cases `/12:b` or `/12:d` are equivalent to these, if we reflect and swap `a` ↔ `b`, `c` ↔ `d`.] (2) `02 aaa`, `23 bbb` [or equivalently `03' bbb`]. (3) `02 aaa`, `13' bbb`, and `\03:a` or `\03:c`. (4) `02 aaa`, and `bbb` in `12`, `12'`, `22`, `22'`, or `13`. (5) `13 aaa`, `02' bbb`, and `\12:a` or `\12:c`. (6) `13 aaa`, `12 bbb`. Each type is easy, yielding $80768+164964+77660+819832+88772+185172 = 1417168$ solutions.

[Notice that the illustrated example of strong symmetry actually tiles the plane without rotation; that is, it has `-04` = `-20`, `-14` = `-30`, `/03` = `/41`, ..., `\10` = `\32`. Exactly 40208 of the essentially distinct solutions satisfy this additional proviso.]

To generate the weak solutions, introduce new secondary items b_{xy}, b'_{xy} for each triangle (x,y) or $(x,y)'$ with $y>1$, representing color changes within the triangle. Typical options are now '`02 aad -02:a /02:a \02:d b02:5`', '`02 aad -02:a /02:d \02:a b02:3`', '`02 aad -02:d /02:a \02:a b02:6`', '`02 abc -02:a /02:b \02:c b02:7`', '`31' bdc -32:c /41:b \31:d b02:7`', '`31' ccd -32:c /41:c \31:d b02:5`'. We may assume that `ddd` is opposite `aaa`, `ccc` is opposite `bbb`. Algorithm C generates each weak-not-strong solution twice, each strong solution once; the six types yield a total of $24516+45818+22202+341301+44690+130676 = 609203$ weak-not-strong solutions.

Turning now to reflections of the hexagon, there are two essentially different possibilities: Top-bottom reflection preserves the values of four edges, but all triangles change; left-right reflection preserves the values of four triangles and two edges. Therefore *strong reflection symmetry is impossible*. (In the first case, all triangles change, hence all colors change. In the second case, two colors must be fixed. With colors `a` and `d` fixed but `b` ↔ `c`, eight triangles `aaa`, `aad`, `abc`, `acb`, `bcd`, `bdc`, `dda`, `ddd` must be fixed.)

Weak symmetry under top-bottom reflection can be assumed as before to take `aaa` to `ddd`, `bbb` to `ccc`. Again there are six types: [1] `02' aaa`, `22' bbb`, `-13:a` or `-13:c`. [2] `02' aaa`, `bbb` in `12'`, `03'`, `13`, `13'`, or `23`. [3] `12' aaa`, `bbb` in `03` or `03'`. [4] `03 aaa`, `23 bbb`, `13:a` or `-13:c`. [5] `03 aaa`, `bbb` in `13` or `13'`. [6] `03' aaa`, `13' bbb`, `-13:a` or `-13:c`. Surprisingly, some placements are "special": They have strong rotational symmetry, as well as weak top-down symmetry! Algorithm C, which generates the special ones once and the others twice, produces respectively (88, 0, 0, 98, 0, 75) + 2(1108, 12827, 8086, 3253, 12145, 4189) solutions. Here are examples of the $88+98+75 = 261$ special placements, which belong simultaneously to types [1] and (5), [4] and (3), [6] and (1):

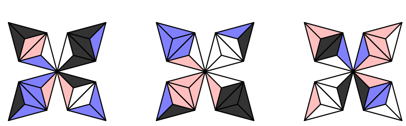


Weak left-right symmetry is similar, but there now are some fixed triangles. If `aaa` is fixed, assume that `ddd` is also fixed; three such types arise, with $46975 + 35375 + 25261 = 107611$ solutions. Otherwise assume that `ddd` is opposite `aaa`; six types of this kind yield (75, 0, 98, 0, 0, 88) strong and (3711, 56706, 5889, 60297, 38311, 9093) non-strong solutions. So there's a grand total of 281618 essentially distinct weak-not-strong placements with left-right symmetry — of which 194 are top-down symmetric too.

[Arrangements that have strong and weak symmetry were first discovered by Kate Jones, who presented them in the 1991 user manual for Multimatch® III, an attractively produced set of triangular tiles.]

130. The nicest coordinate system for an octahedron is probably to number the faces 000, 001, …, 111 in binary, and to let the vertices be {`0**`, `1**`, `*0*`, `*1*`, `**0`, `**1`}; the edges are $\{xy{*}, x{*}y, {*}xy\}$ for $x, y \in \{0, 1\}$. Construct 512 options '`000 aaa *00:a 0*0:a 00*:a`', '`000 aab *00:a 0*0:b 00*:a`', '`000 aab *00:b 0*0:a 00*:a`', …, with face-name items `000`, …, `111` primary and tile-name items `aaa`, …, `ddd` secondary. Algorithm C quickly finds 2723472 solutions, which include 45356 distinct sets of eight. Those 45356 sets become, in turn, new options for Algorithm X (or C), with 24 primary tile-name items; now we get 1615452 solutions, which are the desired partitions.

Many symmetries are present, of course; we'll study how to distinguish nonisomorphic representatives in Section 7.2.3. One of the most interesting solutions,


,

has four color-swap symmetries, with all the solid-color triangles on one octahedron.

131. (a) Each triangle edge is either `a` (straight) or `b` (a wave) or `c` (a hump) or `d` (a dip). We can set this up with options and items as in answer 126, except that the edge-match condition is now `a` ↦ `a`, `b` ↦ `b`, `c` ↦ `d`, `d` ↦ `c`; to get proper matching, the options of ∇ triangles should state the *mate* color, as in '`01' abc -02:a /11:b \01:d`'.

Every solution corresponds to 24 equivalent solutions, because we get a factor of 6 by rotating the hexagon, a factor of two by interchanging humps with dips, and another factor of two by reflection. (Reflection is a bit tricky, because a wave becomes an anti-wave when a piece is flipped over. However, every reflected piece has its own anti-piece, which yields the desired anti-solution.) Thus we can force `aaa` to be in position `02`. Treating `c` and `d` symmetrically as in answer 126 (with $v =$ `c`) produces exactly 2,231,724 canonical solutions and needs only 30 gigamems of running time.

[This puzzle is manufactured by Kadon Enterprises under the name Trifolia®.]

(b) A similar setup, letting `c` and `d` represent 0 spots and 3 spots so that it's easy to treat them symmetrically, now has mates `a` ↦ `b`, `b` ↦ `a`, `c` ↦ `d`, `d` ↦ `c`; hence one option is '`01' abc -02:b /11:a \01:d`'. The boundary colors in directions `/` and `\` are `a`; in direction `-` they are `b`. The solutions to this problem typically form groups of eight (not 24): We can swap `c` ↔ `d`, reflect left-right, reflect top-down, or rotate by 180°;

Jones
Multimatch®
coordinate system
symmetries
contact system for adjacent tiles
dihedral group
chirality
reflection
anti-wave
flipped over
Kadon Enterprises
Trifolia®

the latter two are combined with swapping $\mathtt{a} \leftrightarrow \mathtt{b}$. Without attempting to remove any symmetries, we get 3,419,736,176 solutions, after 20.6 teramems of computation.

Left-right reflection always gives a distinct solution, whether we swap $\mathtt{c} \leftrightarrow \mathtt{d}$ or not (because there are at least eight pieces that stay fixed, and only four places to put them). But the illustrated example shows that some solutions are fixed under 180° rotation; we can find them by adding 15 new primary items, such as `#/23`, and $15 \cdot 4$ new options, such as '`#/23 /23:`x `/20:`x' for $x \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$. Altogether 18656 solutions have that symmetry; such cases form groups of four, not eight. Similarly, 169368 cases turn out to have top-down symmetry. It follows from "Burnside's lemma" that the total number of essentially different solutions is $(3419736176 + 18656 + 169368)/8 = 427490525$.

To double the speed of all these computations, take $v = \mathtt{c}$ in exercise 122.

132. This challenging problem was first resolved by Peter Esser in April 2002, and presented online at `www.polyforms.eu/coloredpolygons/triindex.html#trios24`. [See *JRM* **9** (1977), 209. One can show that the only solutions to the Diophantine equation $d + d(d-1) + d(d-1)(d-2)/3 = m^2$ are $d = 1$, 2, and 24, using advanced methods found in N. P. Smart, *The Algorithmic Resolution of Diophantine Equations* (1998).]

133. This problem is like exercise 126, but considerably simpler because squares are easier than triangles. There are $24 \cdot 81$ options '`00 aaaa h00:a v10:a h01:a v00:a`', ..., '`53 ccba h53:a v63:c h54:c v53:b`', where hxy and vxy denote the horizontal and vertical edges between squares. We save a factor of 4 by limiting `aaaa` to four positions on the border, and another factor of 2 by making `b` and `c` equivalent (exercise 122 with $v = \mathtt{b}$). The resulting 13328 solutions are found in 15 Gμ.

[Today it's easy to count them; but this problem has a tortured history! T. H. O'Beirne missed two of the 20 possible ways to place the internal white edges, when he analyzed the situation by hand in *New Scientist* **9** (2 February 1961), 288–289. A few years later, the problem of solution counts for MacMahon squares was probably the very first large computation ever undertaken at Stanford Artificial Intelligence Laboratory; Gary Feldman found 12261 placements, during a 40-hour computer run (see *Stanford AI Project*, Memo 12 (16 January 1964), 8 pages). That number was believed to be correct until May 1977, when the true value was obtained by H. Fernández Long in Argentina.]

Instead of denoting squares by xy and edges by $\{\mathtt{h}xy, \mathtt{v}xy\}$, it's convenient to use "even/odd coordinates" instead (see exercise 145). In that system, a pair of odd numbers $(2x+1)(2y+1)$ denotes a square, and the edge between two adjacent squares is represented by the midpoint between them. For example, the $24 \cdot 81$ options sketched above would then take the form

'`11 aaaa 01:a 12:a 21:a 10:a`', ..., '`b7 ccba a7:a b8:c c7:c b6:b`'.

Such coordinates are easier to work with under reflection and rotation.

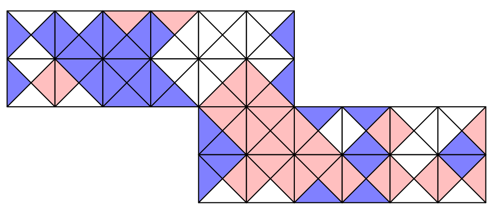
134. (O, P, Q, ..., Z) occur respectively (0, 1672, 22, 729, 402, 61, 36, 48, 174, 259, 242, 0) times, sometimes twice in the same solution; one solution features *four* pentominoes.

[Kate Jones introduced such questions in the Multimatch® I user manual (1991).]

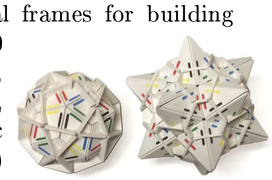
135. Indeed, the total number of solutions is enormous; Monte Carlo estimates predict $\approx 9 \times 10^8$ of them for any fixed placements of `aaaa`, `bbbb`, `cccc` that aren't obviously impossible. Therefore it's natural to impose extra conditions. The elegant wrapping below permutes colors cyclically and has solid colors on every edge of the cube! Investigations by H. L. Nelson, F. Fink, and M. Risueño showed that 61 such solutions are possible; see W. E. Philpott, *JRM* **7** (1974), 266–275. See answer 145 for an even/odd

180° rotation
Burnside's lemma
Esser
Diophantine equation
Smart
history
O'Beirne
Stanford Artificial Intelligence Laboratory
Feldman
Fernández Long
even/odd coordinates
odd/even coordinates, see even/odd
Jones
Multimatch®
Monte Carlo estimates
Fink
Risueño
Philpott
even/odd coordinate system

coordinate system that is useful for representing this problem internally.



(Wrapping the surface of a symmetrical polyhedron is a nice way to avoid awkward boundary conditions when arranging MacMahon-like tiles. Dario Uri devised 39 such problems in 1993, together with ingenious mechanical frames for building the results. Here, for example, is a rhombic triacontahedron (30 rhombuses) and a stellated dodecahedron (60 isosceles triangles), based on all possible ways to put distinct colors from {red, green, blue, yellow, black} on the edges. His report "Tessere di Mac Mahon su superfici tridimensionali" is online at `www.uriland.it`.)



136. The main challenge is to find a good way to represent the faces and edges of a dodecahedron. Perhaps the nicest is to represent the faces by vertices of an icosahedron, with the three-dimensional coordinates $(0,(-1)^b\phi,(-1)^c)\sigma^a$, where $(x,y,z)\sigma=(z,x,y)$; let abc stand for this face, for $0\le a<3$ and $0\le b,c<2$. A face is adjacent to its five nearest neighbors; we can represent the edge between abc and $a'b'c'$ as the midpoint $(abc+a'b'c')/2$. These 30 midpoints have two forms, either $ab=(0,(-1)^b\phi,0)\sigma^a$ or $abcd=\frac{1}{2}((-1)^b,(-1)^c\phi,(-1)^d\phi^2)\sigma^a$. The corresponding XCC problem can now be formulated as usual, with 120 options for each face. For example, a typical option for face `201` is '`201 01243 20:3 1100:0 2001:1 2101:2 1110:4`'.

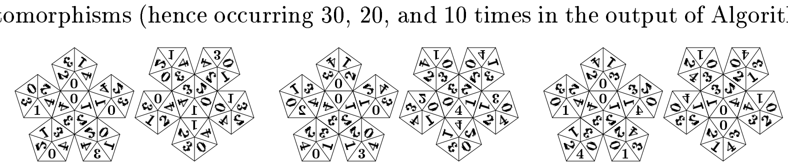
We can force the first tile to be in a particular place by default. Algorithm C needs only 9 megamems to solve the resulting problem, and produces 60 solutions.

Of course many of those solutions are equivalent. There are 120 transformations that preserve the dodecahedron and icosahedron as represented above, generated by three reflection matrices and two orthogonal matrices,

$$D_0=\begin{pmatrix}-1&0&0\\0&+1&0\\0&0&+1\end{pmatrix},\qquad D_1=\begin{pmatrix}+1&0&0\\0&-1&0\\0&0&+1\end{pmatrix},\qquad D_2=\begin{pmatrix}+1&0&0\\0&+1&0\\0&0&-1\end{pmatrix},$$

$$P=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix},\qquad Q=\frac{1}{2}\begin{pmatrix}-1&-\phi&1/\phi\\-\phi&1/\phi&-1\\1/\phi&-1&-\phi\end{pmatrix}.$$

Applying any combination of these, and remapping the colors to agree with the default placement, gives an equivalent solution. It turns out, as Conway discovered by hand(!), that there are just three inequivalent solutions, having respectively 4, 6, and 12 automorphisms (hence occurring 30, 20, and 10 times in the output of Algorithm C):



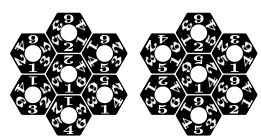
[See M. R. Boothroyd and J. H. Conway, Eureka: *The Archimedeans' Journal* **22** (1959), 15–17, 22–23. Conway has named them the "quintominal dodecahedra."]

polyhedron
Uri
triacontahedron
rhombuses
stellated
dodecahedron
isosceles triangles
icosahedron
coordinates
golden ϕ
automorphisms+
Boothroyd
quintominal dodecahedra

137. (a) This is an easy application of Algorithm C, with $14+12$ items and $7 \cdot (1 + 6 \times 6) = 259$ options. (Clever reasoning also allows it to be established by hand, with a search tree of size 15.)

(b) No. Again Algorithm C gives the answer quickly.

(c) Thousands of random trials indicate that about 93% of the $\binom{120}{7}$ choices have no solution; about 5% have just one solution; about 1% have two solutions; and the remaining 1% have three or more.

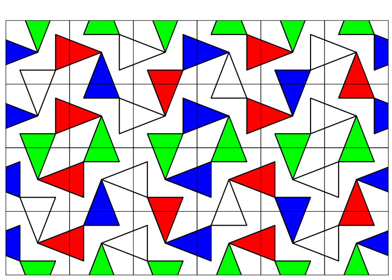


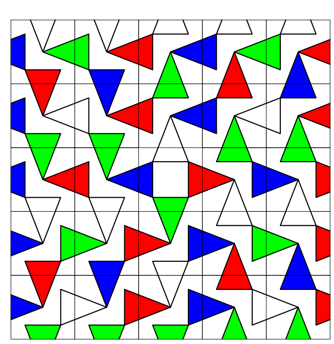
(d) About 0.4% of all cases work, as in the example shown. *Historical notes:* Milton Bradley Company introduced Drive Ya Nuts in 1970; the name of its inventor has unfortunately been forgotten. It was preceded by a much more difficult puzzle with 19 hexagons in three concentric rings, called Super Dom [H. Hydes, *British Patent 149473* (19 August 1920)], and by several similar puzzles [H. Hydes and F. R. B. Whitehouse, *British Patent 173588* (29 December 1921); G. H. Haswell, *U.S. Patent 1558165* (20 October 1925)], featuring both kinds of edge-matching rules.

138. (a) We can name the tiles `ABcd`, `ABdc`, `ACbd`, ..., `DCba`. Assuming that `ABcd` is in the top left corner, a straightforward application of Algorithm C (with 2118 options involving $48+48$ items) will output 42680 solutions, in 13 gigamems. As in other such problems, however, these outputs include many that are essentially the same. Up to 96 equivalent solutions are related by the operations of shifting any cell to the top-left position and/or flipping horizontally and/or flipping vertically, then remapping the colors. For instance, the given example has six automorphisms: We can shift it two columns right, then map $A \mapsto C \mapsto D \mapsto A$, $a \mapsto c \mapsto d \mapsto a$; we can also shift two rows down, reflect left-right, then $A \leftrightarrow D$ and $a \leftrightarrow d$. Hence it contributes $96/6 = 12$ cases to the total of 42680. Altogether there are (79, 531, 5, 351, 6, 68, 12, 4) cases with respectively (1, 2, 3, 4, 6, 8, 12, 24) automorphisms, hence $79+531+5+351+6+68+12+4 = 1056$ essentially different solutions. One with 24 symmetries is shown below (it leads to itself if we move right 1 and down 2, and/or reflect horizontally or vertically).

(b) Now Algorithm C, given 1089 options involving 49+60 items, quickly finds just six solutions — three different pairs related by transposition, each of which is symmetric under 90° rotation, all with heads and tails in the same places.

(c) Take any of the three solutions to (b), reflect it top-down, interchange heads with tails, and swap $B \leftrightarrow D$, $b \leftrightarrow d$. For example, the dual of the given solution is shown below. Alternating all-heads with all-tails, in checkerboard fashion, yields uncountably many tilings of the plane.



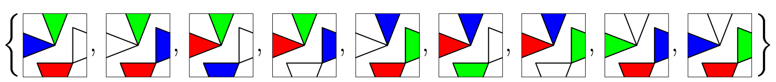


[These tiles are believed to have originated in 1990 with a puzzle called "Super Heads & Tails," designed by Howard Swift and produced in a limited edition.]

random trials
Historical notes
Bradley
Super Dom
Hydes
Patent
Whitehouse
Haswell
flipping
remapping
reflection
automorphisms
automorphisms
90° rotation
checkerboard
Super Heads & Tails
Swift

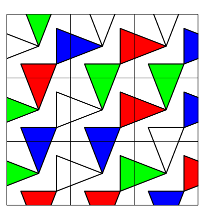
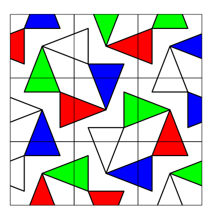
139. (a) Say that two sets of nine are essentially the same if one can be obtained from the other by remapping the colors, and/or reflecting all of the pieces, and/or interchanging heads with tails. For example, $4! \times 2 \times 2 = 96$ different choices of nine are equivalent to the set

reflecting
flipping
Haubrich
historical note

 $(*)$

By considering canonical forms, as in exercise 138(a), we find 14124 equivalence classes, of which (13157, 882, 7, 78) have the respective sizes (96/1, 96/2, 96/3, 96/4).

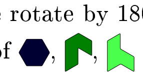
(b) There are exactly (9666, 1883, 1051, 537, 380, 213, 147, 68, 60, 27, 29, 9, 24, 4, 8, 2, 5, 4, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1) classes with (0, 1, 2, ..., 27) solutions; the amazing one with 27 is represented by $(*)$ above. Two of the 1883 puzzles with unique solutions are particularly interesting because they have four automorphisms:

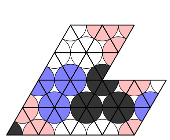
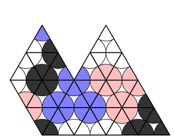
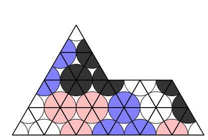



In each case we can flip the pieces and/or swap heads $\leftrightarrow$ tails, then remap the colors to get the original tiles.

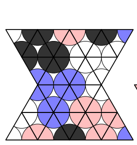
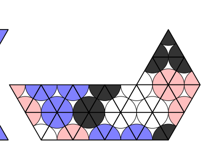
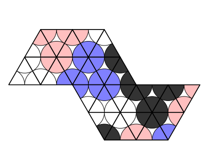
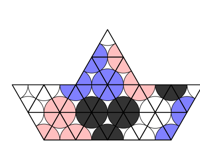
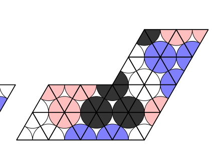
[This problem was first solved by Jacques Haubrich in 1996, who considered color remapping only (hence he had 54498 equivalence classes). Haubrich has collected 435 inequivalent puzzles, from around the world, that consist of nine tiles with two heads opposite two tails. But only 17 of them have all tiles different and all four objects different on each tile; for example, at least one tile such as `ABcb` is usually present. The first "pure" `HHtt` puzzle in his collection was made by the Hoek Loos company in 1974.]

140. (a) We save a factor of 4! by applying exercise 122 with $v = \texttt{a}$. Then Algorithm C gives respectively (10, 5, 6) solutions. The true numbers, however, are (5, 3, 3), because the shapes are symmetrical — and because the middle solution has an additional symmetry: It goes into itself if we rotate by 180° and permute the colors.

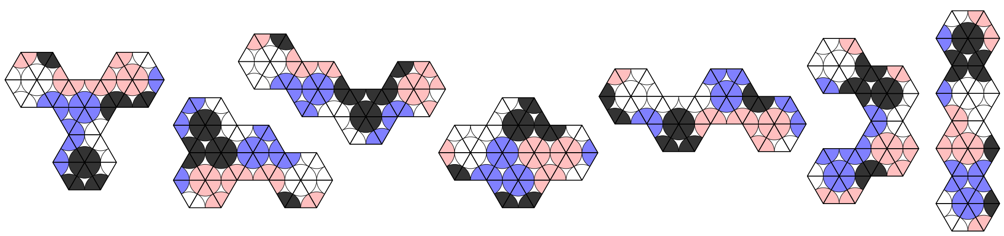
(b) The scaled-up versions of  are impossible. But we have

with respectively (4, 4, 3) solutions; and there are *unique* solutions to the other five:

(c) These shapes, with respectively (7, 9, 48, 2, 23, 28, 18) solutions, are a bit easier to handle. The "wave" has six solutions with central symmetry; the "bar" has four.



[Vertex-colored triangles have been named 'Trioker' by Marc Odier; see French Patent 1582023 (1968), *U.S. Patent 3608906* (1971), and the book *Surprenant Triangles*, which he published with Yves Roussel in 1976. They also are sold as Multimatch® IV.]

141. (a) Using exercise 122 with $v = \mathtt{a}$ yields respectively (138248, 49336, 147708) solutions in (1390, 330, 720) gigamems. Then we divide by (8, 4, 4) to remove symmetries of the board, getting (17281, 12334, 36927) solutions that are essentially distinct. [These numbers were first computed by Toby Gottfried in (1998, 1999, 2002). He had been interested in the puzzle ever since seeing the 5×5 version that was sold by Skor-Mor in 1970 under the name "Nitty Gritty." The puzzle is extremely difficult to solve by hand, in spite of the many solutions; Langford himself was unable to solve the 3×8 case.]

The 12334 solutions for 4×6 include 180 that have matching colors at the left and right. Each of these patterns therefore tiles a "cylinder"; and the 180 form 30 families of 6 that are equivalent to each other by rotating the cylinder. Similarly, 1536 of the 36927 solutions for 3×8 are cylindrical, making 192 families of 8. The example illustrated is one of 42 that have the same solid color at both left and right.

(b) Any solution can be used to tile the plane in combination with its mirror reflections and its 180° rotation (which is a reflection of a reflection).

The 17281 solutions include 209 for which the hole is surrounded by a single color. Six of these have matching colors at two opposite sides; the one illustrated will tile the plane in conjunction with its mate, which is obtained by swapping $\mathtt{b} \leftrightarrow \mathtt{c}$.

The 4×6 example illustrated is the unique solution for which both pairs of opposite sides induce exactly the same color partition (the restricted growth strings 0121120 and 01220). Thus it too will tile the plane together with its $\mathtt{b} \leftrightarrow \mathtt{c}$ mate.

[Vertex-matched squares, with *incomplete* sets of tiles, first appeared in puzzles devised by E. L. Thurston, *U.S. Patents 487797* (1892), *490689* (1893).]

142. Each boundary between the square cells containing octagons now has *two* secondary items that receive color. For example, a typical option for Algorithm C is now '10 $\mathtt{aabc}$ a_{10}:$\mathtt{a}$ r_{10}:$\mathtt{a}$ l_{11}:$\mathtt{b}$ a_{11}:$\mathtt{b}$ b_{21}:$\mathtt{c}$ l_{21}:$\mathtt{c}$ r_{20}:$\mathtt{a}$ b_{20}:$\mathtt{a}$', where a_{xy}, b_{xy}, l_{xy}, and r_{xy} denote the half edges above, below, left, and right of (x, y). The number of solutions, again using exercise 122 with $v = \mathtt{a}$, is $2 \cdot (132046861, 1658603, 119599)$ in cases (i), (ii), (iii), found in (2607, 10223, 77) gigamems. Case (i) includes $2 \cdot (193920, 10512, 96)$ "cylindrical" arrangements in which the colors match at top/bottom, left/right, both; one of the 96 "toroidal" examples is shown. Case (ii) includes $2 \cdot 5980$ cylindrical arrangements that match at left/right. Case (iii) has no cylindrical examples.

[Many other possibilities arise, because neighboring octagons can match without lying in a square grid. Kadon Enterprises offers attractive sets called 'Doris®'.]

143. (a) *simplex*$(8, 6, 8, 2, 0, 0, 0)$; *simplex*$(7, 4, 7, 3, 0, 0, 0)$; *simplex*$(5, 5, 5, 4, 0, 0, 0)$.

central symmetry
Trioker
Odier
Patent
Roussel
Multimatch®
Gottfried
Nitty Gritty
Skor-Mor Corporation
Langford
cylinder
reflections
restricted growth strings
set partition
historical note
Thurston
cylindrical
toroidal
Kadon Enterprises
Doris®

(b, c) Nonnegative integers $x_0x_1x_2x_3x_4x_5$ define such a polygon if and only if the boundary path returns to its starting point, which means that $x_0+x_1=x_3+x_4$ and $x_1+x_2=x_4+x_5$. Rotating by 60° replaces $x_0x_1x_2x_3x_4x_5$ by $x_5x_0x_1x_2x_3x_4$; reflecting left $\leftrightarrow$ right replaces $x_0x_1x_2x_3x_4x_5$ by $x_0x_5x_4x_3x_2x_1$. Hence we get a canonical form by insisting that $x_0 \geq x_3 \geq x_5 \geq x_1$: *Every sequence of nonnegative integers* (a,b,c,d) *with* $a \geq b \geq c \geq d$ *defines the boundary* $x_0x_1x_2x_3x_4x_5$ *of a unique convex triangular polygon*, where $x_0=a$, $x_1=d$, $x_2=a-b+c$, $x_3=b$, $x_4=a-b+d$, $x_5=c$. Furthermore, that polygon contains exactly $N=(a+c+d)^2-b^2-c^2-d^2$ triangles.

Given N, the following algorithm visits all relevant (a,b,c,d). For $c=0, 1, \ldots,$ while $2c^2 \leq N$ do the following: For $d=0, 1, \ldots,$ while $d \leq c$ and $2c(c+2d) \leq N$, let $x=N+c^2+d^2$. If $x \bmod 4 \neq 2$, for every divisor q of x such that $q \equiv x$ (modulo 2) and $q^2 \leq x$, set $a \leftarrow (x/q+q)/2-c-d$ and $b \leftarrow (x/q-q)/2$. Visit (a,b,c,d) if $a \geq b$ and $b \geq c$.

When $N=24$ this algorithm visits six (a,b,c,d), namely $(7,5,0,0)$, $(5,1,0,0)$, $(5,1,1,0)$, $(6,6,2,0)$, $(2,2,2,2)$, $(4,4,3,0)$. The second, fourth, and sixth are the shapes of exercise 140. The other three cannot be tiled properly with Langford's 24 tiles.

[See OEIS sequence A096004, contributed by P. Boddington in 2004.]

(d) Yes. One way is $simplex(a+c+d, a+c, a+d, a-b+c+d, 0, 0, 0)$.

144. The constraints are severe, because a solid color is needed at transitions between regimes. Algorithm C (with $v=$ `a` as in answer 142) quickly finds $2\cdot 102$ solutions to (ii). But surprisingly many arrangements arise in case (i); Algorithm C finds $2\cdot 37586004$ of them, *not* so quickly (643 teramems)!

(These tiles suggest many intriguing questions. For example, suppose we restrict consideration to making a big hexagon from 24 small ones. There are 2^{24} ways to specify whether each position should be matched at vertices or edges; but very few of those specifications are actually realizable. Can the realizable ones be nicely characterized?)

145. Suppose $0 \leq i \leq l$, $0 \leq j \leq m$, and $0 \leq k \leq n$. Let $(2i, 2j, 2k)$ represent vertex (i,j,k); let $(u+v)/2$ represent the edge between adjacent vertices u and v; let $(a+b)/2$ represent the face containing parallel edges a and b; let $(e+f)/2$ represent the cell containing parallel faces e and f. Thus, the triple (x,y,z) represents a vertex, edge, face, or cell when it has respectively 0, 1, 2, or 3 three odd coordinates.

For example, $(2i, 2j{+}1, 2k)$ represents the edge between vertices (i,j,k) and $(i, j{+}1, k)$; $(2i{+}1, 2j, 2k{+}1)$ represents the face whose vertices are (i,j,k), $(i{+}1,j,k)$, $(i,j,k{+}1)$, $(i{+}1,j,k{+}1)$; and $(2i{+}1, 2j{+}1, 2k{+}1)$ represents the cell whose eight vertices are $\bigl(i+(0 \text{ or } 1), j+(0 \text{ or } 1), k+(0 \text{ or } 1)\bigr)$.

Notice that $(a+b)/2$ represents the vertex between adjacent parallel edges a and b; $(e+f)/2$ represents the edge between adjacent parallel faces e and f; $(p+q)/2$ represents the face between adjacent cells p and q.

(We can use a similar convention in two dimensions, as an alternative to the 'H' and 'V' items in situations like answer 109.)

146. (a) Each color occurs four times on the "visible" faces and at most twice on the "hidden" faces. So the five adjacencies account for all six occurrences of five colors.

(b) For every partition of {`a`, `b`, `c`, `d`, `e`, `f`} into three pairs $\{u,u'\}$, $\{v,v'\}$, $\{w,w'\}$, there are two chiral cubes having u opposite u', v opposite v', w opposite w'. Order the colors so that $u<u'$, $u<v$, $v<v'$, $v<w$, $v<w'$; there are 30 ways to do this. The cube named $uu'vv'ww'$ is the one that can be placed with u on top, u' on the bottom, v in front, v' in the back, w at left, w' at the right. For example, the cubes in $(*)$ are named `aebfcd`, `acbfde`, `acbdef`, `afbdec`, `abcedf`, `aebcfd`.

canonical form
OEIS
Boddington
even/odd coordinate system
odd coordinates
chiral

(c) We can set this up for Algorithm C by specifying $6 \cdot 30 \cdot 24$ options, one for each cube position, cube name, and cube placement. There are 6 primary items for the positions; 30 secondary items for the names; $4 \cdot 6$ primary items u_c, d_c, f_c, b_c for colors on the top, bottom, front, and back, where $c \in \{\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{e},\texttt{f}\}$; and 6 secondary items h_k for the colors hidden between positions k and $k+1$. For example, the leftmost cube in $(*)$ corresponds to the option '1 `aebfcd` $u_{\texttt{a}}$ $d_{\texttt{e}}$ $f_{\texttt{b}}$ $b_{\texttt{f}}$ h_0:`c` h_1:`d`'.

If we eliminate all but one option for position 1 (thus saving a factor of 720), there are 2176 solutions. Each solution is, however, potentially equivalent to 95 others, because there are 16 possible rotations/reflections together with 6 cyclic permutations (followed by remapping the colors of the leftmost cube). For example, the solution illustrated has 12 such automorphisms. Further study shows that only 33 solutions are "essentially different" — of which (17, 9, 3, 1, 3) have (1, 2, 4, 6, 12) automorphisms.

(d) Yes, in lots and lots of ways. The $720 \cdot 2176$ solutions obtained *without* fixing the leftmost cube involve 15500 different 6-tuples of cubes; and the exact cover problem for which those 6-tuples are the options has 163,088,368 solutions.

[This problem was posed by Martin Gardner in *Scientific American* **204**, 3 (March 1961), 168–174 (long before the "Instant Insanity" craze), and he extended it to question (c) in *Scientific American* **235**, 3 (September 1978), 26. A solution to (d) that involves five symmetrical arrangements was found by Zoltan Perjés in 1981; see Gardner's book *Fractal Music, Hypercards, and More* (1992), 97.]

147. (a) The "even/odd coordinates" of exercise 145 are ideal for representing the cube positions and the faces between them. For example, the colors in the $1 \times 2 \times 2$ brick that was illustrated with the exercise are nicely represented by the $3 \times 5 \times 5$ array

$$\begin{bmatrix}\texttt{.....}\\ \texttt{.a.a.}\\ \texttt{.....}\\ \texttt{.a.a.}\\ \texttt{.....}\end{bmatrix}\quad \begin{bmatrix}\texttt{.d.d.}\\ \texttt{c.e.c}\\ \texttt{.f.f.}\\ \texttt{c.d.c}\\ \texttt{.e.e.}\end{bmatrix}\quad \begin{bmatrix}\texttt{.....}\\ \texttt{.b.b.}\\ \texttt{.....}\\ \texttt{.b.b.}\\ \texttt{.....}\end{bmatrix},$$

where entry $(0,1,1) = \texttt{a}$, entry $(1,0,1) = \texttt{d}$, entry $(1,1,0) = \texttt{c}$, $\ldots$, entry $(2,3,3) = \texttt{b}$. The cubes in positions $(1,1,1)$, $(1,1,3)$, $(1,3,1)$, $(1,3,3)$ of this example have the respective names `abcedf`, `abcefd`, `abcdfe`, `abcdef`. In a similar way an $l \times m \times n$ brick has colors represented by a $(2l+1) \times (2m+1) \times (2n+1)$ tensor; and the tensor

$$\begin{bmatrix}\texttt{...........}\\ \texttt{.a.a.a.a.a.}\\ \texttt{...........}\\ \texttt{.a.a.a.a.a.}\\ \texttt{...........}\\ \texttt{.a.a.a.a.a.}\\ \texttt{...........}\end{bmatrix} \begin{bmatrix}\texttt{.c.c.c.c.c.}\\ \texttt{b.f.d.f.d.b}\\ \texttt{.e.e.b.b.f.}\\ \texttt{b.c.d.f.e.b}\\ \texttt{.f.f.e.d.d.}\\ \texttt{b.e.d.b.e.b}\\ \texttt{.c.c.c.c.c.}\end{bmatrix} \begin{bmatrix}\texttt{...........}\\ \texttt{.d.b.e.e.e.}\\ \texttt{...........}\\ \texttt{.d.b.c.c.c.}\\ \texttt{...........}\\ \texttt{.d.b.f.f.f.}\\ \texttt{...........}\end{bmatrix} \begin{bmatrix}\texttt{.c.c.c.c.c.}\\ \texttt{b.f.d.b.f.b}\\ \texttt{.e.e.f.d.d.}\\ \texttt{b.c.d.e.f.b}\\ \texttt{.f.f.b.b.e.}\\ \texttt{b.e.d.e.d.b}\\ \texttt{.c.c.c.c.c.}\end{bmatrix} \begin{bmatrix}\texttt{...........}\\ \texttt{.a.a.a.a.a.}\\ \texttt{...........}\\ \texttt{.a.a.a.a.a.}\\ \texttt{...........}\\ \texttt{.a.a.a.a.a.}\\ \texttt{...........}\end{bmatrix}$$

represents a "magnificent brick" whose faces are colored `a`, `b`, `c` (each twice).

(b) Let there be lmn primary items $(2i+1)(2j+1)(2k+1)$ for the cube positions, 30 secondary items for the cube names, and $lm(n+1) + l(m+1)n + (l+1)mn$ secondary items xyz for the cube faces, where $0 \le x \le 2l$, $0 \le y \le 2m$, $0 \le z \le 2n$, $(x \bmod 2) + (y \bmod 2) + (z \bmod 2) = 2$. For example, the option for position 135 in solution (a) is '135 `acbefd` 035:`a` 125:`b` 134:`d` 136:`f` 145:`e` 235:`c`'. We also introduce six primary items to enforce the rule about solid colors on the brick's faces. Each of them has six options, one for each color c; for example, the options for the top face are 'top 101:c 103:c 105:c

automorphisms
essentially different
exact cover problem
Gardner
Perjés
even/odd coordinates
array, 3-dimensional
tensor

107:c 109:c 301:c 303:c 305:c 307:c 309:c'. The number of solutions is reduced by a factor of 720 if we remove all but one of the 720 options for position 111.

It turns out that the brick's face colors have an interesting property in every solution: A repeated face color occurs only on opposite, parallel faces. The example $1\times2\times2$ brick has face colors $\mathtt{ab}\times\mathtt{cc}\times\mathtt{de}$; the $2\times3\times5$ brick in (a) has colors $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cc}$.

A brick is considered to be essentially the same as any other that's obtained from it by rotation, reflection, and/or permutation of colors. The example $1\times2\times2$ brick above has 8 automorphisms; for example, we can reflect top $\leftrightarrow$ bottom and swap $\mathtt{d}\leftrightarrow\mathtt{e}$. The $2\times3\times5$ brick above has 2 automorphisms: The nontrivial one reflects front $\leftrightarrow$ back, top $\leftrightarrow$ bottom, $\mathtt{e}\leftrightarrow\mathtt{f}$.

There's *another* $1\times2\times2$ brick, whose face colors are $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ef}$. It has 16 automorphisms. Thus it occurs only once among the three solutions found by Algorithm C when $(l,m,n)=(1,2,2)$; the other two solutions are equivalent to each other.

There's a *unique* $1\times2\times3$ brick, easily found by hand. It has colors $\mathtt{ab}\times\mathtt{cc}\times\mathtt{dd}$, and 8 automorphisms. (Clearly $1\times m\times n$ is possible only if $mn\le6$.)

The $2\times2\times2$ bricks are especially interesting because MacMahon himself and his friend J. R. J. Jocelyn considered this case (with six different face colors), when they introduced the 30 6-color cubes in U.K. Patent 8275 of 1892. They observed that one can choose any "prototype" cube and replicate it at twice the size, by assembling eight of the other cubes. This can be done in two ways — using, in fact, the same eight cubes. But those two solutions are isomorphic, in 24 different ways. [See *Proc. London Math. Soc.* **24** (1893), 145–155. Their 8-cube puzzle was sold under the name "Mayblox."]

Gerhard Kowalewski, in *Alte und neue mathematische Spiele* (1930), 14–19, found a $2\times2\times2$ brick with face colors $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cd}$. Ferdinand Winter, in *Mac Mahons Problem: Das Spiel der 30 bunten Würfel* (1933), 67–87, found another, with face colors $\mathtt{aa}\times\mathtt{bc}\times\mathtt{de}$. And there's also a fourth solution, having Winter's face colors:

MacMahon

$$\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.c.c.}\\\mathtt{e.b.f}\\\mathtt{.f.e.}\\\mathtt{e.b.f}\\\mathtt{.d.d.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.d.d.}\\\mathtt{.....}\\\mathtt{.c.c.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.c.c.}\\\mathtt{e.a.f}\\\mathtt{.f.e.}\\\mathtt{e.a.f}\\\mathtt{.d.d.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.b.b.}\\\mathtt{.....}\\\mathtt{.b.b.}\\\mathtt{.....}\end{bmatrix};$$

Kowalewski

$$\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.b.b.}\\\mathtt{c.e.d}\\\mathtt{.f.f.}\\\mathtt{c.e.d}\\\mathtt{.b.b.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.d.c.}\\\mathtt{.....}\\\mathtt{.d.c.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.b.b.}\\\mathtt{c.f.d}\\\mathtt{.e.e.}\\\mathtt{c.f.d}\\\mathtt{.b.b.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix};$$

Winter

$$\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.b.b.}\\\mathtt{d.c.e}\\\mathtt{.e.d.}\\\mathtt{d.f.e}\\\mathtt{.c.c.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.f.f.}\\\mathtt{.....}\\\mathtt{.b.b.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.b.b.}\\\mathtt{d.c.e}\\\mathtt{.e.d.}\\\mathtt{d.f.e}\\\mathtt{.c.c.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix};$$

Fourth

$$\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.b.b.}\\\mathtt{d.f.e}\\\mathtt{.e.d.}\\\mathtt{d.f.e}\\\mathtt{.c.c.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.c.c.}\\\mathtt{.....}\\\mathtt{.b.b.}\\\mathtt{.....}\end{bmatrix}\begin{bmatrix}\mathtt{.b.b.}\\\mathtt{d.f.e}\\\mathtt{.e.d.}\\\mathtt{d.f.e}\\\mathtt{.c.c.}\end{bmatrix}\begin{bmatrix}\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\\\mathtt{.a.a.}\\\mathtt{.....}\end{bmatrix}.$$

These solutions have respectively (24, 8, 4, 8) automorphisms; hence Algorithm C finds $48/24+48/8+48/4+48/8=26$ solutions to the case $l=m=n=2$.

Larger cases have solutions that are, perhaps, even more remarkable; but there's room here for only a brief summary. For each feasible case of $l\times m\times n$ bricks with particular face colors, we list the number of different solutions with $(1,2,4,8)$ automorphisms. *Case* $2\times2\times3$*:* $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cc}$, $(0,0,1,0)$; $\mathtt{aa}\times\mathtt{bc}\times\mathtt{dd}$, $(0,2,6,1)$; $\mathtt{aa}\times\mathtt{bc}\times\mathtt{de}$, $(0,1,6,0)$; $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ee}$, $(0,1,2,0)$; $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ef}$, $(0,0,2,0)$. *Case* $2\times2\times4$*:* $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cc}$, $(0,0,1,0)$; $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cd}$, $(0,0,1,0)$; $\mathtt{aa}\times\mathtt{bc}\times\mathtt{dd}$, $(0,3,4,2)$; $\mathtt{aa}\times\mathtt{bc}\times\mathtt{de}$, $(0,11,14,2)$; $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ee}$, $(0,2,2,3)$; $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ef}$, $(0,1,1,1)$. *Case* $2\times2\times5$*:* $\mathtt{aa}\times\mathtt{bc}\times\mathtt{dd}$, $(0,5,4,0)$; $\mathtt{aa}\times\mathtt{bc}\times\mathtt{de}$, $(0,18,9,0)$; $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ee}$, $(0,0,1,0)$; $\mathtt{ab}\times\mathtt{cd}\times\mathtt{ef}$, $(0,2,5,1)$. *Case* $2\times3\times3$*:* $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cc}$, $(2,15,4,0)$; $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cd}$, $(4,8,1,0)$; $\mathtt{aa}\times\mathtt{bc}\times\mathtt{de}$, $(1,4,1,2)$. *Case* $2\times3\times4$*:* $\mathtt{aa}\times\mathtt{bb}\times\mathtt{cd}$,

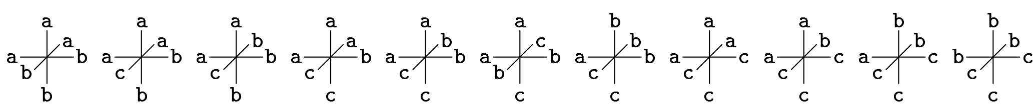
automorphisms
MacMahon
Jocelyn
historical notes
Patent
Mayblox
Kowalewski
Winter
MacMahon

barycentric coordinates
McComb
Scherphuis

$(6,8,1,0)$; $\texttt{aa}\times\texttt{bc}\times\texttt{de}$, $(0,6,0,0)$; $\texttt{ab}\times\texttt{cc}\times\texttt{dd}$, $(0,4,2,0)$; $\texttt{ab}\times\texttt{cc}\times\texttt{de}$, $(0,2,0,0)$; $\texttt{ab}\times\texttt{cd}\times\texttt{ee}$, $(0,2,0,0)$; $\texttt{ab}\times\texttt{cd}\times\texttt{ef}$, $(0,7,0,0)$. *Case* $2\times3\times5$*:* $\texttt{aa}\times\texttt{bb}\times\texttt{cc}$, $(0,2,0,0)$.

(Conspicuous by its absence is the case $l=m=n=3$. There's no $3\times3\times3$ brick, although we can come close: A $3\times3\times3$ without a corner can be made from 26 of the 30; or without the middle cube and the one above it, from 25.)

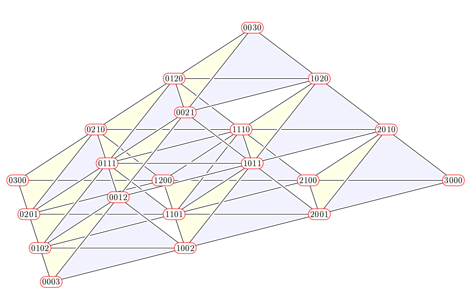
148. There are eleven such cubes, and they can be matched in many pleasant ways:

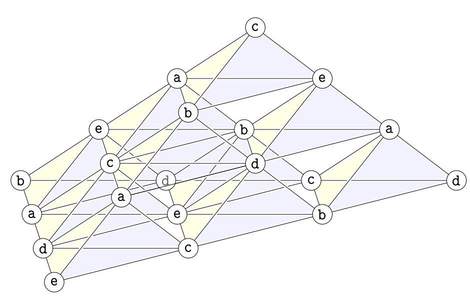


$$\begin{bmatrix}\texttt{.......}\\\texttt{.......}\\\texttt{.......}\\\texttt{...c...}\\\texttt{.......}\\\texttt{.......}\\\texttt{.......}\end{bmatrix}\begin{bmatrix}\texttt{.......}\\\texttt{.......}\\\texttt{...a...}\\\texttt{..a.c..}\\\texttt{...c...}\\\texttt{.......}\\\texttt{.......}\end{bmatrix}\begin{bmatrix}\texttt{.......}\\\texttt{.a.b.a.}\\\texttt{.......}\\\texttt{.b.a.b.}\\\texttt{.......}\\\texttt{.a.b.a.}\\\texttt{.......}\end{bmatrix}\begin{bmatrix}\texttt{.a.a.a.}\\\texttt{a.b.a.c}\\\texttt{.c.b.c.}\\\texttt{a.b.a.c}\\\texttt{.b.a.a.}\\\texttt{a.c.b.c}\\\texttt{.c.c.c.}\end{bmatrix}\begin{bmatrix}\texttt{.......}\\\texttt{.b.c.b.}\\\texttt{.......}\\\texttt{.c.b.c.}\\\texttt{.......}\\\texttt{.b.c.b.}\\\texttt{.......}\end{bmatrix}\begin{bmatrix}\texttt{.......}\\\texttt{.......}\\\texttt{...b...}\\\texttt{..b.c..}\\\texttt{...c...}\\\texttt{.......}\\\texttt{.......}\end{bmatrix}\begin{bmatrix}\texttt{.......}\\\texttt{.......}\\\texttt{.......}\\\texttt{...c...}\\\texttt{.......}\\\texttt{.......}\\\texttt{.......}\end{bmatrix}$$

149. Label the vertices with nonnegative barycentric coordinates $wxyz$, where $w+x+y+z=3$. Also label the ten unit tetrahedra with barycentric coordinates $stuv$, where $s+t+u+v=2$; the vertices $wxyz$ of tetrahedron $stuv$ are then $stuv+\{1000,0100,0010,0001\}$. Introduce ten primary items $stuv$ for the tetrahedra, and ten more $\texttt{abcd}$, $\texttt{abdc}$, $\texttt{abce}$, $\texttt{adec}$, …, $\texttt{bcde}$, $\texttt{bced}$ for the different colorings. And introduce 20 secondary items $wxyz$ for the vertices.

Then the admissible vertex colors are the solutions to the XCC problem with 1200 options '$stuv\ \alpha\ v_1{:}p_1\ \ldots\ v_4{:}p_4$', where α is a coloring, $v_1v_2v_3v_4$ are the vertices of $stuv$, and $p_1p_2p_3p_4$ is an even permutation of α's colors. Curiously, this problem has 2880 solutions (found in 500 Mμ) — and they're all equivalent to the one below, under the $5!\,4!=2880$ automorphisms present.

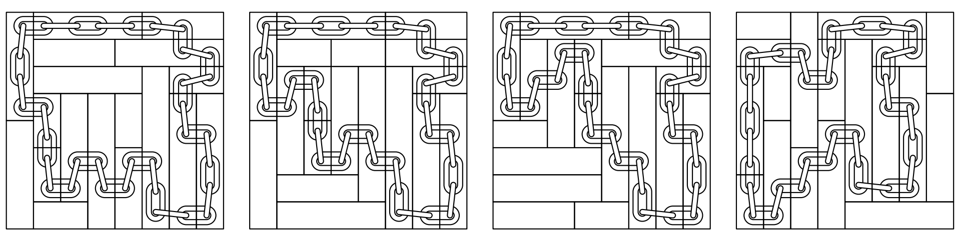




(This problem was posed in 2015 by J. McComb, and solved by J. Scherphuis.)

150. Notice that there are fourteen distinct pieces, with four pairs of two. So we use Algorithm M, with 14 primary items for pieces and 64 for cells. We also introduce secondary items for edges between cells, with colors to indicate the presence or absence of links. The final two pieces must obviously be adjacent, hence we can combine them into a "super-piece" of size 11; then all interfaces between adjacent cells are identical. We can remove symmetry by forcing the super-piece to be in one of 18 positions.

Then 43 solutions are found, in 7 Gμ. Here are some typical examples:



(i) (ii) (iii) (iv)

Solution (i) appears in Hoffmann's *Puzzles Old and New*, puzzle 3–18. Solution (iii) avoids most of the lower left quadrant, and solution (iv) avoids the entire right column. If we ignore blank spaces, the links form eight different paths, all of length 34. Paths (i), (ii), (iii), (iv) occur in respectively 1, 15, 9, 3 of the 43 solutions. [The Endless Chain Puzzle was distributed circa 1887 by Reason Manufacturing Company.]

151. (a) The key idea is to start by *factoring* this problem, by considering only the task of edge-matching between adjacent dominoes, while ignoring the loop details.

Algorithm M applies, with primary items 1–9 and a–i for the distinct on-off patterns of attachment points, as well as primary items ij for each cell to be covered ($0 \le i < 8$, $0 \le j < 9$), and two special primary items H, V. There are 63+64 secondary items h_{ij} and v_{ij}, to indicate path/nopath at internal attachment points. Typical options:

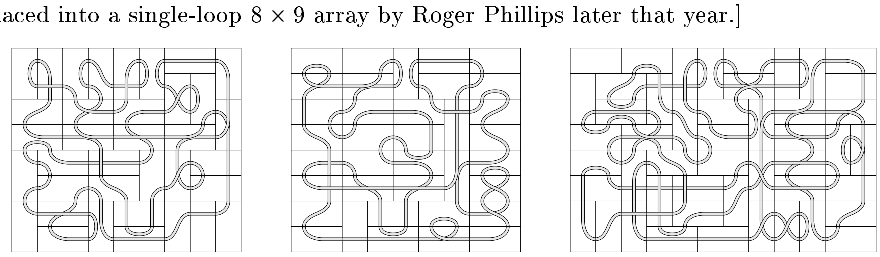
$$\text{`a 10 11 } v_{12}{:}1\ h_{21}{:}1\ h_{20}{:}1,\ h_{10}{:}0\ h_{11}{:}1 \text{ H'},$$
$$\text{`a 11 21 } h_{11}{:}0\ v_{12}{:}0\ v_{22}{:}1,\ h_{31}{:}1\ v_{21}{:}1\ v_{11}{:}1 \text{ V'};$$

the goal is to find an exact cover with multiplicities 1 for patterns 1–9, multiplicities 3 for patterns a–i, and multiplicities 18 for H and V. (There are millions of solutions.)

Once that task is solved, we need to assign the actual dominoes whose subpaths jointly define a single loop. A (nontrivial) program, whose structure has a lot in common with Algorithm X, will find such assignments in microseconds (although a full day might be needed to actually *write* that program).

(b) Now H and V should have multiplicities 32 and 4. (Also, we can save about half of Algorithm M's running time by omitting vertical placements at odd height.) The algorithm finds 6420 solutions; suitable domino assignments are then found in a flash.

[These 36 path dominoes were first studied by Ed Pegg Jr. in 1999, and first placed into a single-loop 8×9 array by Roger Phillips later that year.]



152. This (factored) problem is like the previous one, but with an additional pattern j of multiplicity 11, and without H or V. One needs to be lucky to find a solution; the author struck it rich with Algorithm M after 35.1 Tμ.

[Notice that exactly 32 of the 48 path dominoes have no crossings. Thus it is irresistible to try to place them on a chessboard, so as to form a single noncrossing

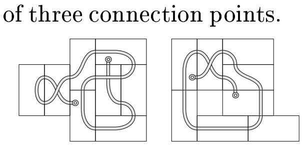
Hoffmann
Reason, Henry, Manufacturing Company
factoring
multiplicities
Pegg
Phillips
author
crossings

loop. Unfortunately, Algorithm M tells us that such a mission is impossible, even with multiple loops, because the corresponding factored problem has no solution. *Something* interesting, however, can surely be done with those 32.]

153. (a) Algorithm M quickly verifies the uniqueness of the solution below, if we add a blank *monomino* of multiplicity 4. ["Line puzzles" like this were invented by Bill Darrah; several of his ingenious designs were made by Binary Arts in 1994 and 1999.]

(b) There are 30 patterns, three for each distinct choice of three connection points.

(c) Trials with random choices of respectively (2, 2, 4) sets of (2, 3, 4) distinct connection points usually give no solutions at all. But one of the author's first 1000 trials was suitable, and it led to a nice puzzle whose solution is shown.



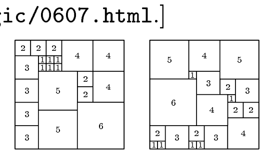
154. The integer solutions to $P(n) = n(n+1)^2(n+2)/12 = m^2$ involve perfect squares u^2 and v^2 with $v^2 \approx 3u^2$. If $|v^2 - 3u^2|$ is sufficiently small, v/u must be a convergent to the continued fraction $\sqrt{3} = 1 + /\!/1, 2, 1, 2, 1, 2, 1, 2, \ldots /\!/$ (see exercise 4.5.3–42).

Pursuing this idea, let $\theta = 2 + \sqrt{3}$, $\hat{\theta} = 2 - \sqrt{3}$, $\langle a_n \rangle = \langle (\theta^n + \hat{\theta}^n)/2 \rangle = \langle 1, 2, 7, 26, 97, \ldots \rangle$ and $\langle b_n \rangle = \langle (\theta^n - \hat{\theta}^n)/(2\sqrt{3}) \rangle = \langle 0, 1, 4, 15, 56, \ldots \rangle$. Notice that $a_n^2 = 3b_n^2 + 1$; $(a_n + 3b_n)^2 = 3(a_n + b_n)^2 - 2$. We find that $P(n)$ is a perfect square if and only if $n = 6b_m^2$ for some m (thus $n = 0$, 6, 96, 1350, 18816, ...) or $n = (a_m + 3b_m)^2$ for some m (thus $n = 1$, 25, 361, 5041, 70225, ...).

[See R. Wainwright, in *Puzzlers' Tribute* (A. K. Peters, 2002), 277–281; also Erich Friedman's survey in `www2.stetson.edu/~efriedma/mathmagic/0607.html`.]

155. (a) Algorithm M finds $8 \cdot 7571$ solutions, in 60 Gμ.

(b) The maximum is 35 (not easy to find!), and the minimum is 5. [This exercise was suggested by Robert Reid, who found a minimum solution by hand in 2000.]



156. At level l of backtracking, branch on all ways to fill the leftmost unfilled cell of the topmost unfilled row. Even though no MRV heuristic is used, this method needs just 2.0 teramems (and negligible memory) to find 18656 solutions. The search tree has 61636037366 nodes.

We can save a factor of 8 by removing symmetry: The 1×1 square can be confined to cells (i, j) with $i < 18$ and $j \geq 35 - i$. Furthermore, if (i, j) is on the diagonal $(j = 35 - i)$, the context of the 1×1 square must be either ╤╧ or ╧╤, and we can insist on the former. Now we find 2332 solutions (and 6975499717 nodes), in just 235 gigamems.

By contrast, the MCC problem (61) for $n = 8$ has 1304 items and 7367 options of total length 205753, when we restrict the options of #1 to $i < 18$ and $j \geq 35 - i$. It needs 490.6 teramems to find 2566 solutions; postprocessing reduces that number to 2332, because 468 of those 2566 have #1 in position (i, j) with $j = 35 - i$.

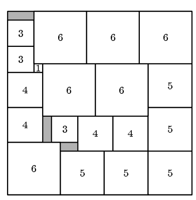
We conclude that a dancing-links approach is decidedly *not* the method of choice for this partridge problem; straightforward backtracking with bitwise operations is more than 2000 times faster! Indeed, we might consider ourselves fortunate to pay "only" a 2000-fold cost penalty, since each of the 841 options for #8 in (61) contributes 65 nodes to doubly linked lists. Such updating and downdating keeps the dancers extremely busy.

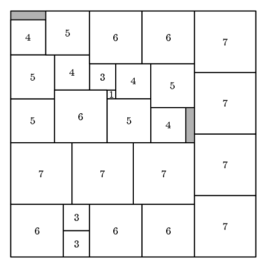
[*Historical notes:* The 2332 solutions for $n = 8$ were first found by Bill Cutler in 1996, using a refinement of the backtrack approach described above. At that time no solutions for $n < 11$ had been known, although Wainwright knew how to solve $12 \leq n \leq 15$ in 1981, and C. H. Jepsen and S. Ahearn had presented constructions for $11 \leq n \leq 33$ in *Crux Mathematicorum* **19** (1993), 189–191. The puzzle can surely be solved for all $n > 7$, but no proof is yet known.]

monomino
multiplicity
Line puzzles
Darrah
Binary Arts
random choices
author
Wainwright
Friedman
Reid
symmetry
dancing links, sometimes slow
Cutler
Wainwright
Jepsen
Ahearn

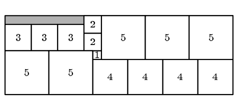
quadrants
octants
dominating sets
queen graph
kernels
maximal independent sets, see kernels

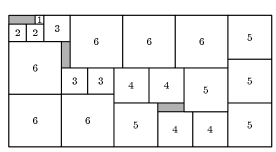
157. Algorithm M readily shows the nonexistence of perfect packings, but the backtrack method of exercise 156 is much better to show that we can't pack all but one 2×2. That method also shows that we *can* pack all but two of them:

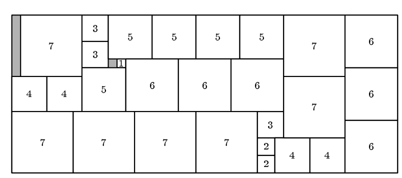




158. The following solutions can be proved optimum with bitwise backtracking as in exercise 156:







159. Replace # by four primary items $\#_0$, $\#_1$, $\#_2$, $\#_3$ representing "quadrants," and use $\#_{2\lfloor i/4\rfloor+\lfloor j/4\rfloor}$ in place of # in (64). Then partition into ten separate cases, in which the multiplicities $m_0m_1m_2m_3$ of $\#_0\#_1\#_2\#_3$ are respectively (2012, 2111, 2120, 3002, 3011, 3020, 3110, 4010, 4001, 5000). (Omit options containing $\#_k$ of multiplicity 0.) These cases produce (134, 884, 33, 23, 34, 1, 16, 0, 22, 0) solutions, in (95, 348, 60, 23, 75, 8, 19, 2, 10, 0) megamems. (Notice that $4\cdot 134+4\cdot 884+8\cdot 33+4\cdot 23+8\cdot 34+8\cdot 1+4\cdot 16+8\cdot 0+4\cdot 22+4\cdot 0=4860$.) The running time has decreased by a factor of 20.

[For larger values of n we could divide the cells into nine regions: eight octants, plus a special region containing the diagonals (and the middle row, column if n is odd).]

160. There are 589 components, among which are 388 isolated vertices and one giant of size 3804. The other 200 components have sizes ranging from 2 to 12. (For example, the first three solutions in (65) belong to the giant component; the other belongs to a component of size 8.)

161. In general, consider the problem of finding all the m-vertex dominating sets of a graph G; the $n \times n$ m-queen problem is the special case where G is the queen graph of order n. Then the options (64) have the form '$\#\ v\ v_1\ \ldots\ v_t$', where $\{v_1,\ldots,v_t\}$ are the vertices adjacent to v, and # is a special primary item of multiplicity m.

Variant (i) is equivalent to asking for all *kernels* of size m (all of the maximal independent sets). Let there be a secondary item e for every edge in G; the options are then '$\#\ v\ v_1\ \ldots\ v_t\ e_1\ \ldots\ e_t$', where e_j is the edge between v and v_j. An 8×8 chessboard has $8\cdot 91=728$ kernels of size 5. (It also has 6912, 2456, and 92 kernels of sizes 6, 7, and 8; see exercise 7.1.4–241(a).)

For variant (ii) we simply shorten v's option to '$\#\ v_1\ \ldots\ v_t$'; some other option must then cover v. Exactly 352 of the 5-queen solutions satisfy (ii).

Variant (iii) seems a bit harder to formulate. Let there be a secondary item $\hat{v}$ for each vertex v. The option for choosing v can then be '$\#\ v\ \hat{v}{:}1\ v_1\ \ldots\ v_t\ \hat{u}_1{:}0\ \ldots\ \hat{u}_s{:}0$', where $\{u_1,\ldots,u_s\}=V\setminus\{v,v_1,\ldots,v_t\}$ is the set of vertices *not* adjacent to v. The 8×8 chessboard has 20 clique-dominators of size 5.

[Chapter 10 of the classic work *Mathematische Unterhaltungen und Spiele* by W. Ahrens (1910) is an excellent survey of early work on queen-domination problems.]

162. Formulate these as MCC problems, by starting with the ordinary options for the n queens problem (see (23)), then adding additional options such as '$\#\ r_j\ c_{k+1}\ a_{j+k+1}\ b_{j-k-1}\ r_{j+1}\ c_{k+3}\ a_{j+k+3}\ b_{j-k-3}\ r_{j+2}\ c_k\ a_{j+k+2}\ b_{j-k+2}\ r_{j+3}\ c_{k+2}\ a_{j+k+5}\ b_{j-k+1}$' to represent a contained $\mathcal{Q}_4$, for $1 \le j, k \le n-3$. Here # is a new primary item, which is given the desired multiplicity.

(a) 15; one can, in fact, get disjoint $\mathcal{Q}_4$ and $\mathcal{Q}_5$ in a $\mathcal{Q}_{15}$.

(b, c) 17. Put a queen in the center, make a pinwheel! [See Ahrens (1910), 258.]

(d) 22; see below. Algorithm M proves $n = 21$ impossible after 1.2 teramems.

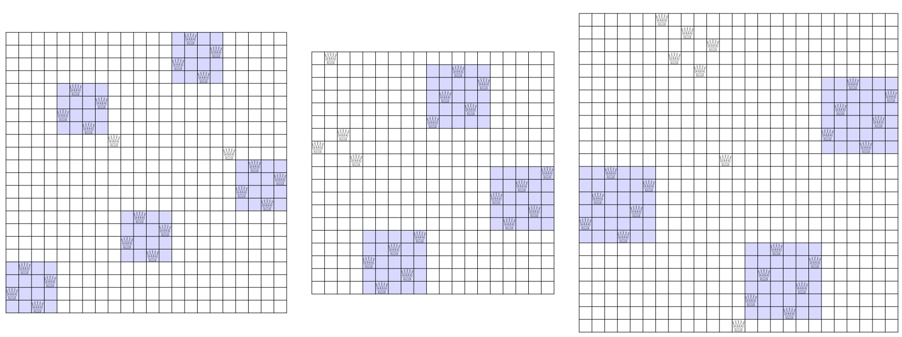
(e) 16; there are four essentially different solutions.

(f) 19; see below. Only 35 Gμ to show that $n = 18$ is too small.

(g) 20(!). Once you know this, Algorithm X will find all 18 solutions in 2 Mμ.

(h) 22; there are 28 essentially different solutions.

(i) 25; see below. (After 6 teramems, we learn that $n = 24$ doesn't work.)



163. Sometimes Algorithm M is called on to choose zero or more items from an empty list. Then it sets $\texttt{FT[}l\texttt{]} \leftarrow i$ and $x_l \leftarrow i$, where i is the item whose list is empty; but step M5 doesn't actually tweak anything. The peculiar rule in (71) ensures that step M8 doesn't actually untweak anything as we backtrack.

164. If $x_j \le N$, node x_j is the header for item x_j; there's no further option for such j.

[A good implementation will also extend answer 12, so that the relative positions of each x_j in the search tree are identified. For this purpose one can add a new array `SCORE`, setting $\texttt{SCORE[}l\texttt{]} \leftarrow \theta_i$ and $\texttt{FT[}l\texttt{]} \leftarrow 0$ at the end of step M3. When printing the jth step x_j of a solution, the old answer 12 is used if $\texttt{FT[}j\texttt{]} = 0$; otherwise that answer is modified as follows: If $x \le N$ and ($x = \texttt{FT[}j\texttt{]}$ or $x = \texttt{TOP(FT[}j\texttt{])}$), print '`null NAME(`$x$`)`'; otherwise print option x as before. Conclude by looping with $i \leftarrow 0$, $q \leftarrow \texttt{FT[}j\texttt{]}$ rather than $i \leftarrow \texttt{TOP(}x\texttt{)}$, $q \leftarrow \texttt{DLINK(}i\texttt{)}$; report '$k$ of $\texttt{SCORE[}j\texttt{]}$' rather than '$k$ of $\texttt{LEN(}i\texttt{)}$'.]

165. (a) To cover 2 of 4, we have 3 choices at the root, then 3 or 2 or 1 at the next level, hence (1, 3, 6) cases at levels (0, 1, 2). To cover 5 of 7, there are (1, 3, 6, 10, 15, 21) cases at levels (0, 1, …, 5). Thus the search profile with item 1 first is (1, 3, 6, $6\cdot3$, $6\cdot6$, $6\cdot10$, $6\cdot15$, $6\cdot21$). The other way is better: (1, 3, 6, 10, 15, 21, $21\cdot3$, $21\cdot6$).

(b) With item 1 first the profile is $(a_0, a_1, \ldots, a_p, a_pa_1, \ldots, a_pa_q)$, where $a_j = \binom{j+d}{d}$. We should branch on item 2 first because $a_{p+1} < a_pa_1$, $a_{p+2} < a_pa_2$, …, $a_q < a_pa_{q-p}$, $a_qa_1 < a_pa_{q-p+1}$, …, $a_qa_{p-1} < a_pa_{q-1}$. (These inequalities follow because the sequence $\langle a_j\rangle$ is strongly log-concave: It satisfies the condition $a_j^2 > a_{j-1}a_{j+1}$ for all $j \ge 1$. See exercise MPR–125.)

Ahrens
MCC problems
Ahrens
profile
log-concave

166. (a) The “monus” operation $x \dot{-} y = \max(x-y, 0)$ is good for situations like this:

$$\theta_p \;=\; \bigl(\mathtt{LEN}(p)+1\bigr) \dot{-} \bigl(\mathtt{BOUND}(p) \dot{-} \mathtt{SLACK}(p)\bigr).$$

(b) It's better to branch on p' (although this may be counterintuitive).

[The author's implementation of step M3 breaks ties by first preferring an item with smaller SLACK, then preferring longer LEN when the SLACKs are equal. Thus, his MRV replaces answer 9 by this: Set $\theta \leftarrow \infty$, $p \leftarrow \mathtt{RLINK}(0)$. While $p \neq 0$, do the following: Set $\lambda \leftarrow \theta_p$; if $\lambda < \theta$ or ($\lambda = \theta$ and $\mathtt{SLACK}(p) < \mathtt{SLACK}(i)$) or ($\lambda = \theta$ and $\mathtt{SLACK}(p) = \mathtt{SLACK}(i)$ and $\mathtt{LEN}(p) > \mathtt{LEN}(i)$) set $\theta \leftarrow \lambda$, $i \leftarrow p$; then set $p \leftarrow \mathtt{RLINK}(p)$.

167. Step M3 isn't precisely defined; therefore *any* change to v_p could possibly affect the behavior. But let's assume that step M3 is implemented as in exercise 166.

Even so, there can be differences. A minor difference arises, for instance, if there are *no* options: A primary item with multiplicity $[0\,..\,1]$ will be inactivated by covering in step M4; with multiplicity $[0\,..\,2]$, it will become inactive at the end of step M5.

There can also be more significant differences. Suppose there's just one option, 'a', and one primary item. If a has multiplicity 1, we simply cover a as in Algorithm X. But if a has multiplicity $[1\,..\,2]$, we'll do some tweaking and untweaking — even entering a new level, and taking a null branch there.

On the other hand, the differences can't get much worse. Let $\mathtt{BOUND}_0(p)$ and $\mathtt{BOUND}_1(p)$ denote the values of $\mathtt{BOUND}(p)$ when the upper bound v_p has respectively been specified as M_p and $M_p+\delta$. If the same options are chosen, we'll have $\mathtt{BOUND}_1(p) = \mathtt{BOUND}_0(p) + \delta$ throughout the algorithm, because $\mathtt{BOUND}(p)$ is adjusted appropriately whenever the algorithm recursively reduces the problem by removing an option. Also $\mathtt{SLACK}_1(p) = \mathtt{SLACK}_0(p) + \delta$. One can then prove, by induction on the computation, that the same options are indeed chosen (possibly with different amounts of tweaking).

Any two values of v_p that are $M_p + 2$ or more will be totally equivalent.

168. Introduce a new primary item '!' and a new secondary item '+'. Replace the two copies of α by '! +:0', '! α', 'α +:1'. [Similarly, three copies of α can be replaced by '! +:0', '! α', '!! ++:0', '!! α +:1', 'α ++:1', after introducing '!!' and '++'.]

169. Let there be one primary item, #, together with one secondary item for each vertex. And let there be one option, '# v v_1:0 … v_d:0' for each vertex v, where v_1 through v_d are the neighbors of v. Finally, let # have multiplicity t. [Notice that the secondary items in this construction are colored either with 0 or not at all!]

170. Introduce the primary item !v for each vertex, and give it $d+1$ options: '# !v v:1 v_1:0 … v_d:0', '!v v:0 v_1:1', '!v v:0 v_1:0 v_2:1', …, '!v v:0 v_1:0 … v_{d-1}:0 v_d:1'.

171. Let there be ten primary items v, for $0 \le v < 10$; also fifteen primary items $\#uv$, with multiplicity $[1\,..\,5]$, for each edge $u \,—\, v$, where the edges are $0 — 1 — 2 — 3 — 4 — 0$, $0 — 5$, $1 — 6$, $2 — 7$, $3 — 8$, $4 — 9$, and $5 — 7 — 9 — 6 — 8 — 5$. Let there be $26 \cdot 10$ secondary items $\mathtt{a}_v$ through $\mathtt{z}_v$, for $0 \le v < 10$; also $26 \cdot 30$ secondary items $\mathtt{a}_{uv}$ through $\mathtt{z}_{uv}$, for $u \not— v$; also a secondary item w for each word in, say, $\mathtt{WORDS}(1000)$. There are 26 options, '$\#uv$ $\mathtt{a}_u$ $\mathtt{a}_v$' through '$\#uv$ $\mathtt{z}_u$ $\mathtt{z}_v$', for each edge. And there are 10 options for each word; for example, the options for `added` are 'v $\mathtt{a}_v$:1 $\mathtt{b}_v$:0 $\mathtt{c}_v$:0 $\mathtt{d}_v$:1 $\mathtt{e}_v$:1 $\mathtt{f}_v$:0 … $\mathtt{z}_v$:0 $\mathtt{a}_{02}$ $\mathtt{a}_{03}$ $\mathtt{a}_{06}$ $\mathtt{a}_{07}$ $\mathtt{a}_{08}$ $\mathtt{a}_{09}$ $\mathtt{d}_{02}$ … $\mathtt{e}_{09}$ `added`', where $0 \le v < 10$.

Every solution to this MCC problem will be obtained 120 times, because the Petersen graph has 120 automorphisms. But symmetry can be broken by choosing the labels of 0, 1, and 3 at levels 0, 1, and 2, and by ordering the label ranks so that $r_0 > r_1$, $r_0 > r_2$, $r_0 > r_3$, $r_0 > r_6$, $r_1 > r_4$, $r_1 > r_5$, $r_3 > r_7$, $r_3 > r_8$, $r_3 > r_9$.

monus
author
multiplicity
MCC problem
automorphisms
symmetry can be broken

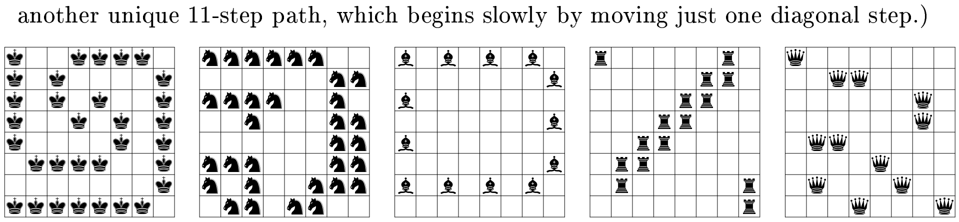
parity
Aztec diamond
double counting+

There are two solutions in WORDS(834), namely **muddy**, **thumb**, **books**, **knock**, **ended**, **apply**, **fifth**, **grass**, **civil**, (**refer** or **fewer**), found in 3.5 Tμ.

172. A construction analogous to answer 170 generates all solutions to the *weaker* problem where connectivity isn't tested; it's easy to remove the unconnected solutions from Algorithm M's output. Consider cycles first: There are $1+\binom{d}{2}$ options for each primary item $!v$, namely '$!v$ v:0' and '# $!v$ v:1 v_1:a_1 $\ldots$ v_d:a_d', where $a_1 \ldots a_d$ is a binary vector with $a_1+\cdots+a_d=2$. For the path problem, the options for the starting vertex should have $a_1+\cdots+a_d=1$, not 2. The options for all other vertices that aren't adjacent to the starting vertex should have d additional options '# E $!v$ v:1 v_1:a_1 $\ldots$ v_d:a_d', with $a_1+\cdots+a_d=1$, where E is a new primary item signifying the end vertex.

(a) Paths of length l are obtained when the multiplicity of # is set to $l+1$.

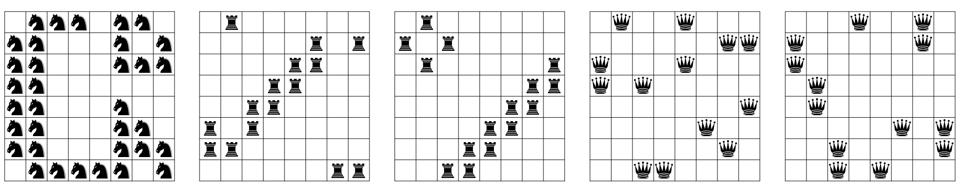
First let's restrict consideration to paths that start in the corner cell $(0,0)$. Then every essentially distinct path occurs twice — reflected about the diagonal. (i) There are 16 distinct snake-in-the-box king paths of length 31 from a given corner, found in 6 Tμ. One of them, illustrated below, also *ends* at a corner; hence it occurs four times, not two — twice in each direction. These paths are optimum, because we can divide the board into sixteen 2×2 subsquares, each of which can contain at most two kings. (ii) A single run, with the multiplicity of # set to $[32\,..\,33]$, suffices to find the 13 distinct knight solutions of length 31 in 58 Gμ, simultaneously showing that length 32 is impossible. One of the most remarkable solutions is shown below. (iii) With bishops we should first eliminate all squares of the wrong parity, because they cannot be connected to the start. Then the 32 solutions of length 12 are found in just 13 Mμ. (It's not difficult to prove by hand that an $n\times n$ board has exactly 2^{n-3} bishop solutions of length $2n-4$, when n is even.) (iv) Rook solutions are even easier to enumerate by hand: There are $(n-1)!^2$ of them, because we always have $n-k$ choices at steps $2k-1$ and $2k$. (Algorithm M finds the $7!^2=25401600$ solutions in 625 Gμ, while generating also 21488110 disconnected impostors.) However, $(n-2)!^2-(n-2)!$ of those solutions are counted twice, because they go from corner to corner and have no symmetry. Hence there are $25401600-517680/2=25142760$ distinct rook solutions of length 14. (v) Finally, there are 134 distinct queen solutions of length 11 — found and proved optimum in 17 Gμ, despite having 16788 options of total length 454380(!). The unique solution that occupies opposite corners is shown here. (You may enjoy finding another unique 11-step path, which begins slowly by moving just one diagonal step.)



Now let's consider paths that start in cell $(0,1)$ and do not end in a corner. (i) Five solutions with 32 kings are found (in 3.7 Tμ); but they all have 3-cycles and are disconnected. (ii) Knights, however, yield a big surprise: There's a unique path of length 33, doubly counted! (Found in 43 Gμ.) (iii) Bishop paths can't have length 12 unless they start or end in a corner. (iv) There are $N=(n-1)!^2-2(n-2)!^2$ solutions where the rook first moves down, and N where it first moves sideways. Of these, $2N_c$ end at $(n-1,n-2)$ and are double-counted by central symmetry, where $N_c=\bigl(2^{\lfloor n/2\rfloor-1}(\lfloor n/2\rfloor-1)!\bigr)^2$; $N_t=2(n-2)!$ end at $(1,0)$ and are *not* double-counted by transposition; N_t end at $(n-2,n-1)$ and aren't double-counted by dual transposition.

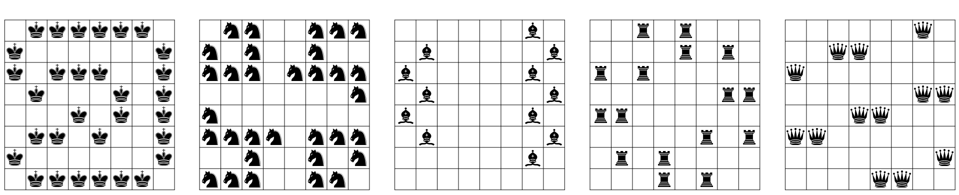
So there are $2N - N_c - (2(n-2)!^2 - N_t) = 47691936$ equivalence classes when $n = 8$.
(v) Another nice surprise greets us, namely a unique queen path of length 12!

The next step is to consider paths that start in $(0, 2)$ and don't end in the 12 types of cells already considered. And so on, for seven more cases. Of course rook counting gets hairier and hairier; we shall omit it. Unexpectedly, there's also another maximum queen path(!). All of these computations are fast, except that the kings need 6.3 Tμ.



(b) Cycles are similar, but symmetry now becomes even trickier. (i) The six distinct 31-cycles of a king are asymmetric, so they each appear eight times when reflected and/or rotated. (ii) But the four distinct 32-cycles of a knight include two that are equivalent to their transpose, and one (shown below) with central symmetry. (iii,v) A bishop has 36 distinct 12-cycles, and a queen has five 13-cycles, all asymmetric.

(iv) A rook, on the other hand, has oodles of 16-cycles, some of which (like the one illustrated) even have 4-fold symmetry under both horizontal and vertical reflection. Every rook snake-in-the-box 16-cycle can be represented uniquely as $(p_0q_0\ p_0q_1\ p_1q_1\ p_1q_2\ \ldots\ p_7q_7\ p_7q_0)$, where $p_0p_1\ldots p_7$ and $q_0q_1\ldots q_7$ are permutations of $\{0, 1, \ldots, 7\}$ with $p_0 = 0$ and $q_0 < q_1$. Consequently there are $8!^2/16 = 101606400$ of them, if symmetry isn't taken into account. That cycle is equivalent to its transpose if and only if $p_j = q_{(k-j) \bmod 8}$ for some k and all j; there are $8!/2 = 20160$ such cases. It is equivalent to its 180° rotation if and only if $p_j + p_{4+j} = q_j + q_{4+j} = 7$ for $0 \le j < 4$; there are $6 \cdot 4 \cdot 2 \cdot 8 \cdot 6 \cdot 4 \cdot 2/2 = 9216$ such cases. And it is equivalent to both, in $6 \cdot 4 \cdot 2 \cdot 8/2 = 192$ cases. Hence by "Burnside's lemma" there are $(101606400 + 0 + 9216 + 0 + 20160 + 0 + 20160 + 0)/8 = 12706992$ equivalence classes of rook cycles.



[T. R. Dawson introduced this problem for knights, and presented an example path of length 31 and an example cycle of length 32, in *L'Echiquier* (2) **2** (1930), 1085; **3** (1931), 1150. C. C. Verbeek posed the problem of maximizing the number of queens such that each is "attacked by exactly two others" in *Elsevier's Weekly* (June 1971); if we allow several queens in the same row, arguing that the first doesn't attack the third, 14 queens are actually possible (see P. Torbijn, *Cubism For Fun* **17** (1991), 19). The name 'snake-in-the-box' was coined by W. H. Kautz, *IRE Trans.* **EC-7** (1958), 177–180, for the case where G is an n-cube. The term 'coil-in-the-box' is often used nowadays for a snake-in-the-box cycle.]

Nikolai Beluhov proved in 2018 that, if $n \ge 6$ is even, all snake-in-the-box king paths of the maximal length $n^2/2 - 1$ on $n \times n$ boards have an interesting structure, which can be characterized completely. In fact, he showed that exactly $2n + (n \bmod 4)/2$

symmetry
central symmetry
180° symmetry
4-fold symmetry
180° rotation
Burnside's lemma
Dawson
Verbeek
Torbijn
Kautz
hypercube
n-cube
coil-in-the-box, see snake-in-the-box cycle
Beluhov

such paths are distinct under symmetry. Furthermore, there are exactly six distinct snake-in-the-box king *cycles* of length $n^2/2-1$, when $n \geq 8$ is a multiple of 4.

With arguments of a different kind, Beluhov has also proved that the longest snake-in-the-box paths and cycles of a *knight*, on an $m \times n$ board, have length $mn/2 - O(m+n)$. [To appear.]

173. (a) Write '$k \mathbin{—} ij$' if clue k is a (knight or bishop) move away from cell (i,j). For each row, column, and box, compute "quotas" r_i, c_j, and b_x, equal to 3 minus the number of pieces already present among the given clues. Also compute the quota p_k for each clue k, equal to the label minus the number of neighboring cells already occupied. There is no solution if any quota is negative.

Say that cell (i,j) of box x is *known* if it is occupied, or if $r_i = 0$ or $c_j = 0$ or $b_x = 0$, or if $p_k = 0$ for some $k \mathbin{—} ij$. Introduce primary items R_i, C_j, B_x, P_k for each row, column, box, or clue with a positive quota, having multiplicities r_i, c_j, b_x, p_k. There is one option for each unknown cell, namely '$R_i\ C_j\ B_x \bigcup\{P_k \mid k \mathbin{—} ij\}$'.

(b, c, d) See Fig. A–3. The knight puzzles with labels ≥ 6, and the bishop puzzles with labels 0, 10, and 12, are due to N. Beluhov; the others represent the author's best early attempts, not necessarily minimum. Solutions can be found in Appendix E.

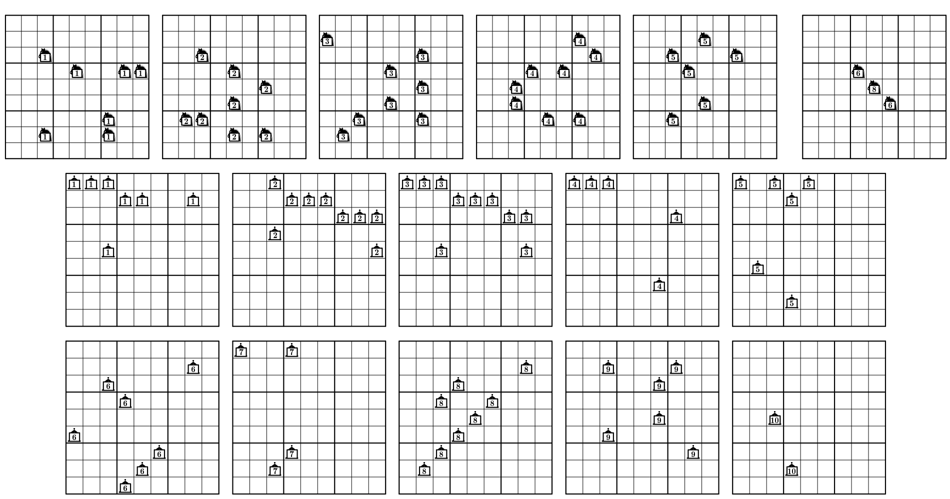
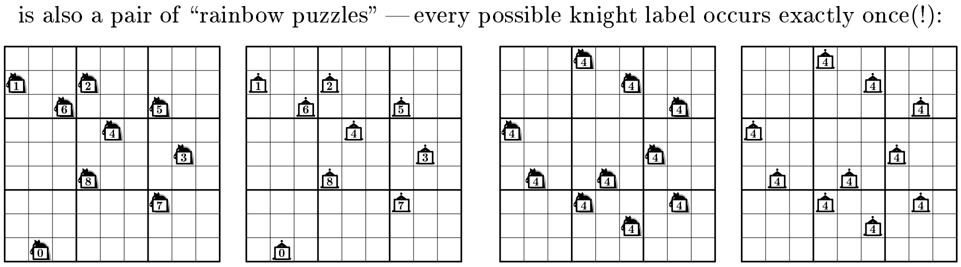


Fig. A–3. A gallery of knight and bishop sudoku puzzles.

[These variants of sudoku were devised by David Nacin and first published in *MAA Focus* **38**, 6 (Dec. 2018/Jan. 2019), 36; see also `quadratablog.blogspot.com`.]

174. Beluhov's remarkable solution, which he obtained with the help of a SAT solver, is also a pair of "rainbow puzzles" — every possible knight label occurs exactly once(!):



Beluhov
author
Nacin
SAT solver
rainbow puzzles

[Also shown are his 10-clue puzzles in which all the labels are equal.]

175. We can allow an option α to be repeated twice by simply replacing it by three options '$\alpha\ x$', '$\#\ x$', '$\#\ \alpha$', where # is a new primary item and x is a new secondary item. (If α contains uncolored secondary items $y_1, \ldots, y_s$, we should first replace them by $y_1{:}c, \ldots, y_s{:}c$, where c is a new color.)

In general if α is the ith option and if $a_i = a + 1 > 1$, replace α by the $2a + 1$ options '$\alpha\ x_{1i}$', '$\#_{1i}\ x_{1i}$', '$\#_{1i}\ \alpha\ x_{2i}$', '$\#_{2i}\ x_{2i}$', '$\#_{2i}\ \alpha\ x_{3i}$', $\ldots$, '$\#_{ai}\ x_{ai}$', '$\#_{ai}\ \alpha$', where $\#_{ti}$ and x_{ti} are new primary and secondary items.

176. (a) Introduce $3N$ items $\{A_j, B_j, \#_j \mid 1 \le j \le N\}$, to be used in M options $\{A_j \mid a_{ij} \ge 1\} \cup \{B_j \mid a_{ij} = 2\}$ for $1 \le i \le M$. (For example, the option for row (2, 1, 0, 2, 0, $\ldots$) would be '$A_1\ B_1\ A_2\ A_4\ B_4$'.) Add $2N$ further options '$\#_j\ A_j$', '$\#_j B_j$' for $1 \le j \le N$. Use Algorithm M with multiplicities $(2, 1, 1)$ for $(A_j, B_j, \#_j)$.

(b) The same construction works, but with multiplicities $(3, 1, 1)$.

(c) Now use $4N$ primary items $\{A_j, B_j, \#_j, \#'_j\}$ and N secondary items x_j. Change the $2N$ special options to '$\#_j\ A_j$', '$\#_j\ B_j\ x_j$', '$\#'_j\ A_j\ x_j$', '$\#'_j\ B_j$', for $1 \le j \le N$. Use multiplicities $(4, 2, 1, 1)$.

(d) With $7N$ primary items $\{A_j, B_j, \#_{1j}, \ldots, \#_{5j}\}$ and $4N$ secondary items $\{x_{1j}, x_{2j}, x_{3j}, x_{4j}\}$, the special options are '$\#_{1j}\ A_j$', '$\#_{1j}\ B_j\ x_{1j}$', '$\#_{2j}\ A_j\ x_{1j}$', '$\#_{2j}\ B_j\ x_{2j}$', $\ldots$, '$\#_{5j}\ A_j\ x_{4j}$', '$\#_{5j}\ B_j$', and the multiplicities are $(11, 5, 1, 1, 1, 1, 1)$.

177. (a) The $2^s 3^t - 1$ nonzero vectors $a_1 \ldots a_s b_1 \ldots b_t$ with $0 \le a_i \le 1$ and $0 \le b_i \le 2$ form the rows of a matrix A. Allow the $2^t - 1$ rows with $a_i = 0$ and $b_i \ne 2$ to be repeated, via answer 175; also encode the 2's via answer 176. That leads to $s{+}3t{+}2^t{-}1$ primary items, $2^t - 1$ secondary items, and a total of $2^s 3^t - 1 + 2t + 2(2^t - 1)$ options. (There are 91914202 multipartitions when $s = t = 5$. Algorithm M generates them at a rate of about 1300 mems per solution; that's only about seven times slower than the special-purpose Algorithm 7.2.1.5M.)

(b) This problem is easier, because we simply disallow using an option twice. That leaves us with $s + 3t$ primary items and $2^s 3^t - 1 + 2t$ options.

(Exercise 7.2.1.5–73 enumerates the number of solutions $P(s, t)$ for part (a). The same argument gives a similar recurrence for the number $Q(s, t)$ of solutions to part (b):

$$Q(s, 0) = \varpi_s; \qquad 2Q(s, t+1) \;=\; Q(s+2, t) + Q(s+1, t) - \sum_k \binom{n}{k} Q(s, k).$$

With this formula one finds quickly, for example, that $Q(5, 5) = 75114998$.)

178. (a) Since $360 = 2^3 \cdot 3^2 \cdot 5$, we need first to extend exercise 176 to matrices of 0s, 1s, 2s, and 3s. Encoding $a_{ij} = 3$ in option i can be done by using items A_j, B_j, C_j. To ensure a total of 3 in that column, let $\#_j$ and $\#'_j$ be new primary items, and give multiplicity $(3, 1, 1, 1, 1)$ to $(A_j, B_j, C_j, \#_j, \#'_j)$; also let x_j be secondary. Then the special options '$\#_j\ A_j$', '$\#_j\ B_j\ x_j$', '$\#'_j\ A_j\ x_j$', '$\#'_j\ C_j$' will fix everything up.

This makes an MCC problem with 29 options, $9 + 1$ items, and 34 solutions.

(b) Now use exercise 175 to allow the options for factors 3 and 2×3 to be repeated at most twice, and to allow the option for factor 2 to be repeated at most thrice. The MCC problem now has 37 options, $13 + 5$ items, and 52 solutions. [These solutions were first studied by John Wallis; see exercise 7.2.1.7–28.]

179. From 1000 + 0110 + 0001 we get four solutions 100000 + {011100, 011100} + {000011, 000011}; from 1110 + 0001 we get two solutions 111100 + {000011, 000011}; and from 1010 + 0101 we get 101000 + 010111.

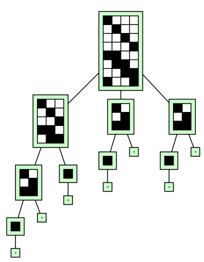
secondary items
color
varpi
0123-mtx
Wallis

restricted growth strings
differential equation
tail
equivalence relations
Moser
Guy

180. The text showed that $o_1 =$ 'i_1' and that i_2 and o_5 exist, when $t = 4$ and $t' \geq 1$. Continuing that example, if $s_2 = 5$ so that $t' \geq 2$, then option o_2 intersects only $\{o_1, \ldots, o_5\}$; hence $o_2 =$ '$i_1\ i_2$', and i_2 cannot occur in *more* than 4 options. Its appearances must therefore be in $\{o_2, o_3, o_4, o_5\}$.

Furthermore, o_3 must be '$i_1\ i_2\ i_3\ \ldots$' for some third item, i_3, since we can't have $o_3 = o_2$. Consequently there's an option $o_6 =$ '$i_2\ i_3\ \ldots$'. And so on.

181. $(c_0, c_1, c_2, c_3, c_4) = (188, 248, 320, 425, 566)/96$, by the initial values in the text.

182. 
(To establish the lower bound in Theorem E, make n copies of this problem, on disjoint four-tuples of items. This yields 7^n solutions, in a search tree with $(5\cdot 7^n - 3)/2$ nodes. Notice that the branching factor never exceeds 3 in this construction.)

183. (Can one, for example, often make the branching factor $t = 4$?)

184. Yes. If we can write $t = a_{n-1}\varpi_{n-1} + a_{n-2}\varpi_{n-2} + \cdots + a_0\varpi_0$, with $0 \leq a_j \leq \binom{n-1}{j}$ for $0 \leq j < n$, we get such a problem by letting the options consist of (i) all $2^{n-1} - 1$ subsets of $\{1, \ldots, n-1\}$; (ii) exactly a_j subsets of $\{1, \ldots, n\}$ of size $n-j$ that contain n.

To write t in that form, suppose $t = \binom{n-1}{n-1}\varpi_{n-1} + \cdots + \binom{n-1}{n-k+1}\varpi_{n-k+1} + a_{n-k}\varpi_{n-k} + t'$, where $0 \leq a_{n-k} < \binom{n-1}{n-k}$ and $0 \leq t' < \varpi_{n-k}$. Then, by induction, we can write $t' = a_{n-k-1}\varpi_{n-1-k} + \cdots + a_0\varpi_0$, with $0 \leq a_j \leq \binom{n-k-1}{j} \leq \binom{n-1}{j}$.

For example, $10000 = 1\cdot 4140 + 6\cdot 877 + (1\cdot 203 + 7\cdot 52 + 2\cdot 15 + (0\cdot 5 + (0\cdot 2 + (1\cdot 1))))$.

185. We get the most solutions when we have the most options, namely the $2^{N_1+N_2} - 2^{N_2}$ subsets that aren't entirely secondary. Then the solutions are the set partitions that include at most one entirely secondary block; and the number of such set partitions is seen to be $\sum_m \left\{ {N_1 \atop m} \right\}(m+1)^{N_2}$, when we consider their restricted growth strings.

186. (a) The list for i consists of all 2^{n-1} subsets that contain i. So there are $\binom{n-1}{k-1}$ operations hide(p) on options p of size k; and $u_n = 1 + \sum_k \binom{n-1}{k-1}(k-1) = (n-1)2^{n-2} + 1$.

(b) The lists get shorter, so the algorithm does $u_{n-1} + \cdots + u_{n-(k-1)}$ updates.

(c) Sum $u_n + \sum_k \binom{n-1}{k-1}(s_{n-1} - s_{n-k})$, where $s_n = \sum_{k=1}^{n} u_k = (n-2)2^{n-1} + n + 1$. For example, $(v_0, v_1, \ldots, v_5) = (0, 1, 3, 12, 57, 294)$; $(x_0, x_1, \ldots, x_5) = (0, 1, 4, 18, 90, 484)$.

187. (a) We have $X'(z) = \sum_n x_{n+1} z^n/n! = V'(z) + e^z X(z)$, where $V(z) = \sum_n v_n z^n/n!$. The given function solves this differential equation and has $X(0) = 0$.

(b) Similarly, we have $T'_{r,s}(z) = e^z T_{r,s}(z) + z^r$ and $T_{r,s}(0) = 0$.

(c) Integrate by parts.

(d) For example, $T_{1,3}(z) = 4e^{e^z-1} + 2T_{0,0}(z) - ze^{2z} - (2z+1)e^z - 2z - 3$, by (c).

188. By induction, $\widehat{\varpi}_{nk}$ is the number of n-element, single-tail set partitions (equivalence relations) for which $n > 1$ and $1 \not\equiv 2$, ..., $1 \not\equiv k$. (For example, if we know that 22 single-tail partitions of $\{1, 2, 3, 4, 5\}$ have $1 \not\equiv 2$, and that 6 such partitions of $\{1, 2, 3, 4\}$ have $1 \not\equiv 2$, then 6 single-tail partitions of $\{1, 2, 3, 4, 5\}$ must have $1 \not\equiv 2$ and $1 \equiv 3$; hence 16 of them have $1 \not\equiv 2$ and $1 \not\equiv 3$.) Therefore $\widehat{\varpi}_{nn} = \widehat{\varpi}_{n-1}$, for all $n \geq 1$.

[Leo Moser played with this triangular array in 1968 and found the generating function $\sum_n \widehat{\varpi}_n z^n/n! = e^{e^z}\int_0^z e^{-e^t}\,dt$; he showed his results to R. K. Guy, who told

N. J. A. Sloane; see OEIS sequences A046936 and A298804. If we start with '0, 0, 1' on the diagonal instead of '0, 1', we get Gould's $\langle a_{n2}\rangle = \langle 0,0,1,1,4,14,54,233,\ldots\rangle$; etc.]

189. (a) $|e^{e^z}| = |e^{x\cos y + ix\sin y}| = \exp(x\cos y)$; $|e^{-e^z}| = \exp(-x\cos y)$.

(b) $\bigl|\int_0^\theta \exp(-e^{\xi e^{i\phi}})\,d(\xi e^{i\phi}) + \int_\xi^\infty e^{-e^t}dt\bigr| = O(\xi\exp(-e^x\cos y)) + O(\exp(-e^\xi))$; $|e^{e^z}| = O(\exp(e^\xi))$; and we have $x = \xi\cos\theta \ge \xi - 2.25/\xi$, $\cos y \ge \cos\frac{3}{2}$.

(c) We have $\int_z^\infty e^{-e^t}dt = \int_0^\infty e^{-e^t}dt - \int_0^1 e^{-e^{uz}}d(uz) = \hat g/3 - I$. Let $\max|e^{e^{uz}}|$ for $0 \le u \le 1$ be $\exp(-e^{u_0x}\cos u_0y)$. If $\cos u_0y \ge 0$ we have $|I| = O(\xi)$. Otherwise if $\cos y - \cos u_0y \le 1$ we have $|e^{e^z}I| \le \xi\exp(e^x\cos y - e^{u_0x}\cos u_0y) \le \xi\exp(e^x\cos y - e^x\cos u_0y) \le \xi\exp(e^x)$. Otherwise we use a more delicate argument: Since $\cos(a-b) - \cos(a+b) = 2(\sin a)(\sin b)$, we have $|\sin\frac{u_0-1}{2}y| = \frac{1}{2}|(\cos y - \cos u_0y)/\sin\frac{u_0+1}{2}y| \ge \frac{1}{2}$, hence $u_0 \le 1 - \pi/(3y)$. And in this range, $u_0x \le x - \frac{\pi}{3}x/y = \xi\cos\theta - \frac{\pi}{3}\cot\theta \le \xi - c\xi^{1/3} + O(1)$, where $c^3 = \frac{3}{8}\pi^2$.

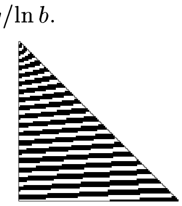
The desired bound now holds in each case because $x = \xi\sqrt{1-\sin^2\theta} \le \xi - 9/(8\xi)$.

(d) If $\frac{\pi}{2} \le \theta \le \pi$, $|e^{e^z}|\exp(-e^{u_0x}\cos u_0y) = O(1)$. Since $\rho_{n-1}/(n-1)! = \frac{1}{2\pi i}\oint R(z)\,dz/z^n$, and since $\varpi_{n-1}/(n-1)! = \Theta\bigl(e^{e^\xi}/(\xi^{n-1}\sqrt{\xi n})\bigr)$ by 7.2.1.5–(26), we have $|\rho_{n-1}/\varpi_{n-1}| = O\bigl(\sqrt{\xi n}\exp(-c_2e^\xi/\xi)\bigr)$ for all $c_2 < \frac{9}{8}$. And $-e^\xi/\xi = -n/\xi^2 < -n/\ln^2 n$.

[These results, and considerably more, were proved by W. Asakly, A. Blecher, C. Brennan, A. Knopfmacher, T. Mansour, and S. Wagner, *J. Math. Analysis and Applic.* **416** (2014), 672–682. In particular, they proved that a_{nk}/ϖ_n rapidly approaches the constant $\hat g_k = \int_0^\infty t^{k-1}e^{1-e^t}\,dt/k! = \int_0^\infty e^{-x}\ln^k(1+x)\,dx/k!$, for all $k > 0$.]

Historical notes: Leonhard Euler computed the constant $\hat g$ when he argued that this value can be assigned to the divergent series $\sum_{n=0}^\infty(-1)^n n!$ [*Novi Comment. Acad. Sci. Pet.* **5** (1754), 205–237]. Benjamin Gompertz, who did not know the constant $\hat g$ explicitly, studied the probability distributions $F(x) = 1 - a^{1-b^x}$ for $a, b > 0$ and $x \ge 0$ [*Philos. Trans.* **115** (1825), 513–585]. His name came to be associated with $\hat g$ because, for example, a random variable with $a = e$ in his distribution has $\mathrm{E}\,X = \hat g/\ln b$.

190. Empirically, these signs are essentially periodic, but with a slowly increasing period length as n grows. For example, the signs for $4000 \le n \le 4100$ are $+^2-^4+^4-^5+^4-^4+^5-^4+^4-^5+^4-^4+^4-^5+^4-^4+^5-^4+^4-^5+^4-^4+^4-^5$. The quantities $\widehat\varpi_{nk} - \hat g\varpi_{nk}$ for $1 \le k \le n \le 100$ have the interesting sign pattern shown at the right. (See exercise 188.) Complex variables are evidently interacting here somehow!



191. The mean is $G'(1) = 1 + \hat g$; the variance is $G''(1) + G'(1) - G'(1)^2 = 2\hat g_2 + \hat g - \hat g^2 \approx 0.773$. [Incidentally, $G(z)$ can also be written $e\Gamma(1+z) - \sum_{k=1}^\infty(-1)^k ez/((k+z)k!)$.]

192. Let $\xi e^\xi = n$ as in 7.2.1.5–(24). Then, when $x = e^\xi - 1 + t$ and t is small, we have $e^{-x}(\ln(1+x))^n \approx A\exp(-(1+\xi)t^2/(2n))$, where $A = \exp(n\ln\xi + 1 - e^\xi)$. Trading tails and integrating over $-\infty < t < \infty$ gives $\hat g_n \sim A\sqrt{2\pi n/(1+\xi)}/n!$.

193. At level 0, when given the complete graph K_{t+1}, the algorithm does $t+1$ updates when covering i in step X4, and t updates when covering each of t values of j in step X5. Thus $U(t+1) = 1 + t + t^2 + tU(t-1)$.

194. (a) In general we have $X(2q+1) = (2q)(2q-2)\ldots(2)\bigl(a_0 + a_2/2 + a_4/(2\cdot4) + \cdots + a_{2q}/(2\cdot4\cdot\ldots\cdot(2q))\bigr) = 2^q q!S - R$, where $S = \sum_{n\ge0} a_{2n}/(2^n n!)$ and $R = a_{2q+2}/(2q+2) + a_{2q+4}/((2q+2)\cdot(2q+4)) + \cdots$. Hence when $a_t = 1$ we have $S = e^{1/2}$ and $0 < R < 1$. [This result was noticed in 1999 by Michael Somos; see OEIS A010844.]

(b) In general, $X(2q) = \bigl((2q)!/(2^q q!)\bigr)S - R$, where $S = X(0) + a_1 + a_3/3 + a_5/(3\cdot5) + a_7/(3\cdot5\cdot7) + \cdots$ and $R = a_{2q+1}/(2q+1) + a_{2q+3}/((2q+1)\cdot(2q+3)) + \cdots$.

Sloane
OEIS
Gould
Asakly
Blecher
Brennan
Knopfmacher
Mansour
Wagner
Euler
divergent series
Gompertz
Gamma function
Trading tails
Somos
OEIS

Perfect matchings
$K_{n,n}$
Dworkin
quasi-independent
MRV heuristic

When $a_t = X(0) = 1$, $S - 1 = 1 + 1/3 + 1/(3\cdot 5) + \cdots = e^{1/2}\operatorname{erf}\bigl(\sqrt{1/2}\bigr)/\bigl((\frac{1}{2})^{1/2}/(\frac{1}{2})!\bigr)$, and $0 < R < 1$. So the answer is $\bigl\lfloor (1 + \sqrt{e\pi/2}\operatorname{erf}(\sqrt{1/2}))(2q)!/(2^q q!)\bigr\rfloor$.

(c) $2^q q! C - 2q + O(1)$, where $C = \sum_{n\ge 0}(1 + 2n + 4n^2)/(2^n n!) = 5e^{1/2} \approx 8.24361$.

(d) $((2q)!/(2^q q!))C' - 2q + O(1)$, where $C' = 3 + 5\sqrt{e\pi/2}\operatorname{erf}\bigl(\sqrt{1/2}\bigr) \approx 10.05343$.

195. Assume that $q, r > 1$, and let v be the unique vertex of degree 2. The algorithm will try to match v with the vertex at its left; that leaves a problem of matching the independent graphs K_{2q} and K_{2r}. If $q \le r$, each matching of K_{2q} will initiate a computation of the matchings of K_{2r}; otherwise each matching of K_{2r} will initiate the matchings of K_{2q}. So the running time of this phase will be C' updates per solution, where C' is the constant of answer 194(d) and there are $(2q)!(2r)!/(2^q q!\,2^r r!)$ solutions.

The algorithm will also try to match v with the vertex at its right. That leaves a problem of independently matching K_{2q+1} and K_{2r-1}, and there are no solutions. The running time of this phase will be C times $\min\bigl(2^q q!, 2^{r-1}(r-1)!\bigr)$, where C is the constant of answer 194(c). (Curiously, it's actually negligible compared to the other phase.)

196. (a) $b_1 \ldots b_9 = 135778899$. (Draw the bipartite graph, and rotate it 180°.)

(b) Let $\bar{k} = n + 1 - k$ for $1 \le k \le n$. Then '$X_j\ Y_k$' is a dual option if and only if '$Y_{\bar{j}}\ X_{\bar{k}}$' is an original option; $q_1 \ldots q_n$ is the inverse of an original solution if and only if $\bar{q}_n \ldots \bar{q}_1$ is a dual solution.

(c) $1 + a_1(n+1)$, because each Y_k for $1 \le k \le a_1$ appears in n options.

(d) $a_1(a_2-1)(a_3-2)\ldots(a_n-n+1)$. [This number must therefore be equal to $b_1(b_2-1)(b_3-2)\ldots(b_n-n+1)$ — and that's *not* an obvious fact!]

(e) Let $\Pi_j = \prod_{i=1}^{j}(a_i - i + 1)$. From (c), the answer is $1 + \bigl(\sum_{j=1}^{n}(n+3-j)\Pi_j\bigr) - \Pi_n$.

(f) $1 + \bigl(\sum_{j=1}^{n}(n+3-j)n^{\underline{j}}\bigr) - n! \approx (4e-1)n!$. [Perfect matchings of $K_{n,n}$.]

(g) $6\cdot 2^n - 2n - 7$, because $\Pi_j = 2^j$ for $1 \le j < n$, and $\Pi_n = 2^{n-1}$.

(h) Now $\Pi_n = \lfloor\frac{n+1}{2}\rfloor^{\underline{}}\lfloor\frac{n+2}{2}\rfloor^{\underline{}}$; and the total number of updates, divided by Π_n, is therefore $6 + 4/1! + 5/2! + \cdots + O(n^2/(n/2)!) \approx 4e - 1$.

(i) If $b_1 < a_1$, the first branch is on Y_n, not X_1; and $1 + b_1(n+1)$ updates are made at root level. (The example problem in (a) branches on Y_9, then X_2, then Y_8, etc.)

197. (a, b). Induction; σ_{st} can in fact be any permutation that takes $s \mapsto t$ and doesn't increase any other element.

(c) $C(a_1, \ldots, a_n) = \prod_{j=1}^{n}(z + a_j - j)$, by (a), since we gain a cycle in that product representation if and only if $t_j = j$. $I(a_1, \ldots, a_n) = \prod_{j=1}^{n}(1 + z + \cdots + z^{a_j - j})$, by (b). [See exercise 7.2.1.5–29; also M. Dworkin, *J. Combinatorial Theory* **B71** (1997), 17–53.]

198. (a) If $s > a_r$ we have $\pi_{rs} = 0$. Otherwise let q be the smallest j with $a_j \le s$; then $q \le r$. Each permutation of $P(a_1, \ldots, a_n)$ with $p_r = s$ corresponds to one of $P(a'_1, \ldots, a'_{r-1}, a'_{r+1}, \ldots, a'_n)$, where $a'_j = a_j - [j \ge q]$. Thus $(a_r + 1 - r)\pi_{rs} = \prod_{j=q}^{r-1}(a_j - j)/(a_j + 1 - j)$.

(b) We have $q' \ge q$ when $s' > s$. Consequently $\pi_{rs}/\pi_{rs'} = \prod_{j=q}^{q'-1}(a_j - j)/(a_j + 1 - j)$ for all $r \ge q'$, if $\pi_{rs'} > 0$. [In such cases the parameters r and s are said to be "quasi-independent."]

$$\begin{pmatrix}
\frac12 & \frac12 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
\frac18 & \frac18 & \frac14 & \frac14 & \frac14 & 0 & 0 & 0 & 0\\
\frac18 & \frac18 & \frac14 & \frac14 & \frac14 & 0 & 0 & 0 & 0\\
\frac1{12} & \frac1{12} & \frac16 & \frac16 & \frac16 & \frac13 & 0 & 0 & 0\\
\frac1{12} & \frac1{12} & \frac16 & \frac16 & \frac16 & \frac13 & 0 & 0 & 0\\
\frac1{24} & \frac1{24} & \frac1{12} & \frac1{12} & \frac1{12} & \frac16 & \frac12 & 0 & 0\\
\frac1{72} & \frac1{72} & \frac1{36} & \frac1{36} & \frac1{36} & \frac1{18} & \frac16 & \frac13 & \frac13\\
\frac1{72} & \frac1{72} & \frac1{36} & \frac1{36} & \frac1{36} & \frac1{18} & \frac16 & \frac13 & \frac13\\
\frac1{72} & \frac1{72} & \frac1{36} & \frac1{36} & \frac1{36} & \frac1{18} & \frac16 & \frac13 & \frac13
\end{pmatrix}$$

199. Assume by symmetry that $m \le \lceil n/2 \rceil$. With the MRV heuristic it's not difficult to see that every branch at level l for $l < m$ is on some a_i for $i \le m$, with exactly $(n-l)(m-1-l)$ descendants. Hence there are $n^{\underline{l}}(m-1)^{\underline{l}}$ nodes on level l. The total number of nodes when $m \approx n/2$ is huge, $\Theta\bigl((n-2)!\bigr)$; and there are no solutions.

200. (a) When all n^3 options are present, $\det Q(X) = \sum \mathrm{sign}(p) v_{1p_1q_1} \ldots v_{np_nq_n}$, summed over all permutations $p = p_1 \ldots p_n$ and all n-tuples $q = q_1 \ldots q_n$ with $q_j \notin X$. Summing $(-1)^{|X|} \det Q(X)$ yields $\sum \mathrm{sign}(p) v_{1p_1q_1} \ldots v_{np_nq_n}$ where both p and q are permutations. (This is essentially an application of the inclusion-exclusion principle.) Set $v_{ijk} \leftarrow 0$ if option 'a_i b_j c_k' isn't present.

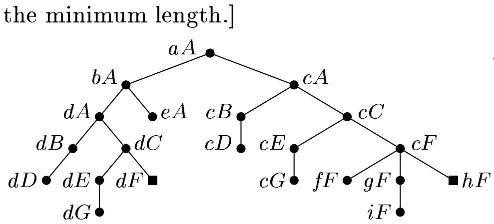
(b) Assign a random integer in $[0\,..\,p)$ to each of the M given options, where p is a prime greater than $2M$, and evaluate $s = S \bmod p$. If $s \neq 0$, S is nonzero. If $s = 0$, S is nonzero with probability less than $1 - (1-1/p)^M < M/p < 1/2$, by exercise 4.6.1–16, because S is linear in each variable. Repeating r times will fail with probability $< 2^{-r}$.

[In practice, 2^n is often an overestimate because many of the determinants are obviously zero. For example, if $Q(X)$ has an all-zero row or column, so does $Q(X')$ for all $X' \supseteq X$. This method shines on unsolvable examples such as those of exercise 199. Björklund's paper, *STACS* **27** (2010), 95–106, has more general results.]

201. (a) "Given n people seated at a circular table, how many seating arrangements do not require anybody to move more than one place left or right?"

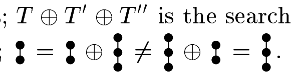
(b) Two solutions in which everybody moves, plus L_n solutions (a Lucas number) in which at least one person remains in the same seat.

(c) An interesting recursive structure leads to the answer $5L_{n+2} + 10n - 33$. [This analysis depends on using the given ordering to break ties in step X3 when several lists have the minimum length.]

202. 

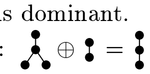
.

203. (a) Yes; $T \oplus T' \oplus T''$ is the search tree corresponding to $A \oplus A' \oplus A''$.

(b) No; .

204. By definition of $T \oplus T'$, we have $\mathrm{subtree}(\alpha\alpha') = \mathrm{subtree}(\alpha) \oplus \mathrm{subtree}(\alpha')$. Hence $\deg(\alpha\alpha') = \min(\deg(\alpha), \deg(\alpha'))$.

Let $\mathrm{ancestors}(\alpha) = \{\alpha_0, \ldots, \alpha_l\}$ and $\mathrm{ancestors}(\alpha') = \{\alpha'_0, \ldots, \alpha'_{l'}\}$. Suppose $\alpha\alpha'$ is dominant in $T \oplus T'$ and $\deg(\alpha\alpha') = d$. If $0 \le k < l$, some ancestor $\alpha_k\alpha'_{k'}$ of $\alpha\alpha'$ has $\deg(\alpha_k) = \deg(\alpha_k\alpha'_{k'}) < d$; hence α is dominant. Similarly, α' is dominant.

We've proved the "only if" part, but the converse is false: .

205. The first statement follows easily from the definition (see exercise 202). Suppose $\alpha\alpha' = \alpha_l\alpha'_{l'} \in T \oplus T'$, as in answer 204, where neither α nor α' is dominant, and where $l + l'$ is minimum. Then $l > 0$ and $l' > 0$, because α_0 and α'_0 are dominant.

Assume that the parent of $\alpha\alpha'$ is $\alpha\alpha'_{l'-1}$. Then $\alpha'_{l'-1}$ is dominant, and α_l isn't. So there's a $k < l$ such that $\deg(\alpha_k) = \max(\deg(\alpha_0), \ldots, \deg(\alpha_l))$. Hence there's a maximum $k' < l'$ such that $\alpha_k\alpha'_{k'}$ is an ancestor of $\alpha\alpha'$. Then $\deg(\alpha'_{k'}) \le \deg(\alpha'_{l'-1}) < \deg(\alpha)$, and $\alpha_k\alpha'_{k'+1}$ is also an ancestor. But $\alpha_k\alpha'_{l'}$ isn't. Contradiction.

A similar contradiction arises when the parent of $\alpha\alpha'$ is $\alpha_{l-1}\alpha'$.

206. Replace each solution node of T by a copy of T'.

207. (a) If $\lambda_j = 4$ we now prefer the 5-way branch on i, because $\lambda'_i = 7/2 < 11/3 = \lambda'_j$. If $\lambda_j = 3$ we prefer $\min(i,j)$, because $\lambda'_i = 3 = \lambda'_j$. If $\lambda_j = 2$ we still prefer the binary branch on j to the ternary branch on i. And if $\lambda_j = 1$ or 0 we certainly prefer j.

inclusion-exclusion principle
prime
Björklund
circular table
Lucas number

stamp (time stamp)
short float
multiplicity
nonsharp preference
bitriples
biquadruples

(b) Include two new fields, `ACT` and `STAMP`, initially zero, in each item node. (They can share an octabyte, if `ACT` is a short float and `STAMP` is a tetrabyte.) A global variable `TIME` serves as the "convenient clock." Another global, `BUMP` (which is a short float, initially 10^{-32}), is the amount by which we advance activity scores. Whenever i is covered or uncovered, or whenever $\mathtt{LEN}(i)$ is changed, we check to see if $\mathtt{STAMP}(i) = \mathtt{TIME}$; if not, we set $\mathtt{ACT}(i) \leftarrow \mathtt{ACT}(i) + \mathtt{BUMP}$ and $\mathtt{STAMP}(i) \leftarrow \mathtt{TIME}$.

The "clock" advances at the beginning of steps X4, X5, X6, and X7. This means that $\mathtt{TIME} \leftarrow (\mathtt{TIME}+1) \bmod 2^{32}$ and $\mathtt{BUMP} \leftarrow \mathtt{BUMP}/\rho$. (Furthermore, if $\mathtt{BUMP} \ge 10^{29}$, we divide `BUMP` and *all* `ACT` fields by 10^{64}, to avoid overflow. We limit ρ to be at most .999, so that each α_i is at most 1000.)

These changes allow us to replace the definition of λ in step X3 (answer 9) by $\lambda \leftarrow \bigl(\mathtt{LEN}(p) \le 1?\ \mathtt{LEN}(p)\colon 1 + \mathtt{LEN}(p)/(1 + \mu\,\mathtt{ACT}(p)/\mathtt{BUMP})\bigr)$.

(c) Consider (90) first. After branching on 00 and trying option '00 01', we have $\alpha_{00} = \alpha_{02} = \rho$, $\alpha_{01} = 1 + \rho$, $\alpha_{04} = \alpha_{05} = \alpha_{06} = 1$, and the other α's are zero. We want $\lambda'_{05} = 1 + 3/(1 + \mu\alpha_{05})$ to be less than $\lambda'_{10} = 1 + 2$; that is, $\mu > 1/2$. Later, after trying option '00 02', we'll have $\alpha_{05} > 1$ and $\alpha_{06} > 1$; again, item 01 isn't chosen.

Problem (92) is trickier. After trying '00 01', the nonzero α's are $\alpha_{00} = \alpha_{02} = \rho$, $\alpha_{01} = 1 + \rho$, and $\alpha_{03} = \alpha_{04} = \alpha_{05} = 1$. We'll prefer the 3-way branch on 02 to the 2-way branch on 20 if $\mu > 1/(2\rho)$; and we'll even prefer the 4-way branch on 04 (or 05) to that 2-way branch, if $\mu > 1$. In either case we'll reach a solution to problem 0 before starting on problem 1. The same calculations then take us to problem 2 only when problem 1 has been solved; etc. (Furthermore, when coming back down there will be no incentive to go back up. In fact, 4-way branches will be done on the items $k3$ because of their high activity scores.)

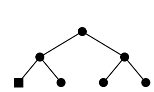
(d) The normal Algorithm X finds all 212 solutions in 96 Gμ, with a 55-meganode search tree. This modification finds them in 51 Gμ, if we set $\mu = 1/8$ and $\rho = .99$, with a 26-meganode search tree. (With $\mu = 1/2$ and $\rho = .9$, the time is 62 Gμ. In long runs, the α scores tend to approach $1/(1-\rho)$; so increases in ρ usually imply decreases in μ.)

208. The original problem has primary items ij for $\mathtt{0} \le i, j \le \mathtt{e}$, and eight kinds of options '$\{ij + \delta \mid \delta \in S_k\}$' for all cells $ij + \delta$ that are in range, where $S_0 = \{01, 11, 21, 31, 10\}$, $S_1 = \{00, 01, 02, 03, 11\}$, $S_2 = \{00, 10, 20, 30, 21\}$, $S_3 = \{10, 11, 12, 13, 02\}$, $S_4 = \{01, 11, 21, 31, 20\}$, $S_5 = \{00, 01, 02, 03, 12\}$, $S_6 = \{00, 10, 20, 30, 11\}$, $S_7 = \{10, 11, 12, 13, 01\}$. Options that involve the center cell 77 come only from S_0.

The modified problem adds secondary items V_{ij} and H_{ji}, for $\mathtt{0} \le i \le \mathtt{b}$, $\mathtt{1} \le j \le \mathtt{d}$. It inserts $V_{i(j-1)}$, $H_{(i+1)j}$, $V_{i(j+1)}$, respectively into the options with S_4, S_5, S_6, S_7.

(The 16 solutions to this problem represent $2^2 + 2^4 + 2^5 + 2^2 + 2^3 + 2^2 + 2^5 + 2^3 + 2^5 + 2^3 + 2^2 + 2^4 + 2^3 + 2^4 + 2^2 + 2^4 = 212$ solutions to the original. We're lucky that none of those solutions has an 'H' that includes 77.)

209. With the modified options '0 1 A', '0 2 B', '1 4 5 B', '2 3 4 A', obtained from the bipairs ('0 1', '2 3 4'; '0 2', '1 3 4') and ('0 1', '2 4 5'; '0 2', '1 4 5'), we get the balanced search tree shown here.



210. Add a new primary item #A and give it multiplicity $[0\,..\,2]$. Insert it into options α', β', γ'. Then use the nonsharp preference variant of Algorithm M.

211. No bipairs. (But Langford has bitriples, and all three have "biquadruples.")

212. (a) Order the options first by their smallest item, and secondly by lexicographic order among those with the same smallest item.

(b) Yes. For example, we can let $1 < 2$, and $1 < 4 < 0 < 5$.

prefix of a string
reachability problem
acyclic digraph
sinks
2DM
4-cycles
author

213. Yes, *provided* that we regard a proper prefix of a string as lexicographically *larger* than that string (contrary to the conventions of a dictionary). Otherwise the condition fails when α is a prefix of α' (although exercise 212 remains valid).

Suppose the items of α and β are respectively represented by the digits j and k in $\mathrm{rgs}(\pi)$, the restricted growth string of π. Then j will also represent α' in $\mathrm{rgs}(\pi')$, and both strings will be equal up to the point where j first appears.

Let β' be represented by k' in $\mathrm{rgs}(\pi')$; then $k' > j$. Consider the leftmost place where $\mathrm{rgs}(\pi)$ differs from $\mathrm{rgs}(\pi')$. If that digit is j in $\mathrm{rgs}(\pi)$, it is k' in $\mathrm{rgs}(\pi')$. Otherwise it is k in $\mathrm{rgs}(\pi)$; but then it is j in $\mathrm{rgs}(\pi')$, and α is a prefix of α'.

214. We can find all solutions Σ that reduce to a given strong solution Σ_0, by repeatedly reversing the construction in the proof of Theorem S — replacing joint occurrences of α and β by joint occurrences of α' and β', for all canonical bipairs, in all possible ways. (It's a reachability problem: to find all nodes of an acyclic digraph, given the sinks.)

Notice that different strong solutions can lead to the same nonstrong solution. For example, in the 2DM problem with options $\{\mathtt{xX}, \mathtt{xY}, \mathtt{yX}, \mathtt{yY}, \mathtt{yZ}, \mathtt{zY}, \mathtt{zZ}\}$, where uv stands for 'u v', we might have the canonical bipairs $(\mathtt{yX}, \mathtt{xY}; \mathtt{yY}, \mathtt{xX})$, $(\mathtt{yZ}, \mathtt{zY}; \mathtt{yY}, \mathtt{zZ})$. The strong solutions $\{\mathtt{xY}, \mathtt{yX}, \mathtt{zZ}\}$ and $\{\mathtt{xX}, \mathtt{yZ}, \mathtt{zY}\}$ both lead to the nonstrong $\{\mathtt{xX}, \mathtt{yY}, \mathtt{zZ}\}$. (However, in that same problem, we could have made the bipairs $(\mathtt{yX}, \mathtt{xY}; \mathtt{yY}, \mathtt{xX})$, $(\mathtt{yY}, \mathtt{zZ}; \mathtt{yZ}, \mathtt{zY})$ canonical. Then there would have been only one strong solution.)

215. (a) This is the number of 4-cycles, of which there are $3\binom{2q+1}{4}$: Four vertices $i < j < k < l$ can form three 4-cycles, with either j or k or l opposite i.

(b) For convenience, denote options by ij instead of 'i j'. If $i < j < k < l$, we exclude (i, j, k, l) unless $\min(ij, ik, il, jk, jl, kl) = ij$ or kl. We exclude (i, k, j, l) unless $\min(ij, ik, il, jk, jl, kl) = ik$ or jl. We exclude (i, l, j, k) unless $\min(ij, ik, il, jk, jl, kl) = il$ or jk. Hence exactly two of the three possibilities are excluded.

(c) When $i < j < k < l$ they are (i, k, j, l) and (i, l, j, k).

(d) The root has $2q$ children, branching on 0. All of them are leaves except for the branch '0 1'. That one has $2q - 2$ children, all of which are leaves except for the branch '2 3'. And so on, with $2(q - l)$ nodes on level $l > 0$.

(e) For $i < j$, use only (i, j, k, l) for $i < k < \min(j, l)$ and (k, l, i, j) for $k < i < l$.

(f) Put '1 $2q$' first, then '2 $2q{-}1$', ..., then 'q $q{+}1$', then the others. When we branch on '0 k' at the root, for $1 \le k \le 2q$, no options remain for item $2q + 1 - k$.

(g) '0 k' and '$2q{+}1{-}k$ l' are excluded, for all $l \notin \{0, k, 2q{+}1 - k\}$. (Altogether $(q - 1)(q - 3)$ cases.) [Is it perhaps feasible to order the options *dynamically*?]

216. The search tree is almost always smaller than that of answer 215(c), which in fact has the worst case on every level. But it rarely seems to go below half of the worst-case size. (The author discovered the trick of answer 215(f) by studying randomly generated examples that had unusually small trees.)

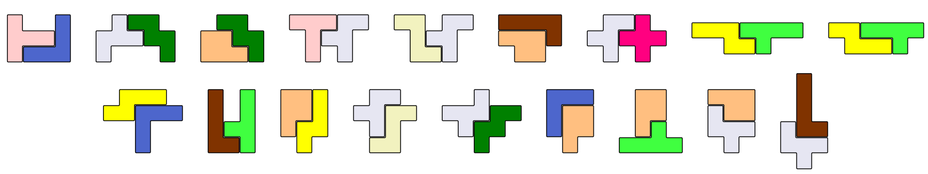
Algorithm X needs 540 Gμ to prove that K_{21} has no perfect matching. It has potentially $2\binom{21}{4} = 11970$ excludable quadruples. We can use Algorithm 3.4.2S to sample just m of them; then the running time for $m =$ (2000, 4000, 6000, 8000, and 10000) decreases to about (40 Gμ, 1.6 Gμ, 145 Mμ, 31 Mμ, 12 Mμ), respectively.

217. Each delta $\alpha - \alpha'$ has k positive terms and k negative terms; we can assume that $1 \le k \le 4$. Furthermore it suffices to work with "normalized" deltas, which are lexicographically smallest under rotation, reflection, and negation. The pentominoes (O, P, ..., Z) have (10, 64, 81, 73, 78, 25, 23, 24, 22, 3, 78, 24) normalized deltas, of which (1, 7, 3, 3, 2, 0, 1, 0, 1, 0, 4, 0) have $k = 1$. Two of the deltas are shared by four different pentominoes: $00{+}01{-}23{-}33$ (Q, S, W, Z); $00{-}02$ (P, Q, R, Y). Eleven are shared by three.

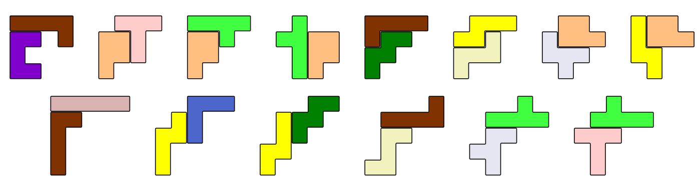
A common delta is necessary but not sufficient; if $\alpha - \alpha' = \beta' - \beta$, we still need to fill in cancelled terms that don't clash. For example, $00 - 23$ is common to Q and W, but it doesn't yield a bipair. Furthermore (although the exercise didn't state this!), we don't want the 10-cell region to have a hole; the delta $00+01-12-22$ is common to P, U, and Y, but only PY makes a useful bipair. A delta can arise in more than one way: From $00+01+02+03-20-21-22-23$ we can make a Q with either 10 or 13, and a Y with either 11 or 12; symmetry (and hole removal) yields only one bipair, not four.

Miller
naked pairs
sudoku
spacers
commit'

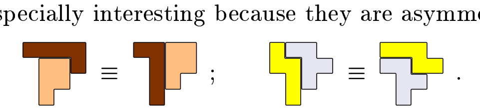
The complete catalog has 34 essentially distinct entries. Eighteen of them



have 10-cell shapes with left-right symmetry. Fourteen have transposition symmetry:



The other two are especially interesting because they are asymmetric:



(These two each lead to eight varieties when rotated and reflected, not just four. See J. C. P. Miller in Eureka: *The Archimedeans' Journal* **23** (1960), 14–15.)

218. If the only options involving p are '$p\ i{:}0$' and '$p\ i{:}1$', we can't eliminate item i. [But if they all involve, say, $i{:}0$, we *could* eliminate it; Algorithm P doesn't go that far.]

219. If option o contains i, but neither p nor q, it can be in a solution only with two other options $\{o', o''\}$ that contain $\{p, q\}$. But o' and o'' must then both contain j. [This argument is like the "naked pairs" of sudoku lore. It's tempting to go further, by also eliminating items i and j; but that could increase the number of solutions.]

220. Let the option be '$i_1\ i_2[{:}c_2]\ \ldots\ i_t[{:}c_t]$'. We've already covered item $i = i_1$, which is represented by node x. Nodes $x+1$, $x+2$, ... represent the other items, possibly with spacers that were inserted when this option was shortened (see exercise 222). We want to commit $i_2, \ldots, i_t$, and to determine whether this causes `LEN(`p`)` to become 0 for some primary $p \notin \{i_2, \ldots, i_t\}$. The tricky part is to be sure that $p \notin \{i_2, \ldots, i_t\}$; to accomplish this, we set `COLOR(`i_j`)` $\leftarrow x$ for $1 < j \le t$. [In detail: Set $p \leftarrow x+1$; while $p > x$, set $j \leftarrow$ `TOP(`p`)`, and if $j \le 0$ set $p \leftarrow$ `ULINK(`p`)`, otherwise set `COLOR(`j`)` $\leftarrow x$, $p \leftarrow p+1$.]

Then we make a second pass over the option: Set $p \leftarrow x+1$. While $p > x$, set $j \leftarrow$ `TOP(`p`)`, and if $j \le 0$ set $p \leftarrow$ `ULINK(`p`)`, otherwise commit$'(p, j)$ and set $p \leftarrow p+1$. Here commit$'(p, j)$ emulates (54): Set $c \leftarrow$ `COLOR(`p`)`, $q \leftarrow$ `DLINK(`j`)`; while $q \ne j$, hide$'''(q)$ unless `COLOR(`q`)` $= c > 0$, and set $q \leftarrow$ `DLINK(`q`)`. And hide$'''(p)$ is just like hide(p), but it detects blocking if `LEN(`y`)` becomes 0 for some $y \le N_1$ with `COLOR(`y`)` $\ne x$.

Finally, a third pass undoes our changes: Set $p \leftarrow x-1$. While $p \ne x$, set $j \leftarrow$ `TOP(`p`)`, and if $j \le 0$ set $p \leftarrow$ `DLINK(`p`)`, otherwise uncommit$'(p, j)$ and set $p \leftarrow p-1$. Here uncommit$'(p, j)$ undoes commit$'(p, j)$ in the obvious way.

It is possible to switch immediately from committing to uncommitting as soon as blocking is detected, by jumping into the middle of a loop (see answer 122).

jumping into the middle of a loop
spacer
author

221. While $S > 0$, set $x \leftarrow S$, $S \leftarrow \mathtt{TOP}(x)$, $\mathtt{TOP}(x) \leftarrow i$, and do the following: Set $q \leftarrow x$; while $q \geq x$, set $j \leftarrow \mathtt{TOP}(q)$, and if $j \leq 0$ set $q \leftarrow \mathtt{ULINK}(q)$; otherwise if $j \leq N_1$ and $\mathtt{LEN}(j) = 1$, go to P9; otherwise set $u \leftarrow \mathtt{ULINK}(q)$, $d \leftarrow \mathtt{DLINK}(q)$, $\mathtt{ULINK}(d) \leftarrow u$, $\mathtt{DLINK}(u) \leftarrow d$, $\mathtt{LEN}(j) \leftarrow \mathtt{LEN}(j) - 1$, $q \leftarrow q + 1$.

222. Set $p \leftarrow \mathtt{DLINK}(i)$, and do the following steps while $p \neq i$: Set $p' \leftarrow \mathtt{DLINK}(p)$, $q \leftarrow p + 1$. While $q \neq p$, set $j \leftarrow \mathtt{TOP}(q)$, and if $j \leq 0$ set $q \leftarrow \mathtt{ULINK}(q)$; otherwise if $j = S$, exit this loop; otherwise set $q \leftarrow q + 1$. Then if $q \neq p$, set $\mathtt{ULINK}(p) \leftarrow p + 1$, $\mathtt{DLINK}(p) \leftarrow p - 1$, $\mathtt{TOP}(p) \leftarrow 0$ (thereby making a spacer); otherwise set $q \leftarrow p + 1$ and perform the loop in answer 221 while $q \neq p$ (instead of while $q \geq x$). Finally set $p \leftarrow p'$.

223. In accordance with the conventions of exercise 8, we first declare the items of the reduced problem: For $1 \leq i \leq N$, output the distinguishing mark for secondary items, if $i = N_1 + 1$; and output the name of item i, if $\mathtt{LEN}(i) > 0$ or $i = N = 1$. Then we output the remaining options: For $1 \leq i \leq N$, if $\mathtt{LEN}(i) > 0$, set $p \leftarrow \mathtt{DLINK}(i)$ and do the following while $p \neq i$: Set $q \leftarrow p - 1$ and while $\mathtt{DLINK}(q) = q - 1$ set $q \leftarrow q - 1$. If $\mathtt{TOP}(q) \leq 0$ (hence i was the leftmost item to survive, in the option following the spacer node q), output the option as explained below. Then set $p \leftarrow \mathtt{DLINK}(p)$ and repeat.

To output the (possibly shortened) option that follows node q, set $q \leftarrow q + 1$; then, while $\mathtt{TOP}(q) \geq 0$, output the name of item $\mathtt{TOP}(q)$ if $\mathtt{TOP}(q) > 0$, followed by $\mathtt{:}c$ if $\mathtt{COLOR}(q) = c > 0$, and set $q \leftarrow q + 1$. (Afterwards, $-\mathtt{TOP}(q)$ is the number of the corresponding option in the original input.)

224. Use $3n-3$ items $p_1, x_1, i_1, \ldots, p_{n-1}, x_{n-1}, i_{n-1}$ (in that order), with the options '$i_{n-k}\ p_k\ x_k$', '$i_{n-k}\ p_k\ x_{k+1}$', '$i_{n-k}\ x_k$', '$i_{n-k}\ p_{k+1}$', for $1 \leq k < n - 1$, and also '$i_1\ p_{n-1}\ x_{n-1}$', '$i_1\ x_{n-1}$'. During round k, for $1 \leq k < n$, item i_{n-k} is forced by p_k.

225. Some options, like 'Z 01 02 11 20 21' and 'U 30 31 41 50 51', are obviously useless because they cut off a region of fewer than five cells. More of these options are discarded in the larger problem — but only because of piece U. Eight options, like 'O 10 11 12 13 14', are useless because they block a corner cell.

The smaller problem also has numerous options like 'P 02 12 13 22 23', which turn out to be useless because they block piece X. (That piece has been confined to just eight placements, in order to break symmetry. It has more freedom in the larger problem, and can't be blocked there.) Round 2 also discovers that options like 'O 22 23 24 25 26' would block X, since round 1 has disabled one of X's eight choices.

226. Since $\Sigma_1' = \sum_{k=1}^{2n}(2n+1-k)a_k$, it's clear that $\Sigma_1 + \Sigma_1' = (2n+1)\sum_{k=1}^{2n} a_k = (2n+1)(n+1)n$. Similarly $S + S' = (2n+1)\sum_{k=1}^{2n} a_k^2 = (2n+1)^2(n+1)n/3$.

The relation $\Sigma_2' - (2n+1)\Sigma_1' = \Sigma_2 - (2n+1)\Sigma_1$ holds for *any* sequence $a_1 \ldots a_{2n}$.

227. (a) $\$(ij^2 + ik^2)$. (b) $\$(i^2j + i^2k)$. [$\$(C - ij^2 - ik^2)$, for large C, will *maximize* Σ_2.]

228. Well, it certainly surprised the author. Intuitively, we expect small $\Sigma_1 = \sum k a_k$ to be correlated with small $\Sigma_2 = \sum k^2 a_k$, but not nearly so well. For some mysterious reason, Langford pairings with the same Σ_1 tend to have the same Σ_2, and vice versa!

That's not always true. For example, 2 8 6 2 3 5 7 4 3 6 8 5 4 1 7 1 and 3 5 7 4 3 8 6 5 4 1 7 1 2 6 8 2 have the same Σ_1 but different Σ_2; 1 5 1 7 4 8 9 5 11 4 10 7 6 3 8 2 9 3 2 6 11 10 and 1 4 1 6 7 10 4 5 9 11 6 8 7 5 2 3 10 2 9 3 8 11 have the same Σ_2 but different Σ_1. Yet such exceptions are rare. When $n = 7$, the four pairings that have $\Sigma_1 = 444$ are the same as the four that have $\Sigma_2 = 4440$; the six pairings that have the larger value $\Sigma_1 = 448$ are the same as the six that have $\Sigma_2 = 4424$, which is *smaller* than 4440. What is going on?

minimax
author
rounding errors
truncation errors
rooks

The special nature of Langford pairings does allow us to prove certain curious facts. For example, let j_k be the index of the first occurrence of k. The other occurrence is at $j_k + k + 1$; hence $\sum_{k=1}^{n} j_k = (3n-1)n/4$. Also $\sum_{k=1}^{n} j_k^2 = (4n^2-1)n/3 - \frac{1}{2}\Sigma_1$.

229. These pairings can be found by Algorithm 7.2.2L (or its reverse-order variant). But we can also find them via dancing links, using the sharp minimax modification of Algorithm X (or C) in exercise 85: Order options (16) so that '$i\ s_j\ s_k$' precedes '$i'\ s_{j'}\ s_{k'}$' when $j' < j$, or when $j' = j$ and $i' < i$ (for lex max) or $i < i'$ (for lex min). Then repeatedly (i) use the minimax algorithm to fill the smallest undetermined slot s_j; (ii) move the option that minimally covered s_j to the front of the list, and remove all other options that involve s_j.

Thus we find 1 2 1 3 2 4 8 3 12 13 4 10 14 15 16 8 9 6 11 5 7 12 10 13 6 5 9 14 7 15 11 16 in sixteen such steps, all of which are easy (and need less than 110 Kμ) except for the placements of 8 in s_7 (4.5 Mμ) and 12 in s_9 (500 Kμ). The total time (6 Mμ) includes 465 Kμ just for inputting the data in step X1. After placing 8 items, only 12 solutions remain, so it's slightly faster to switch gears when finishing. (This pairing has $\Sigma_1 =$ \$5240, $\Sigma_2 =$ \$119192, $S =$ \$60324; somewhat high but not extreme.)

The lexicographic maximum turns out to be (108) — partially explaining why it is so "remarkable." It can be obtained in the same fashion, in fewer than 2 Mμ.

230. Assume that all solutions to the exact cover problem contain the same number of options, d. (For example, $d = 16$ in Fig. 74.) Then we can replace each cost \$$c$ by the complementary cost, \$$(C - c)$, where C is sufficiently large to make this nonnegative. Solve the problem with the complementary costs; then subtract its total cost from Cd. [It's convenient to implement a special version of Algorithm X$^\$$ that does this automatically, with appropriate changes to the presentation of intermediate and final results.]

231. (a) `MAPLE` `ARRAY` `SMOKE` `TYPES` (\$139); (b) `HAPPY` `EXILE` `ALLOW` `PELTS` (\$176); (c) `JAMBS` `EQUIP` `TUMOR` `SASSY` or `MAGMA` `EQUIP` `OUNCE` `WAKED` (\$197).

Algorithm X$^\$$ needs 6 Gμ, 80 Gμ, and 483 Gμ to find these; Algorithm X needs 5 Gμ, 95 Gμ, and 781 Gμ to visit all solutions, of which there are 27, 8017, and 310077. (Section 7.2.2's trie-based methods are *much* faster: They need just 12 Mμ, 628 Mμ, 13 Gμ.)

232. No. The author actually did just that, in his first experiments; and he was lucky, because the algorithm not only gave the same optimum placements, it also took significantly less time — only 1.5 Gμ and 0.2 Gμ. However, the minimum cost (\$182) and maximum cost (\$202) were just \$1 different from the next-best costs! The effects of 16 rounding errors, each potentially changing the result by nearly \$1, could have invalidated everything. [Therefore the author used \$$\lfloor 2^{32} d(i,j)\rfloor$ when preparing Fig. 74.]

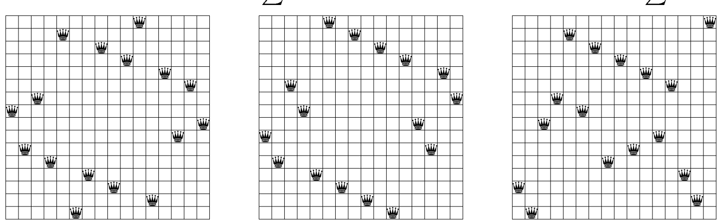
233. With costs \$$\lfloor 2^{32} \ln d(i,j)\rfloor$, we get the same answers (but faster: $1.2 + 0.2$ Gμ).

234. By that measure, *every* placement of n nonattacking queens (or rooks!) costs

$$\sum_{k=1}^{n}\bigl((k-c)^2 + (p_k - c)^2\bigr) = 2\sum_{k=1}^{n}(k-c)^2 = \frac{n(n^2-1)}{6}, \qquad \text{where } c = (n+1)/2.$$

235. Now the roles are reversed: We're more interested in the periphery than in the center, and the minimum is easier to compute than the maximum. The minimum cost, \$127760, is achievable in four ways, each symmetric; hence we must take $K = 17$, not $K = 9$. This computation took only 1.3 Gμ. (The two examples below have different sets of distances, which coincidentally yield the same total cost.) But there's a unique way to get the maximum cost, \$187760, discovered (with $K = 9$) in 9.7 Gμ:

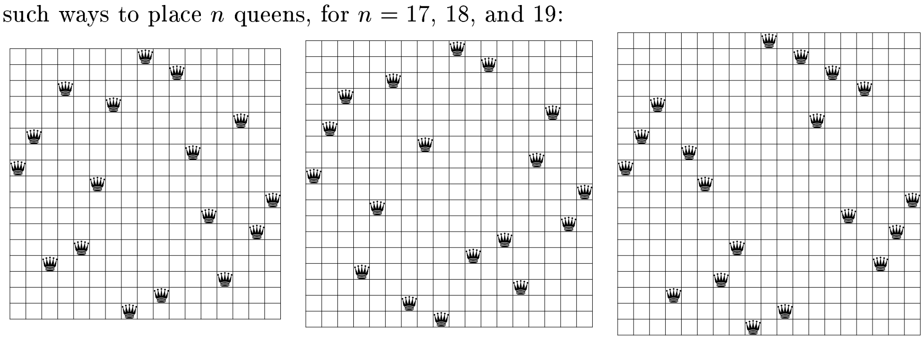
Minimum $\sum d^4$ Maximum $\sum d^4$



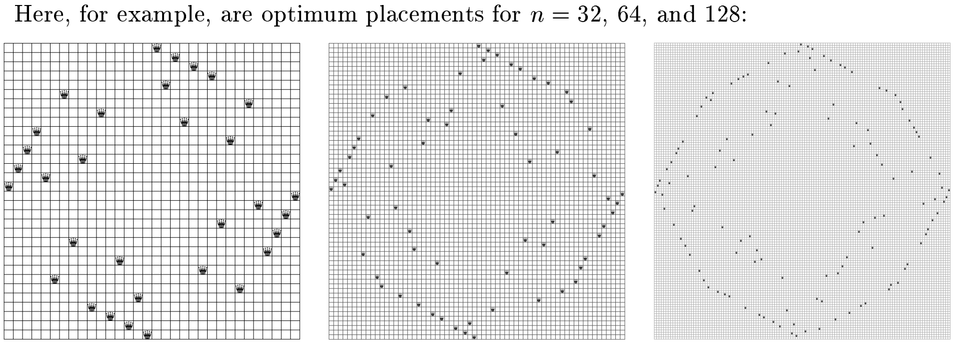
236. The idea is first to minimize the longest distance; then, placing a queen at that distance in all possible ways, to minimize the next-longest distance; and so on. In other words, if the options are in nondecreasing order by cost, it's almost like the search for lexicographically minimax solutions, iteratively as in answer 229.

However, there's a catch: Many options have the same cost. Different orderings of equal-cost options can lead to wildly different lex-min solutions. For example, suppose there are four options, '1' for \$1, '2' for \$2, '1 3' for \$3, and '2 3' for \$3. In that order, the minimax solution omits the final option and costs $3^N + 2^N$, which is not optimum.

The solution is to add to each option a primary item describing its cost, and to use Algorithm M iteratively by specifying the number of queens of highest costs, keeping this as low as possible until the problem has no solutions. Here are the best such ways to place n queens, for $n = 17$, 18, and 19:



The author was able to reach $n = 47$ with dancing-links-based methods, in an afternoon. But he knew that integer programming is significantly faster for "linear" applications such as the n queens problem (see answer 36). So he enlisted the help of Matteo Fischetti; and sure enough, Matteo was able to extend the results dramatically. Here, for example, are optimum placements for $n = 32$, 64, and 128:



author
integer programming
Fischetti

It appears likely that these optimum queen placements have rotational symmetry only when $n = 1$, 4, 5, 16, and 32. But the solutions for $n = 64$ and 128 do have 2^6 and 2^{12} equivalent mates, because they contain respectively 6 and 12 "tiltable squares" in the sense of exercise 7.2.2–11(c).

(The limiting behavior may not "kick in" until N is quite large. For example, the optimum solution when $n = 16$ and $N = 20$ is *not* the symmetrical one illustrated; the placements 8 11 4 7 5 12 1 16 14 2 15 10 3 13 6 9 have total cost $\approx 2.08 \times 10^{21}$, which beats $\approx 2.09 \times 10^{21}$. The limiting shape turns out to be optimum if and only if $N \geq 21$.)

237. False. For example, the square shown here is the smallest of ≈ 3 billion solutions for which $02 \equiv 20$, $03 \equiv 30$, $12 \equiv 21$, $13 \equiv 31$, $42 \equiv 24$, $43 \equiv 34$.

$$\begin{bmatrix} 1&2&1&1&3\\ 1&1&0&0&3\\ 1&0&1&0&3\\ 1&0&3&6&9\\ 3&1&3&9&1 \end{bmatrix}$$

238. $\begin{bmatrix} 1&1&3\\ 3&0&7\\ 1&3&9 \end{bmatrix} \begin{bmatrix} 2&1&1&1\\ 1&0&3&1\\ 1&1&9&3 \end{bmatrix} \begin{bmatrix} 2&1&2&1&1\\ 1&0&3&0&1\\ 1&1&3&9&3 \end{bmatrix} \begin{bmatrix} 1&1&1&2&1&1\\ 1&0&0&1&0&3\\ 3&3&1&1&7&1 \end{bmatrix} \begin{bmatrix} 1&1&1&1&2&1&1\\ 1&0&0&0&4&0&3\\ 3&1&9&3&1&7&1 \end{bmatrix};$

$\begin{bmatrix} 9&9&7\\ 7&8&7\\ 7&3&3 \end{bmatrix} \begin{bmatrix} 8&9&9&9\\ 8&6&9&9\\ 7&7&1&7 \end{bmatrix} \begin{bmatrix} 9&8&9&9&9\\ 9&8&8&9&7\\ 7&7&3&1&7 \end{bmatrix} \begin{bmatrix} 9&8&9&9&9&9\\ 9&8&8&5&7&9\\ 7&7&3&3&7&1 \end{bmatrix} \begin{bmatrix} 9&8&9&9&9&9&9\\ 9&8&6&8&5&9&7\\ 7&7&7&3&3&1&7 \end{bmatrix}.$

The problems for $n = 7$ have 1759244 options; yet they were solved in 20 Gμ without preprocessing. Special methods would, however, be required for $n \geq 8$.

239. Introduce primary items k and jk, for $1 \leq k \leq n$ and for all j with $k \in S_j$. When $S_j = \{k_1, \ldots, k_t\}$, there's an option '$jk_1\ \ldots\ jk_t$' of cost w_j, together with t options '$k_i\ jk_i$' of "infinitesimal" cost ϵ^i for $1 \leq i \leq t$; also t "slack" options 'jk_i' of cost 0.

For example, suppose the only sets that cover 1 are S_1, S_2, S_3, S_4; and suppose that an optimum set cover uses S_2 and S_4 but neither S_1 nor S_3. Then a maximum-cost solution to this exact cover problem will use option '11 . . . ' of cost w_1, '31 . . . ' of cost w_3, '1 21' of cost ϵ^2, and '41' of cost 0 (because the alternative with '21' and '1 41' has smaller cost ϵ^4).

[See M. Gondran and M. Minoux, *Graphs and Algorithm* (1984), exercise 10.35. When finding the k best solutions instead of a single optimum, all solutions that become identical when ϵ is set to zero should be counted just once.]

240. Add {`WY`, `CO`, `NM`} and either `ID` or `UT` or `AZ`. Or add {`ID`, `UT`, `CO`, `OK`}. Or add {`SD`, `MO`} and either {`IA`, `OK`} or {`NE`, `AR`} (a surprise to the author when he posed this problem).

241. No, although it does find the cases where regions of fewer than 6 vertices are cut off. Round 1 discovers that New England can be shrunk to a single item; then Round 2 is able to remove options such as '`LA AR TN VA MD PA`'. Altogether 3983 options and 5 items are removed, at a cost of 8 Gμ.

242. Before visiting a solution in step R2$'$, use depth-first search to find the connected components of the residual graph. Reject the solution if any such component has a size d for which $d < L \cdot \lceil d/(U-1) \rceil$.

243. Let $W = w_1 + \cdots + w_n$ be the sum of all weights. Then we have $\sum_{k=1}^{d}(x_k - r)^2 = \sum_{k=1}^{d} x_k^2 - 2rW + r^2 d$, because $\sum_{k=1}^{d} x_k = W$ in an exact cover problem.

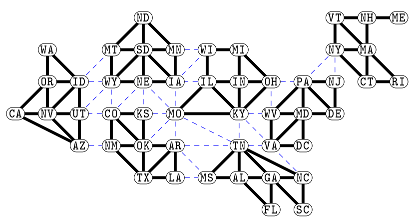
244. True: Let G have m edges and n vertices. A solution with k edges between vertices of the same option has total interior cost $n(t-1) - 2k$, total exterior cost $2(m-k)$.

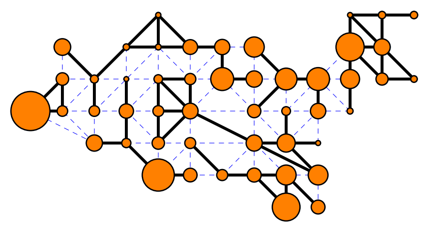
[But answer 246 shows that this can fail with options of different sizes.]

245. For (a), exercise 242 gives $42498 - 25230 = 17268$ options of size 7. Minimum cost \$58 is discovered in 101 Mμ. For (b), there are $1176310 - 1116759 = 59551$ options

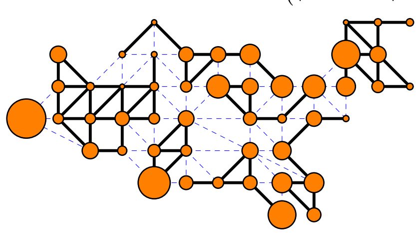
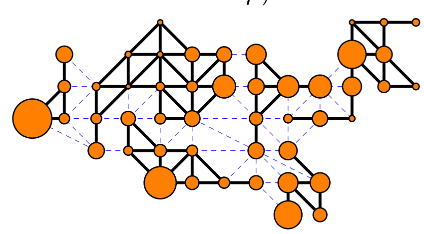
rotational symmetry
slack
Gondran
Minoux
author
New England
depth-first search

with population in [43 . . 45] million. In the optimum solution shown below, which was found in 7.7 Gμ, all populations lie in the range [43.51 . . 44.24] million.

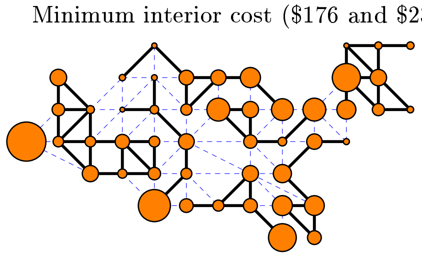
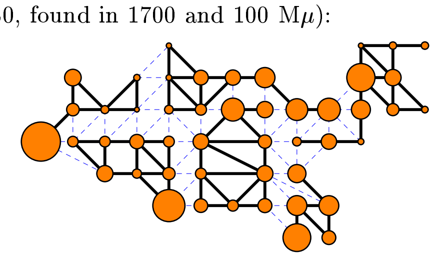




246. Minimum exterior cost (\$90 and \$74, found in 612 and 11 Mμ):

Minimum interior cost (\$176 and \$230, found in 1700 and 100 Mμ):

247. Use the procedure of answer 8 for raw data entry, but also set `COST(`j`)` $\leftarrow$ ⟨the cost of the current option⟩ for $p < j \le p + k$ at the beginning of step I5.

Then assign taxes "greedily" by doing the following for $k = 1, 2, \ldots, n$: If item k has no options, terminate with an unsolvable problem. Otherwise let c be the minimum cost of k's options, and set `COST(`k`)` $\leftarrow c$; this is the "tax" on k. If $c > 0$, subtract c from the cost of every option on k's list; this will affect all nodes of those options.

(The modified costs will be used internally. But all results reported to the user should be expressed in terms of the original costs, by adding the taxes back in.)

After all taxes have been assigned, sort the options by their (new) costs. (The "natural list merge sort," exercise 5.2.4–12, works well for this purpose, with the `COST` fields in spacer nodes serving as links.)

Finally, achieve (118) by re-inserting all nodes, in order of cost.

[Taxes could be assessed in many other ways. In general we seek real numbers u_1, $\ldots$, u_n such that $c_j \ge \sum\{u_i \mid \text{item } i \text{ in option } j\}$ for $1 \le j \le m$, where $u_1 + \cdots + u_n$ is maximum. This is a linear programming problem, which happens to be dual to the (fractional) exact cover problem of minimizing $c_1x_1 + \cdots + c_mx_m$ such that $x_1, \ldots, x_m \ge 0$ and $\sum\{x_j \mid \text{item } i \text{ in option } j\} = 1$ for $1 \le i \le n$. An "optimum" taxation scheme, found by a linear programming solver, might make Algorithm C$^\$$ significantly faster than it is with the greedy scheme above, even on highly nonlinear XCC problems; careful tests have not yet been made. See M. Gondran and J. L. Laurière, *Revue Française d'Automatique, Informatique et Recherche Opérationnelle* **8**, V-1 (1974), 27–40.]

greedily
sort
list merge sort
spacer
linear programming problem
dual
exact cover problem, fractional
Gondran
Laurière

248. Set $t \leftarrow \infty$, $c \leftarrow 0$, $j \leftarrow \texttt{RLINK}(0)$, and do the following while $j > 0$: Set $p \leftarrow \texttt{DLINK}(j)$ and $c' \leftarrow \texttt{COST}(p)$. If $p = j$ or $c' \geq \vartheta$, go to C8$^\$$. Otherwise set $s \leftarrow 1$, $p \leftarrow \texttt{DLINK}(p)$, and loop as follows: If $p = j$ or $\texttt{COST}(p) \geq \vartheta$, exit the loop; otherwise if $s = t$, set $s \leftarrow s + 1$ and exit; otherwise if $s \geq L$, set $s \leftarrow \texttt{LEN}(j)$ and exit; otherwise set $s \leftarrow s + 1$, $p \leftarrow \texttt{DLINK}(p)$, and continue. After exiting the loop, if $s < t$ or ($s = t$ and $c < c'$), set $t \leftarrow s$, $i \leftarrow j$, and $c \leftarrow c'$. Finally set $j \leftarrow \texttt{RLINK}(j)$.

[The author uses $L = 10$. He considered doing a complete search, thereby avoiding the frequent updates to LEN in (13), (15), etc.; but that turned out to be a bad idea.]

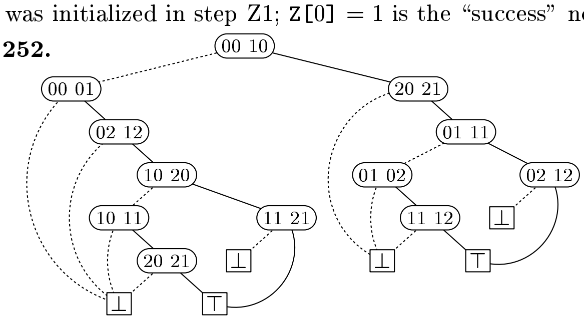
249. After we've seen t costs, we know only that the remaining $dk - t$ are nonnegative. The following algorithm sorts incoming costs into the rightmost positions of a buffer $b_0 b_1 \ldots b_{dk-1}$, maintaining the best possible lower bound l: Set $l = t = 0$. When seeing a new cost c, set $p \leftarrow t$, $r \leftarrow 1$, and do this while $rp > 0$: Set $x \leftarrow b_{dk-p}$. If $c \leq x$, set $r \leftarrow 0$. Otherwise if $p \bmod k = 0$, set $l \leftarrow l + x - y$; set $y \leftarrow b_{dk-p-1} \leftarrow x$, $p \leftarrow p - 1$. After $rp = 0$, set $b_{dk-p-1} \leftarrow c$, $t \leftarrow t + 1$; if $p \bmod k = 0$, set $l \leftarrow l + c - y$. Stop if $l \geq \theta$.

250. Keep a separate "accumulator" for each character in Z, and another for z if it is present. Look at each active item i: If $\texttt{NAME}(i)$ begins with a character of Z, add $\texttt{COST}(\texttt{DLINK}(i))$ to the appropriate accumulator. Otherwise if $z = 1$, add that cost to the accumulator for z. Otherwise if $z > 1$, use exercise 249 to accumulate costs that are separated by z. If any of the accumulators becomes $\geq T - C_l$, go to C8$^\$$.

(When Z or z hints are given, step C1$^\$$ should verify that they are legitimate.)

251. When all items have been covered, step Z2 will see the signature $\texttt{S}[0] = 0$, which was initialized in step Z1; $\texttt{Z}[0] = 1$ is the "success" node '$\top$'.

252.



Notice that this free ZDD is *not* ordered, because '02 12' appears above '20 21' in the left branch but below '20 21' in the right branch.

253. Introduce a global variable COUNT; also auxiliary variables $c_0 c_1 \ldots$ indexed by the current level l; also integer variables $\texttt{C}[t]$ indexed by cache location t. Set $\texttt{COUNT} \leftarrow 0$ and $\texttt{C}[0] \leftarrow 1$ in step Z1. If a cache hit occurs in Z2, set $\texttt{COUNT} \leftarrow \texttt{COUNT} + \texttt{C}[t]$; otherwise set $c_l \leftarrow \texttt{COUNT}$. Set $\texttt{C}[m_l] \leftarrow \texttt{COUNT} - c_l$ in step Z7.

254. (a) If the options include d different colors for item i, a subproblem has $d + 2$ distinct cases: Either item i does not appear in any remaining options, or its list has not been purified, or its list has been purified to a particular color. So we reserve $\lceil \lg(d + 2) \rceil$ bits for i in the signature. If, for example, $d = 4$, those three bits will contain one of the codes 000, 001, 010, 011, 100, 101.

[In order to recognize the relevant case, Algorithm Z's version of the 'purify' operation in (55) should set $\texttt{COLOR}(i) \leftarrow c$ in the header node for i; the 'unpurify' in (57) should set $\texttt{COLOR}(i) \leftarrow 0$; and step Z1 should set $\texttt{COLOR}(i) \leftarrow 0$. That initialization step should also remap i's colors so that they appear internally as 1, 2,…, d.]

(b) In large problems σ will occupy several octabytes. Give each item i a new field $\texttt{SIG}(i)$, which is an index to a code table, and a new field $\texttt{WD}(i)$. If $\texttt{LEN}(i) \neq 0$, item i will contribute $\texttt{CODE}[\texttt{SIG}(i) + \texttt{COLOR}(i)]$ to octabyte $\texttt{WD}(i)$ of σ.

author
free ZDD
ordered ZDD
purify
header node
unpurify

[If hashing is used for the cache lookup in step Z2, the `CODE` table can also contain random bits, for convenience in computing a good hash function.]

(c) Operation hide$'(p)$ doesn't remove node q from list `TOP`(q), if that list has been purified. But if `TOP`(q) is included in the signature, we'll *never* get a cache hit for solutions with different colors, even when subproblems don't actually depend on those colors. Therefore we need to know when a secondary item has no active options in its list.

(d) The trick is to decrease `LEN`(i), while still retaining the nodes on list i. If `LEN`(i) becomes zero, when i is a secondary item, we can then remove it from the list of active secondary items (whose head is $N+1$, by answer 8).

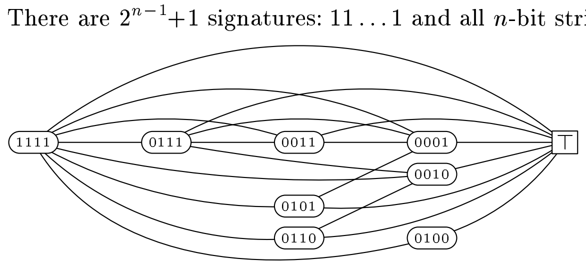
[We can also use this trick in the 'hide' routine: Let hide$''(p)$ be like hide(p) except that `DLINK`(u) and `ULINK`(d) remain unmodified when `COLOR`$(q)<0$; `LEN`(x) is decreased as usual.] Of course unpurify and unhide$''$ should undo purify and hide$''$.

Some delicate maneuvers are needed to avoid deactivating a secondary item twice, and to reactivate it at precisely the right time when unpurifying. (The author's implementation temporarily sets the `LEN` to -1.)

255. Let $V_n=\sum_{k=0}^{n}(n-1-2k)\binom{n-k}{k}$, $W_n=\sum_{k=0}^{n}\bigl(1+(n-1-2k)+(n-1-2k)^2\bigr)\binom{n-k}{k}$. Using the fact that $\sum_{k=0}^{n}\binom{k}{r}\binom{n-k}{k}=[z^n]\,z^{2r}/(1-z-z^2)^{r+1}$, we obtain the closed forms $V_n=\bigl((n-5)F_{n+1}+2(n+1)F_n\bigr)/5$ and $W_n=\bigl((5n^2+7n+25)F_{n+1}-6(n+1)F_n\bigr)/25$. (See the derivation of 1.2.8–(17).) When N is even, Algorithm Z performs W_N-1 updates and outputs a ZDD with V_N+2 nodes. When N is odd, it performs W_N updates and outputs the trivial ZDD '$\bot$'.

256. Let $T(N)$, $Z(N)$, and $C(N)$ be the time, ZDD size, and cache size needed for K_N. With (89) the algorithm first spends $T(2q)+T(2r)$ time to create a ZDD of size $Z(2q)+Z(2r)$. Then it spends $\min(T(2q+1),T(2r-1))$ time to learn that no more ZDD nodes are desirable. The cache size is $C(2q)+C(2r)+\min(C(2q+1),C(2r-1))$.

257. (a) There are $2^{n-1}+1$ signatures: $11\ldots1$ and all n-bit strings beginning with 0.
(b)



Each nonzero signature σ has $2^{\nu\sigma-1}$ branches.

258. See (84). Now it's $V_n=v_n+\sum_{k=1}^{n-1}\binom{n-1}{k}v_k=((72n-342)5^n+(375n-875)4^n+600\cdot3^n+1800n2^n+1550)/3600$. [For example, $V_{16}=40454337297$; $\varpi_{16}=10480142147$.]

259. (a) The signatures at level l are $\{X_{l+1},\ldots,X_n\}$ together with all $\binom{n}{l}$ l-element subsets of $\{Y_1,\ldots,Y_n\}$. So there are 2^n of them; also $2+\sum_{l=0}^{n}(n-l)\binom{n}{l}=n2^{n-1}+2$ ZDD nodes; and $((n^2+3n+4)2^n-4)/4$ updates.

(b) Now the signatures are $\{X_{l+1},\ldots,X_n\}$ plus l-element subsets of $\{Y_1,\ldots,Y_{l+1}\}$. So we get $\binom{n}{2}+1$ cache memos; n^2+2 ZDD nodes; $(2n^3+15n^2+n)/6$ updates.

260. The ménage problem, with $\approx n!/e^2$ solutions, leads to unexpected running times: We seem to get roughly order $n^{3/2}\rho^n$ updates, where $\rho\approx3.1$; but better results are obtained for $n\ge13$ when the MRV heuristic is *not* used in step Z3! Then the running time may well be $\Theta(ne^n)$, although the ZDD size apparently grows as $n\rho^n$ with $\rho\approx2.56$.

The other problem, with L_n+2 solutions, needs just $6n+9$ memos, $8n-9$ ZDD nodes, and $34n-58$ updates.

261. (a) Introduce primary items v^- and v^+ for each vertex v, representing the possibility of passing through v; but omit v^- for $v \in S$, and v^+ for $v \in T$. Also introduce secondary items v, whose color (if nonzero) represents the path number. The main options are 'u^+ v^- $u{:}k$ $v{:}k$', for each arc $u \longrightarrow v$ and for $1 \le k \le m$. There also are options 'v^- $v{:}0$' for all $v \notin S$, and 'v^+ $v{:}0$' for all $v \notin T$.

Moreover, we need a way to number each path canonically, so that we don't get $m!$ equivalent solutions. (The method of exercise 122 does not work with Algorithm Z.) If $S = \{s_1, \ldots, s_p\}$, introduce primary items x_k and secondary items y_k for $1 \le k \le p$, with the following options: 'x_k $s_k{:}0$ $y_{k-1}{:}j$ $y_k{:}j$' and 'x_k $s_k{:}(j{+}1)$ $y_{k-1}{:}j$ $y_k{:}(j{+}1)$', for $1 \le k \le p$ and $0 \le j < k$. [Omit the item $y_{k-1}{:}j$ when $k = 1$; omit options with $y_p \ne m$.]

Many of these options can never be used. Algorithm P readily removes them.

(b) Remove unreachable vertices and unreachable arcs from G, if necessary, so that the only sources and sinks are $S = \{s_1, \ldots, s_k\}$ and $T = \{t_1, \ldots, t_k\}$. Then use items v^-, v^+, v and the main options of the construction in part (a); but omit any option that specifies $s_j{:}k$ or $t_j{:}k$ for $j \ne k$.

(c) This is a trick question, because each path contains exactly one vertex on the diagonal. The problem therefore factors neatly into two independent subproblems. It suffices to find $n-1$ vertex-disjoint paths from $S = \{(0,1), \ldots, (0,n{-}1), (1,n), \ldots, (n{-}1,n)\}$ to $T = \{(1,1), \ldots, (n{-}1,n{-}1)\}$ in the digraph with vertices (i,j) for $0 \le i \le j \le n$, $(i,j) \notin \{(0,0), (0,n), (n,n)\}$, and arcs $(i,j) \longrightarrow (i+1,j)$, $(i,j) \longrightarrow (i,j-1)$.

If this problem has P_n solutions, given by a ZDD Z with M_n nodes, the original problem has P_n^2 solutions, given by a ZDD Z'' with $2M_n$ nodes. We obtain Z'' by replacing $\top$ in Z with the root of Z', where Z' specifies the reflections of the paths of Z.

Algorithm Z needs just 7 gigamems to find $P_{16} = 992340657705109416$ and $M_{16} = 3803972$. (In fact, P_n is known to be $\prod_{1\le i\le j\le k\le n}(i{+}j{+}k{-}1)/(i{+}j{+}k{-}2)$, the number of plane partitions that are totally symmetric: N. Beluhov [to appear] has found a nice way to glue six triangular diagrams together, in kaleidoscope fashion, which establishes a one-to-one correspondence linking these paths to symmetrical diamond tilings like those of exercise 262(b).)

(d) There are exactly 47356 solutions. Algorithm C finds them in 278 Gμ, without preprocessing; but it needs only 760 Mμ, after Algorithm P has removed redundant options. Algorithm Z, by contrast, handles the problem in 92 Gμ, using 7 gigabytes of memo-cache memory (without preprocessing); 940 Mμ and 90 megabytes (with). Hence Algorithm Z is undesirable for problem (d), but essential for problem (c).

262. (a) The ordering of the primary items — the cells of S_n — is critical: Rowwise ordering (left-to-right, top-to-bottom) causes exponential growth; but columnwise ordering (top-to-bottom, left-to-right) yields *linear* ZDD size, and $\Theta(n^2)$ running time.

Furthermore, it turns out to be better *not* to use the MRV heuristic, when $n \ge 18$. Then the number of ZDD nodes is $154440n - 2655855$ for all $n \ge 30$. Only 2.2 Gμ are needed for $n = 32$. There are $68719476736 = (\sqrt{2}\,)^{72}$ solutions for S_{16}, via exercise 7.1.4–208; for S_{32} there are $152326556015596771390830202722034115329 \approx 1.552^{200}$.

(An Aztec diamond of order m has exactly $2^{m(m+1)/2}$ domino tilings; moreover, as $m \to \infty$, the dominoes at the corners are q.s. aligned, except within an "arctic circle" of radius $m/\sqrt{2}$. See W. Jockusch, J. Propp, and P. Shor, arXiv:math/9801068 [math.CO] (1995), 44 pages; H. Cohn, N. Elkies, and J. Propp, *Duke Math. J.* **85** (1996), 117–166. See also D. Grensing, I. Carlsen, and H.-Chr. Zapp, *Philos. Mag.* **A41** (1980), 777–781.)

[Tilings of the more general shapes S_{mn} considered here, where we replace 16 by $2m$ and 7 by $m{-}1$, are more mysterious. M. Ciucu observes that $R_{(2m)(n-2m)} \subseteq S_{mn} \subseteq$

factors
plane partitions
totally symmetric
Beluhov
preprocessing
Rowwise ordering
columnwise ordering
MRV heuristic
Aztec diamond
arctic circle
Jockusch
Propp
Shor
Cohn
Elkies
Propp
Grensing
Carlsen
Zapp
Ciucu

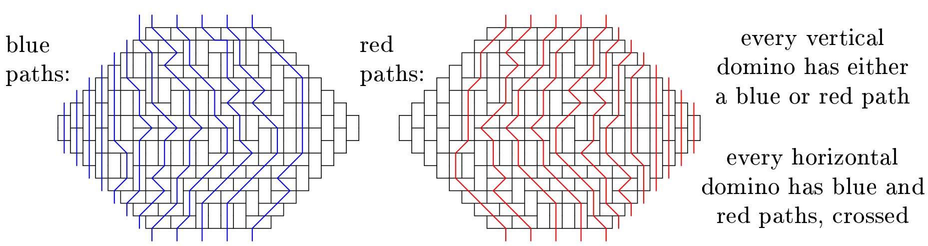
Stanley
vertex-disjoint paths
Randall
convex triangular regions
plane partitions
Klarner
reverse plane partition
Cohn
Larsen
Propp
ellipse
Boutillier

$R_{(2m)(n+2m)}$, where R_{kn} is a $k \times n$ rectangle; furthermore both $R_{(2m)(n+2m)} \setminus S_{mn}$ and $S_{mn} \setminus R_{(2m)(n-2m)}$ are tilable. Richard Stanley has shown, in *Discrete Applied Math.* **12** (1985), 81–87, that $R_{(2m)n}$ has $\sim a_{2m}\mu_{2m}^{n+1}$ tilings, for fixed m as $n \to \infty$, where

$$a_k = \frac{(1+\sqrt{2})^{k+1} - (1-\sqrt{2})^{k+1}}{2\sqrt{2}}, \quad \mu_{2m} = \prod_{j=1}^{m}\Bigl(\cos\theta_j + \sqrt{1+\cos^2\theta_j}\Bigr), \quad \theta_j = \frac{j\pi}{2m+1}.$$

Hence S_{mn} has $\Theta(\mu_{2m}^n)$ tilings in that limit. But if $m = \alpha n$ as $n \to \infty$, the limiting "arctic curve" outside which dominoes tend to be frozen remains to be discovered.]

There is, incidentally, a beautiful connection between domino tilings and vertex-disjoint paths, discovered by D. Randall (unpublished):



(b) In this case the triangle coordinates of answer 124 yield linear growth if we use items (x, y) for $0 \le x < n+8$, $0 \le y < 16$, $x+y \ge 8$; $(x, y)'$ for $0 \le x < n+8$, $0 \le y < 16$, $7 \le x+y < n+15$. The options are '(x, y) (x', y')', where $(x', y') = (x, y) - \{(0,0), (0,1), (1,0)\}$ and both items exist. Then the ZDD size (without MRV) turns out to be $257400n - 1210061$, for all $n \ge 7$.

The convex triangular regions that can be tiled with diamonds are precisely those that have equally many Δ and ∇ triangles, namely the generalized hexagons T_{lmn} with sides (l, m, n, l, m, n) for some $l, m, n \ge 0$. These tilings are equivalent to plane partitions that fit in an $l \times m \times n$ box. In fact you can "see" this equivalence, because the diagrams resemble cubies packed into a corner of the box! (David Klarner made this discovery in the 1970s, but didn't publish it.) Therefore every tiling of T_n has respectively $(1, 2, \ldots, 8, 7, \ldots, 1)$ vertical diamonds in rows $(1, 2, \ldots, 15)$, hence 64 in all; and these occurrences are nested. For example, the middle diagram corresponds to the reverse plane partition shown here. (See exercise 5.1.4–36, from which it follows that the generalized hexagon T_{lmn} has exactly $\Pi_{lmn} = \prod_{i=1}^{l}\prod_{j=1}^{m}\prod_{k=1}^{n}(i+j+k-1)/(i+j+k-2)$ tilings. In particular, we have $\Pi_{888} =$ 5055160684040254910720; $\Pi_{88(16)} =$ 2065715788914012182693991725390625.)

```
00012457
1134569c
12368abc
25789bbc
4578accc
459aaccc
569aaccc
bbbbcccc
```

[In *New York J. of Math.* **4** (1998), 137–165, H. Cohn, M. Larsen, and J. Propp studied random tilings of T_{lmn} when l, m, and n approach infinity with constant scaling, and conjectured that they are q.s. "frozen" outside of the largest enclosed ellipse. See also the more general results of C. Boutillier, *Annals of Probability* **37** (2009), 107–142.]

263.

	parameters	solutions	items	options	Alg C time	space	Alg Z time	space	ZDD
(a)	organ-pipe order	14772512	$32+58$	256	40 Gμ	23 KB	55 Gμ	4.1 GB	56M
(b)	6×10	2339	$72+0$	2032	4.1 Gμ	230 KB	3.1 Gμ	23 MB	11K
(b)	8×8, square	16146	$77+1$	2327	20 Gμ	264 KB	14 Gμ	101 MB	59K
(b)	8×8, straight	24600	$77+1$	2358	36 Gμ	267 KB	26 Gμ	177 MB	93K
(b)	8×8, skew	23619	$77+1$	2446	28 Gμ	275 KB	20 Gμ	137 MB	84K
(b)	8×8, ell	60608	$77+1$	2614	68 Gμ	291 KB	44 Gμ	276 MB	183K
(b)	8×8, tee	25943	$77+1$	2446	35 Gμ	275 KB	25 Gμ	166 MB	92K
(c)	`aaa` placed	987816	$49+42$	1514	25 Gμ	149 KB	18 Gμ	646 MB	2.2M

(d)	$(7,0,3)$	137216	$64+128$	3970	8.5 Gμ	642 KB	1.7 Gμ	20 MB	210K
(d)	$(7,3,4)$	41280	$70+140$	4762	3.2 Gμ	769 KB	1.0 Gμ	13 MB	122K
(e)	p=6, WORDS(1200)	1	$12+1230$	14400	17 Gμ	2 MB	25 Gμ	91 MB	14
(f)	kill symmetry	44*	$12+36$	1188	1.3 Gμ	110 KB	0.9 Gμ	10 MB	186
(g)	unmodified	18	$1165+66$	4889	202 Gμ	509 MB	234 Gμ	8.9 GB	2049
(g)	modified	1	$1187+66$	5143	380 Gμ	537 MB	424 Gμ	15 GB	336
(g)	preprocessed	18	$446+66$	666	223 Mμ	66 KB	1.8 Mμ	136 KB	574

* includes solutions that touch all cells

264. Let the primary items be linearly ordered, and let $r(o)$ be the smallest primary item in option o. If $(\bar{o}?\ l{:}\ h)$ is a ZDD node, every option o' in the subZDD rooted at h has $r(o') > r(o)$, because o covers $r(o)$ and smaller items have already been covered. Moreover, if $l \neq 0$, the option o' in node l has $r(o') = r(o)$; and o' precedes o in the input.

Thus, if we use a stable sorting algorithm to sort the options by decreasing $r(o)$, the ZDD will respect the reverse of this ordering. [This result was proved by Nishino, Yasuda, Minato, and Nagata in their original paper. Unfortunately, the algorithm is usually too slow without MRV, except in special situations like those of exercise 262.]

265. Every solution below any given ZDD node covers the same primary items. If all items are primary, no two visible nodes have the same signature. And the nodes of the chain below every visible node are distinct, because they branch on different options.

Now suppose we have three primary items $\{p,q,r\}$, and one secondary item s, with options 'p', '$p\ r$', '$p\ s$', '$q\ r$', '$q\ s$'. If we don't use MRV, we'll branch on p. Choice 1, 'p', leads to a subproblem with signature 0111 that outputs $I_2 = (\overline{q\,r}?\ 0{:}\ 1)$, $I_3 = (\bar{p}?\ 0{:}\ 2)$. Choice 2, '$p\ r$', leads to a subproblem with signature 0101 that outputs $I_4 = (\overline{q\,s}?\ 0{:}\ 1)$, $I_5 = (\overline{p\,r}?\ 3{:}\ 4)$. Choice 3, '$p\ s$', leads to a subproblem with signature 0110 that outputs $I_6 = (\overline{q\,r}?\ 0{:}\ 1)$, $I_7 = (\overline{p\,s}?\ 5{:}\ 6)$. And $I_6 = I_2$.

A similar example, with items $\{q_1,q_2,q_3,r_1,r_2,r_3\}$ in place of $\{q,r\}$, and with 23 options 'p', '$p\ r_i$', '$p\ s$', '$q_i\ q_j$', '$r_i\ r_j$', '$q_i\ r_i$', '$q_i\ r_j$', '$q_j\ r_i$', '$q_i\ s$', for $1 \le i < j \le 3$, fails when MRV dictates the choices.

266. Let the given shape be specified as a set of integer pairs (x,y). These pairs might simply be listed one by one in the input; but it's much more convenient to accept a more compact specification. For example, the utility program with which the author prepared the examples of this book was designed to accept UNIX-like specifications such as '`[14-7]2 5[0-3]`' for the eight pairs $\{(1,2),\ (4,2),\ (5,2),\ (6,2),\ (7,2),\ (5,0),\ (5,1),\ (5,3)\}$. (Notice that a pair is included only once, if it's specified more than once.) The range $0 \le x,y < 62$ has proved to be sufficient in almost all instances, with such integers encoded as single "extended hexadecimal digits" `0`, `1`, ..., `9`, `a`, `b`, ..., `z`, `A`, `B`, ..., `Z`. The specification '`[1-3][1-k]`' is one way to define a 3×20 rectangle.

Similarly, each of the given polyominoes is specified by stating its piece name and a set T of typical positions that it might occupy. Such positions (x,y) are specified using the same conventions that were used for the shape; they needn't lie within that shape.

The program computes *base placements* by rotating and/or reflecting the elements of that set T. The first base placement is the shifted set $T_0 = T - (x_{\min}, y_{\min})$, whose coordinates are nonnegative and as small as possible. Then it repeatedly applies an elementary transformation, either $(x,y) \mapsto (y, x_{\max} - x)$ or $(x,y) \mapsto (y,x)$, to every existing base placement, until no further placements arise. (That process becomes easy when each base placement is represented as a sorted list of packed integers $(x \ll 16) + y$.) For example, the typical positions of the straight tromino might be specified as '`1[1-3]`'; it will have two base placements, $\{(0,0),(0,1),(0,2)\}$ and $\{(0,0),(1,0),(2,0)\}$.

stable sorting
Nishino
Yasuda
Minato
Nagata
visible nodes
signature
MRV
author
UNIX
extended hexadecimal digits
hexadecimal notation, extended
base placements
sorted
packed integers
straight tromino

After digesting the input specifications, the program defines the items of the exact problem, which are (i) the piece names; (ii) the cells xy of the given shape.

Finally, it defines the options: For each piece p and for each base placement T' of p, and for each offset (δ_x, δ_y) such that $T' + (\delta_x, \delta_y)$ lies fully within the given shape, there's an option that names the items $\{p\} \cup \{(x+\delta_x, y+\delta_y) \mid (x,y) \in T'\}$.

(The output of this program is often edited by hand, to take account of special circumstances. For example, some items may change from primary to secondary; some options may be eliminated in order to break symmetry. The author's implementation also allows the specification of secondary items with color controls, along with base placements that include such controls.)

Historical notes: Early algorithms for polyomino packing failed to realize the essentially unity between cells to be covered and pieces to be covered; their treatment of cells was quite different from their treatment of pieces. The fact that both cells and pieces are primary items of a "pure" exact cover problem was first noticed in connection with the Soma cube, by C. Peter-Orth [*Discrete Mathematics* **57** (1985), 105–121]. The base placements of tiles that are to be translated (but not rotated or reflected) are called "aspects" in *Tilings and Patterns* by Grünbaum and Shephard (1987).

267. RUSTY. [Leigh Mercer posed a similar question to Martin Gardner in 1960.]

268. As in the 3×20 example considered in the text, we can set up an exact cover problem with $12 + 60$ items, and with options for every potential placement of each piece. This gives respectively (52, 292, 232, 240, 232, 120, 146, 120, 120, 30, 232, 120) options for pieces (O, P, ..., Z) in Conway's nomenclature, thus 1936 options in all.

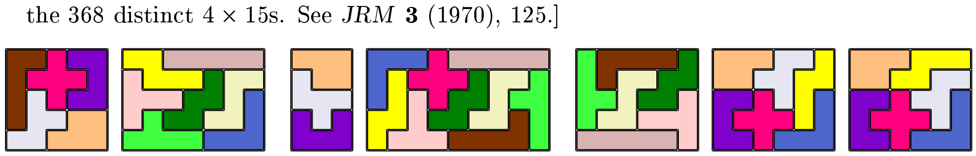
To reduce symmetry, we can insist that the X occurs in the upper left corner; then it contributes just 10 options instead of 30. But some solutions are still counted twice, when X is centered in the middle row. To prevent this we can add a *secondary item* 's': Append 's' to the five options that correspond to those centered appearances; also append 's' to the 60 options that correspond to placements where the Z is flipped over.

Without those changes, Algorithm X would use 10.04 Gμ to find 4040 solutions; with them, it needs just 2.93 Gμ to find 1010.

This approach to symmetry breaking in pentomino problems is due to Dana Scott [Technical Report No. 1 (Princeton University Dept. of Electrical Engineering, 10 June 1958)]. Another way to break symmetry would be to allow X anywhere, but to restrict the W to its 30 *unrotated* placements. That works almost as well: 2.96 Gμ.

269. There's a unique way to pack P, Q, R, U, X into a 5×5 square, and to pack the other seven into a 5×7. (See below.) With independent reflections, together with rotation of the square, we obtain 16 of the 1010. There's also a unique way to pack P, R, U into a 5×3 and the others into a 5×9 (noticed by R. A. Fairbairn in 1967), yielding 8 more. And there's a unique way to pack O, Q, T, W, Y, Z into a 5×6, plus two ways to pack the others via a bipair, yielding another 16. (These paired 5×6 patterns were apparently first noticed by J. Pestieau; see answer 286.) Finally, the packings in the next exercise give us 264 decomposable 5×12s altogether.

[Similarly, C. J. Bouwkamp discovered that S, V, T, Y pack uniquely into a 4×5, while the other eight can be put into a 4×10 in five ways, thus accounting for 40 of the 368 distinct 4×15s. See *JRM* **3** (1970), 125.]



secondary
break symmetry
author
color controls
Historical notes
cells versus pieces
pieces versus cells
Soma cube
Peter-Orth
aspects
Grünbaum
Shephard
Mercer
Gardner
Conway
secondary item
flipped over
Scott
break symmetry
Fairbairn
bipair
Pestieau
Bouwkamp

270. Without symmetry reduction, 448 solutions are found in 1.24 Gμ. But we can restrict X to the upper left corner, as in answer 268, flagging its placements with 's' when centered in the middle row or middle column (but not both). Again the 's' is appended to flipped Z's. Finally, when X is placed in dead center, we append *another* secondary item 'c', and append 'c' to the 90 rotated placements of W. This yields 112 solutions, after 0.35 Gμ.

Or we could leave X unhindered but curtail W to 1/4 of its placements. That's easier to do (although not *quite* as clever) and it finds those 112 in 0.44 Gμ.

Incidentally, there *aren't* actually any solutions with X in dead center.

271. The exact cover problem analogous to that in exercise 268 has 12 + 60 items and (56, 304, 248, 256, 248, 128, 1152, 128, 128, 32, 248, 128) options. It finds 9356 solutions after 16.42 Gμ of computation, without symmetry reduction. But if we insist that X be centered in the upper left quarter, by removing all but 8 of its placements, we get 2339 solutions after just 4.11 Gμ. (The alternative of restricting W's rotations is *not* as effective in this case: 5.56 Gμ.) These solutions were first enumerated by C. B. and Jenifer Haselgrove [Eureka: *The Archimedeans' Journal* **23** (1960), 16–18].

272. (a) Obviously only $k = 5$ is feasible. All such packings can be obtained by omitting all options of the cover problem that straddle the "cut." That leaves 1507 of the original 2032 options, and yields 16 solutions after 104 Mμ. (Those 16 boil down to just the two 5×6 decompositions that we already saw in answer 269.)

(b) Now we remove the 763 options for placements that don't touch the boundary, and obtain just the two solutions below, after 100 Mμ. (This result was first noticed by Tony Potts, who posted it to Martin Gardner on 9 February 1960.)

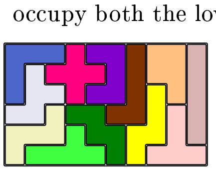
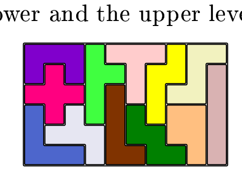
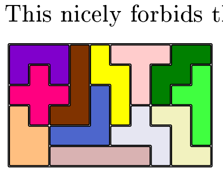
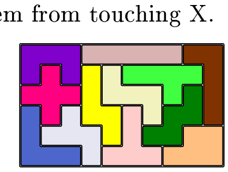
(c) With 1237 placements/options, the *unique* solution is now found after 83 Mμ.

(d) There are respectively (0, 9, 3, 47, 16, 8, 3, 1, 30, 22, 5, 11) solutions for pentominoes (O, P, Q, ..., Z). (The I/O pentomino can be "framed" by the others in 11 ways; but all of those packings also have at least one other interior pentomino.)

(e) Despite many ways to cover all boundary cells with just seven pentominoes, none of them lead to an overall solution. Thus the minimum is eight; 207 of the 2339 solutions attain it. To find them we might as well generate and examine all 2339.

(f) The question is ambiguous: If we're willing to allow the X to touch unnamed pieces at a corner, but not at an edge, there are 25 solutions (8 of which happen to be answers to part (a)). In each of these solutions, X also touches the outer boundary. (The cover and frontispiece of Clarke's book show a packing in which X doesn't touch the boundary, but it *doesn't* solve this problem: Using Golomb's piece names, there's an edge where X meets I, and there's a point where X meets P.) There also are two packings in which the edges of X touch only F, N, U, and the boundary, but not V.

On the other hand, there are just 6 solutions if we allow only F, N, U, V to touch X's corner points. One of them, shown below, has X touching the short side and seems to match the quotation best. These 6 solutions can be found in just 47 Mμ, by introducing 60 secondary items as sort of an "upper level" to the board: All placements of X occupy the normal five lower-level cells, plus up to 16 upper-level cells that touch them; all placements of F, N, U, V are unchanged; all placements of the other seven pieces occupy both the lower and the upper level. This nicely forbids them from touching X.

flipped Z's
Haselgrove, Colin
Haselgrove, Jenifer
Potts
Gardner
Golomb
secondary items

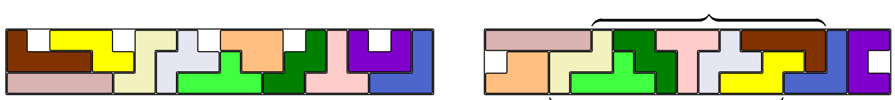
273. (a) We could set this up as twelve separate exact cover problems, one for each pentomino omitted. But it's more interesting to consider all cases simultaneously, by giving a "free pass" to one pentomino as follows: Add a new primary item '#', and twelve new options '# O', '# P', ..., '# Z'. The sixty items ij are demoted to secondary status.

To remove symmetry, delete 3/4 of the options for piece V; also make its new option '# V s', and add 's' to 3/4 of the options for piece W, where 's' is a new secondary item. That makes a total of 1194 options, involving $13 + 61$ items.

If Algorithm X branches first on #, the effect is equivalent to 12 separate runs; the search tree has 7.9 billion nodes, and the run time is 16.8 teramems. But if we use the nonsharp preference heuristic (see answer 10), the algorithm is able to save some time by making decisions that are common to several subcases. Its search tree then has 7.3 billion nodes, and the run time is 15.1 teramems. Of course both methods give the same answer, which is huge: 118,034,464.

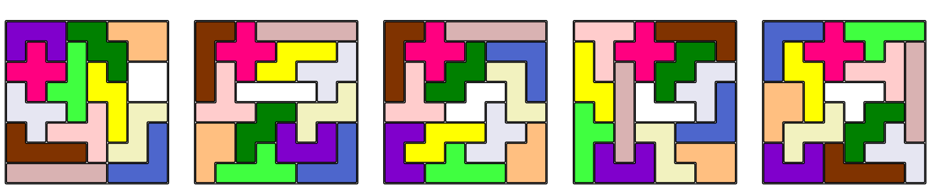
(b) Now keep items ij primary, but introduce 60 new secondary items ij'. There are 60 new options 'ij ij' $(i{+}1)j'$ $i(j{+}1)'$ $(i{+}1)(j{+}1)'$', where we omit items containing $(i{+}1)$ when $i = 2$ or $(j{+}1)$ when $j = 19$. This problem has 1254 options involving 73+61 items. Its search tree (with deprecated # branching) has about 950 million nodes; it finds 4,527,002 solutions, after about 1.5 teramems of computation.

A related, but much simpler, problem asks for packings in which exactly one hole appears in each of the column pairs {1, 2}, {5, 6}, {9, a}, {d, e}, {h, i}. That one has 1224 options, 78+1 items, 20 meganodes, 73 gigamems, and 23642 solutions. Here's one:



(c) A setup like the one in (a) yields 1127 options, $13+58$ items, 1130 meganodes, 2683 gigamems, 22237 solutions. (One of the noteworthy solutions is illustrated above.)

274. Restrict X to five essentially different positions; if X is on the diagonal, also keep Z unflipped by using the secondary item 's' as in answer 268. There are respectively (16146, 24600, 23619, 60608, 25943) solutions, found in (20.3, 36.3, 28.0, 68.3, 35.2) Gμ.

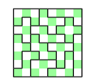


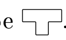
In each case the tetromino can be placed anywhere that doesn't immediately cut off a region of one or two squares. [The twelve pentominoes first appeared in print when H. E. Dudeney published *The Canterbury Puzzles* in 1907. His puzzle #74, "The Broken Chessboard," presented the first solution shown above, with pieces checkered in black and white. That parity restriction, with the further condition that no piece is turned over, would reduce the number of solutions to only 4, findable in 120 Mμ.]

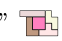
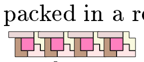
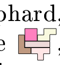
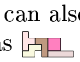
The 60-element subsets of the chessboard that *can't* be packed with the pentominoes have been characterized by M. Reid in *JRM* **26** (1994), 153–154.

The earliest known polyomino puzzle appeared in P. F. Catel's *Verzeichniß von sämmtlichen Waaren* (Berlin, 1785), #11: 4 Z pentominoes + 4 ells make a 6×6 square.

symmetry
nonsharp preference heuristic
flipped
Dudeney
checkered
parity
one-sided pentominoes
historical notes
Reid
Catel

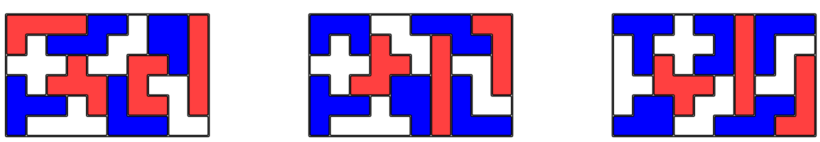


275. Yes, in seven essentially different ways. To remove symmetry, we can make the O vertical and put the X in the right half. (The pentominoes will have a total of $6 \times 2 + 5 \times 3 + 4 = 31$ black squares; therefore the tetromino *must* be .)

276. These shapes can't be packed in a rectangle. But we can use the "supertile"  to make an infinite strip $\cdots$  $\cdots$. [See B. Grünbaum and G. C. Shephard, *Tilings and Patterns* (1987), 508.] We can also tile the plane with a supertile like , or even use a generalized torus such as  (see exercise 7–137). That supertile was used in 2009 by George Sicherman to make tetromino wallpaper.

277. The 2339 solutions contain 563 that satisfy the "tatami" condition: No four pieces meet at any one point. Each of those 563 leads to a simple 12-vertex graph coloring problem; for example, the SAT methods of Section 7.2.2.2 typically need at most two or three kilomems to decide each case.

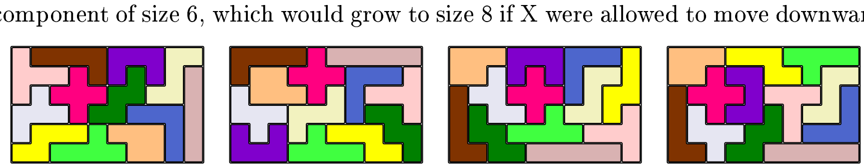
It turns out that exactly 94 are three-colorable, including the second solution to exercise 272(b). Here are the three for which W, X, Y, Z all have the same color:



278. The 2339 solutions in answer 271 restrict X to the upper left quarter; we must be careful not to include bipairs that might swap X out of that region. One way (see exercise 212) is to order the items: Put X first, then the other piece names, then the place names from 00 to 59. All swaps involving X will then move it up or left.

The 34 bipairs of the catalog now result in an exact cover problem with the same primary items and options as before, but with 2804 new secondary items. They limit the number of solutions to 1523; but the running time increases to 4.26 Gμ.

[The proof idea of Theorem S yields an interesting directed acyclic graph with 2339 vertices and 937 arcs. It has 1528 source vertices, 1523 sink vertices, and 939 isolated vertices (both sources and sinks). If we ignore the arc directions, there are 1499 components, of which the largest has size 10. That component contains the leftmost solution below, which belongs to four different bipairs. There also are two components of size 8, with three nonoverlapping bipairs. The rightmost solution belongs to a component of size 6, which would grow to size 8 if X were allowed to move downward.]

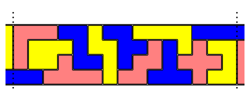
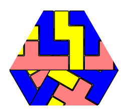
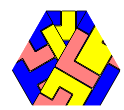


279. It's also possible to wrap *two* cubes of size $\sqrt{5}\times\sqrt{5}\times\sqrt{5}$, as shown by F. Hansson; see *Fairy Chess Review* **6** (1947–1948), problems 7124 and 7591. A full discussion appears in *FGbook*, pages 685–689.

280. (Notice that width 3 would be impossible, because every faultfree placement of the V needs width 4 or more.) We can set up an exact cover problem for a 4×19 rectangle in the usual way; but then we make cell $(x, y+15)$ identical to $(3-x, y)$ for $0 \le x < 4$ and $0 \le y < 5$, essentially making a half-twist when the pattern begins to wrap around. There are 60 symmetries, and care is needed to remove them properly. The easiest way is to put X into a fixed position, and allow W to rotate at most 90°.

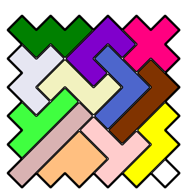
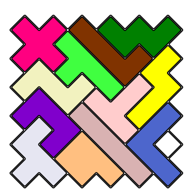
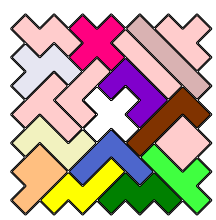
symmetry
Grünbaum
Shephard
torus
torus, generalized
Sicherman
wallpaper
tatami
SAT
directed acyclic graph
source vertices
sink vertices
isolated vertices
Hansson
symmetries

This exact cover problem has 850 solutions, 502 of which are faultfree. Here's one of the 29 strongly three-colorable ones, shown before and after its ends are joined:

 top:  bottom: 

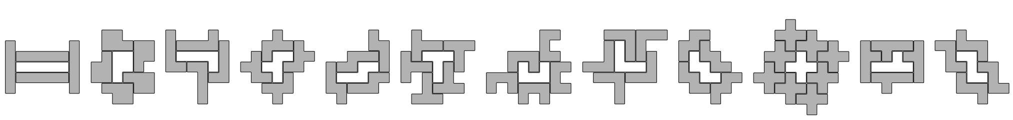
281. Both shapes have 8-fold symmetry, so we can save a factor of nearly 8 by placing the X in (say) the north-northwest octant. If X thereby falls on the diagonal, or in the middle column, we can insist that the Z is not flipped, by introducing a secondary item 's' as in answer 270. Furthermore, if X occurs in dead center — this is possible only for shape (i) — we use 'c' as in that answer to prohibit also any rotation of the W.

Thus we find (a) 10 packings, in 3.5 Gμ; (b) 7302 packings, in 353 Gμ; for instance

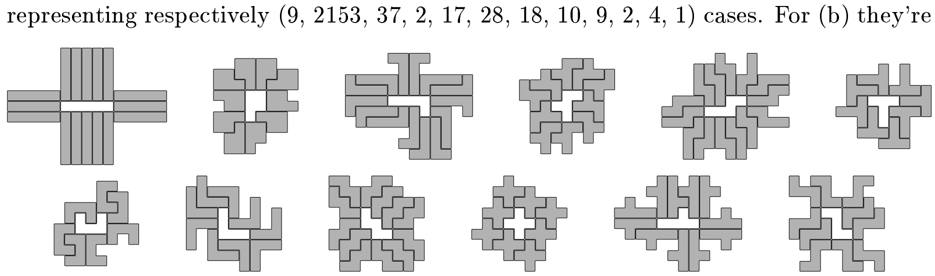
 ,  ;  .

It turns out that the monomino must appear in or next to a corner, as shown. [The first solution to shape (i) with monomino in the corner was sent to Martin Gardner by H. Hawkins in 1958. The first solution of the other type was published by J. A. Lindon in *Recreational Mathematics Magazine* #6 (December 1961), 22. Shape (ii) was introduced and solved much earlier, by G. Fuhlendorf in *The Problemist: Fairy Chess Supplement* **2**, 17 and 18 (April and June, 1936), problem 2410.]

282. It's easy to set up an exact cover problem in which the cells touching the polyomino are primary items, while other cells are secondary, and with options restricted to placements that contain at least one primary item. Postprocessing can then remove spurious solutions that contain holes. Typical answers for (a) are



representing respectively (9, 2153, 37, 2, 17, 28, 18, 10, 9, 2, 4, 1) cases. For (b) they're

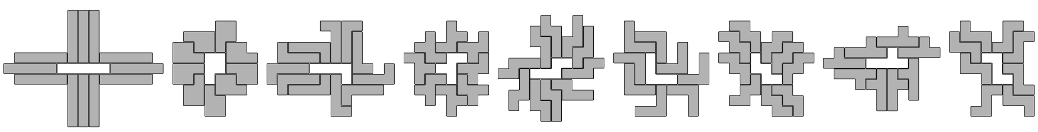


representing (16, 642, 1, 469, 551, 18, 24, 6, 4, 2, 162, 1). The total number of fences is respectively (3120, 1015033, 8660380, 284697, 1623023, 486, 150, 2914, 15707, 2, 456676, 2074), after weeding out respectively (0, 0, 16387236, 398495, 2503512, 665, 600, 11456, 0, 0, 449139, 5379) cases with holes. (See *MAA Focus* **36**, 3 (June/July

three-colorable
flipped
secondary item
Gardner
Hawkins
Lindon
Fuhlendorf

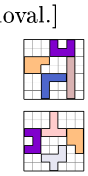
2016), 26; **36**, 4 (August/September 2016), 33.) Of course we can also make fences for one shape by using *other* shapes; for example, there's a beautiful way to fence a Z with 12 Ps, also a unique way to fence one pentomino with only *three* copies of another.

283. The small fences of answer 282(a) already meet this condition — except for the X, which has *no* tatami fence. The large fences for T and U in 365(b) are also good. But the other nine fences can no longer be as large:



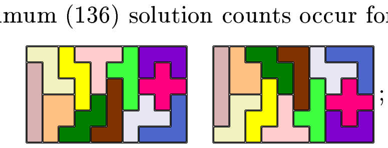
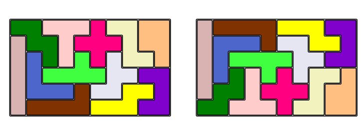
[The tatami condition can be incorporated into the exact cover problem by introducing a secondary item $/ij$ for each interior point ij. Add this item to every placement option that has a convex corner at ij and occupies either the cell to the northeast or the cell to the southwest. However, for this exercise it's best simply to apply the tatami condition directly to each ordinary solution, before postprocessing for hole-removal.]

284. This problem is readily solved with the "second death" algorithm of exercise 19, by letting the four designated piece names be the *only* primary items. The answers to both (a) and (b) are unique. [See M. Gardner, *Scientific American* **213**, 4 (October 1965), 96–102, for Golomb's conjectures about minimum blocking configurations on larger boards.]



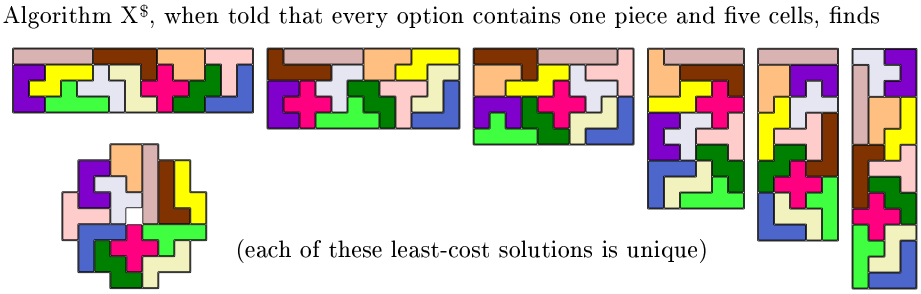
285. This exercise, with 3 × 30, 5 × 18, 6 × 15, and 9 × 10 rectangles, yields four increasingly difficult benchmarks for the exact cover problem, having respectively (46, 686628, 2567183, 10440433) solutions. Symmetry can be broken as in answer 270. The 3 × 30 case was first resolved by J. Haselgrove; the 9 × 10 packings were first enumerated by A. Wassermann and P. Östergård, independently. [See *New Scientist* **12** (1962), 260–261; J. Meeus, *JRM* **6** (1973), 215–220; and *FGbook* pages 455, 468–469.] Algorithm X needs (.006, 5.234, 15.576, 63.386) teramems to find them.

286. Two solutions are now equivalent only when related by 180° rotation. Thus there are $2 \cdot 2339/64 = 73.09375$ solutions per problem, on average. The minimum (42) and maximum (136) solution counts occur for the cases

(a)  ; (b)  .

[In *U.S. Patent 2900190* (1959, filed 1956), J. Pestieau remarked that these 64 problems would give his pentomino puzzle "unlimited life and utility."]

287. Let $c = (12, 11, \ldots, 1)$ for pieces (O, P, ..., Z) when assigning costs to each option. Algorithm X$^{\$}$, when told that every option contains one piece and five cells, finds

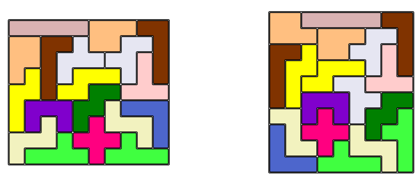


(each of these least-cost solutions is unique)

second death
Gardner
pentominoes, shortest games
benchmarks
Haselgrove
Wassermann
Östergård
Meeus
180° rotation
central symmetry, see 180° rotation
Patent
Pestieau

in respectively (1.5, 3.4, 3.3, 2.9, 3.2, 1.4, 1.1) Gμ. The corresponding times for Algorithm X are (3.7, 10.0, 16.4, 16.4, 10.0, 3.7, 2.0) Gμ. (However, we could reduce symmetry when applying Algorithm X, then calculate the values of four or eight different reflections or rotations whenever a solution is found; that would often be faster.)

288. When symmetry is removed efficiently, Algorithm X needs 63 Tμ to visit all of the essentially different solutions. But Algorithm X$^\$$ wins this competition, by discovering

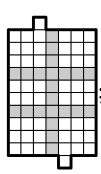
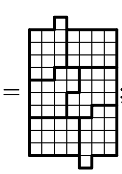
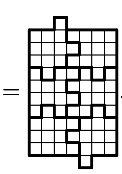


(which both are uniquely optimum) in 28.9 Tμ and 25.1 Tμ, respectively.

289. (a) One of the $8\cdot 2422\cdot 85\cdot 263\cdot 95\cdot 224\cdot 262\cdot 226\cdot 228\cdot 96\cdot 105\cdot 174$ solutions is shown in Fig. A–4. (It isn't hard to keep pentominoes of the same shape from touching.)

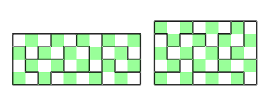
(b) Now there are $1472\cdot 5915\cdot 596\cdot 251\cdot 542\cdot 204\cdot 170\cdot 226\cdot 228\cdot 96\cdot 651\cdot 316$.

(c) The first seven columns left of the middle line can yield six 12-cell regions only by using all 72 cells. Thus the problem *factors* neatly into ten independent problems of the form (i). That problem has 7712 solutions with six connected regions; Algorithm X$^\$$ needs a search tree of only 622 nodes to determine that there are just 11 minimum-perimeter solutions. Three of them are symmetrical; and the nicest is shown in (ii). (And two of the solutions, such as (iii), *maximize* the total perimeter.)

(i) = ; (ii) = ; (iii) = .

Unfortunately (36) can't be expanded into the desired 720-cell shape based on (ii), because the scaled-up Q can't be packed. But the *alternative* form of (36) does lead to $16\cdot 2139\cdot 6\cdot 97\cdot 259\cdot 111\cdot 44\cdot 64\cdot 79\cdot 12\cdot 17\cdot 111$ solutions, such as the one in Fig. A–4.

290. There are no ways to fill 2×20; $66\cdot 4$ ways to fill 4×10; $84\cdot 4$ ways to fill 5×8. None of the solutions are symmetrical. [See R. K. Guy, *Nabla* **7** (1960), 99–101.]



291. The puzzles for January, April, September, and December (say) are equivalent; thus only $4\cdot 31 = 124$ puzzles need to be solvable, not 366. Only 53 of the 220 pentomino triples are unsuitable: First reject all 55 that include X, and all 10 that are subsets of $\{O, R, S, W, Z\}$; then restore $P\{O, Q, S, T, U, V, Y\}X$ and ORS, OSW, RSW; then reject RTZ and TWZ. Of the remaining 167 triples, PQV is by far the easiest: Every PQV puzzle has at least 1778 solutions! The hardest is QTX, which allows only about 33 solutions per day, on average. [This puzzle was designed by Marcel Gillen, © 2018, who made it with pentominoes R, U, W for the 2018 International Puzzle Party.]

292. Most of the hexominoes will have three black cells and three white cells, in any "checkering" of the board. However, eleven of them (shown as darker gray in the illustration) will have a two-to-four split. Thus the total number of black cells will always be an even number between 94 and 116, inclusive. But a 210-cell rectangle

symmetry
factors
min-cost solutions
max-cost sols
Guy
Gillen
black cells
checkering
parity

parity
exact cover
factoring
bipair of hexominoes

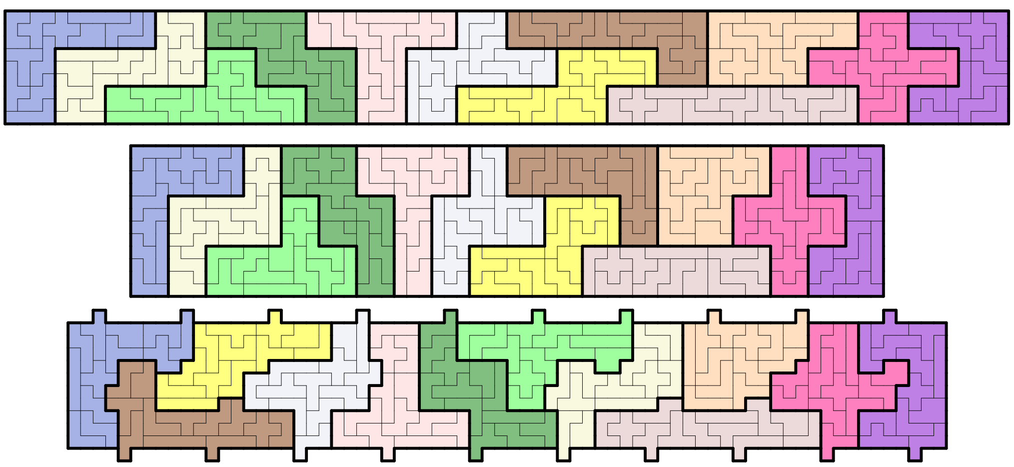
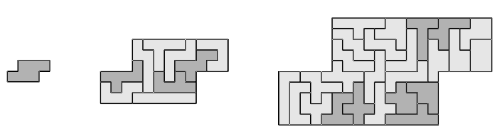


Fig. A–4. Pentominoes of pentominoes.

always contains exactly 105 black cells. [See *The Problemist: Fairy Chess Supplement* **2**, 9–10 (1934–1935), 92, 104–105; *Fairy Chess Review* **3**, 4–5 (1937), problem 2622.]

Benjamin's triangular shape, on the other hand, has $1+3+5+\cdots+19 = 10^2 = 100$ cells of one parity and $\binom{21}{2} - 10^2 = 110$ of the other. It can be packed with the 35 hexominoes in a huge number of ways, probably not feasible to count exactly.

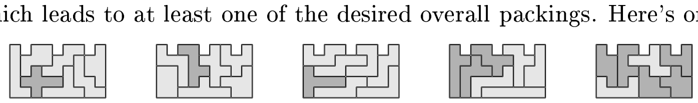
293. The parity considerations in answer 292 tell us that this is possible only for the "unbalanced" hexominoes, such as the one shown. And in fact, Algorithm X readily finds solutions for all eleven of those, too numerous to count. Here's an example:



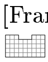
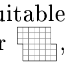
[See *Fairy Chess Review* **6** (April 1947) through **7** (June 1949), problems 7252, 7326, 7388, 7460, 7592, 7728, 7794, 7865, 7940, 7995, 8080. See also the similar problem 7092.]

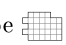
294. Each castle must contain an odd number of the eleven unbalanced hexominoes (see answer 292). Thus we can begin by finding all sets of seven hexominoes that can be packed into a castle: This amounts to solving $\binom{11}{1} + \binom{11}{3} + \binom{11}{5} + \binom{11}{7} = 968$ exact cover problems, one for each potential choice of unbalanced elements. Each of those problems is fairly easy; the 24 balanced hexominoes provide secondary items, while the castle cells and the chosen unbalanced elements are primary. In this way we obtain 39411 suitable sets of seven hexominoes, with only a moderate amount of computation.

That gives us *another* exact cover problem, having 35 items and 39411 options. This secondary problem turns out to have exactly 1201 solutions (found in just 115 Gμ), each of which leads to at least one of the desired overall packings. Here's one:

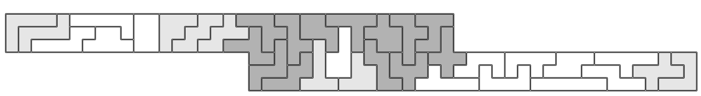


In this example, two of the hexominoes in the rightmost castle can be flipped vertically; and of course the entire contents of each castle can independently be flipped horizontally. Thus we get 64 packings from this particular partition of the hexominoes (or maybe $64 \cdot 5!$, by permuting the castles), but only two of them are "really" distinct. Taking multiplicities into account, there are 1803 "really" distinct packings altogether.

[Frans Hansson found the first way to pack the hexominoes into five equal shapes, using  as the container; see *Fairy Chess Review* **8** (1952–1953), problem 9442. His container admits 123189 suitable sets of seven, and 9298602 partitions into five suitable sets instead of only 1201. Even more packings are possible with the container , which has 202289 suitable sets and 3767481163 partitions!]

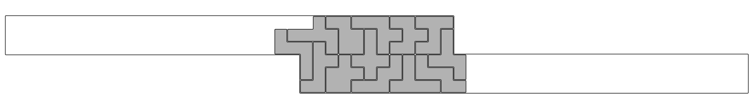
In 1965, M. J. Povah packed all of the hexominoes into containers of shape , using *seven* sets of *five*; see *The Games and Puzzles Journal* **2** (1996), 206.

295. By exercise 292, m must be odd, and less than 35. F. Hansson posed this question in *Fairy Chess Review* **7** (1950), problem 8556. He gave a solution for $m = 19$,

,

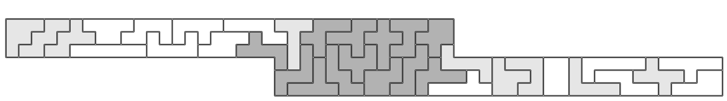
and claimed without proof that 19 is maximum. The 13 dark gray hexominoes in this diagram cannot be placed in either "arm"; so they must go in the center. (Medium gray indicates pieces that have parity restrictions in the arms.) Thus we cannot have $m \geq 25$.

When $m = 23$, there are 39 ways to place all of the hard hexominoes, such as

.

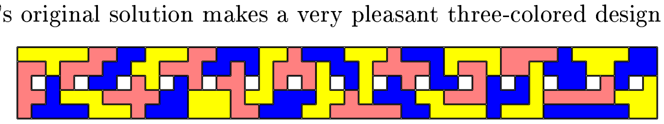
However, none of these is completable with the other 22; hence $m \leq 21$.

When $m = 21$, the hard hexominoes can be placed in 791792 ways, without creating a region whose size isn't a multiple of 6 and without creating more than one region that matches a particular hexomino. Those 791792 ways have 69507 essentially distinct "footprints" of occupied cells, and the vast majority of those footprints appear to be impossible to fill. But in 2016, George Sicherman found the remarkable packing

,

which not only solves $m = 21$, it yields solutions for $m = 19$, 17, 15, 11, 9, 7, 5, and 3 by simple modifications. Sicherman also found separate solutions for $m = 13$ and $m = 1$.

296. Stead's original solution makes a very pleasant three-colored design:



[See *Fairy Chess Review* **9** (1954), 2–4; also *FGbook*, pages 659–662.]

This problem is best solved via the techniques of dynamic programming (Section 7.7), *not* with Algorithm X, because numerous subproblems are equivalent.

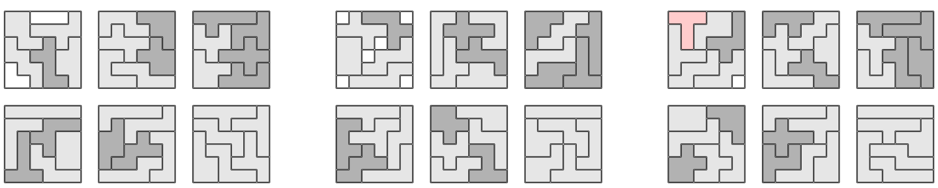
297. Yes — in fact, there are so many ways, further conditions ought to be imposed. Torbijn's original quest, to leave a hexomino-shaped "hole" in one square, turns out to have been impossible. But there's a nice alternative: We can add the two *trominoes*.

A. van de Wetering showed in 1991 that exactly 13710 sets of six hexominoes can fit into a single square. [See *JRM* **23** (1991), 304–305.] Similarly, exactly 34527 sets of five hexominoes will fit, when supplemented by two trominoes that both occupy two

Hansson
Povah
Hansson
Sicherman
strongly three-colorable
dynamic programming
trominoes
van de Wetering

black cells. So we're left with a secondary covering problem, with 35 primary items and 48237 options, as in answer 294. That problem has 163 solutions (found in 3 Tμ).

Another alternative, also suggested by van de Wetering, is to place six empty cells symmetrically. He also was able to add a monomino and one of the pentominoes: The secondary covering problems associated with pentominoes (O, P, ..., Z) turn out to have (94, 475, 1099, 0, 0, 2, 181, 522, 0, 0, 183, 0) solutions.



298. Make options for the pentominoes in cells xy for $0 \le x < 8$, $0 \le y < 10$ as in exercise 266, and also for the tetrominoes in cells xy for $1 \le x < 7$, $1 \le y < 9$. In the latter options include also items xy':0 for all cells xy *in* the tetromino, as well as xy':1 for all other cells xy *touching* the tetromino, where the items xy' for $0 \le x < 8$ and $0 \le y < 10$ are secondary. We can also assume that the center of the X pentomino lies in the upper left corner. There are 168 solutions, found after 1.5 Tμ of computation. (Another way to keep the tetrominoes from touching would be to introduce secondary items for the *vertices* of the grid. Such items are more difficult to implement, however, because they behave differently under the rotations of answer 266.)

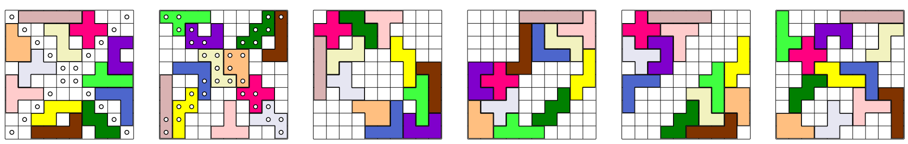
[Many problems that involve placing the tetrominoes and pentominoes together in a rectangle were explored by H. D. Benjamin and others in the *Fairy Chess Review*, beginning already with its predecessor *The Problemist: Fairy Chess Supplement* (1936), problem 2171. But this question seems to be new; it was inspired by Michael Keller's 15×18 pentomino + hexomino construction in *World Game Review* **9** (1989), 3. See also P. Torbijn's elegant 13×23 packing of all the n-ominoes for $1 \le n \le 6$, in *Cubism For Fun* **25**, part 1 (1990), 11.]

299. P. J. Torbijn and J. Meeus [*JRM* **32** (2003), 78–79] have exhibited solutions for rectangles of sizes 6×45, 9×30, 10×27, and 15×18; thus intuition suggests that enormously many solutions ought to be possible for this case too. But Peter Esser has surprisingly proved that *no* packing of the 35 hexominoes into a 5×54 rectangle will occupy all 114 of the border cells. Indeed, the pieces can individually occupy at most (6, 5, 5, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, $5+x_{24}$, $4+x_{25}$, $4+x_{26}$, $4+x_{27}$, $3+x_{28}$, $3+x_{29}$, $3+x_{30}$, $2+x_{31}$, $2+x_{32}$, $4+2x_{33}$, $3+2x_{34}$, $3+2x_{35}$) border cells, respectively, under an appropriate numbering of the pieces, where $x_k = 1$ only if piece k is in a corner. Since there are only four corners, we can occupy at most $6+5+\cdots+4+3+3+(1+2+2+2) = 114$ border cells — but only if $x_{33} = x_{34} = x_{35} = 1$. Unfortunately, those last three pieces (namely , ,) can't simultaneously occupy corners.

300. Make options as usual (exercise 266), but also include 100 new options 'xy Rx Cy' for $0 \le x, y < 10$. Then use Algorithm M, assigning multiplicity 4 to each Rx and Cy. Remove symmetry by confining X to the upper left corner, and by insisting that O be horizontal. (a) One of the 31 solutions (found in 12 Gμ) is shown below. (b) This case has 5347 solutions (found in 4.6 Tμ); and if we insist on filling also all cells just *above* the diagonals, the solution turns out to be *unique* (see below). (c) Instead of focusing on diagonals, Aad van de Wetering noticed that we can require the empty spaces to be *symmetrical*. For example, there are 1094 solutions (found in 19.2 Tμ)

black cells
parity
monomino
pentominoes
Benjamin
Keller
Torbijn
Torbijn
Meeus
Esser
multiplicity
van de Wetering
symmetrical

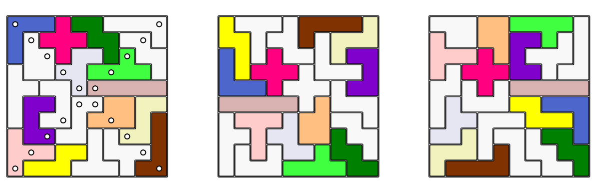
whose empty spaces are diagonally symmetric. Three of them, like the one shown here, are also rather close (92%) to being *centrally symmetric* (that is, under 180° rotation).



Three others, like the fourth example above, leave a 4×4 hole in the corner. Moreover, there are 98 solutions (found in 3.2 Tμ) whose empty spaces have 100% central symmetry. One of them has a large "moat" between two blocks of pentominoes; another has connected pentominoes, with holes of size at least 6.

Furthermore, van de Wetering reported that he had found "by accident" a solution where each of the four 5×5 quadrants of the 10×10 contained exactly three pentominoes. This additional stipulation is, indeed, easy to add to our MCC formulation: We omit options that cross quadrant boundaries, append a new item Q_t to each option in the tth quadrant, and give multiplicity 3 to each Q_t. It turns out that there actually are 1,124,352 inequivalent solutions(!), found by Algorithm M in 23 Tμ.

But van de Wetering also discovered a class of solutions that's even more interesting: He packed the empty spaces entirely with "shadow" pentominoes, all different!



To obtain such remarkable solutions, use primary items $\#xy$, $!xy$, $\#\mathrm{R}x$, and $\#\mathrm{C}y$ for $0 \le x, y < 10$, as well as O, P, ..., Z; use secondary items xy as well as O′, P′, ..., Z′. Items $\#\mathrm{R}x$ and $\#\mathrm{C}y$ have multiplicity 4. Specify two options for each pentomino placement, such as 'V !00 00:1 !01 01:1 !02 02:1 !10 10:1 !20 20:1' for V in the corner and 'V′ !00 00:0 !01 01:0 !02 02:0 !10 10:0 !20 20:0' for its shadow in that place. Also specify 200 further options, '$\#xy$ $\#\mathrm{R}x$ $\#\mathrm{C}y$ xy:0' and '$\#xy$ xy:1', for $0 \le x, y < 10$. Algorithm M with the nonsharp heuristic will then make intelligent choices. There are (amazingly) 357 solutions, found in 322 teramems with a search tree of 32 giganodes. The first solution above is one of six that cover exactly six cells of each main diagonal, answering a question that had been posed by Aad Thoen. The second solution is one of two for which all seven of the "unambiguously named pentominoes" T, U, V, W, X, Y, Z are among the shadows. The third solution is one of two that respects 5×5 quadrants. [*Note:* A similar question, but with *identical* polyominoes, was Erich Friedman's "problem of the month" in May 2007; see `www2.stetson.edu/~efriedma/mathmagic/0507.html`.]

301. (a) Algorithm M produces $4 \cdot 13330$ solutions when we specify the desired multiplicities for cell items. Symmetry under reflection can be removed by restricting, say, `W` to only 1/4 of its options.

(b) Consider the conflict graph on vertices `O`, `P`, ..., `Z`, defined by declaring pieces to be adjacent when they appear in the same cell. We can achieve $\le d$ levels if and only if we can color that graph with $\le d$ colors. The conflict graph for the given arrangement has the 4-clique {`Q`, `X`, `Y`, `Z`}; so it can't be 3-colored.

(c, d) A SAT solver such as Algorithm 7.2.2.2D quickly determines that exactly (587, 12550, 193) of the conflict graphs for the 13330 distinct solutions to (a) have

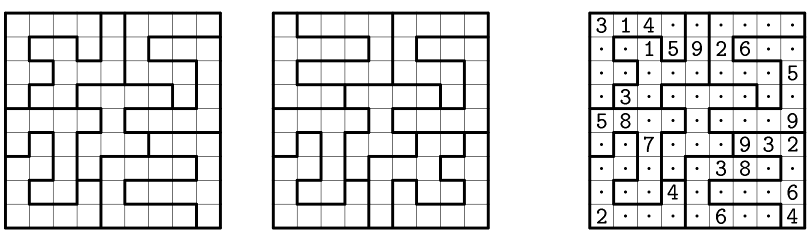
centrally symmetric
quadrants
van de Wetering
shadow
nonsharp heuristic
Thoen
Friedman
multiplicities
Symmetry under reflection
conflict graph
color that graph
clique
SAT solver

chromatic numbers (3, 4, 5). The first example below can be (uniquely) 3-colored `OVYZ` | `PRWX` | `QSTU`; the second example has the clique {`Q`, `R`, `S`, `W`, `Y`}.

OU	XY	UW	WZ	SZ
OUX	UXY	UXY	SWZ	SVW
OT	XY	RZ	SZ	VW
ORT	RTY	RTV	PSV	PQV
OT	QR	PQ	PQ	PQ

QY	QR	TU	TX	TU
RSY	QRY	RUX	UTX	UVX
SY	QW	RW	TX	VZ
SWY	QSW	PVZ	PVZ	PVZ
OW	OS	OZ	OP	OP

302. (a) There are 94. (But 16 of them have interior "holes" and can't be used in (b).)
(b) The two solutions are related by rotating four of the pieces:



(c) Sixteen different jigsaw sudoku diagrams can be used. The first of them collaborates with π as shown above; the others probably do too. [Appendix E has the answer. This exercise was suggested by E. Timmermans, *Cubism For Fun* **85** (2011), 4–9.]

303. (a) Represent the tree as a sequence $a_0a_1\ldots a_{2n+1}$ of nested parentheses; then $a_1\ldots a_{2n}$ will represent the corresponding root-deleted forest, as in Algorithm 7.2.1.6P. The left boundary of the corresponding parallomino is obtained by mapping each '`(`' into N or E, according as it is immediately followed by '`(`' or '`)`'. The right boundary, similarly, maps each '`)`' into N or E according as it is immediately *preceded* by '`)`' or '`(`'. For example, the parallomino for forest 7.2.1.6–(2) is shown below with part (d).

(b) This series $wxy + w^2(xy^2 + x^2y) + w^3(xy^3 + 2x^2y^2 + x^3y) + \cdots$ can be written $wxyH(w, wx, wy)$, where $H(w,x,y) = 1/(1 - x - y - G(w,x,y))$ generates a sequence of "atoms" corresponding to places x, y, G where the juxtaposed boundary paths have the respective forms $^{\mathtt{E}}_{\mathtt{E}}$, $^{\mathtt{N}}_{\mathtt{N}}$, or $^{\mathtt{N}}_{\mathtt{E}}\langle\text{inner}\rangle^{\mathtt{E}}_{\mathtt{N}}$. The area is thereby computed by diagonals between corresponding boundary points. (In the example from (a), the area is 1+1+1+1+2+2+2+2+2+2+2+2+2+2+1+1; there's an "outer" G, whose H is $xyxyGy$, and an "inner" G, whose H is $xyyxyxxyy$.) Thus we can write G as a continued fraction,

$$G(w,x,y) = wxy/(1-x-y-wxy/(1-wx-wy-w^3xy/(1-w^2x-w^2y-w^5xy/(\cdots)))).$$

[A completely different form is also possible, namely $G(w,x,y) = x\frac{J_1(w,x,y)}{J_0(w,x,y)}$, where

$$J_0(w,x,y) = \sum_{n=0}^{\infty} \frac{(-1)^n y^n w^{n(n+1)/2}}{(1-w)(1-w^2)\ldots(1-w^n)(1-xw)(1-xw^2)\ldots(1-xw^n)};$$

$$J_1(w,x,y) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} y^n w^{n(n+1)/2}}{(1-w)(1-w^2)\ldots(1-w^{n-1})(1-xw)(1-xw^2)\ldots(1-xw^n)}.$$

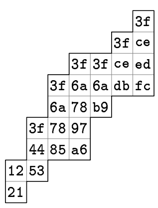
This form, derived via *horizontal* slices, disguises the symmetry between x and y.]

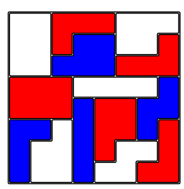
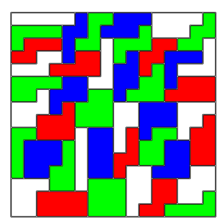
(c) Let $G(w,z) = G(w,z,z)$. We want $[z^n]\,G'(1,z)$, where differentiation is with respect to the first parameter. From the formulas in (b) we know that $G(1,z) =$

Timmermans
nested parentheses
forest
continued fraction
Bessel functions, gen'lized

$z(C(z)-1)$, where $C(z)=\bigl(1-\sqrt{1-4z}\bigr)/(2z)$ generates the Catalan numbers. Partial derivatives $\partial/\partial w$ and $\partial/\partial z$ then give $G'(1,z)=z^2/(1-4z)$ and $G_\prime(1,z)=1/\sqrt{1-4z}-1$.

(d) This problem has four symmetries, because we can reflect about either diagonal. When $n=5$, Algorithm X finds 801×4 solutions, of which 129×4 satisfy the tatami condition, and 16×4 are strongly three-colorable. (The tatami condition is easily enforced via secondary items in this case, because we need only stipulate that the upper right corner of one parallomino doesn't match the lower left corner of another.) When $n=6$ there are oodles and oodles of solutions. All of the trees/parallominoes thereby appear together in an attractive compact pattern.



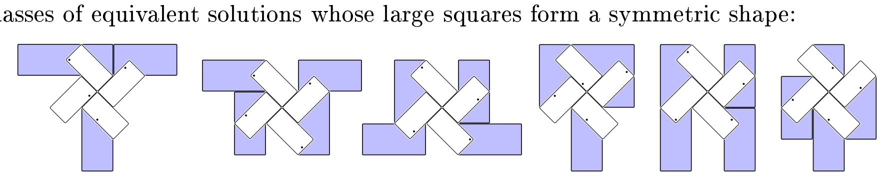
[*References:* J. Levine, *Scripta Mathematica* **24** (1959), 335–338; D. A. Klarner and R. L. Rivest, *Discrete Math.* **8** (1974), 31–40; E. A. Bender, *Discrete Math.* **8** (1974), 219–226; I. P. Goulden and D. M. Jackson, *Combinatorial Enumeration* (New York: Wiley, 1983), exercise 5.5.2; M.-P. Delest and G. Viennot, *Theoretical Comp. Sci.* **34** (1984), 169–206; W.-J. Woan, L. Shapiro, and D. G. Rogers, *AMM* **104** (1997), 926–931; P. Flajolet and R. Sedgewick, *Analytic Combinatorics* (2009), 660–662.]

304. E. D. Demaine and M. L. Demaine [*Graphs and Combinatorics* **23** (2007), Supplement, 195–208] show the NP-completeness also of several other related problems, such as to exactly pack given boxes of sizes $\{1\times x_1,\ \ldots,\ 1\times x_n\}$ into a given rectangle.

305. A scheme of "even/odd coordinates" (see exercise 145 and answer 133) works beautifully to represent the space occupied by a windmill domino: Encode the large square in row i and column j by the ordered pair $(2i{+}1)(2j{+}1)$; encode the small "tilted" square that overlaps two adjacent large squares by the midpoint between them. Then, for example, '15' is the large square in row 0 and column 2; '25' is the small tilted square whose top and bottom halves are the bottom and top quarters of 15 and 35. Large squares have area 4; small tilted squares have area 2; the encoding of each square specifies the coordinates of its center point. The relevant coordinates xy in an $m\times n$ box satisfy $0<x<2n$ and $0<y<2m$, where x and y are integers that aren't both even.

Therefore the possible placements of the leftmost windmill domino are either $\{13,15,12,23\}+(2k,2l)$, $\{33,53,23,32\}+(2k,2l)$, $\{33,31,34,23\}+(2k,2l)$, or $\{31,11,41,32\}+(2k,2l)$, where k and l are nonnegative integers.

(a) Here it suffices to use a 5×5 box, and to require that the small squares of each option are either `{34,45}`, `{47,56}`, `{76,65}`, or `{63,54}`. Each piece has exactly four such options; for example, if we call the leftmost piece '`0`', its options are '`0 35 37 34 45`', '`0 57 77 47 56`', '`0 53 33 63 54`', '`0 75 73 76 65`'. The problem has $183\cdot 4$ solutions, in groups of four that are related by 90° rotation. Here are six of the eight classes of equivalent solutions whose large squares form a symmetric shape:

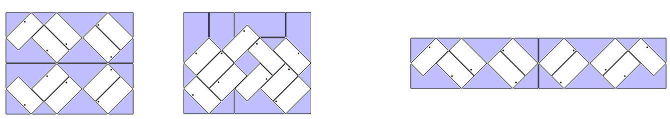


Catalan numbers
tatami
strongly three-colorable
secondary items
geek art
Levine
Klarner
Rivest
Bender
Goulden
Jackson
Delest
Viennot
Woan
Shapiro
Rogers
Flajolet
Sedgewick
Demaine
Demaine
even/odd coordinates
base placements

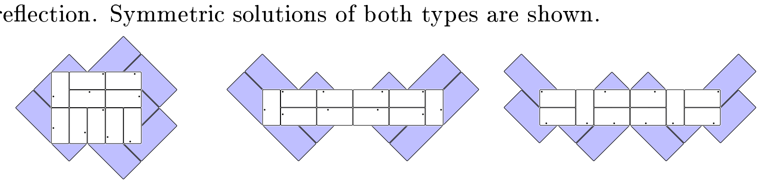
Nonsharp branching
multiplicity

(b) Algorithm X quickly finds $501484 = 4\cdot 2 + 125369\cdot 4$ solutions, including four classes that are symmetric under reflection and 125369 unsymmetric classes. One of the symmetric examples is shown below; also one of the 164 asymmetric classes whose small squares do at least form a symmetric shape.

(c) The $288 = 4\cdot 2 + 70\cdot 4$ solutions include four symmetric classes (like the one shown) and 70 that have no symmetry.



(d) We can set this up as a 7×7 problem in which the small squares form a rectangle whose corners are $\{\texttt{47}, \texttt{74}, \texttt{8b}, \texttt{b8}\}$. It has $2696\cdot 2$ solutions, all asymmetric; $95\cdot 2$ of them fit in a 5×5 box, and $3\cdot 2$ of them have large squares that form the symmetric shape shown. (e) Now there are two possibilities: We might have an 8×8 box, with small squares in the rectangle whose corners are $\{\texttt{34}, \texttt{43}, \texttt{cd}, \texttt{dc}\}$; or we might have a 9×9 box, with small squares confined to the rectangle $\{\texttt{45}, \texttt{54}, \texttt{de}, \texttt{ed}\}$. The first case has $69120 = 4\cdot 2 + 17278\cdot 4$ solutions, four with reflective symmetry; the second case has a whopping $157398 = 75\cdot 2 + 39312\cdot 4$ solutions, with 75 classes unchanged under reflection. Symmetric solutions of both types are shown.

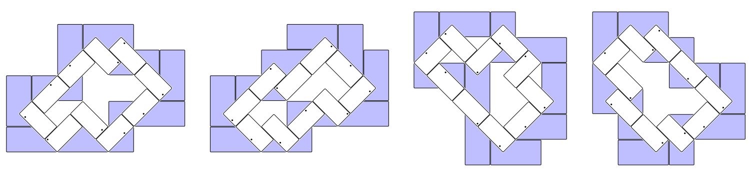


306. Introduce items `0` to `9` and xy as in the previous answer, as well as `p`xy and `#`xy; again x and y aren't both even, and $0 < x < 2n$, $0 < y < 2m$. Here `p`xy and `#`xy are primary, but the xy items are secondary. Options of the first kind, like '`0 p35 35`:1 `p37 37`:1 `p34 34`:1 `p45 45`:1', specify placement of a piece. Options of the second kind, '`p`xy xy:0', allow square xy to be empty. Options of the third kind, either '`#`xy xy:0' or '`#`xy xy:1 $(x{-}2)y$:a $(x{+}2)y$:b $x(y{-}2)$:c $x(y{+}2)$:d' for binary variables a, b, c, d with $a+b+c+d=2$, and where both x and y are odd, enforce the snake condition for large squares. Options of the fourth kind, either '`#`xy xy:0' or '`#`xy xy:1 $(x{-}1)(y{-}1)$:a $(x{-}1)(y{+}1)$:b $(x{+}1)(y{-}1)$:c $(x{+}1)(y{+}1)$:d' and where $x+y$ is odd, enforce the snake condition for small squares. Nonsharp branching (exercise 10) should be used.

Those options unfortunately produce a huge number of spurious solutions containing 4-cycles. One can rule out the 4-cycle whose large squares have a given $x'y'$ as midpoint by using Algorithm M and introducing a new primary item `#`$x'y'$ whose multiplicity is $[0\,..\,3]$. (Notice that x' and y' are both even.) This primary item is appended to every option of type 3 that begins with '`#`xy xy:1', where xy is one of the four squares touching point $x'y'$. The 4-cycles of small squares can be ruled out similarly, with new primary items `#`xy`!`, where $x+y$ is even.

Every snake-in-the-box cycle of 20 large squares will fit into a box of size 3×9, 4×8, 5×7, or 6×6; and Algorithm M finds respectively $(0, 0, 9\cdot 4, 8\cdot 8)$ solutions in those four cases. Six of the eight 6×6 equivalence classes are, however, spurious solutions, because their small squares form an 8-cycle and a 12-cycle instead of a single 20-cycle. Thus there are eleven essentially different solutions. Two of each size are shown below. [The middle two examples show two of the large squares touching at a corner. The definition of snake-in-the-box cycles allows this to happen; but five of the eleven solutions don't have this "defect." See *Cubism For Fun* **41** (October 1996), 30–32.]

Factoring
one-sided versions
bipair
convex polygons

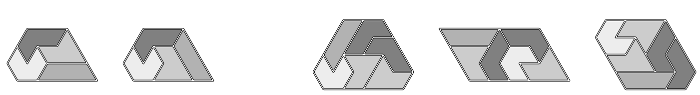


307. "Factoring" with the residues $(i-j) \bmod 3$ and $(i+j) \bmod 3$, we see that the domino must go into adjacent cells with $(i-j) \bmod 3 \neq 1$ and $(i+j) \bmod 3 \neq 2$. That means either $\{(3i,3j),(3i,3j+1)\}$ or $\{(3i+1,3j+2),(3i+2,3j+2)\}$. Conversely, it's easy to insert straight trominoes after placing a domino into any of those cell pairs.

308. (a) Each shape now has integer pairs of the forms (x,y) and $(x,y)'$. One elementary transformation, which rotates by 60°, takes $(x,y) \mapsto (x+y, x_{\max}-x)'$ and $(x,y)' \mapsto (x+y+1, x_{\max}-x)$; its result should be adjusted afterwards so that the coordinates are as small as possible while remaining nonnegative. The other elementary transformation, which is a reflection, simply takes $(x,y) \mapsto (y,x)$ and $(x,y)' \mapsto (y,x)'$.

For convenience, let's write just xy for (x,y). One tetriamond is the triangle of size 2, $\{00, 01, 10, 00'\}$. It has two base placements; the other one is $\{01', 10', 11', 11\}$. Another tetriamond is "straight," $\{00, 00', 10, 10'\}$, and it has six base placements. (Three of them, such as $\{00, 00', 01, 01'\}$, involve reflection; hence that tetriamond has two one-sided versions.) The remaining tetriamond is "bent," $\{00', 01, 10, 10'\}$, a hexagon minus a diamond. Its six base placements are all obtained by rotation.

(b) Four of the 20-iamonds are convex, namely those parameterized by $(6,4,0,0)$, $(10,10,1,0)$, $(4,2,1,0)$, and $(5,5,2,0)$ in the notation of exercise 143. But only $(4,2,1,0)$ can be packed with the four pentiamonds — in fact in two ways, differing by a bipair.



(c) The convex 30-iamonds $(15,15,1,0)$ and $(7,7,1,1)$ cannot be packed. But $(4,2,1,1)$, $(5,5,3,0)$, $(3,3,3,1)$ have respectively 3, 1, and 4 distinct solutions.

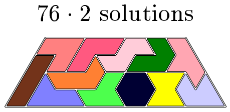
309. (a) (A, ..., L) have respectively (6, 3, 6, 1, 6, 6, 12, 12, 6, 12, 12, 12) placements.

(The hexiamonds have also been given *descriptive* names: A = lobster (or heart); B = butterfly (or spool); C = chevron (or bat); D = hexagon; E = crown (or boat); F = snake (or wave); G = hook (or shoe); H = signpost (or pistol or airplane); I = bar (or rhomboid); J = crook (or club or ladle); K = yacht (or steps); L = sphinx (or funnel).)

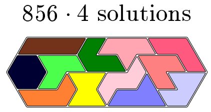
(b) Hexiamonds K and L are special, because they contain four triangles of one kind (Δ or ∇) and two of the other (∇ or Δ). The other hexiamonds are balanced, with three of each kind.

Eleven convex polygons are 72-iamonds, by exercise 143. Those with height less than 4, namely $(36,36,1,0)$, $(19,17,0,0)$, $(18,18,2,0)$, and $(12,12,3,0)$, are unsolvable. So is $(9,3,0,0)$, which is out of balance by 6. The other six are solvable; for example,

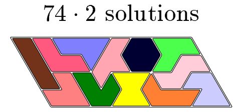
$(11,7,0,0)$
$76 \cdot 2$ solutions



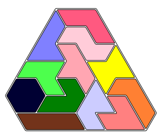
$(8,8,2,2)$
$856 \cdot 4$ solutions



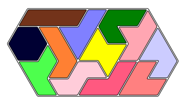
$(9,9,4,0)$
$74 \cdot 2$ solutions



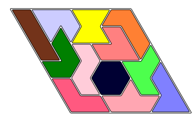
$(6,2,2,1)$
$5885 \cdot 2$ solutions



$(6,6,3,2)$
$5916 \cdot 2$ solutions



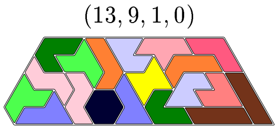
$(6,6,6,0)$
$156 \cdot 4$ solutions



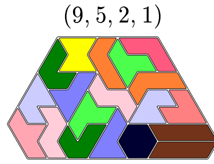
The shape $(6,2,2,1)$ is out of balance by 4. Consequently we can restrict K and L to about half of the positions where they would otherwise fit. The running time to find all solutions (without removing symmetry) thereby decreases, from 168 Gμ to 135 Gμ; thus the parity theory helps here, but not as much as might be expected.

What about the one-sided hexiamonds (with "flipped" versions of F through L, making 19 in all)? There are six convex polygons made up of $6 \cdot 19 = 114$ triangles, and again the small-height ones $(57,57,1,0)$, $(28,28,1,1)$, $(19,19,3,0)$ are unsolvable. The case $(13,9,1,0)$ has 1,687,429 solutions (found by Algorithm X in 11 Tμ). Shape $(8,8,3,3)$ has 4,790,046 distinct solutions (103 Tμ); $(9,5,2,1)$ has 17,244,919 (98 Tμ).

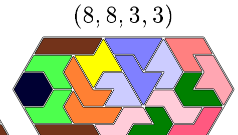
$(13,9,1,0)$



$(9,5,2,1)$



$(8,8,3,3)$



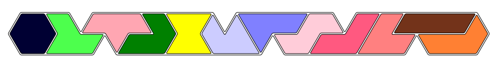
Historical notes: T. Scrutchin [*U.S. Patent 895114* (1908)] described an early puzzle based on assembling checkered polyiamonds of sizes 3–7 into a large equilateral triangle. The complete set of hexiamonds was perhaps first invented by Charles H. Lewis, who submitted a paper about them to the *American Mathematical Monthly* in April 1958. His paper wasn't judged worthy of publication; but a copy survives in the files of Martin Gardner, to whom he had sent a preprint. (He'd been inspired by Martin's exposition of polyominoes in December 1957.) Lewis named his pieces *hexotinoes*, and said that they belonged to the family of "polotinoes," which began with the monotino, the dotino, the trotino, three tetrotinoes, and four pentotinoes. He knew the parity rule, and he exhibited one of the ways to pack all 12 hexotinoes into a 6×6 rhombus.

Other people came up with similar ideas independently a few years later. It was T. H. O'Beirne who coined the names "polyiamond" and "hexiamond" — to the eternal dismay of language purists — first in letters to Richard Guy in 1960, then in his popular weekly columns in *New Scientist* [**12** (1961), 261, 316–317, 379, 706–707]. He introduced an intriguing problem about packing the one-sided hexiamonds into the rosette shape formed by 19 hexagons (12 surrounding 6 surrounding 1); see pages 452–455 of *FGbook* for details. Martin Gardner wrote about the subject in *Scientific American* **211**, 6 (December 1964), 123–130, and hexiamonds were soon sold as pleasing puzzles in Japan, Germany, the USA, and elsewhere. The 24 heptiamonds also have many aficionados, but they are beyond the scope of this book.

The earliest papers about hexiamonds considered mostly standard shapes like parallelograms, or shapes that are decidedly non-convex. Polygon $(6,2,2,1)$ above, the "diaper," may have first appeared as problem 130 in the Russian magazine *Nauka i Zhizn'* #6 (1969), 146; #7 (1969), 101; Michael Beeler enumerated its solutions in HAKMEM (M.I.T. A.I. Laboratory, 1972), Hack 112. Polygon $(6,6,3,2)$ has apparently not occurred previously in print, although it has more solutions than the others.

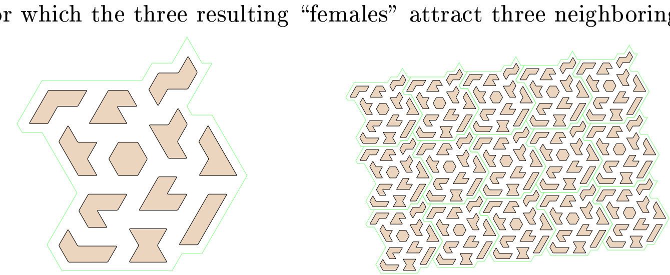
one-sided
Historical notes
Scrutchin
Patent
Lewis
Gardner
hexotinoes
O'Beirne
Guy
rosette
heptiamonds
Beeler
HAKMEM

310. The container holds $4m+2$ triangles; $m = 18$ doesn't work, so we need at least six empty cells. The author's favorite way constrains them to be well-separated "teeth":



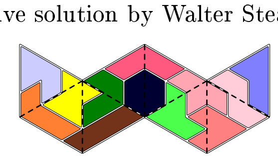
311. H. Postl found a nice proof that N must be at least 190: Replace hexiamonds A, G, K by the heptiamond that includes a hexagon. The twelve resulting pieces contain 75 triangles; enlarge them by appending quarter-size triangles around all the edges. This adds 91 trapezoids and 163 quarter-triangles. The latter must occupy at least $91 + (163-91)/3 = 115$ triangles, because we can't fill a triangle without using a trapezoid.

Exercise 7–137 explains how to obtain many generalized toruses that are composed of 95 rhombuses; so we might as well make the repeating pattern as square as possible by choosing $(a,b,c,d) = (11,-4,-1,9)$, as in the solution below. There are (astonishingly) 321530 such packings, each of which represents 24 different solutions when the heptiamonds revert to $\{\mathrm{A},\mathrm{G},\mathrm{K}\}$. The example shown is one of only 1768 solutions for which the three resulting "females" attract three neighboring "males."



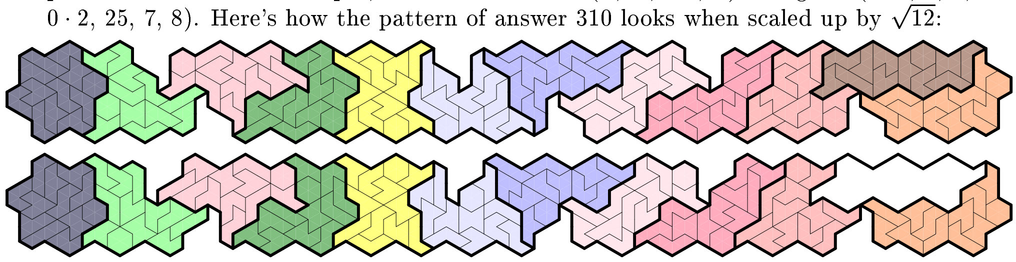
[The smallest region for *pentomino* wallpaper has 143 cells. See A. Thoen and A. van de Wetering, *Facets of Pentominoes* (2018), 95.]

312. Adrian Struyk wrapped the octahedron with hexiamonds, and showed it to Martin Gardner in 1964. An attractive solution by Walter Stead (1970, unpublished),

,

doesn't bend any piece in more than two places. (Incidentally, Thijs Notenboom showed in 1967 how to wrap the *icosahedron* with the four *pentiamonds*.)

313. The whirled versions of Pieces (A, ..., L) can be packed in respectively (13, $2\cdot2$, 10, $55\cdot6$, 19, $10\cdot2$, 9, 10, $10\cdot2$, 18, 6, 20) ways. But with flipped whirls, the one-sided pieces lead to different shapes, and the counts for (F, G, ..., L) change to ($6\cdot2$, 7, 8, $0\cdot2$, 25, 7, 8). Here's how the pattern of answer 310 looks when scaled up by $\sqrt{12}$:



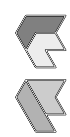
[The "whirl" in this exercise is the case $n = 3$ of an n-whirl, which has n^2+3 triangles for $n \ge 2$. In 1936, Maurits Escher visited the Alhambra and saw a pattern

author
Postl
heptiamond
generalized toruses
rhombuses
squarish triangles
sex
pentomino wallpaper
Thoen
Wetering
Struyk
Gardner
Stead
Notenboom
icosahedron
pentiamonds
one-sided pieces

related to the whirl tessellation. He was subsequently inspired to develop it much further; see *The World of M. C. Escher* (1971), plates 84 and 199.]

314. To make the same shape from two pairs $\{a,b\}$ and $\{c,d\}$ of polyiamonds (or polyominoes, etc.), choose an n-celled region A into which any solution will fit. Use four primary items $\{a,b,c,d\}$ and $6n$ secondary items 0α, 1α, $a\alpha$, $b\alpha$, $c\alpha$, $d\alpha$ for each cell α. For each placement '$a\ \alpha_1\ \ldots\ \alpha_s$' in A, and each of the 2^s sequences $q_1\ldots q_s$ with $q_k\in\{c,d\}$, create the option '$a\ 0\alpha_1\ q_1\alpha_1\ \ldots\ 0\alpha_s\ q_s\alpha_s\ a\beta_1\ \ldots\ a\beta_{n-s}$', where $\{\beta_1,\ldots,\beta_{n-s}\}=A\setminus\{\alpha_1,\ldots,\alpha_s\}$. Also create similar options for each placement of b, c, d, with the roles of $(0,a,c,d)$ replaced respectively by $(0,b,c,d)$, $(1,c,a,b)$, $(1,d,a,b)$.

Choose one of $\{a,b,c,d\}$ (one-sided if possible) and restrict it to a single placement. For the pentiamond problem, the author chose the piece a that includes a tetrahedron, and placed it in the center of a 70-iamond A. There are three separate cases, depending on which piece is called b; they yielded three huge exact cover problems, each of which had 15300 options of length 76 (thus total length 1.2 million). Yet Algorithm X solved each problem in at most 1.5 Gμ, including 0.3 Gμ just to load the data.

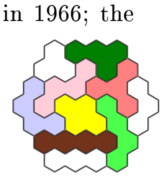


The answer, as Sicherman observed, is unique. [See Ed Pegg Jr.'s blog, `www.mathpuzzle.com/30November2008.html`. Solomon Golomb, in *Recreational Math. Mag.* #5 (October 1961), 3–12, had shown that the twelve pentominoes can be partitioned into three sets of four, each of which make congruent pairs.]

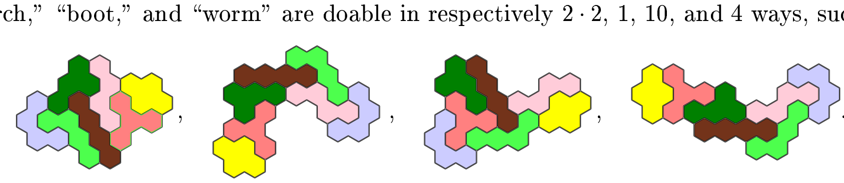
315. Proceed as in answer 308, but simply let $(x,y)\mapsto(x+y,x_{\max}-x)$; ignore $(x,y)'$.

[There's also an even/odd coordinate system for hexagons, with hexagon xy represented by $(2x+1,2y+1)$, and the edge between adjacent hexagons represented by their average. Then 60° rotation takes $(x,y)\mapsto(x+y-1,x_{\max}-x+1)$.]

316. There are $12290\cdot 12$ solutions, and it's not hard to find one by hand. (The first solutions were discovered independently by T. Marlow and E. Schwartz in 1966; the total number was found by K. Noshita in 1974.) The example shown here has the trihexes "maximally separated." [The seven tetrahexes pack the rhomboid $\{xy\mid 0\le x<4,\ 0\le y<7\}$ in $9\cdot 2$ ways, and the skew triangle $\{xy\mid 0\le x<7,\ x\le y<7\}$ in $5\cdot 2$ ways; but they can't pack the triangle $\{xy\mid 0\le x<7,\ 0\le y<7-x\}$.]



317. The scaled-up "bar," "wave," and "propeller" cannot be packed. But the "bee," "arch," "boot," and "worm" are doable in respectively $2\cdot 2$, 1, 10, and 4 ways, such as



[This problem was introduced by E. Schwartz in 1966 and independently by G. Edgar in 1967, who showed their solutions to Martin Gardner. Edgar pointed out that the rosettes can actually be placed in *two* ways — either rising or falling slightly from left to right when put together. The three one-sided tetrahexes therefore lead to *distinct* scaled-up shapes. Only one of those two is packable, for the boot and the worm; *both* are impossible for the wave. The slight tilting accounts for some of the remarkable properties of R. W. Gosper's "flowsnake" fractal; see M. Gardner, *Scientific Amer.* **235**, 6 (December 1976), 124–128, 133; A. Vince, *SIAM J. Discrete Math.* **6** (1993), 501–521.]

318. The "holes" in the T-grid correspond to vertices of the infinite triangular grid; and every hexagon of the T-grid is inside exactly one of the triangles made by those

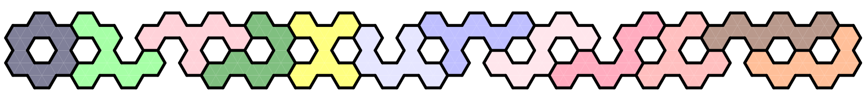
Escher
Alhambra
tessellation
polyforms
one-sided
author
Pegg Jr.
Golomb
pentominoes
even/odd coordinate
Marlow
Schwartz
Noshita
propeller
Schwartz
Edgar
Gardner
Gosper
flowsnake
fractal
Vince

hexiamond
represent
H-grid
kingwise connected
author
polytans
tangram
O'Beirne
lexicographically

vertices. More formally, we can let

$$\Delta\ (x,y) \leftrightarrow \text{hexagon } (x-y, x+2y+1);\quad \nabla\ (x,y)' \leftrightarrow \text{hexagon } (x-y, x+2y+2).$$

Adjacent triangles correspond to adjacent hexagons. The hexiamond hexahexes are

.

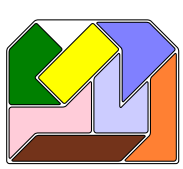
319. One way is to replace each square by a 3×3 array, representing ◺, ◸, ◹, ◿ by $\substack{\text{ooo}\\ \text{xoo}\\ \text{oxo}}$, $\substack{\text{oxo}\\ \text{xoo}\\ \text{ooo}}$, $\substack{\text{oxo}\\ \text{oox}\\ \text{ooo}}$, $\substack{\text{ooo}\\ \text{oox}\\ \text{oxo}}$. But it uses only 4 pixels out of 9. A more compact scheme is able to use 4 pixels out of every 8: We rotate the pieces by 45° and represent ◁, △, ▷, ▽ by $\substack{\text{xo}\\ \text{xo}}$, $\substack{\text{xx}\\ \text{oo}}$, $\substack{\text{ox}\\ \text{ox}}$, $\substack{\text{oo}\\ \text{xx}}$, separated by $\substack{\text{oo}\\ \text{oo}}$. For example, the 14 tetraboloes take the following forms:

```
·AA·    BB··    ·C···    ·DD·    E····    F·····    ·GG
·AA·    BB··    ·C···    D··D    E····    F··FF·    G··
A··A    ··BB    C····    D··D    ·EE··    ·FF··F    G··
A··A    ··BB    C··CC    ·DD·    ·EE··    ·····F    ·GG
                ·CC··                     ···EE               ·GG

··HH··    II··II··    JJ··JJ·    ·K··    ···L    ··MM··    NN····
HH··HH    ··II··II    ··JJ··J    ·K··    LL·L    ··MM··    ··NN··
····HH                ······J    K··K    ·LL·    MM··MM    ··NN··
                                 K··K    ·LL·              ····NN
                                 ·KK·
```

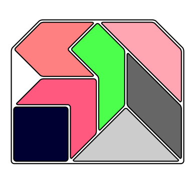
This scheme sets up a one-to-one correspondence between n-aboloes and $2n$-ominoes on the "H-grid," which is the set of all pixels (x,y) with $\lfloor x/2\rfloor + \lfloor y/2\rfloor$ even. (Each $2n$-omino is kingwise connected; it actually consists of n dominoes.)

Formally speaking, let's divide every square cell into four quarters, by cutting at the diagonals. Then every n-abolo occupies $2n$ quarters; and the (north, east, south, west) quarters of cell (x,y), in polyabolo coordinates, correspond respectively to cells $(2x-2y, 2x+2y) + \bigl((0,1),(1,1),(1,0),(0,0)\bigr)$ of the H-grid.

[After first seeing the H-grid versions of the tetraboloes, the author felt a foolish but irresistible urge to pack them into a 10×12 box, putting seven of them in the H-grid and the other seven in the complementary H-grid, leaving eight vacant pixels at the sides. This corresponds to putting the tetraboloes into two layers of a certain frame that's capable of holding 29 halfsquares. It turned out that there are $305\cdot 8$ ways to do this (found by Algorithm X in 10 Gμ). For example:



```
·DD··MMLLMM·
DGGDCBBMMLLN
DGGDCBBMMLLN
GDDBBCHKLNNJ
GCCBBCHKLNNJ
·GGCCFKHNKJE
·FFAAFKHNKJE
FAAFFHHKKEEJ
FAAIIHHIIEEJ
·IIAAIIEEJJ·
```



Nowadays, polyaboloes are often called "polytans," based on their connection to classical tangram puzzles from 18th-century China. T. H. O'Beirne introduced polyaboloes in *New Scientist* **13** (18 January 1962), 158–159.]

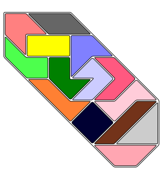
320. Every convex polyabolo can be characterized by six more-or-less independent parameters: We start with an $m\times n$ rectangle, then cut off triangles of sizes a, b, c, d at the lower left, lower right, upper right, and upper left corners, where $a+b\le n$, $b+c\le m$, $c+d\le n$, and $d+a\le m$. The number of halfsquares is $N = 2mn - a^2 - b^2 - c^2 - d^2$. To avoid duplicates, we require $m\le n$, and insist that (a,b,c,d) be lexicographically greater than or equal to (b,a,d,c), (c,d,a,b), (d,c,b,a). Furthermore, if $m=n$, this 4-tuple (a,b,c,d) should also be lexicographically greater than or equal to (a,d,c,b), (b,c,d,a), (c,b,a,d), (d,a,b,c).

The smallest positive area achievable with $m < n$ is $2m(n-m)$ halfsquares; and when $m = n$ the smallest is $2n-1$. Thus we must have $n \le (N+2)/2$, and it's feasible to backtrack through a finite number of cases.

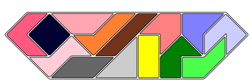
There are 63 solutions when $N = 56$. But most of them are unpackable, because of an important property noted by T. H. O'Beirne in 1962: Exactly five of the tetraboloes, namely $\{\mathrm{E}, \mathrm{G}, \mathrm{J}, \mathrm{K}, \mathrm{L}\}$, have an odd number of unmatched $\sqrt{2}$ sides in each direction. It follows that $a+c$ (and $b+d$) must be odd.

Just 10 of the 63 solutions pass this extra test. Two of those ten — $(1\times 29; 1, 1, 0, 0)$ and $(3\times 11; 3, 1, 0, 0)$ — don't work. But the other eight are achievable:

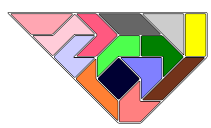
$(7\times 7; 5, 1, 4, 0)$ 1836 solutions



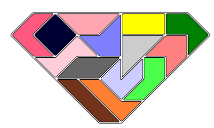
$(3\times 11; 2, 2, 1, 1)$ $236 \cdot 2$ solutions



$(5\times 9; 5, 3, 0, 0)$ 772 solutions

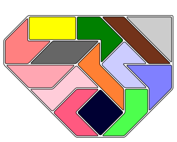


$(5\times 9; 4, 4, 1, 1)$ $747 \cdot 2$ solutions

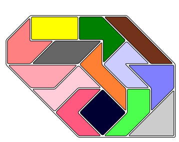


*

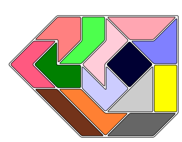
$(5\times 7; 3, 2, 0, 1)$ 5365 solutions



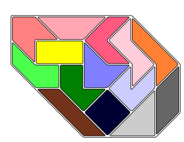
$(5\times 7; 3, 0, 2, 1)$ 5274 solutions



$(5\times 7; 3, 1, 0, 2)$ 4828 solutions



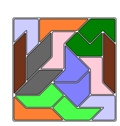
$(5\times 7; 3, 1, 2, 0)$ 4454 solutions



Most of them were cracked by E. S. Ainley in 1965; but H. Picciotto found '$*$' in 1989.

[This enumeration problem was first studied by F. T. Wang and C.-C. Hsiung, *AMM* **49** (1942), 596–599, who proved that there are 20 convex 16-aboloes. The totals for general N are OEIS sequence A245676, contributed by E. Fox-Epstein in 2014.]

321. [In a letter to Martin Gardner dated 12 March 1967, O'Beirne said that he now knew of 13 solutions, with help from several readers. "Are these the lot?" The answer is yes: The total is indeed 13. The solution shown here leads to three of the others, via tricky rearrangements.]



322. (i) We can reduce polysticks to (disconnected) polyominoes, by 3-fold enlargement: Let vertex ij of a square grid correspond to pixel $(3i)(3j)$; and let the line segment between adjacent vertices ij — $i'j'$ correspond to the two pixels between $(3i)(3j)$ and $(3i')(3j')$. Placements can intersect each other only at internal pixels where two parallel segments touch; we can prevent crossing by making such pixels secondary.

For example, to pack the 6×6 array in the example, we use the pixels xy for $0 \le x, y \le 18$, where x or y is a multiple of 3; item xy is secondary if 3 divides *both* x and y. One of the options for the T-shaped tetrastick is '04 05 07 08 16 26 36 46 56'; one of the options for the V-shaped tetrastick is '34 35 36 37 38 49 59 69 79 89'. The secondary item 36 ensures that these options won't both be chosen simultaneously.

(ii) Instead of scaling up by 3, we can scale up by 2, as in the even/odd coordinate system, by letting vertex ij correspond to pixel $(2i)(2j)$. Then segment ij — $i'j'$ corresponds to pixel $(i+i')(j+j')$; and the 6×6 example involves primary items xy for $0 \le x, y \le 12$ with $x+y$ odd, together with secondary items xy with x and y both even. The example T and V options in this scheme become '03 05 14 24 34' and '23 24 25 36 46 56'; now it's the secondary item 24 that keeps them from interacting.

O'Beirne
Ainley
Picciotto
Wang
Hsiung
OEIS
Fox-Epstein
Gardner
even/odd coordinate system

Scheme (i) can be used without change to answer 266. Scheme (ii) is almost twice as fast; but answer 266 must then be modified so that it never shifts by odd amounts. (Notice, for example, that the O and X tetrasticks each have only one base placement in scheme (ii), namely '01 10 12 21' and '12 21 23 32'. Shifting by 11 would change O to X and vice versa!) Thus, 90° rotation must be redefined as $(x, y) \mapsto (y, x_{\max} + (x_{\max} \,\&\, 1) - x)$, in the modified answer 266; also, δ_x and δ_y must be even.

[Polysticks were named and explored by B. R. Barwell in *JRM* **22** (1990), 165–175. They had actually been studied in the 1940s by H. D. Benjamin and T. R. Dawson in the 1940s, who already knew how to pack the pieces for $n \le 4$ into a 6×6 grid; see G. P. Jelliss, *JRM* **29** (1998), 140–142. See also *FGbook*, pages 457–472.]

90° rotation
Barwell
Benjamin
Dawson
Jelliss
spin
Owen
tetrominoes
dual
Keller

323. (a) For example, the vertices (m, n) of an ordinary square grid can be skewed to

$$(m,n)' \;=\; (m - (n \bmod 2)\epsilon,\, n - (m \bmod 2)\epsilon), \quad \text{where } \epsilon \text{ is the degree of skew.}$$

Notice that each square of the skewed grid has a clockwise or counterclockwise "spin."

(b) There's a nice way to represent each square as a 5-pixel cross, and each rhombus as a 3-pixel diagonal. For example, here are pixel equivalents of the tetraskews:

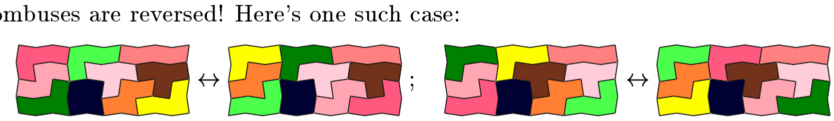
```
                     k · · · · · ·    · · l · · · ·    q · · Q ·    · S s · · · ·
· I · · i I · · i    · k · · · · ·    · l · · · · ·    · q Q Q Q    S S S s · · ·
I I I i I I I i ·    · K k · k K ·    l L l · · L ·    · Q q Q q    · S · S s · s
· I i · · I i · ·    K K K k K K K    L L L l L L L    Q Q Q q ·    · · S S S s ·
                     · K k · · K ·    · L · · l L ·    · Q q · ·    · · · S s · ·

· · t T · · t    · · u U · · u    v · · V v · ·    · · · Y · · ·    · · · Z · · z
· t T T T t ·    · u U U U u ·    · v V V V v ·    · · Y Y Y · ·    · · Z Z Z z ·
t · t T t · ·    u · · U u U ·    · · v V · V v    · Y y Y · Y ·    · Z z Z z · ·
· · · t · · ·    · · · · U U U    · · · · V V V    Y Y Y y Y Y Y    Z Z Z z · · ·
· · · · t · ·    · · · · · U ·    · · · · · V ·    · Y · · y Y ·    · Z · · z · ·
```

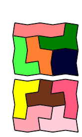
(Lowercase letters indicate the rhombuses here only for clarity; all pixels are either "in" or "out." The shapes fit together only when squares and rhombuses alternate properly.)

(c) The 4×10 frame in the example has 486 solutions; the analogous 5×8 frame has 572; these were first enumerated by Brendan Owen in 2000. There are 3648 ways to fit the pieces into a 2×21 frame, but 2×20 is too tight.

However, those counts can be divided by 2, because solutions to this problem come in pairs. Consider an arrangement of ten *unskewed* tetrominoes that involves one square, one straight, two skews, two tees, and four ells. It can be skewed in four ways, because we have two choices for which cells should be rhombuses and two choices for the spins; and it will be a valid skewed solution if and only if the resulting ten tetraskews are distinct. Changing the spins of a valid solution always gives another valid solution in which K $\leftrightarrow$ L, S $\leftrightarrow$ Z, U $\leftrightarrow$ V are swapped. Every solution therefore has a *dual*, which looks rather different but is well defined.

For example, the 486 solutions to the 4×10 rectangle problem correspond to exactly 226 unskewed arrangements that are distinct under reflections, 17 of which actually yield *two* dual pairs of skewed solutions, in which the roles of squares and rhombuses are reversed! Here's one such case:





[Michael Keller named the polyskews in 1993, and found a way to pack the tetraskews into two 4×5 frames, thus solving the 4×10 and 5×8 rectangles simultaneously. (See *World Game Review* **12** (1994), 12. That problem has just 24 solutions.) Generalizations to 3D await investigation.]

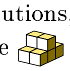
References: Polyforms live on many excellent and well-illustrated websites — notably `puzzler.sourceforge.net` by David Goodger; `www.polyforms.eu` by Peter Esser; `www.iread.it/lz/polymultiforms2.html` by Livio Zucca; `userpages.monmouth.com/~colonel/polycur.html` by George Sicherman; `www.recmath.org/PolyPages/` by Andrew Clarke; `abarothsworld.com/Puzzles.htm` by Abaroth. In particular, Abaroth's page "Squaring the Hexagon" discusses many ways to reduce one polyform to another. See also Ed Pegg Jr.'s chapter in *Tribute to a Mathemagician* (2005), 119–125.

324. The same ideas apply, but with three coordinates instead of two, and with the elementary transformations $(x,y,z) \mapsto (y, x_{\max} - x, z)$, $(x,y,z) \mapsto (y,z,x)$.

Pieces (1, 2, …, 7) have respectively (12, 24, 12, 12, 12, 12, 8) base placements, leading to $144 + 144 + 72 + 72 + 96 + 96 + 64$ options for the $3 \times 3 \times 3$ problem.

325. It's tempting, but wrong, to try to compute the Somap by considering only the 240 solutions that have the tee in a fixed position and the claw restricted; the pairwise semidistances between these special solutions will miss many of the actual adjacencies. To decide if $u \mathrel{—} v$, one must compare u to the 48 solutions equivalent to v.

(a) The strong Somap has vertex degrees $7^1\,6^7\,5^{19}\,4^{31}\,3^{59}\,2^{63}\,1^{45}\,0^{15}$; so an "average" solution has $(1\cdot 7 + 7\cdot 6 + \cdots + 15\cdot 0)/240 \approx 2.57$ strong neighbors. (The unique vertex of degree 7 has the level-by-level structure $\substack{333\\534\\552}\,\substack{114\\674\\652}\,\substack{174\\772\\662}$ from bottom to top.) This graph has two edges between $\substack{333\\534\\552}\,\substack{614\\114\\752}\,\substack{664\\762\\772}$ and $\substack{333\\534\\554}\,\substack{614\\114\\752}\,\substack{662\\762\\772}$, so it's actually a multigraph.

The full Somap has vertex degrees $21^2\,18^1\,16^9\,15^{13}\,14^{10}\,13^{16}\,12^{17}\,11^{12}\,10^{16}\,9^{28}\,8^{26}\,7^{25}\,6^{26}\,5^{16}\,4^{17}\,3^3\,2^1\,1^1\,0^1$, giving an average degree ≈ 9.14. (Its unique isolated vertex is $\substack{333\\432\\462}\,\substack{455\\466\\762}\,\substack{115\\715\\772}$, and its only pendant vertex is $\substack{333\\435\\222}\,\substack{755\\465\\462}\,\substack{771\\711\\466}$. Two other noteworthy solutions, $\substack{333\\436\\466}\,\substack{412\\452\\762}\,\substack{115\\755\\772}$ and $\substack{333\\436\\466}\,\substack{412\\455\\765}\,\substack{112\\752\\772}$, are the only ones that contain the two-piece substructure . There are 14 instances of repeated edges.)

(b) The Somap has just two components, namely the isolated vertex and the 239 others. The latter has just three bicomponents, namely the pendant vertex, its neighbor, and the 237 others. Its diameter is 8 (or 21, if we use the edge lengths 2 and 3).

The strong Somap has a much sparser and more intricate structure. Besides the 15 isolated vertices, there are 25 components of sizes $\{8 \times 2, 6 \times 3, 4, 3 \times 5, 2 \times 6, 7, 8, 11, 16, 118\}$. Using the algorithm of Section 7.4.1, the large component breaks down into nine bicomponents (one of size 2, seven of size 1, the other of size 109); the 16-vertex component breaks into seven; and so on, totalling 58 bicomponents altogether.

(One can also consider "physical" Somaps with 480 vertices, by saying that solutions are equivalent under rotation but not reflection. There are no repeated edges. The degree sequences are $7^2 6^{14} \ldots 0^{30}$ and $21^4 18^2 \ldots 0^2$, double what we had before.)

[The Somap was first constructed by R. K. Guy, J. H. Conway, and M. J. T. Guy, without computer help. It appears on pages 910–913 of Berlekamp, Conway, and Guy's *Winning Ways*, where all of the strong links are shown, and where enough other links are given to establish near-connectedness. Each vertex in that illustration has been given a code name; for example, the five special solutions mentioned in part (a) have code names B5f, R7d, LR7g, YR3a, and R3c, respectively.]

326. Let the cubie coordinates be $51z$, $41z$, $31z$, $32z$, $33z$, $23z$, $13z$, $14z$, $15z$, for $z \in \{1,2,3\}$. Replace matrix A of the exact cover problem by a simplified matrix A' having only items $(1,2,3,4,5,6,7,\mathrm{S})$, where S is the sum of all items xyz of A where $x \cdot y \cdot z$ is odd. Any solution to A yields a solution to A' with item sums $(1,1,1,1,1,1,1,10)$. But that's impossible, because the S counts of pieces (1, …, 7) are at most (1, 2, 2, 1, 1, 1, 1). [See the Martin Gardner reference in answer 333.]

Goodger
Esser
Zucca
Sicherman
Clarke
Abaroth
reduce one polyform to another
Pegg
multigraph
pendant vertex: of degree 1
repeated edges
diameter
Guy
Conway
Guy
Berlekamp
Conway
Guy
Gardner

327. (a) The solution counts, ignoring symmetry reduction, are: 4×5 corral (2), gorilla (2), smile (2), 3×6 corral (4), face (4), lobster (4), castle (6), bench (16), bed (24), doorway (28), piggybank (80), five-seat bench (104), piano (128), shift 2 (132), 4×4 coop (266), shift 1 (284), bathtub (316), shift 0 (408), grand piano (526), tower 4 (552), tower 3 (924), canal (1176), tower 2 (1266), couch (1438), tower 1 (1520), stepping stones (2718). So the 4×5 corral, gorilla, and smile are tied for hardest, while stepping stones are the easiest. (The bathtub, canal, bed, and doorway each have four symmetries; the couch, stepping stones, tower 4, shift 0, bench, 4×4 coop, castle, five-seat bench, piggybank, lobster, piano, gorilla, face, and smile each have two. To get the number of *essentially distinct* solutions, divide by the number of symmetries.)

(b) Notice that the stepping stones, canal, bed, and doorway appear also in (a). The solution counts are: W-wall (0), almost W-wall (12), bed (24), apartments 2 (28), doorway (28), clip (40), tunnel (52), zigzag wall 2 (52), zigzag wall 1 (92), underpass (132), chair (260), stile (328), fish (332), apartments 1 (488), goldfish (608), canal (1176), steps (2346), stepping stones (2718); hence “almost W-wall” is the hardest of the possible shapes. Notice that the stepping stones, chair, steps, and zigzag wall 2 each have two symmetries, while the others in Fig. 75(b) all have four. The $3\times3\times3$ cube, with its 48 symmetries, probably is the easiest possible shape to make from the Soma pieces.

[Piet Hein himself published the tower 1, shift 2, stile, and zigzag wall 1 in his original patent; he also included the bathtub, bed, canal, castle, chair, steps, stile, stepping stones, shift 1, five-seat bench, tunnel, W-wall, and both apartments in his booklet for Parker Brothers. Parker Brothers distributed four issues of *The SOMA® Addict* in 1970 and 1971, giving credit for new constructions to Noble Carlson (fish, lobster), Mrs. C. L. Hall (clip, underpass), Gerald Hill (towers 2–4), Craig Kenworthy (goldfish), John W. M. Morgan (face, gorilla, smile), Rick Murray (grand piano), and Dan Smiley (doorway, zigzag wall 2). Sivy Farhi published a booklet called *Somacubes* in 1977, containing the solutions to more than one hundred Soma cube problems including the bench, the couch, and the piggybank.]

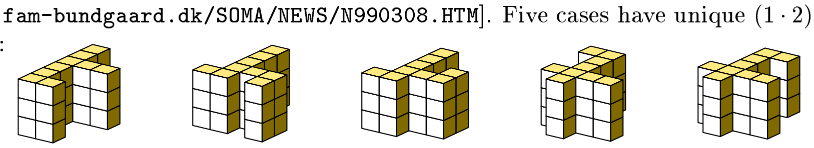
328. By eliminating symmetries, there are (a) 421 distinct cases with cubies omitted on both layers, and (b) 129 with cubies omitted on only one layer. All are possible, except in the one case where the omitted cubies disconnect a corner cell. The easiest of type (a) omits $\{\texttt{000}, \texttt{001}, \texttt{200}\}$ and has 3599 solutions; the hardest omits $\{\texttt{100}, \texttt{111}, \texttt{120}\}$ and has $45 \cdot 2$ solutions. The easiest of type (b) omits $\{\texttt{000}, \texttt{040}, \texttt{200}\}$ and has 3050 solutions; the hardest omits $\{\texttt{100}, \texttt{110}, \texttt{140}\}$ and has $45 \cdot 2$ solutions. (The two examples illustrated have $821 \cdot 2$ and $68 \cdot 4$ solutions. Early Soma solvers seem to have overlooked them!)

329. (a) The 60 distinct cases are all quite easy. The easiest has 3497 solutions and uses $\{\texttt{002}, \texttt{012}, \texttt{102}\}$ on the top level; the hardest has 268 solutions and uses $\{\texttt{002}, \texttt{112}, \texttt{202}\}$.

(b) Sixteen of the 60 possibilities are disconnected. Three of the others are also impossible — namely those that omit $\{\texttt{01}z, \texttt{13}z, \texttt{21}z\}$ or $\{\texttt{10}z, \texttt{11}z, \texttt{12}z\}$ or $\{\texttt{10}z, \texttt{11}z, \texttt{13}z\}$. The easiest has 3554 solutions and omits $\{\texttt{00}z, \texttt{01}z, \texttt{23}z\}$; the hardest of the possibles has only 8 solutions and omits $\{\texttt{00}z, \texttt{12}z, \texttt{13}z\}$.

(The two examples illustrated have $132 \cdot 2$ and $270 \cdot 2$ solutions.)

330. T. Bundgård and C. McFarren found in 1999 that all but 216 are realizable [`www.fam-bundgaard.dk/SOMA/NEWS/N990308.HTM`]. Five cases have unique $(1 \cdot 2)$ solutions:



symmetries
Hein
Parker Brothers
Carlson
Hall
Hill
Kenworthy
Morgan
Murray
Smiley
Farhi
symmetries
Bundgård
McFarren

331. Every polycube has a minimum enclosing box for which it touches all six faces. If those box dimensions $a\times b\times c$ aren't too large, we can generate such polycubes uniformly at random in a simple way: First choose 27 of the abc possible cubies; try again if that choice doesn't touch all faces; otherwise try again if that choice isn't connected.

For example, when $a=b=c=4$, about 99.98% of all choices will touch all faces, and about 0.1% of those will be connected. This means that about $.001\binom{64}{27}\approx 8\times 10^{14}$ of the 27-cubie polycubes have a $4\times 4\times 4$ bounding box. Of these, about 5.8% can be built with the seven Soma pieces.

But most of the relevant polycubes have a larger bounding box; and in such cases the chance of solvability goes down. For example, $\approx 6.2\times 10^{18}$ cases have bounding box $4\times 5\times 5$; $\approx 3.3\times 10^{18}$ cases have bounding box $3\times 5\times 7$; $\approx 1.5\times 10^{17}$ cases have bounding box $2\times 7\times 7$; and only 1% or so of those cases are solvable.

Section 7.2.3 will discuss the enumeration of polycubes by their size.

332. Each interior position of the penthouse and pyramid that might or might not be occupied can be treated as a secondary item in the corresponding exact cover problem. We obtain $10\cdot 2$ solutions for the staircase; $(223, 286)\cdot 8$ solutions for the penthouse with hole at the (bottom, middle); and $32\cdot 2$ solutions for the pyramid, of which $2\cdot 2$ have all three holes on the diagonal and $3\cdot 2$ have no adjacent holes.

333. A full simulation of gravity would be quite complex, because pieces can be prevented from tipping with the help of their neighbors above and/or at their side. If we assume a reasonable coefficient of friction and an auxiliary weight at the top, it suffices to define stability by saying that a piece is stable if and only if at least one of its cubies is immediately above either the floor or a stable piece.

The given shapes can be packed in respectively $202\cdot 2$, $21\cdot 2$, $270\cdot 2$, $223\cdot 8$, and $122\cdot 2$ ways, of which $202\cdot 2$, $8\cdot 2$, $53\cdot 2$, $1\cdot 8$, and $6\cdot 2$ are stable. Going from the bottom level to the top, the layers $\substack{4\cdot\cdot7\\ \cdot\cdot\cdot\cdot\\ 3\cdot\cdot6}\ \substack{4477\\ 5567\\ 3366}\ \substack{54\cdot\cdot\\ 5\cdot\cdot\cdot\\ 31\cdot\cdot}\ \substack{22\cdot\cdot\\ 21\cdot\cdot\\ 21\cdot\cdot}$ give a decently stable cot; a fragile vulture comes from $\substack{2\cdot\cdot\cdot\\ \cdot\cdot\cdot7\\ 3\cdot\cdot\cdot}\ \substack{2447\\ 3177\\ 3566}\ \substack{2244\\ 1156\\ 3556}$; a delicate mushroom comes from $\substack{\cdot\cdot\cdot\\ \cdot7\cdot\\ \cdot\cdot\cdot}\ \substack{572\\ 577\\ 466}\ \substack{552\\ 3\cdot6\\ 446}\ \substack{322\\ 311\\ 341}$; and a delicate cantilever from $\substack{\cdot\cdot\cdot\\ 222\\ \cdot\cdot\cdot}\ \substack{\cdot\cdot\cdot\\ 255\\ \cdot\cdot\cdot}\ \substack{\cdot5\cdot\\ \cdot5\cdot\\ \cdot7\cdot}\ \substack{634\\ 674\\ 177}\ \substack{333\\ 664\\ 114}$. The author's cherished set of Skjøde Skjern Soma pieces, made of rosewood and purchased in 1967, includes a small square base that nicely stabilizes both mushroom and cantilever. The vulture needs a book on top.

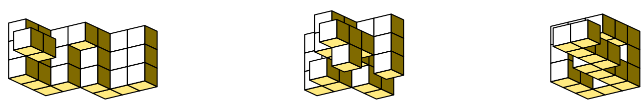
[The casserole and cot are due respectively to W. A. Kustes and J. W. M. Morgan. The mushroom, which is hollow, is the same as B. L. Schwartz's "penthouse," but turned upside down; John Conway noticed that it then has a unique stable solution. See Martin Gardner, *Knotted Doughnuts* (1986), Chapter 3.]

334. Infinitely many cubies lie behind a wall; but it suffices to consider only the hidden ones whose distance is at most $27-v$ from the v visible ones. For example, the W-wall has $v=25$, and the two invisible cubies are $\{332, 331\}$ if we use the coordinates of answer 326. We're allowed to use any of $\{241, 242, 251, 252, 331, 332, 421, 422, 521, 522\}$ at distance 1, and $\{341, 342, 351, 352, 431, 432, 531, 532, 621, 622\}$ at distance 2. (The stated projection doesn't have left-right symmetry.) The X-wall is similar, but it has $v=19$ and potentially (9, 7, 6, 3, 3, 2, 1) hidden cubies at distances 1 to 7 (omitting cases like 450, which is invisible at distance 2 but "below ground").

Using secondary items for the optional cubies, we must examine each solution to the exact cover problem and reject those that are disconnected or violate the gravity constraint of exercise 333. Those ground rules yield 282 solutions for the W-wall, 612 for the X-wall, and a whopping 1,130,634 for the cube itself. (These solutions fill respectively 33, 275, and 13842 different sets of cubies.) Here are examples of some of

secondary item
author
Skjøde Skjern
Knutsen, see Skjøde Skjern
Kustes
Morgan
Schwartz
Conway
Gardner

the more exotic shapes that are possible, as seen from behind and below:



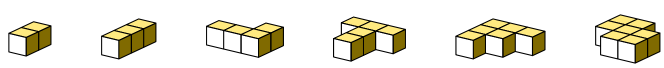
There also are ten surprising ways to make the cube façade if we allow hidden "underground" cubies: The remarkable construction $\begin{smallmatrix}\cdot\cdot\cdot\cdot\cdot\\ \cdot\cdot\cdot\cdot\cdot\\ \cdot\cdot\cdot55\end{smallmatrix}$ $\begin{smallmatrix}446\cdot\cdot\\ 7\cdot65\cdot\\ 2\cdot\cdot5\cdot\end{smallmatrix}$ $\begin{smallmatrix}744\cdot\cdot\\ 7766\cdot\\ 2\cdot\cdot\cdot\cdot\end{smallmatrix}$ $\begin{smallmatrix}333\cdot\cdot\\ 231\cdot\cdot\\ 211\cdot\cdot\end{smallmatrix}$ raises the entire cube one level *above* the floor, and is gravitationally stable, by exercise 333's criteria! Unfortunately, though, it falls apart — even with a heavy book on top.

[The false-front idea was pioneered by Jean Paul Francillon, whose construction of a fake W-wall was announced in *The SOMA® Addict* **2**, 1 (spring 1971).]

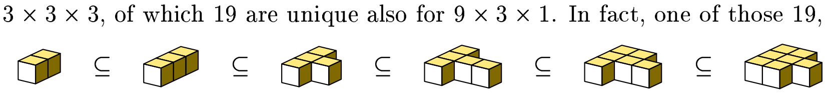
335. (a) Each of 13 solutions occurs in 48 equivalent arrangements. To remove the symmetry, place piece 7 horizontally, either (i) at the bottom or (ii) in the middle. In case (ii), add a secondary 's' item as in answer 268, and append 's' also to all placements of piece 6 that touch the bottom more than the top. Run time: 400 Kμ.

[This puzzle was number 3–39 in Hoffmann's *Puzzles Old and New* (1893). Another $3 \times 3 \times 3$ polycube dissection of historical importance, "Mikusinski's Cube," was described by Hugo Steinhaus in the 2nd edition of his *Mathematical Snapshots* (1950). That one consists of the ell and the two twist pieces of the Soma cube, plus the pentacubes B, C, and f of exercise 340; it has 24 symmetries and just two solutions.]

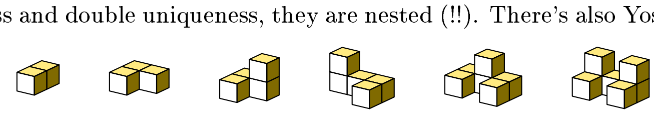
(b) Yes: Michael Reid, circa 1995, found the remarkable set



which also makes $9 \times 3 \times 1$ uniquely(!). George Sicherman carried out an exhaustive analysis of all relevant flat polyominoes in 2016, finding exactly 320 sets that are unique for $3 \times 3 \times 3$, of which 19 are unique also for $9 \times 3 \times 1$. In fact, one of those 19,



is the long-sought "Holy Grail" of $3 \times 3 \times 3$ cube decompositions: Its pieces not only have flatness and double uniqueness, they are nested (!!). There's also Yoshiya Shindo's



known as the "Neo Diabolical Cube" (1995); notice that it has 24 symmetries, not 48.

336. This piece can be modeled by a polycube with $20 + 20 + 27 + 3$ cubies, where we want to pack nine of them into a $9 \times 9 \times 9$ box. Divide that box into 540 primary cells (which must be filled) and 189 secondary cells (which will contain the 27 cubies of the simulated dowels). Answer 324 now yields an exact cover problem with 1536 options; and Algorithm X needs only 33 Mμ to discover 24 solutions, all equivalent by symmetry. (Or we could modify answer 324 so that all offsets have multiples of 3 in each coordinate; then there would be only 192 options, and the running time would go down to 8 Mμ.) One packing is $\begin{smallmatrix}122\\123\\443\end{smallmatrix}$ $\begin{smallmatrix}567\\163\\849\end{smallmatrix}$ $\begin{smallmatrix}557\\867\\899\end{smallmatrix}$, with dowels at $\begin{smallmatrix}010\\400\\030\end{smallmatrix}$ $\begin{smallmatrix}070\\529\\080\end{smallmatrix}$ $\begin{smallmatrix}000\\600\\000\end{smallmatrix}$.

One might be tempted to factor this problem, by first looking at all ways to pack nine solid bent trominoes into a $3 \times 3 \times 3$ box. That problem has 5328 solutions, found in about 5 Mμ; and after removing the 48 symmetries we're left with just 111 solutions, into which we can try to model the holes and dowels. But such a procedure is rather complicated, and it doesn't really save much time, if any.

gravitationally stable
Francillon
Hoffmann
Mikusinski's Cube
Steinhaus
pentacubes
Reid
Sicherman
Holy Grail
Shindo
Neo Diabolical Cube
factor
ell trominoes, see bent trominoes
bent trominoes
right trominoes, see bent trominoes
solid bent trominoes
L-cube puzzle

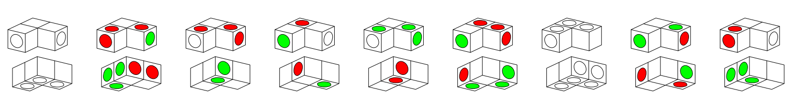
Ronald Kint-Bruynseels, who designed this remarkable puzzle, also found that it's possible to drill holes in the solid cubies, parallel to the other two, without destroying the uniqueness of the solution(!). [*Cubism For Fun* **75** (2008), 16–19; **77** (2008), 13–18.]

337. Let's use even/odd coordinates as in exercise 145, so that each final face has one coordinate in $\{0, 6\}$ and two coordinates in $\{1, 3, 5\}$. The first goal has red spots on faces 330, 105, 501, 015, 033, 051, 611, 615, 651, 655, 161, 165, 363, 561, 565, 116, 136, 156, 516, 536, 556. The other goal has green spots on 19 of those 21 faces; but 303 replaces 033 and 633 replaces 363. (For simplicity, we'll ignore alternative setups; there actually are sixteen ways to put spots on dice, not just two.)

Nine bent tricubes will pack a $3 \times 3 \times 3$ cube in 5328 ways. (They fall into 111 equivalence classes of size 48, under rotation and reflection; but that fact is irrelevant here.) Take any such solution and color its 54 external faces with the red solution. Then see if its pieces can be rearranged to give the green solution.

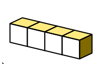
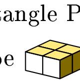
Notice that each bent tricube has fourteen square faces; but the two "inner" faces are never visible in the final assembly. That assembly will specify from 2 to 7 of the 12 potential faces, leaving 5 to 10 faces unconstrained. Altogether we'll have 21 faces specified red, 33 specified blank, and 54 still free.

It turns out that 371 of the 5328 red solutions can be rearranged into green solutions; in fact one case leads to 6048 different green solutions! And there are 52 combinations of red+green solutions that leave 18 faces unspecified, such as this:



We're free to put anything we like on those 18 faces — giving red or green spots that are false clues, and/or concealing a *third* pattern that the puzzler is challenged to achieve.

(The classic "Spots Puzzle" in Hoffmann's *Puzzles Old and New* (1893), No. 3–17, distributed by E. Wolff & Son's pencil company, assembled a single die from *straight* tricubes. Lavery's elegant "Twice Dice" was produced by Pentangle Puzzles in 1990.)

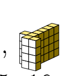
338. The straight tetracube  and the square tetracube , together with the size-4 Soma pieces in (39), make a complete set.

We can fix the tee's position in the twin towers, saving a factor of 32; and each of the resulting 40 solutions has just one twist with the tee. Hence there are five inequivalent solutions, and $5 \cdot 256$ altogether.

The double claw has $63 \cdot 6$ solutions. But the cannon, with $1 \cdot 4$ solutions, can be formed in essentially only one way. (*Hint:* Both twists are in the barrel.)

There are no solutions to 'up 3'. But 'up 4' and 'up 5' each have $218 \cdot 8$ solutions (related by turning them upside down). Gravitationally, four of those 218 are stable for 'up 5'; the stable solution for 'up 4' is unique, and unrelated to those four.

References: Jean Meeus, *JRM* **6** (1973), 257–265; Nob Yoshigahara, *Puzzle World No. 1* (San Jose: Ishi Press International, 1992), 36–38.

339. All but 48 are realizable. The unique "hardest" realizable case, , has $2 \cdot 2$ solutions. The "easiest" case is the $2 \times 4 \times 4$ cuboid, with $11120 = 695 \cdot 16$ solutions.

340. (a) A, B, C, D, E, F, a, b, c, d, e, f, j, k, l, . . . , z. (It's a little hard to see why reflection doesn't change piece 'l'. In fact, S. S. Besley once patented the pentacubes under the impression that there were 30 different kinds! See *U.S. Patent 3065970* (1962), where Figs. 22 and 23 illustrate the same piece in slight disguise.)

Kint-Bruynseels
even/odd coordinates
Spots Puzzle
Hoffmann
historical notes
Wolff
Twice Dice
Pentangle Puzzles
straight tetracube
square tetracube
Meeus
Yoshigahara
cuboid
Besley
Patent
Historical notes

Historical notes: R. J. French, in *Fairy Chess Review* **4** (1940), problem 3930, was first to show that there are 23 different pentacube shapes, if mirror images are considered to be identical. The full count of 29 was established somewhat later by F. Hansson and others [*Fairy Chess Review* **6** (1948), 141–142]; Hansson also counted the $35 + 77 = 112$ mirror-inequivalent hexacubes. Complete counts of hexacubes (166) and heptacubes (1023) were first established soon afterwards by J. Niemann, A. W. Baillie, and R. J. French [*Fairy Chess Review* **7** (1948), 8, 16, 48].

(b) The cuboids $1\times3\times20$, $1\times4\times15$, $1\times5\times12$, and $1\times6\times10$ have of course already been considered. The $2\times3\times10$ and $2\times5\times6$ cuboids can be handled by restricting X to the bottom upper left, and sometimes also restricting Z, as in answers 268 and 270; we obtain 12 solutions (in 350 Mμ) and 264 solutions (in 2.5 Gμ), respectively.

The $3\times4\times5$ cuboid is more difficult. Without symmetry-breaking, we obtain 3940×8 solutions in about 200 Gμ. To do better, notice that O can appear in four essentially different positions. With four separate runs we can find $5430/2 + 1348/4 + 716/2 + 2120/4 = 3940$ solutions, in $35.7 + 10.0 + 4.5 + 7.1 \approx 57$ Gμ.

[The fact that solid pentominoes will fill these cuboids was first demonstrated by D. Nixon and F. Hansson, *Fairy Chess Review* **6** (1948), problem 7560 and page 142. Exact enumeration was first performed by C. J. Bouwkamp in 1967; see *J. Combinatorial Theory* **7** (1969), 278–280, and *Indagationes Math.* **81** (1978), 177–186.]

(c) Almost *any* subset of 25 pentacubes can probably do the job. But a particularly nice one is obtained if we simply omit o, q, s, and y, namely those that don't fit in a $3\times3\times3$ box. R. K. Guy proposed this subset in *Nabla* **7** (1960), 150, although he wasn't able to pack a $5\times5\times5$ at that time.

The same idea occurred independently to J. E. Dorie, who trademarked the name "Dorian cube" [*U.S. Trademark 1,041,392* (1976)].

An amusing way to form such a cube is to make 5-level prisms in the shapes of the P, Q, R, U, and X pentominoes, using pieces $\{a,e,j,m,w\}$, $\{f,k,l,p,r\}$, $\{A,d,D,E,n\}$, $\{c,C,F,u,v\}$, $\{b,B,t,x,z\}$; then use the packing in answer 269(!). This solution can be found with six very short runs of Algorithm X, taking only 300 megamems overall.

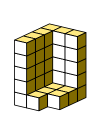
Another nice way, due to Torsten Sillke, is more symmetrical: There are 70,486 ways to partition the pieces into five sets of five that allow us to build an X-prism in the center (with piece x on top), surrounded by four P-prisms.

One can also assemble a Dorian cube from five cuboids, using one $1\times3\times5$, one $2\times2\times5$, and three $2\times3\times5$s. Indeed, there are zillions more ways, too many to count.

341. (a) Make an exact cover problem in which a and A, b and B, ..., f and F are required to be in symmetrical position; there are respectively $(86, 112, 172, 112, 52, 26)$ placements for such 10-cubie "super-pieces." Furthermore, the author decided to force piece m to be in the middle of the top wall. Solutions were found immediately! So piece x was placed in the exact center, as an additional desirable constraint. Then there were exactly 20 solutions; the one below has also n, o, and u in mirror-symmetrical locations.

(b) The super-pieces now have $(59, 84, 120, 82, 42, 20)$ placements; the author also optimistically forced j, k, and m to be symmetrical about the diagonal, with m in the northwest corner. A long and apparently fruitless computation (34.3 teramems) ensued; but — hurrah — two closely related solutions were discovered at the last minute.

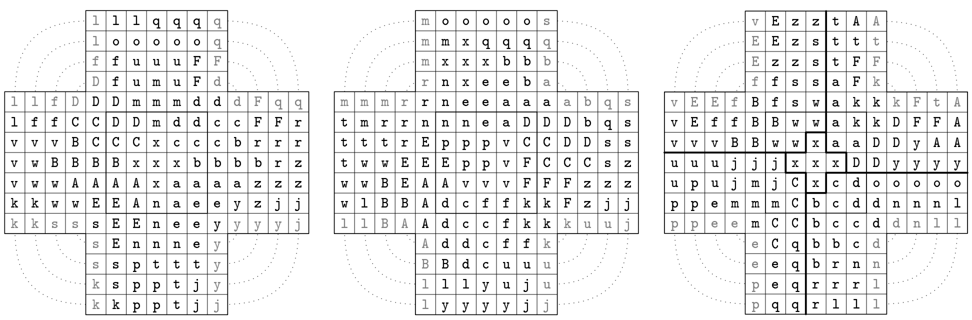
(c) This computation, due to Torsten Sillke [see *Cubism For Fun* **27** (1991), 15], goes much faster: The quarter-of-a-box shown here can be packed with seven non-x pentacubes in 55356 ways, found in 1.3 Gμ. As in answer 294, this yields a new exact cover problem, with 33412 different options.



French
Hansson
chirality
hexacubes
heptacubes
Niemann
Baillie
cuboids
symmetry-breaking
Nixon
Hansson
Bouwkamp
Guy
Dorie
Dorian cube
pentominoes
Sillke
partition
author
geek art
Sillke

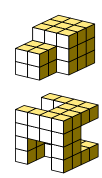
Künzell
Farhi
Farhi
weighted factorization

Another 11.8 Gμ then yields seven suitable partitions into four sets of seven, one of which is illustrated below. [See also *Cubism For Fun* **49** (1999), 26.]



(a) (b) (c)

342. As in previous exercises, the key is to reduce the search space drastically, by asking for solutions of a special form. (Such solutions aren't unlikely, because pentacubes are so versatile.) Here we can break the given shape into four pieces: Three modules of size 3^3+2^3 to be packed with seven pentacubes, and one of size $4^3-3\cdot 2^3$ to be packed with eight pentacubes. The first problem has 13,587,963 solutions, found with 2.5 Tμ of computation; they involve 737,695 distinct sets of seven pentacubes. The larger problem has 15,840 solutions, found with 400 Mμ and involving 2075 sets of eight. Exactly covering those sets yields 1,132,127,589 suitable partitions; the first one found, $\{a, A, b, c, j, q, t, y\}$, $\{B, C, d, D, e, k, o\}$, $\{E, f, l, n, r, v, x\}$, $\{F, m, p, s, u, w, z\}$, works fine. (We need only one partition, so we needn't have computed more than a thousand or so solutions to the smaller problem.)



Pentacubes galore: Since the early 1970s, Ekkehard Künzell and Sivy Farhi have independently published booklets that contain hundreds of solved pentacube problems.

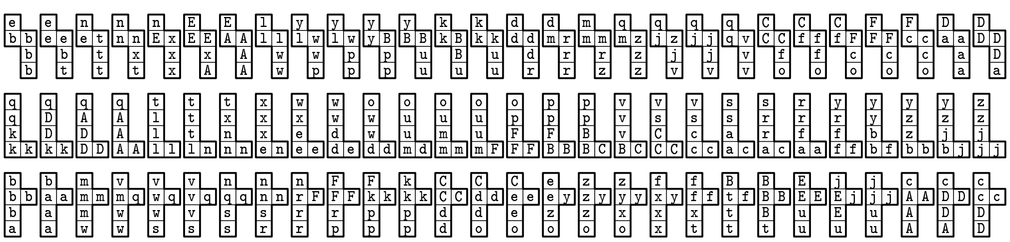
343. We can use an instructive variety of methods to deduce that the tallest towers have heights $(h_{\rm O}, h_{\rm P}, \ldots, h_{\rm Z}) = (12, 29, 28, 28, 29, 25, 26, 23, 24, 17, 28, 27)$: Case O is trivial. A perfect tower for P was published by S. Farhi in *Pentacubes*, 5th edition (1981), Fig. 78. And it's easy to show that $h_{\rm W} \le 24$, because r, t, v, x, z can't be placed.

Factorization yields most of the upper bounds. For example, let the cells of a tower for R be $\{00k, 01k, 11k, 12k, 21k \mid 1 \le k \le h\}$, and add a new "weight" column to the exact-cover matrix, representing the sum of all items/columns $00k$ and $12k$. (Thus the option 'y 212 311 312 412 512' has weight 4.) An exact cover by disjoint options/rows will then make the new column sum $2h$. But the maximum weights of the pentacubes (a, A, …, f, F, j, k, …, z) are respectively (1, 1, 1, 1, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 1, 1, 3, 5, 3, 4, 2, 3, 0, 3, 0, 0, 0, 4, 0). Their sum is 57; hence $h_{\rm R} \le 57/2 < 29$.

Similar arguments prove that $h_{\rm U} < 27$, $h_{\rm V} < 24$, $h_{\rm X} < 18$, $h_{\rm Z} < 28$. But case T is more complicated. Let's introduce a column for the weights $(100\cdot 00k) + (100\cdot 02k) + (10\cdot 11k) + (101\cdot 21k)$, and compute the 29 maximum weights (312, 312, 310, 310, 311, 311, 221, 221, 210, 210, 220, 220, 220, 210, 211, 210, 310, 505, 323, 414, 300, 323, 400, 400, 400, 300, 200, 414, 400). The heaviest 27 sum to 8296, which is less than $311\cdot 27$; hence $h_{\rm T} < 27$. And if $h_{\rm T} = 26$, further study shows that we must omit x and two of $\{e, E, k, m\}$. Moreover, each piece must use an option of maximum weight, except that c and C should use weight 310. These restrictions narrow down the search considerably; Algorithm X is able to prove that $h_{\rm T} < 26$ in 11 Tμ (and Algorithm M in 7.6 Tμ).

It's difficult to prove that $h_Q < 29$, and even harder to prove that $h_Y < 29$. But in both cases a suitable weighted factorization makes the calculations feasible. (See `www.math.uni-bielefeld.de/~sillke/POLYCUBE/TOWER/pentacube`.)

Such weights also greatly accelerate the *successful* searches, for towers of maximum height. Here are some that were hardest-to-find (add piece 's' atop the first one):



344. Reduce the placements that occupy the center cell from 72 to 3. That problem has 2528 solutions, found by Algorithm X in 25 Gμ; and those solutions form 1264 mirror-symmetric pairs. [See C. J. Bouwkamp and D. A. Klarner, *JRM* **3** (1970), 10–26.]

345. A variation of even/odd coordinates works nicely: Let the pieces fill 13 cells like $(x,y,z)+\{(\pm1,\pm1,\pm3),(1,\pm1,\pm1)\}$, xyz odd, where the items (x,y,z) for $0 \le x,y \le 10$ and $0 \le z \le 6$ are primary for x,y,z even and secondary for x,y,z odd. The solution is unique. [This puzzle, marketed as "Vier Farben Block," was designed by T. Geerinck in 2004.]

```
001122 001122 001122 001122
001122 888899 888899 001122
334444 834894 83b89b 33bbbb
334444 a34a94 a3ba9b 33nnnn
556677 aaaa99 aaaa99 556677
556677 556677 556677 556677
```

346. (a) Shifting by multiples of $(0,1,1)$ gives N disjoint tripods whose corners are on layer 0 of the torus, filling all cells of that layer except for a (possibly broken) diagonal, and also filling all cells of such a diagonal on layer 1. We can plug the holes on layer 0 by appropriately placing N tripods whose corners are on layer $N-1$. And so on.

(b) Here's a way to pack twelve of them into a $3\times6\times6$ torus. (Is 7/9 optimum?)

```
012600    066678    0..6..
112371    917778    .1..7.
222348    9a2888    ..2..8
933345    9ab399    9..3..
0a4445    aab64a    .a..4.
01b555    bbb675    ..b..5
```

(c) Place 13 tripods in a $6\times6\times6$ torus, with corners at $(0,0,0)$, $(0,1,1)$, $(0,2,2)$, $(1,1,3)$, $(1,2,4)$, $(2,3,2)$, $(2,4,4)$, $(3,3,3)$, $(3,4,5)$, $(4,4,0)$, $(4,5,1)$, $(5,0,5)$, $(5,5,3)$.

(d) One can place $2r(l,m,n)$ nonoverlapping tripods in a $2l\times2m\times2n$ torus, by putting the tripod corners at the positions of the pod corners, plus $(0,0,0)$ and (l,m,n).

(e) With one primary item '#' and lmn secondary items xyz, and with options such as '# 123 023 103 113 120 121 122' (one for each pod with $0\le x<l$, $0\le y<m$, $0\le z<n$), we can find solutions with t pods by giving multiplicity t to #. Furthermore we can save time by letting the items 000 and $(l-1)(m-1)(n-1)$ be primary, because those two pods can be assumed to be present. In this way we find $444\mapsto8$, $445\mapsto9$, $446\mapsto9$, $455\mapsto10$, $456\mapsto10$, $466\mapsto12$, $555\mapsto11$, $556\mapsto12$, $566\mapsto13$, $666\mapsto14$. (Algorithm M can determine that $r(6,6,6)<15$ in reasonable time, 253 Gμ, despite its rather weak heuristics for pruning the search. But the SAT solver Algorithm 7.2.2.2C solves this problem in only 2 Gμ; it can also establish that $r(7,7,7)=19$ in 169 Gμ, while Algorithm M as it stands would be hopeless for that task.)

Bouwkamp
Klarner
even/odd coordinates
Vier Farben Block
Geerinck
torus, 3D
SAT solver

[*Notes:* Sherman Stein initiated the study of tripods (actually an n-dimensional generalization called "semicrosses") in *IEEE Trans.* **IT-30** (1984), 356–363; see also his paper with W. Hamaker on pages 364–368. They proved that the function $r(n) = r(n,n,n)$ is $\Omega(n^{1.516})$, and that $r(l,n,n)/n$ approaches a limit as $n \to \infty$. The initial values $(r(1),\ldots,r(9)) = (1,2,5,8,11,14,19,23,28)$ were found by C. Morgan, in an undergraduate project at the University of Warwick in 2000; see also S. Szabó, *Ann. Univ. Sci. Budapestiensis, Sect. Computatorica* **41** (2013), 307–322. With extensive computations, P. R. J. Östergård and A. Pöllänen have proved that $r(10) = 32$ and (surprisingly) that $r(11) = 38$ [*Discrete and Computational Geometry* (online 18 June 2018)]. See also A. Tiskin, *Discrete Math.* **307** (2007), 1973–1981, who showed among other things that $r(12) \geq 43$, $r(n) = \Omega(n^{1.534})$, $r(n) = O(n^2/(\log n)^{1/15})$.]

347. Fourteen proofs have been given by S. Wagon, *AMM* **94**, (1987), 601–617. [For generalizations, see R. J. Bower and T. S. Michael, *Math. Magazine* **79** (2006), 14–30.]

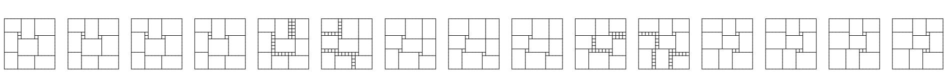
348. See F. W. Barnes's complete solution, *Discrete Mathematics* **133** (1994), 55–78.

349. Let $t = s/4$. Each brick of an m-brick packing contains at least one of the 27 "special points" $\{(it,jt,kt) \mid 0 < i,j,k < 4\}$, because a, b, and c exceed t. Hence $m \leq 27$.

In a packing with $m = 27$, each of the "special lines" l_{*jk}, l_{i*k}, l_{ij*} with two coordinates fixed will be totally full, because the bricks collectively occupy $27(a+b+c)$ units of space on those lines. The special lines also intersect the bricks in 27 segments of each length a, b, c; hence each special line has a segment of each length.

Thus we're led to solve an XCC problem with primary items p_{ijk}, l_{*jk}, l_{i*k}, l_{ij*} and secondary items x_{ijk}, y_{ijk}, z_{ijk}, and with options like 'p_{ijk} $x_{ijk}{:}\pi_1$ $y_{ijk}{:}\pi_2$ $z_{ijk}{:}\pi_3$' and 'l_{i*k} $y_{i1k}{:}\pi_1$ $y_{i2k}{:}\pi_2$ $y_{i3k}{:}\pi_3$', where $\pi_1\pi_2\pi_3$ is a permutation of $\{a,b,c\}$. That problem has 7712 solutions, when we fix one of the six options for p_{111}.

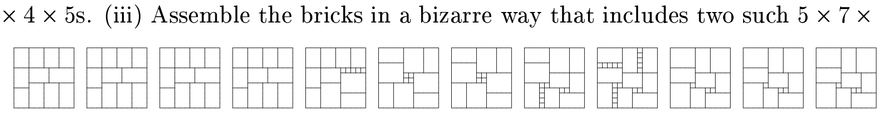
Only 168 of those solutions, in 21 equivalence classes under the 48 symmetries of the cube, actually pack properly when $(a,b,c) = (2,3,4)$. And it can be shown that those 21 solutions will solve Hoffman's problem for arbitrary (a,b,c). Here, for example, is the unique solution that is "self-dual" — isomorphic to itself when $a \leftrightarrow c$:



[See Hoffman's exposition in *The Mathematical Gardner* (1981), 212–225.]

350. Set this up for Algorithm M with 28 instances of a $3\times4\times5$ brick and 48 instances of a single cubie. We can omit all options where a brick lies 1 or 2 units from a face but not on the face, because the brick could move outward in such solutions. We can also force the placement of a brick at corner $(0,0,0)$. Furthermore, an empty corner would imply at least 27 cubies there; hence we can omit placing a cubie in any corner except $(11,11,11)$. This problem, with 715 options of size 61 and 1721 options of size 2, has 112 solutions(!), found in 440 Gμ. (The author's first attempt, in 2004, took much longer.)

There are three species of solutions: (i) Pack seven bricks into $5\times7\times12$; arrange four of those in a pinwheel (see exercise 365), leaving a $2\times2\times12$ hole. (ii) Pack 12 into $5\times12\times12$; add a pinwheel of four $5\times7\times7$s, each of which is a pinwheel of four $3\times4\times5$s. (iii) Assemble the bricks in a bizarre way that includes two such $5\times7\times7$s:



Types (i), (ii), (iii) contribute 6+10+4 nonisomorphic solutions. [George Miller's puzzle with bricks of tricolored faces is called Perfect Packing, because 28 is a perfect number.]

Stein
semicrosses
Hamaker
Morgan
Szabó
Östergård
Pöllänen
Tiskin
Wagon
Bower
Michael
Barnes
XCC problem
self-dual
author
pinwheel
Miller
Perfect Packing

351. (Generalizing exercise 349, Hoffman observed that such a construction would yield a nice geometrical way to prove the inequality $(abcde)^{1/5} \le (a+b+c+d+e)/5$.)

352. None. But any eleven of the "hypersolid pentominoes" can easily be squeezed in;

for example,
```
Q X W W .   T S S S U   S S Z R U   Q Q Q Q .
X X X W W   T T T . U   Z Z Z R R   O O O O O
. X P P W   T P P P U   Z Y R R U   Y Y Y Y .
```
is one way to pack all but `V`.

353. There are exactly 9 (including a mirror pair). They pack a $3 \times 3 \times 3$ cube in $48 \cdot 8789$ ways, such as $\substack{000\\112\\123} \mid \substack{434\\523\\567} \mid \substack{548\\786\\876}$. [See J. Lou, Danish patent 126840 (1973).]

354. (a) Let cell (x, y) of a polyomino correspond to $(-x, x, y, -y)$. Let cell (x, y) of a polyhex, as represented in exercise 315, correspond to $(0, x, y, -x-y)$.

(b) A polysphere is planar if and only if the differences between its adjacent cells lie in a plane. Each of those differences has the form $e_{ij} = e_i - e_j$, where $e_1 = (1, 0, 0, 0)$, $\ldots$, $e_4 = (0, 0, 0, 1)$. Three such differences can't be linearly independent yet lie in a plane; the linearly dependent cases are polyominoes and/or polyhexes.

(c) Every connected graph has at least one vertex whose removal doesn't disconnect the graph. So the result follows by induction on n.

(d) An orthogonal matrix fixes $w + x + y + z$ if and only if its row and column sums are 1. The matrices (i) T and (ii) R below respectively rotate by 120° about $x = y = z$ and by 90° about $(x = y) \wedge (w = z)$.

$$T = \begin{pmatrix} 1&0&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 0&1&0&0 \end{pmatrix};\quad R = \frac{1}{2}\begin{pmatrix} 1&-1&1&1\\ 1&1&1&-1\\ -1&1&1&1\\ 1&1&-1&1 \end{pmatrix};\quad R^2 = \begin{pmatrix} 0&0&0&1\\ 0&0&1&0\\ 0&1&0&0\\ 1&0&0&0 \end{pmatrix};\quad H = \frac{1}{6}\begin{pmatrix} 5&-1&-1&3\\ -1&-1&5&3\\ -1&5&-1&3\\ 3&3&3&-3 \end{pmatrix}.$$

(e) The matrices (i) R^2 and (ii) H above respectively rotate by 180° about $(x = y) \wedge (w = z)$ and about $(x = y) \wedge (w = 3z - 2x)$. Thus H can be used when $z = 0$.

(f) Suppose $V' = \{v'_1, \ldots, v'_n\}$ is a rotation of $V = \{v_1, \ldots, v_n\} \subset S$, where $v_k = (w_k, x_k, y_k, z_k)$, $v'^T_k = (w'_k, x'_k, y'_k, z'_k)^T = Qv^T_k$, $v_1 = v'_1 = (0, 0, 0, 0)$, and $v_2 = e_{12} = (1, -1, 0, 0)$. The matrix $Q = (q_{ij})$ is orthogonal, with row and column sums and determinant 1. By applying an even permutation to the coordinates of v' and the rows of Q, we can assume without loss of generality that $v'_2 = e_{12} = v_2$. Hence $q_{k1} = q_{k2} + \delta_{k1} - \delta_{k2}$, $q_{11} = q_{22}$. If $Q \ne I$ we have $v_p - v_q = e_{ij} \ne e_{i'j'} = v'_p - v'_q$ for some p, q, i, j, i', and j', with $i < j$. By orthogonality, $e_{12} \cdot e_{ij} = e_{12} \cdot e_{i'j'} \in \{-1, 0, +1\}$.

If $e_{12} \cdot e_{ij} = 1$, there are six cases, depending on (i, j, i', j'): $(1, 3, 1, 4)$ implies $Q = TH$; $(1, 4, 1, 3)$ implies $Q = HT^2$; $(1, 3, 4, 2)$ implies $Q = T^2RT$ or THR^3T^2; $(1, 4, 3, 2)$ implies $Q = T^2RT$ or HR; $(1, 3, 3, 2)$ and $(1, 4, 4, 2)$ are impossible.

If $e_{12} \cdot e_{ij} = 0$, we have $(i, j, i', j') = (3, 4, 4, 3)$ and Q is forced to be TR. Finally, the case (i, j, i', j') for $e_{12} \cdot e_{ij} = -1$ is the same as the case (i, j, j', i') for $e_{12} \cdot e_{ij} = +1$.

Note: Some authors represent S as the set of integer triples (X, Y, Z) with $X + Y + Z$ even. The Hadamard transform provides an isomorphism between these representations: If $-2M$ is the upper left 4×4 submatrix of 7.2.1.1–(21), we have $M^2 = I$, $\det M = 1$, and M takes $(-x-y-z, x, y, z) \mapsto (0, x+z, y+z, x+y) = (0, X, Y, Z)$.

355. (a) Normalize the given polysphere by subtracting $(x_{\min}, y_{\min}, z_{\min})$, to get *its* base placement. Then, for each base placement P, form up to three others until no more can be formed: (i) Replace each xyz by yzx. (ii) Replace each xyz by $(x + y + z)(t - z)(t - x)$, for some large t; then normalize. (iii) If $z = 0$ in each cell of P, replace each $xy0$ by $yx0$.

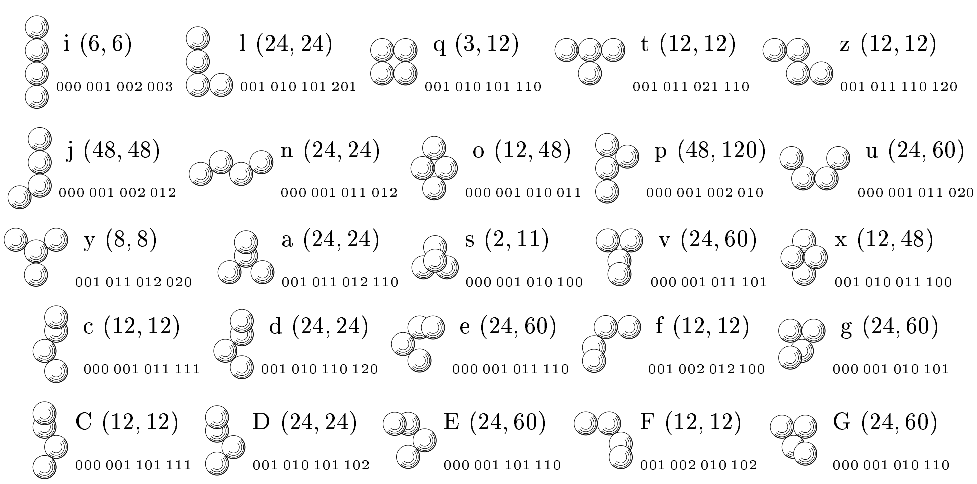
[The (X, Y, Z) representation mentioned in answer 354 suggests "polyjubes" — George Sicherman's name for the sets of *edge-connected cubes* that don't touch face-to-face. Transformation (iii) does not apply to polyjubes; hence there are 5 trijubes and

Hoffman
arithmetic/geometric mean inequality
hypersolid pentominoes
Lou
patent
row and column sums
Hadamard transform
polyjubes
Sicherman
edge-connected cubes

28 tetrajubes. Polyjubes are also equivalent to "polyrhons" — the connected sets of rhombic dodecahedra, which are the Voronoi regions of the face-centered cubic lattice. See S. Coffin, *The Puzzling World of Polyhedral Dissection* (1990), Figure 167.]

(b) Phenalene has eight base placements; in lexicographic order they are $\{000, 001, 010\}$, $\{000, 001, 100\}$, $\{000, 010, 100\}$, $\{001, 010, 011\}$, $\{001, 010, 100\}$, $\{001, 100, 101\}$, $\{010, 100, 110\}$, $\{011, 101, 110\}$. The straight trisphere has six base placements, namely $\{000, 001, 002\}$, $\{000, 010, 020\}$, $\{000, 100, 200\}$, $\{002, 011, 020\}$, $\{002, 101, 200\}$, $\{020, 110, 200\}$. The bent trisphere has twelve, from $\{001, 010, 101\}$ to $\{011, 100, 110\}$. And phenanthrene has twenty-four, from $\{000, 001, 011\}$ to $\{020, 101, 110\}$.

(c) There are 853 connected subsets, with 475 different base placements. (Each placement with $\max(x + y + z) = (1, 2, 3)$ occurs respectively $(10, 4, 1)$ times.) They form 25 distinct tetraspheres — five from tetrominoes and six additional planar pieces from tetrahexes, plus four nonplanar nonchiral pieces and five chiral pairs:



Each piece has been given an identifying letter. This chart shows the number of base placements and the number of occurrences in $simplex(3, 3, 3, 3, 3, 0, 0)$, as well as the lexicographically smallest base placement. Notice that j and p have 48 base placements, while a polycube can have at most 48. Piece s is $simplex(1, 1, 1, 1, 1, 0, 0)$, a tetrahedron with four equidistant spheres. Piece x is perhaps the most fascinating to play with.

[The tetraspheres were first enumerated by K. Takizawa; then T. Sillke enumerated the nonplanar polyspheres of larger sizes. See B. Wiezorke, *Cubism For Fun* **25**, part 3 (1990), 10–17; G. Bell, *Cubism For Fun* **81** (2010), 18–23; OEIS A038174.]

356. (a) The n-tetrahedron, which is the same as $simplex(n-1, n-1, n-1, n-1, n-1, 0, 0)$, has base placement $\{xyz \mid x, y, z \geq 0,\ x+y+z < n\}$; $\binom{n+2}{3}$ cells. (It has one other base placement, namely $\{(n-1-x)(n-1-y)(n-1-z) \mid x, y, z \geq 0,\ x+y+z < n\}$.)

One of the 12 base placements of the $m \times n$ roof is $\{x(y+k)(m-1-y) \mid k \geq 0,\ 0 \leq x < n-k,\ 0 \leq y < m-k\}$. If $m \leq n$, it has $m(m+1)(3n-m+1)/6$ cells.

The stretched $m \times n$ roof is based on slicing the face-centered cubic lattice into layers with constant $y-z$. (Each cell has two neighbors on its own layer, four neighbors on each adjacent layer, and two neighbors that are two layers away.) One of its 12 base placements is $\{(x+m-1-y)(y+k)y \mid k \geq 0,\ 0 \leq x < n-k,\ 0 \leq y < m-k\}$.

polyrhons
rhombic dodecahedra
Voronoi regions
Coffin
lexicographic order
tetrahedron
Takizawa
Sillke
Wiezorke
Bell
simplex

(b) Let's call the four shapes T_4, $R_{3\times 4}$, $S_{3\times 4}$, and $S_{4\times 3}$. Here are the stats:

Shape	Total	All planar	Mixed		All nonplanar	
	multisets(sets)	(balanced)	(balanced)	(chiral)	(balanced)	(chiral)
T_4	2952(1211)	174(34)	308(115)	2442(1062)	2(0)	26(0)
$R_{3\times 4}$	11531(6274)	372(69)	1250(583)	9818(5608)	3(0)	88(14)
$S_{3\times 4}$	1184(480)	51(6)	108(48)	1014(426)	1(0)	10(0)
$S_{4\times 3}$	266(52)	2(0)	27(8)	234(44)	1(0)	2(0)

For example, $\{\mathrm{j,j,p,p,t}\}$ is one of 174 multisets of five planar pieces that can make T_4. [In fact, the solution is unique — and $\{\mathrm{j,j,p,p,t}\}$ also uniquely solves $R_{3\times 4}$ and $S_{3\times 4}$! G. Bell used this fact as the basis for his elegant Triple Pyradox puzzle; see *Cubism For Fun* **94** (2014), 10–13.] Of those 174 cases, 34 have five *different* pieces; for instance, $\{\mathrm{n,o,p,u,y}\}$ is one of only seven that contains y, the "propeller."

Many other suitable sets of five mix planar places with nonplanar ones. Of these, 115 (like $\{\mathrm{g,G,i,s,x}\}$) are closed under reflection; that one has 24 solutions, all essentially the same. The other 1062 form 531 mirror-image pairs (like $\{\mathrm{d,e,f,G,i}\}$ and $\{\mathrm{D,E,F,g,i}\}$); every solution for a chiral set has 12 equivalents, not 24.

Algorithm M discovers all such solutions quickly, if we assign multiplicity $[0\,..\,5]$ to each piece. There are respectively (88927, 77783, 3440, 996) solutions to (T_4, $R_{3\times 4}$, $S_{3\times 4}$, $S_{4\times 3}$), without symmetry removal; they're found in (840, 607, 48, 13) Mμ.

Six of the multisets — three mirror pairs — are actually able to make *all four* shapes. These versatile combinations of pieces are $\{\mathrm{e,g,g,p,p}\}$ and $\{\mathrm{E,G,G,p,p}\}$, $\{\mathrm{g,j,p,p,p}\}$ and $\{\mathrm{G,j,p,p,p}\}$, $\{\mathrm{g,p,p,p,p}\}$ and $\{\mathrm{G,p,p,p,p}\}$.

There's an obvious, yet interesting, way to make T_4 with the "pure" multiset $\{\mathrm{s,s,s,s,s}\}$. The only other pure multiset that works is $\{\mathrm{p,p,p,p,p}\}$, which is able to form both T_4 and $R_{3\times 4}$, as well as many other shapes noted by W. Schneider in 1995.

[A 2×7 roof also has 20 cells. So we might want to consider additional stats:

$R_{2\times 7}$	3940(1628)	608(116)	1296(512)	1970(1000)	14(0)	52(0)
$S_{2\times 7}$	426(84)	58(4)	48(20)	306(60)	2(0)	12(0)
$S_{7\times 2}$	4(0)	0(0)	0(0)	0(0)	2(0)	2(0)

The long and skinny $S_{7\times 2}$ can be made in only two ways, both with x in the middle, surrounded by g's or G's. The set $\{\mathrm{i,j,n,o,p}\}$ packs both $S_{2\times 7}$ and $S_{7\times 2}$, as well as T_4.]

(c) Let's name the trispheres 1, 2, 3, 4, according to the squared distance between the two farthest-apart cells; thus the pieces in exercise 355 are 2, 4, 1, 3. The pyramid P_4 is buildable from 296 such multisets, many of which allow huge numbers of solutions. (For example, each of the ten multisets that contain $\{1,1,2,2,3,3,4,4\}$ leads to more than 30,000 solutions; $\{1,1,2,2,2,3,3,4,4,4\}$ has more than 120,000!) Most interesting are the cases with unique solution ($\{2, 2, 4, 4, 4, 4, 4, 4, 4, 4\}$‡, $\{1, 1, 1, 4, 4, 4, 4, 4, 4, 4\}$, $\{1, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$), or with just two solutions ($\{2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$†, $\{1, 1, 3, 3, 3, 3, 3, 3, 3, 3\}$, $\{2, 4, 4, 4, 4, 4, 4, 4, 4, 4\}$‡); † = noted by L. Gordon (1986); ‡ = noted by J. Becker (2009). The stretched pyramid S_4 has 213 such multisets, all of which can also make P_4. Unique solutions occur for $\{1, 1, 1, 3, 4, 4, 4, 4, 4, 4\}$ and $\{1, 3, 3, 3, 3, 3, 3, 3, 4, 4\}$; almost for $\{3, 3, 3, 3, 3, 3, 3, 3, 4, 4\}$.

Historical notes: The first polysphere puzzle may have been "Pyramystery," copyright by Piet Hein in 1967 when his Soma cube was becoming popular. Pyramystery had the six pieces $\{1,1,3,4,\mathrm{o,p}\}$; Hein knew that it could form T_4, as well as two copies of T_3, and several planar designs. A similar puzzle of unknown origin, called

Bell
Pyradox
propeller
multiplicity
symmetry removal
Schneider
Gordon
Becker
Pyramystery
Hein
Soma

Kugelpyramide, may have been created earlier, because it was seen by B. Wiezorke in 1968. Kugelpyramide's pieces, $\{1,3,4,4,\mathrm{o},\mathrm{p}\}$ were slightly different. With either Pyramystery or Kugelpyramide one can make T_4, T_3+T_3, $R_{3\times4}$, $R_{2\times7}$, and $S_{2\times7}$; and with the not-thought-of pieces $\{1,2,3,4,\mathrm{o},\mathrm{p}\}$, one could have made also $S_{3\times4}$ but not T_3+T_3. The first puzzle to mix polyomino-type polyspheres with polyhex-type polyspheres — a nonobvious possibility — was Tetra, by A. Kuwagaki and S. Takenaka; see *Sugaku Seminar* **11**, 7 (July 1972), cover, 34–38; also *U.S. Patent 3837652* (1974). That patent describes making P_3 from the dispheres and trispheres, and making the 44-ball octahedron $P_4P_3^R$ from the planar tetraspheres $\{\mathrm{i},\mathrm{j},\mathrm{l},\mathrm{n},\mathrm{o},\mathrm{p},\mathrm{q},\mathrm{t},\mathrm{u},\mathrm{y},\mathrm{z}\}$. In those early days, the stretched roofs and pyramids weren't known to be possible; they were first introduced by Leonard Gordon, in his WARP-30 puzzle (Kadon Enterprises, 1986).

(d) The unique base placement is $\{xyz \mid x,y,z\in\{0,1,2,3\}, x\neq y\neq z\neq x\}$. Stats are 95(0): 5(0) 13(0) 70(0) 3(0) 4(0). Only pieces a, c, d, q, u will fit in this shape. Here's how to make it with $\{\mathrm{a},\mathrm{a},\mathrm{c},\mathrm{d},\mathrm{u},\mathrm{u}\}$, $\{\mathrm{c},\mathrm{c},\mathrm{c},\mathrm{C},\mathrm{C},\mathrm{C}\}$, or $\{\mathrm{u},\mathrm{u},\mathrm{u},\mathrm{u},\mathrm{u},\mathrm{u}\}$:

```
a2a2   a1  a2    a1a1               C2c3   C2  c3    C2c3               u5u3   u1  u2    u3u3
c  a2  c    u2          a1u1  ;  C2  c3 c1    C3         C2c3  ;  u5  u6 u1    u6           u2u2
 d d          c    u2  c   u1     C1c2         c1    C3 c1  C3     u5u5          u1    u6 u1  u6
        d d     d   u2   u1u1            C1c2     C1  c2   C1c2               u4u4   u4  u6   u4u2
```

(Note that $\{\mathrm{q},\mathrm{q},\mathrm{q},\mathrm{q},\mathrm{q},\mathrm{q}\}$ is trivial.) This is a hollow object that can't stand on its own.

357. Truncated octahedra are the Voronoi regions of the "body-centered cubic lattice," which is less tight than the face-centered cubic lattice: It can be represented as the set of all integer triples (x,y,z) with $x \bmod 2 = y \bmod 2 = z \bmod 2$. Two truncated octahedra whose centers are two such points are adjacent if and only the distance between those points is either $\sqrt{3}$ (eight neighbors, joined at hexagonal faces) or 2 (six neighbors, joined at square faces). There are 2 displatts, 6 trisplatts, and 44 tetrasplatts — including 9 chiral pairs. [See M. Owen and M. Richards, *Eureka* **47** (1987), 53–58.]

Base placements can be found almost as in exercise 324, except that we must set $(x,y,z)\mapsto(y, 2\lceil x_{\max}\rceil - x, z)$. Furthermore, each base placement should be normalized, by adding $(\pm1,\pm1,\pm1)$ if needed, so that $x_{\min}+y_{\min}+z_{\min}\le 1$.

[One might also consider truncating further, leaving only the union of four small hexagonal prisms between diametrically opposite hexagonal faces. This yields a subfamily of polysplatts called "polycrunches" — named and enumerated by G. Sicherman: Adjacent crunches, with centers $\sqrt{3}$ apart, are pasted together where the prisms meet. The polycrunch family has 1 monocrunch, 1 dicrunch, 3 tricrunches, and 14 tetracrunches (including 2 chiral pairs). The tricrunches have respectively (4, 12, 12) base placements; the tetracrunches have respectively (4, 6, 6, 8, 12, 12, 12, 12, 24, …, 24).]

358. This fascinating packing is considerably more difficult than the other. For example, there are *six* distinct trihexaspheres, having respective angles of (60°, 90°, $\arccos(-1/3)\approx 109.5°$, 120°, $\arccos(-5/6)\approx 146.4°$, 180°) and respective maximum squared distances (1, 2, 8/3, 3, 11/3, 4). G. Bell has discovered a convenient way to represent *magnified* polyhexaspheres *within* the face-centered cubic lattice: Consider the subset $\widehat{S}$ of S whose elements have the special form $\alpha j+\beta k+\gamma_l$ for integers j, k, l, where $\alpha=(0,3,-3,0)$, $\beta=(0,0,3,-3)$, $\gamma_{2l}=(6l,-2l,-2l,-2l)$, and $\gamma_{2l+1}=(6l+3,-3-2l,-2l,-2l)$. Two cells of $\widehat{S}$ are called adjacent if the distance between them is $\sqrt{18}$. Thus each cell v of layer l has six neighbors $v\pm\{(0,3,-3,0),(0,0,3,-3),(0,-3,0,3)\}$ on the same level; three neighbors $v+A[l\text{ even}]+B[l\text{ odd}]$ on level $l+1$, where $A=\{(3,-3,0,0),\ (3,0,-3,0),\ (3,0,0,-3)\}$ and $B=\{(3,1,-2,-2),\ (3,-2,1,-2),\ (3,-2,-2,1)\}$; and three neighbors $v-A[l\text{ odd}]-B[l\text{ even}]$ on level $l-1$.

Kugelpyramide
Wiezorke
Tetra
Kuwagaki
Takenaka
Patent
octahedron
Gordon
WARP-30
Kadon Enterprises
Voronoi
body-centered cubic lattice
face-centered cubic lattice
chiral pairs
Owen
Richards
Base placements
polycrunches
Sicherman
triagonal neighbors
trihexaspheres
Bell
face-centered cubic lattice

X pentomino
Bloch

All of the tetraspheres are tetrahexaspheres, because they fit on at most two levels. But many of the pentaspheres, for example the planar one for pentomino T, are not pentahexaspheres. A polyomino polyhexasphere exists if and only if the polyomino fits in a $2\times k$ box: Connected subsets of $\{(0,0,3k,-3k),(-3,-1,2{+}3k,2{-}3k)\}$ are OK.

The matrices T and R^2HR^2 of answer 354 are rotations of $\widehat{S}$. Therefore we can obtain equivalent base placements in the manner of answer 355, replacing each xyz by either yzx or $(y+\frac{2}{3}w)(x+\frac{2}{3}w)(z+\frac{2}{3}w)$, where $w=-x-y-z$. Normalize a placement by adding or subtracting 666 or $3\bar{3}0$ or $03\bar{3}$ or $\bar{3}03$. But the analysis is still incomplete: Are further transformations of base placements needed? How many n-hexaspheres are possible, for $n=4$, 5, ...? [See *Cubism For Fun* **106** (2018), 24–29.]

359. First we realize that every edge of the square must touch at least three pieces; hence the pieces must in fact form a 3×3 arrangement. Consequently any correct placement would also lead to a placement for nine pieces of sizes $(17-k)\times(20-k)$, ..., $(24-k)\times(25-k)$, into a $(65-3k)\times(65-3k)$ box. Unfortunately, however, if we try, say, $k=16$, Algorithm X quickly gives a contradiction.

But aha — a closer look shows that the pieces have *rounded corners*. Indeed, there's just enough room for pieces to get close enough together so that, if they truly were rectangles, they'd make a 1×1 overlap at a corner.

So we can take $k=13$ and make nine pieces of sizes 4×7, ..., 11×12, consisting of rectangles *minus* their corners. Those pieces can be packed into a 26×26 square, as if they were polyominoes (see exercise 266), but with the individual cells of the enclosing rectangle treated as secondary items because they needn't be covered. (Well, the eight cells adjacent to corners can be primary.) We can save a factor of 8 by insisting that the 9×11 piece appear in the upper left quarter, with its long side horizontal.

Algorithm X solves that problem in 620 gigamems — but it finds 43 solutions, most of which are unusable, because the missing corners give too much flexibility. The unique correct solution is easily identified, because a 1×1 overlap between rectangles in one place must be compensated by a 1×1 empty cell between rectangles in another. The resulting cross pattern (like the X pentomino) occurs in just one of the 43.

360. Let there be mn primary items p_{ij} for $0\le i<m$ and $0\le j<n$, one for each cell that should be covered exactly once. Also introduce m primary items x_i for $0\le i<m$, as well as n primary items y_j for $0\le j<n$. The exact cover problem has $\binom{m+1}{2}\cdot\binom{n+1}{2}$ options, one for each subrectangle $[a\,.\,.\,b)\times[c\,.\,.\,d)$ with $0\le a<b\le m$ and $0\le c<d\le n$. The option for that subrectangle contains $2+(b-a)(d-c)$ items, namely x_a, y_c, and p_{ij} for $a\le i<b$, $c\le j<d$. The solutions correspond to reduced decompositions when we insist that each x_i be covered $[1\,.\,.\,n]$ times and that each y_j be covered $[1\,.\,.\,m]$ times. (We can save a little time by omitting x_0 and y_0.)

The 3×5 problem has 20165 solutions, found in 18 Mμ. They include respectively (1071, 3816, 5940, 5266, 2874, 976, 199, 22, 1) cases with (7, 8, ..., 15) subrectangles.

[See C. J. Bloch, *Environment and Planning* **B6** (1979), 155–190, for a complete catalog of all reduced decompositions into at most seven subrectangles.]

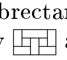
361. The minimum is $m+n-1$. Proof (by induction): The result is obvious when $m=1$ or $n=1$. Otherwise, given a decomposition into t subrectangles, $k\ge 1$ of them must be confined to the nth column. If two of those k are contiguous, we can combine them; the resulting dissection of order $t-1$ reduces to either $(m-1)\times n$ or $m\times n$, hence $t-1\ge(m-1)+n-1$. On the other hand if none of them are contiguous, the reduction of the first $n-1$ columns is $m\times(n-1)$; hence $t\ge m+(n-1)-1+k$.

Close examination of this proof shows that a reduced decomposition has minimum order t if and only if its boundary edges form $m-1$ horizontal lines and $n-1$ vertical lines that don't cross each other. (In particular, the "tatami condition" is satisfied; see exercise 7.1.4–215.) See C. F. Earl, *Environment and Planning* **B5** (1978), 179–187.

tatami condition
Earl
Scherer
Graham

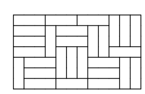
362. Simply remove the offending subrectangles, so that the cover problem has only $\bigl(\binom{m+1}{2}-1\bigr)\bigl(\binom{n+1}{2}-1\bigr)$ options. Now there are 13731 3×5 solutions, found in 11 Mμ, and (410, 1974, 3830, 3968, 2432, 900, 194, 22, 1) cases with (7, 8, ..., 15) subrectangles.

363. Introduce additional primary items X_i for $0<i<m$, to be covered $[1\,..\,n-1]$ times, as well as Y_j for $0<j<n$, to be covered $[1\,..\,m-1]$ times. Then add items X_i for $a<i<b$ and Y_j for $c<j<d$ to the constraint for subrectangle $[a\,..\,b)\times[c\,..\,d)$.

Now the 3×5 problem has just 216 solutions, found in 1.9 megamems. They include (66, 106, 44) instances with (7, 8, 9) subrectangles. Just two of the solutions are symmetric under left-right reflection, namely  and its top-bottom reflection.

364. We can delete non-tromino options from the exact cover problem, thereby getting all faultfree tromino tilings that are reduced. If we also delete the constraints on x_i and y_j — and if we require X_i and Y_j to be covered $[1\,..\,n]$ and $[1\,..\,m]$ times instead of $[1\,..\,n-1]$ and $[1\,..\,m-1]$ — we obtain *all* of the $m\times n$ faultfree tromino tilings.

It is known that such nontrivial tilings exist if and only if $m,n\ge7$ and mn is a multiple of 3. [See K. Scherer, *JRM* **13** (1980), 4–6; R. L. Graham, *The Mathematical Gardner* (1981), 120–126.] So we look at the smallest cases in order of mn: When $(m,n)=(7,9)$, $(8,9)$, $(9,9)$, $(7,12)$, $(9,10)$, we get respectively $(32,32)$, $(48,48)$, $(16,16)$, $(706,1026)$, $(1080,1336)$ solutions. Hence the assertion is false; a smallest counterexample is shown.



365. Augment the exact cover problem of answer 362 by introducing $\binom{m+1}{2}+\binom{n+1}{2}-2$ secondary items x_{ab} and y_{cd}, for $0\le a<b\le m$ and $0\le c<d\le n$, $(a,b)\ne(0,m)$, $(c,d)\ne(0,n)$. Include item x_{ab} and y_{cd} in the option for subrectangle $[a\,..\,b)\times[c\,..\,d)$. Furthermore, cover x_i $[1\,..\,m-i]$ times, not $[1\,..\,n]$; cover y_j $[1\,..\,n-j]$ times.

366. The hint follows because $[a\,..\,b)\times[0\,..\,d)$ cannot coexist motleywise with its left-right reflection $[a\,..\,b)\times[n-d\,..\,n)$. Thus we can forbid half of the solutions.

Consider, for example, the case $(m,n)=(7,7)$. Every solution will include x_{67} with some y_{cd}. If it's y_{46}, say, left-right reflection would produce an equivalent solution with y_{13}; therefore we disallow the option $(a,b,c,d)=(6,7,4,6)$. Similarly, we disallow $(a,b,c,d)=(6,7,c,d)$ whenever $7-d<c$.

Reflection doesn't change the bottom-row rectangle when $c+d=7$, so we haven't broken all the symmetry. But we can complete the job by looking also at the top-row rectangle, namely the option where x_{01} occurs with some $y_{c'd'}$. Let's introduce new secondary items t_1, t_2, t_3, and include t_c in the option that has x_{67} with $y_{c(7-c)}$. Then we include t_1, t_2, and t_3 in the option that has x_{01} with $y_{c'd'}$ for $c'+d'>7$. We also add t_1 to the option with x_{01} and y_{25}; and we add both t_1 and t_2 to the option with x_{01} and y_{34}. This works beautifully, because no solution can have $c=c'$ and $d=d'$.

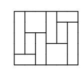
In general, we introduce new secondary items t_c for $1\le c<n/2$, and we disallow all options $x_{(m-1)m}\;y_{cd}$ for which $c+d>n$. We put t_c into the option that contains $x_{(m-1)m}\;y_{c(n-c)}$; t_1 thru $t_{\lfloor(n-1)/2\rfloor}$ into the option that contains $x_{01}\;y_{c'd'}$ when $c'+d'>n$; and t_1 thru $t_{c'-1}$ into the option that contains $x_{01}\;y_{c'(n-c')}$. (Think about it.)

For example, when $m=n=7$ there now are 717 options instead of 729, 57 secondary items instead of 54. We now find 352546 solutions after only 13.2 gigamems of computation, instead of 705092 solutions after 26.4. The search tree now has just 7.8 meganodes instead of 15.7.

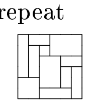
faultfree
Postl
nested motley dissections
pinwheel

(It's tempting to believe that the same idea will break top-bottom symmetry too. But that would be fallacious: Once we've fixed attention on the bottommost row while breaking left-right symmetry, we've lost all symmetry between top and bottom.)

367. From any $m \times n$ dissection of order t we get two $(m{+}2) \times (n{+}2)$ dissections of order $t + 4$, by enclosing it within two $1 \times (m{+}1)$ tiles and two $1 \times (n{+}1)$ tiles. So the claim follows by induction and the examples in exercise 365, together with a 5×6 example of order 10 — of which there are 8 symmetrical instances such as the one shown here. (This construction is faultfree, and it's also "tight": The order of every $m \times n$ dissection is at least $m + n - 1$, by exercise 361.)

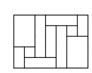


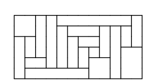
In general, Helmut Postl observes that we can create nested motley dissections by motley-dissecting *any* subrectangle of a motley dissection (taking care not to repeat any internal boundary coordinates) and reducing the result. For example, one of the $2 \cdot (6+3+3+3+1+9+3) = 56$ ways to nest a pinwheel within the second motley 4×4 is shown here.

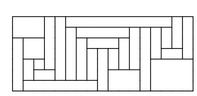


368. The number of subrectangles $[a\,..\,b) \times [c\,..\,d)$ that have either $c = k$ or $d = k$, given k, is ≥ 2 when $k \in \{0, n\}$ and ≥ 3 when $0 < k < n$. Hence $2t \geq 2 + 3(n-1) + 2$.

369. All 214 of the $5{\times}7$ motley dissections have order 11, which is far short of $\binom{6}{2} - 1 = 14$; and there are no $5{\times}8$s, $5{\times}9$s, or $5{\times}10$s. Surprisingly, however, 424 of the 696 dissections of size $6{\times}12$ do have the optimum order 20, and $7{\times}17$ dissections with the optimum order 27 also exist. Examples of these remarkable patterns are shown. (The case $m = 7$ is still not fully explored except for small n. For example, the total number of motley $7{\times}17$ dissections is unknown. No $7{\times}18$s exist, by exercise 368. If we restrict attention to *symmetrical* dissections, the maximum orders for $5 \leq m \leq 8$ are 11 ($5{\times}7$); 19 ($6{\times}11$); 25 ($7{\times}15$); 33 ($8{\times}21$).)







370. The basic idea is to combine complementary options into a single option whenever possible. More precisely: (i) If $a + b = m$ and $c + d = n$, we retain the option as usual; it is self-complementary. (ii) Otherwise, if $a + b = m$ or $c + d = n$, reject the option; merging would be non-motley. (iii) Otherwise, if $a + b > m$, reject the option; we've already considered its complement. (iv) Otherwise, if $b = 1$ and $c + d < n$, reject the option; its complement is illegal. (v) Otherwise, if $b > m/2$ and $c < n/2$ and $d > n/2$, reject the option; it intersects its complement. (vi) Otherwise merge the option with its complement. For example, when $(m, n) = (4, 5)$, case (i) arises when $(a, b, c, d) = (1, 3, 2, 3)$; the option is '$x_1$ y_2 p_{12} p_{22} x_{13} y_{23}' as in answer 366. Case (ii) arises when $(a, b, c, d) = (1, 3, 0, 1)$. Case (iii) arises when $(a, b) = (2, 3)$. Case (iv) arises when $(a, b, c, d) = (0, 1, 0, 1)$; the complement $(3, 4, 4, 5)$ isn't a valid subrectangle in answer 366. Case (v) arises when $(a, b, c, d) = (1, 3, 1, 3)$; cells p_{22} and p_{23} occur also in the complement $(1, 3, 2, 4)$. And case (vi) arises when $(a, b, c, d) = (0, 1, 4, 5)$; the merged option is the union of 'x_0 y_4 p_{04} x_{01} y_{45} t_1 t_2' and 'x_3 y_0 p_{30} x_{34} y_{01}'. (Well, x_0 and y_0 are actually omitted, as suggested in answer 360.)

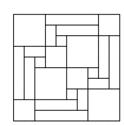
Size 8×16 has (6703, 1984, 10132, 1621, 47) solutions, of orders (26, …, 30).

371. (a) Again we merge compatible options, as in answer 370. But now $(a, b, c, d) \to (c, d, n-b, n-a) \to (n-b, n-c, n-b, n-a) \to (n-b, n-a, c, d)$, so we typically must merge *four* options instead of two. The rules are: Reject if $a = n - 1$ and $c + d > n$, or $c = n - 1$ and $a + b < n$, or $b = 1$ and $c + d < n$, or $d = 1$ and $a + b > n$. Also reject if (a, b, c, d) is lexicographically greater than any of its three successors. But accept, without merging, if $(a, b, c, d) = (c, d, n{-}b, n{-}a)$. Otherwise reject if $b > c$ and $b{+}d > n$, or if $b > n/2$ and $c < n/2$ and $d > n/2$, because of intersection. Also reject if $a+b=n$ or $c+d=n$, because of the motley condition. Otherwise merge four options into one.

For example, the merged option when $n=4$ and $(a,b,c,d)=(0,1,2,4)$ is 'x_0 y_2 p_{02} p_{03} x_{01} y_{24} t_1 x_2 y_3 p_{23} p_{33} x_{24} y_{34} x_3 y_0 p_{30} p_{31} x_{34} y_{24} p_{00} p_{10} x_{02} y_{01}', except that x_0 and y_0 are omitted. Notice that it's important not to include an item x_i or y_j twice, when merging in cases that have $a=c$ or $b=d$ or $a=n-d$ or $b=n-c$.

(b) With bidiagonal symmetry it's possible to have $(a,b,c,d)=(c,d,a,b)$ but $(a,b,c,d)\neq(n-d,n-c,n-b,n-a)$, or vice versa. Thus we'll sometimes merge two options, we'll sometimes merge four, and we'll sometimes accept without merging. In detail: Reject if $a=n-1$ and $c+d>n$, or $c=n-1$ and $a+b>n$, or $b=1$ and $c+d<n$, or $d=1$ and $a+b<n$. Also reject if (a,b,c,d) is lexicographically greater than any of its three successors. But accept, without merging, if $a=c=n-d=n-b$. Otherwise reject if $b>c$ or $b>n-d$ or $a+b=n$ or $c+d=n$. Otherwise merge two or four distinct options into one.

Examples when $n=4$ are: 'x_1 y_1 p_{11} p_{12} p_{21} p_{22} x_{13} y_{13}'; 'x_0 y_3 p_{03} x_{01} y_{34} t_1 x_3 y_0 p_{30} x_{34} y_{01}'; 'x_0 y_2 p_{02} x_{34} y_{23} t_1 x_1 y_3 p_{13} x_{12} y_{34} x_3 y_1 p_{31} x_{34} y_{12} x_2 y_0 p_{20} x_{23} y_{01}'; again with x_0 and y_0 suppressed.



(c) The unique solution for $n=10$ is shown. [The total number of such patterns for $n=(10, 11, \ldots, 16)$ turns out to be $(1, 0, 3, 6, 28, 20, 354)$. All 354 of the 16×16 solutions are found in only 560 megamems; they have orders 34, 36, and 38–44. Furthermore the number of $n\times n$ motley dissections with symmetry (a), for $n=(3, 4, 5, \ldots, 16)$, turns out to be $(1, 0, 2, 2, 8, 18, 66, 220, 1024, 4178, 21890, 102351, 598756, 3275503)$, respectively. Algorithm M needs 3.3 teramems when $n=16$; those patterns have orders $4k$ and $4k+1$ for $k=8, 9, \ldots, 13$.]

372. (a) The $r=n+1$ rows of B contain r +'s and r −'s; hence each vertex of H has in-degree 1 and out-degree 1. By connectedness, H must be an oriented r-cycle, $\vec{C_r}$.

(b) Now $n=r-m=2$; the $n+1$ rows of B must be $\pm\{1\bar{1}100, 00\bar{1}1\bar{1}, \bar{1}10\bar{1}1\}$.

(c) Proceed in two stages: First find all vectors orthogonal to A; then choose $n+1$ of them. For example, given the A for $\vec{K_4}$, there are options '$\#_j$ x_j:+', '$\#_j$ x_j:0', '$\#_j$ x_j:-', for $0\le j<6$. Then there are options such as '0 x_0:+ x_3:0 x_5:-', '0 x_0:0 x_3:- x_5:-', …, for the ways to be orthogonal to row 0 of A; and so on. (A row with 3 nonzero entries leads to 7 such options; a row with 4 nonzero entries leads to 19; etc.)

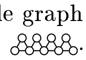
In the second stage, use all the nonzero solutions to the first stage to define an exact cover problem with corresponding options. The idea is to cover $\mathrm{P_j}$ and $\mathrm{N_j}$ for $0\le j<r$, where the option corresponding to a solution has $\mathrm{P_j}$ for all j with x_j:+ and $\mathrm{N_j}$ for all j with x_j:-. We also insert the special item # into each option, and give it multiplicity $n+1$, so that exactly $n+1$ options will be chosen. For example, the option that corresponds to the solution with x_0:+, x_1:+, x_2:0, x_3:-, x_4:0, x_5:0 is '# $\mathrm{P_0}$ $\mathrm{P_1}$ $\mathrm{N_3}$'.

Discard any solutions to the second stage that don't define a connected digraph.

(d) Stage 1 of the method outlined in (c) creates an XCC problem with $15+10$ items and 125 options; it has 219 solutions, one of which has all x_j:0. Stage 2 creates an MCC problem with 21 items and 218 options. Unfortunately it has no solutions that involve $n+1=7$ options. (It does have 195 solutions that involve 6; but they don't count.)

(e) Suppose horizontal segment i touches vertical segment j. Then j lies on opposite sides of two subrectangles, both above i or both below i. Hence exactly zero or two terms of every $a_i\cdot b_j$ are nonzero, and the signs cancel when there are two. (The same construction works when four subrectangles meet at a point, if we simply split either the horizontal segment or the vertical segment into two pieces at that point.)

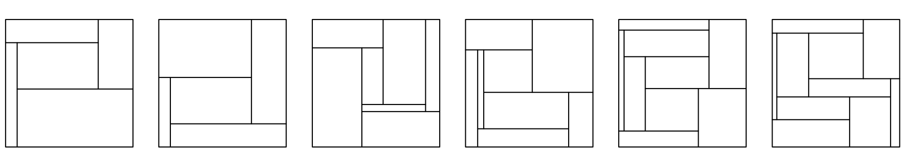
Notes: This theory was introduced by N. Biggs, *Proc. Cambridge Phil. Soc.* **65** (1969), 399–408, who also proved considerably more. For example, since AA^T is the Laplacian matrix of G, he was able to show that orthogonal digraphs G and H always have the same number of spanning trees (see exercise 7.2.1.6–104). If G is planar, and has no vertices of degree 1, it is orthogonal to its dual graph H, because we can rotate the direction of each edge by 90° after interchanging the roles of vertices and faces.

A single graph G might have many nonisomorphic orthogonal mates. (Consider, for example, .) But uniqueness might be provable if G is three-connected. No examples of *nonplanar* graphs with orthogonal mates are known.

373. The reduction of a perfectly decomposed rectangle is a motley dissection. Thus we can find all perfectly decomposed rectangles by "unreducing" all motley dissections.

For example, the only motley dissection of order 5 is the 3×3 pinwheel. Thus the perfectly decomposed $m \times n$ rectangles of order 5 with integer dimensions are the positive integer solutions to $x_1+x_2+x_3=m$, $y_1+y_2+y_3=n$ such that the ten values x_1, x_2, x_3, x_1+x_2, x_2+x_3, y_1, y_2, y_3, y_1+y_2, y_2+y_3 are distinct. Those equations are readily factored into two easy backtrack problems, one for m and one for n, each producing a list of five-element sets $\{x_1, x_2, x_3, x_1+x_2, x_2+x_3\}$; then we search for all pairs of disjoint solutions to the two subproblems. In this way we quickly see that the equations have just two essentially different solutions when $m = n = 11$, namely $(x_1, x_2, x_3) = (1, 7, 3)$ and $(y_1, y_2, y_3) = (2, 4, 5)$ or $(5, 4, 2)$. The smallest perfectly decomposed squares of order 5 therefore have size 11×11, and there are two of them (shown below); they were discovered by M. van Hertog, who reported them to Martin Gardner in May 1979. (Incidentally, a 12×12 square can also be perfectly decomposed.)

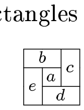
There are no solutions of order 6. Those of orders 7, 8, 9, 10 must come respectively from motley dissections of sizes 4×4, 4×5, 5×5, and 5×6. By looking at them all, we find that the smallest $n \times n$ squares respectively have $n = 18$, 21, 24, and 28. Each of the order-t solutions shown here uses rectangles of dimensions $\{1, 2, \ldots, 2t\}$, except in the case $t = 9$: There's a *unique* perfectly decomposed 24×24 square of order 9, and it uses the dimensions $\{1, 2, \ldots, 17, 19\}$.



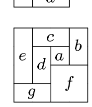
[W. H. Cutler introduced perfectly decomposed rectangles in *JRM* **12** (1979), 104–111.]

374. (a) False (but close). Let the individual dimensions be $z_1, \ldots, z_{2t}$, where $z_1 \le \cdots \le z_{2t}$. Then we have $\{w_1, h_1\} = \{z_1, z_{2t}\}$, $\{w_2, h_2\} = \{z_2, z_{2t-1}\}$, ..., $\{w_t, h_t\} = \{z_t, z_{t+1}\}$; consequently $z_1 < \cdots < z_t \le z_{t+1} < \cdots < z_{2t}$. But $z_t = z_{t+1}$ is possible.

(b) False (but close). If the reduced rectangle is $m \times n$, one of its subrectangles might be $1 \times n$ or $m \times 1$; a motley dissection must be *strict*.



(c) True. Label the rectangles $\{a, b, c, d, e\}$ as shown. Then there's a contradiction: $w_b > w_d \iff w_e > w_c \iff h_e < h_c \iff h_d < h_b \iff w_b < w_d$.



(d) The order can't be 6, because the reduction would then have to be a pinwheel together with a 1×3 subrectangle, and the argument in (c) would still apply. Thus the order must be 7, and we must show that the second dissection of exercise 365 doesn't work. Labeling its regions $\{a, \ldots, g\}$ as shown, we have $h_d > h_a$; hence $w_a > w_d$. Also $h_e > h_b$; so $w_b > w_e$. Oops: $w_f > w_g$ and $h_f > h_g$.

Biggs
Laplacian matrix
spanning trees
planar
dual graph
three-connected
pinwheel
factored
backtrack
van Hertog
Gardner
Cutler
strict

NP-hard
linear programming

In the other motley 4×4 dissection of exercise 365 we obviously have

$$w_4 < w_5, \quad w_4 < w_6, \quad w_6 < w_7, \quad h_4 < h_3, \quad h_3 < h_1, \quad h_4 < h_2;$$

therefore $h_4 > h_5$, $h_4 > h_6$, $h_6 > h_7$, $w_4 > w_3$, $w_3 > w_1$, $w_4 > w_2$. Now $h_5 < h_6 \iff w_5 > w_6 \iff w_2 > w_3 \iff h_2 < h_3 \iff h_6 + h_7 < h_5$. Hence $h_5 < h_6$ implies $h_5 > h_6$; we must have $h_5 > h_6$, thus also $h_2 > h_3$. Finally $h_2 < h_1$, because $h_7 < h_5$.

(e) The condition is clearly necessary. Conversely, given any such pair of solutions, the rectangles $w_1 \times \alpha h_1, \ldots, w_t \times \alpha h_t$ are incomparable for all large enough α.

[Many questions remain unanswered: Is it NP-hard to determine whether or not a given motley dissection supports an incomparable dissection? Is there a motley dissection that supports incomparable dissections having two different permutation labels? Can a *symmetric* motley dissection ever support an incomparable dissection?]

375. (a) By exercise 374(d), the widths and heights must satisfy

$$\begin{aligned} &w_5 = w_2 + w_4, \quad && w_6 = w_3 + w_4, \quad && w_7 = w_1 + w_3 + w_4;\\ &h_3 = h_4 + h_5, && h_2 = h_4 + h_6 + h_7, && h_1 = h_4 + h_5 + h_6. \end{aligned}$$

To prove the hint, consider answer 374(a). Each z_j for $1 \le j \le t$ can be either h or w; then z_{2t+1-j} is the opposite. So there are 2^t ways to shuffle the h's and w's together.

For example, suppose all the h's come first, namely $h_7 < \cdots < h_1 \le w_1 < \cdots < w_7$:

$$\begin{gathered} 1 \le h_7,\ \ h_7 + 1 \le h_6,\ \ h_6 + 1 \le h_5,\ \ h_5 + 1 \le h_4,\ \ h_4 + 1 \le h_4 + h_5,\\ h_4 + h_5 + 1 \le h_4 + h_6 + h_7,\ \ h_4 + h_6 + h_7 + 1 \le h_4 + h_5 + h_6,\\ h_4 + h_5 + h_6 \le w_1,\ \ w_1 + 1 \le w_2,\ \ w_2 + 1 \le w_3,\ \ w_3 + 1 \le w_4,\\ w_4 + 1 \le w_2 + w_4,\ \ w_2 + w_4 + 1 \le w_3 + w_4,\ \ w_3 + w_4 + 1 \le w_1 + w_3 + w_4. \end{gathered}$$

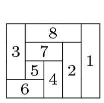
The least perimeter in this case is the smallest value of $w_1 + w_2 + w_3 + w_4 + h_7 + h_6 + h_5 + h_4$, subject to those inequalities; and one easily sees that the minimum is 68, achieved when $h_7 = 2$, $h_6 = 3$, $h_5 = 4$, $h_4 = 5$, $w_1 = 12$, $w_2 = 13$, $w_3 = 14$, $w_4 = 15$.

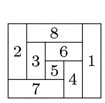
Consider also the alternating case, $w_1 < h_7 < w_2 < h_6 < w_3 < h_5 < w_4 \le h_4 < w_2{+}w_4 < h_4{+}h_5 < w_3{+}w_4 < h_4{+}h_6{+}h_7 < w_1{+}w_3{+}w_4 < h_4{+}h_5{+}h_6$. This case turns out to be infeasible. (Indeed, any case with $h_6 < w_3 < h_5$ requires $h_4 + h_5 < w_3 + w_4$, hence it needs $h_4 < w_4$.) Only 52 of the 128 cases are actually feasible.

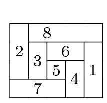
Each of the 128 subproblems is a classic example of linear programming, and a decent LP solver will resolve it almost instantly. The minimum perimeter with seven subrectangles is 35, obtained uniquely in the case $w_1 < w_2 < w_3 < h_7 < h_6 < h_5 < h_4 \le w_4 < w_5 < w_6 < w_7 < h_3 < h_2 < h_1$ (or the same case with $w_4 \leftrightarrow h_4$) by setting $w_1 = 1$, $w_2 = 2$, $w_3 = 3$, $h_7 = 4$, $h_6 = 5$, $h_5 = 6$, $h_4 = w_4 = 7$. The next-best case has perimeter 43. In one case the best-achievable perimeter is 103!

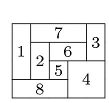
To find the smallest square, we simply add the constraint $w_1{+}w_2{+}w_3{+}w_4 = h_7{+}h_6 + h_5 + h_4$ to each subproblem. Now only four of the 128 are feasible. The minimum side, 34, occurs uniquely when $(w_1, w_2, w_3, w_4, h_7, h_6, h_5, h_4) = (3, 7, 10, 14, 6, 8, 9, 11)$.

(b) With eight subrectangles the reduced pattern is 4×5. We can place a 4×1 column at the right of either the 4×4 pattern or its transpose; or we can use one of the first two 4×5 patterns in exercise 365. (The other six patterns can be ruled out, using arguments similar to those of answer 374.) The labeled diagrams are



,


,


,


.

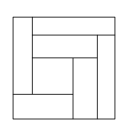
For each of these four choices there are 256 easy subproblems to consider. The best perimeters are respectively (44, 44, 44, 56); the best square sizes are respectively — and surprisingly — (27, 36, 35, 35). [With eight subrectangles we can dissect a significantly smaller square than we can with seven! Furthermore, no smaller square can be incomparably dissected, integerwise, because nine subrectangles would be too many.] One way to achieve perimeter 44 is with $(w_1, w_2, w_3, w_4, w_5, h_8, h_7, h_6, h_4) = (4, 5, 6, 7, 8, 1, 2, 3, 8)$ in the third diagram. The only way to achieve a square of side 27 is with $(w_1, w_2, w_3, w_4, w_5, h_8, h_7, h_6, h_4) = (1, 3, 5, 7, 11, 4, 6, 8, 9)$ in the first diagram.

These linear programs usually have integer solutions; but sometimes they don't. For example, the optimum perimeter for the second diagram in the case $h_8 < h_7 < w_1 < h_6 < w_2 < w_3 < w_4 < h_5$ turns out to be 97/2, achievable when $(w_1, w_2, w_3, w_4, w_5, h_8, h_7, h_6, h_4) = (7, 11, 13, 15, 17, 3, 5, 9, 17)/2$. The minimum rises to 52, if we restrict to integer solutions, achieved by $(w_1, w_2, w_3, w_4, w_5, h_8, h_7, h_6, h_4) = (4, 6, 7, 8, 9, 1, 3, 5, 9)$.

[The theory of incomparable dissections was developed by A. C. C. Yao, E. M. Reingold, and B. Sands in *JRM* **8** (1976), 112–119. For generalizations to three dimensions, see C. H. Jepsen, *Mathematics Magazine* **59** (1986), 283–292.]

376. This is an incomparable dissection in which exercise 374(d) applies. Let's try first to solve the equations $a(x+y+z) = bx = c(w+x) = d(w+x+y) = (a+b)w = (b+c)y = (b+c+d)z = 1$, by setting $b = x = 1$. We find successively $c = 1/(w+1)$, $a = (1-w)/w$, $y = (w+1)/(w+2)$, $d = (w+1)/(w(w+2))$, $z = (w+1)(w+3)/((w+2)(w+4))$. Therefore $x+y+z-1/a = (2w+3)(2w^2+6w-5)/((w-1)(w+2)(w+4))$, and we must have $2w^2+6w = 5$. The positive root of this quadratic is $w = (\surd-3)/2$, where $\surd = \sqrt{19}$.

Having decomposed the rectangle $(a+b+c+d)\times(w+x+y+z)$ into seven different rectangles of area 1, we normalize it, dividing (a, b, c, d) by $a+b+c+d = \frac{7}{15}(\surd+1)$ and dividing (w, x, y, z) by $w+x+y+z = \frac{5}{6}(\surd-1)$. This gives the desired tiling (shown), with rectangles of dimensions $\frac{1}{14}(7-\surd)\times\frac{1}{15}(7+\surd)$, $\frac{5}{42}(-1+\surd)\times\frac{1}{15}(1+\surd)$, $\frac{5}{21}\times\frac{3}{5}$, $\frac{1}{21}(8-\surd)\times\frac{1}{15}(8+\surd)$, $\frac{1}{21}(8+\surd)\times\frac{1}{15}(8-\surd)$, $\frac{5}{42}(1+\surd)\times\frac{1}{15}(-1+\surd)$, $\frac{1}{14}(7+\surd)\times\frac{1}{15}(7-\surd)$.



[See W. A. A. Nuij, *AMM* **81** (1974), 665–666. To get eight different rectangles of area 1/8, we can shrink one dimension by 7/8 and attach a rectangle $(1/8)\times 1$. Then to get nine of area 1/9, we can shrink the *other* dimension by 8/9 and attach a $(1/9)\times 1$ sliver. And so on. The eight-rectangle problem also has two other solutions, supported by the third and fourth 4×5 patterns in exercise 375(b).]

377. (a) We can obtain $h\times w$ except when w is odd and h is not a multiple of 3. For if w is even, we can concatenate $w/2$ instances of size $h\times 2$; if h is a multiple of 3, we can concatenate $h/3$ instances of size $3\times w$; otherwise we can't use concatenation to obtain w as the sum of two even numbers, or h as the sum of two multiples of 3.

(b) The shapes 2×3, 2×4, 2×5, 3×4, 3×5, 3×6, 3×7 are necessary and sufficient. (And then $\Lambda(S) = \{\,h\times w \mid h > 1, w > 3\} \cup \{\,2h\times 3 \mid h \ge 1\}$.)

(c) $S = \{2\times 4, 3\times 8, 4\times 2, 8\times 3\}$.

(d) $h\times w \in S$ if and only if $h = an'$ for some a with $\lfloor m/n'\rfloor < a < 2\lfloor m/n'\rfloor + 2$ and $w = bn''$ for some b with $\lfloor m/n''\rfloor < b < 2\lfloor m/n''\rfloor + 2$, where $n' = n/\gcd(n, w)$ and $n'' = n/\gcd(n, h)$.

378. Consider first a one-dimensional analog: If A is a set of positive integers, let $\Lambda(A)$ be the integers obtainable by adding together one or more elements of A. We can prove that any set B of positive integers has a subset A such that $B \subseteq \Lambda(A)$. For if B is empty, there's nothing to prove; otherwise let $b = \min(B)$. Let q_r be the smallest

integer programming
Yao
Reingold
Sands
Jepsen
Nuij

element of B such that $q_r \bmod b = r$, for $0 \le r < b$, or let q_r be undefined if no such element exists. Then every element of B is some q_r plus a multiple of $q_0 = b$.

Therefore in two dimensions, there's a finite set $X = \{h_1{\times}w_1, \ldots, h_t{\times}w_t\} \subseteq T$ such that the width of every element of T is in $\Lambda(X^*)$, where $X^* = \{w_1, \ldots, w_t\}$ is the set of widths in X. Let $p = h_1 \ldots h_t$ be the product of all heights in X. It follows that $p{\times}w \in \Lambda(X)$ whenever $h{\times}w \in T$.

For $0 \le r < p$, let T_r be the elements $h{\times}w$ of T with $h \bmod p = r$, and let Q_r be a finite subset of T_r such that every element of T_r has a width in $\Lambda(Q_r^*)$. Let q be the largest height of any element of any Q_r. Notice that if $h{\times}w \in T$ with $h > q$, and if $h'{\times}w' \in Q_{h \bmod p}$, we have $h{\times}w' \in \Lambda(X \cup Q_r)$, because $p{\times}w' \in \Lambda(X)$ and $h - h'$ is a positive multiple of p. Hence $h{\times}w \in \Lambda\bigl(\{h{\times}w' \mid h'{\times}w' \in Q_r\}\bigr) \subseteq \Lambda(X \cup Q_r)$.

Finally, for $1 \le i \le q$, let T'_i be the elements $h{\times}w$ of T with $h = i$, and let P_i be a finite subset of T'_i such that every element of T'_i has a width in $\Lambda(P_i^*)$. Then every element of T belongs to $\Lambda(X \cup Q_0 \cup \cdots \cup Q_{p-1} \cup P_1 \cup \cdots \cup P_q)$.

[This argument extends to any number of dimensions. See N. G. de Bruijn and D. A. Klarner, *Philips Research Reports* **30** (1975), 337*–343*; Michael Reid, *J. Combinatorial Theory* **A111** (2005), 89–105.]

379. A 2×5 packing is obvious; thus the basis contains 2×5 (and 5×2). The case $5{\times}w$ and $w > 2$ has a packing only if $h{\times}(w-2)$ does. The case $h = 3$ is clearly impossible.

The case $h = 7$ is more interesting: 7×10 follows by concatenation, while 7×15 has 80 distinct and easily found solutions. Hence the basis contains 7×15 and 15×7.

This basis is complete: We've shown that if h is not a multiple of 5, $h{\times}w$ is possible whenever w is a multiple of 5, except when $h = 1$ or $h = 3$ or ($h = 5$ and w is odd) or (h is odd and $w = 5$). If h and w are both multiples of 5, $h{\times}w$ is possible except when h or w equals 5 and the other is odd. [See W. R. Marshall, *J. Comb. Theory* **A77** (1997), 181–192; M. Reid, *J. Comb. Theory* **A80** (1997), 106–123.]

380. The minimum basis consists of 15×15 (see Fig. 73) plus 39 pairs $\{h{\times}w, w{\times}h\}$, where $(h,w) \in \{(5,10),\ (9,20),\ (9,30),\ (9,45),\ (9,55),\ (10,14),\ (10,16),\ (10,23),\ (10,27),\ (11,20),\ (11,30),\ (11,35),\ (11,45),\ (12,50),\ (12,55),\ (12,60),\ (12,65),\ (12,70),\ (12,75),\ (12,80),\ (12,85),\ (12,90),\ (12,95),\ (13,20),\ (13,30),\ (13,35),\ (13,45),\ (14,15),\ (15,16),\ (15,17),\ (15,19),\ (15,21),\ (15,22),\ (15,23),\ (17,20),\ (17,25),\ (18,25),\ (18,35),\ (22,25)\}$. (This problem has a long history, going back to the discovery by David Klarner that ten *one-sided* Y pentominoes can be packed uniquely in a 5 × 10 box [*Fibonacci Quarterly* **3** (1965), 20]. Klarner eventually found 14 of the 39 basic pairs by hand, including the difficult case (12, 80). The other nine cases $(12, w)$ were found by J. Bitner [*JRM* **7** (1974), 276–278], using a frontier-transition method that works much faster than Algorithm X in cases where h is much less than w. The complete set was nailed down by T. Sillke in 1992 [unpublished], then independently by J. Fogel, M. Goldenberg, and A. Liu [*Mathematics and Informatics Quarterly* **11** (2001), 133–137].)

381. Algorithm X quickly finds examples for $n = 7$, 11, 12, 13, 15, 16, 17; hence it's possible for all $n \ge 11$. [J. B. Kelly discovered the case $n = 7$ in *AMM* **73** (1966), 468. Are *all* packable rectangles consequences of this basis?]

382. Let the back corner in the illustration be the point 777, and write just '*abcdef*' instead of $[a\,..\,b){\times}[c\,..\,d){\times}[e\,..\,f)$. The subcuboids are 670517 (270601) 176705 (012706) 051767 (060127), 561547 (260312) 475615 (122603) 154756 (031226), 351446 (361324) 463514 (243613) 144635 (132436), 575757 (020202), 454545 (232323) — with the 11 mirror images in parentheses — plus the central cubie 343434. Notice that each of the 28 possible intervals is used in each dimension, except $[0\,..\,4)$, $[1\,..\,6)$, $[2\,..\,5)$, $[3\,..\,7)$, $[0\,..\,7)$.

de Bruijn
Klarner
Reid
Marshall
Reid
Bitner
frontier
Sillke
Fogel
Goldenberg
Liu
Kelly
basis

I started from a central cube and built outwards, all the while staring at the 24-cell in Hilbert's Geometry and the Imagination.
— SCOTT KIM, letter to Martin Gardner (December 1975)

Hilbert
KIM
Gardner
Postl
puzzle
Postl
author
Cutler
Kim

383. (Solution by Helmut Postl.) We can use the 7-tuples $(2, 10, 27, 17, 11, 20, 5)$, $(1, 14, 18, 8, 21, 24, 6)$, $(3, 19, 16, 7, 34, 9, 4)$ to "unreduce" the 1st, 2nd, 3rd coordinates. For example, subcuboid 670517 becomes $5 \times (1{+}14{+}18{+}8{+}21) \times (19{+}16{+}7{+}34{+}9{+}4)$. The resulting dissection, into blocks of sizes $1 \times 70 \times 87$, $2 \times 77 \times 88$, $3 \times 80 \times 86$, $4 \times 67 \times 91$, $5 \times 62 \times 89$, $6 \times 79 \times 90$, $7 \times 8 \times 17$, $9 \times 51 \times 65$, $10 \times 38 \times 71$, $11 \times 21 \times 34$, $12 \times 15 \times 22$, $13 \times 25 \times 30$, $14 \times 39 \times 66$, $16 \times 18 \times 27$, $19 \times 33 \times 75$, $20 \times 47 \times 61$, $23 \times 32 \times 48$, $24 \times 36 \times 76$, $26 \times 37 \times 50$, $28 \times 40 \times 43$, $29 \times 31 \times 42$, $35 \times 44 \times 53$, $41 \times 45 \times 54$, makes a fiendishly difficult puzzle.

How were those magic 7-tuples discovered? An exhaustive search such as that of exercise 374 was out of the question. Postl first looked for 7-tuples that led to very few dimensions in the "popular" ranges $[13\,..\,23]$ and $[29\,..\,39]$. With luck, a large set of other 7-tuples would lead to no conflict in the 23 relevant subtotals; and with further luck, some of those wouldn't conflict with each other.

(Postl also proved that no $91 \times 91 \times 91$ decomposition is possible.)

384. The exact cover problem of answer 365 is readily extended to 3D: The option for every admissible subcuboid $[a\,..\,b) \times [c\,..\,d) \times [e\,..\,f)$ has $6 + (b-a)(d-c)(f-e)$ items, namely x_a y_c z_e x_{ab} y_{cd} z_{ef} and the cells p_{ijk} that are covered.

We can do somewhat better, as in exercise 366: Most of the improvement in that answer can be achieved also 3Dwise, if we simply omit cases where $a = l-1$ and either $c + d > m$ or $e + f > n$. Furthermore, if $m = n$ we can omit cases with $(e, f) < (c, d)$.

Without those omissions, Algorithm M handles the case $l = m = n = 7$ in 98 teramems, producing 2432 solutions. With them, the running time is reduced to 43 teramems, and 397 solutions are found.

(The $7 \times 7 \times 7$ problem can be factored into subproblems, based on the patterns that appear on the cube's six visible faces. These patterns reduce to 5×5 pinwheels, and it takes only about 40 Mμ to discover all 152 possibilities. Furthermore, those possibilities reduce to only 5 cases, under the 48 symmetries of a cube. Each of those cases can then be solved by embedding the 5×5 reduced patterns into 7×7 unreduced patterns, considering $15^3 = 3375$ possibilities for the three faces adjacent to vertex 000. Most of those possibilities are immediately ruled out. Hence each of the five cases can be solved by Algorithm C in about 70 Gμ — making the total running time about 350 Gμ. However, this 120-fold increase in speed cost the author two man-days of work.)

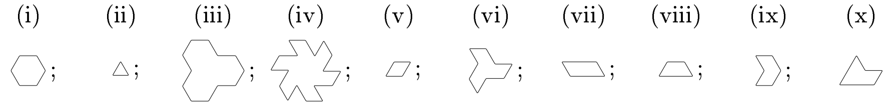
All three methods showed that, up to isomorphism, exactly 56 distinct motley cubes of size $7 \times 7 \times 7$ are possible. Each of those 56 dissections has exactly 23 cuboids. Nine of them are symmetric under the mapping $xyz \mapsto (7-x)(7-y)(7-z)$; and one of those nine, namely the one in exercise 382, has six automorphisms.

[These runs confirm and slightly extend the work of W. H. Cutler in *JRM* **12** (1979), 104–111. His computer program found exactly 56 distinct possibilities, when restricting the search to solutions that have exactly 23 cuboids.]

385. No; there are infinitely many. For example, Postl has constructed a primitive $11 \times 11 \times 13$ by pasting Kim's $7 \times 7 \times 7$ to its mirror image, perturbing a few planes normal to the splice, and reducing.

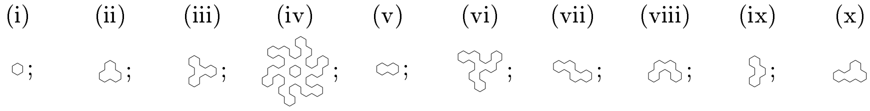
386. The twelve possible symmetries can be represented as the permutations of $\{0, 1, 2, 3, 4, 5\}$ defined by $x \mapsto (ax + b) \bmod 6$, where $a = \pm 1$ and $0 \le b < 6$; let's denote that permutation by b or $\bar{b}$, according to the sign of a. There are ten symmetry classes, depending on the automorphisms that are present: (i) all twelve; (ii) $\{0, \overline{0}, 2, \overline{2}, 4, \overline{4}\}$; (iii) $\{0, \overline{1}, 2, \overline{3}, 4, \overline{5}\}$; (iv) $\{0, 1, 2, 3, 4, 5\}$; (v) $\{0, 3, \overline{0}, \overline{3}\}$ or $\{0, 3, \overline{2}, \overline{5}\}$ or $\{0, 3, \overline{4}, \overline{1}\}$; (vi) $\{0, 2, 4\}$; (vii) $\{0, 3\}$; (viii) $\{0, \overline{0}\}$ or $\{0, \overline{2}\}$ or $\{0, \overline{4}\}$; (ix) $\{0, \overline{1}\}$ or $\{0, \overline{3}\}$ or $\{0, \overline{5}\}$; (x) $\{0\}$.

(i) (ii) (iii) (iv) (v) (vi) (vii) (viii) (ix) (x)



full triaxial-a triaxial-b 60° biaxial 120° 180° axial-a axial-b none

(i) (ii) (iii) (iv) (v) (vi) (vii) (viii) (ix) (x)



full triaxial-a triaxial-b 60° biaxial 120° 180° axial-a axial-b none

(Types (ii), (iii) and (viii), (ix) depend on whether a reflection is left-right or top-down. Notice that there are $12/k$ base placements when there are k automorphisms.)

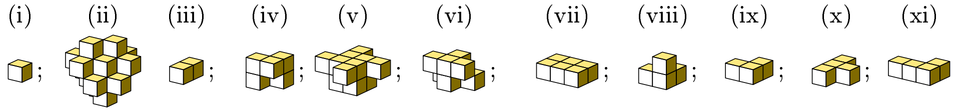
387. The 24 potential symmetries S can be represented as signed permutations of $\{\pm1, \pm2, \pm3\}$, meaning that coordinates are permuted and/or complemented. Using the notation of answer 7.2.1.2–20, they are 123, $1\overline{2}\overline{3}$, $\overline{1}2\overline{3}$, ..., $\overline{3}\overline{2}\overline{1}$, where the number of inversions of the permutation plus the number of complementations is even.

Each of those symmetries is a rotation in 3-space about some line through the origin. (After a polycube has been rotated by one of its symmetries, we should shift the result, if necessary, to bring it into the original position.) For example, $\overline{1}32$ takes $(x, y, z) \mapsto (c - x, z, y)$; it's a rotation of 180° about the diagonal line $x = c/2$, $y = z$. It's a symmetry of the bent tricube $\{000, 001, 010\}$ when $c = 0$; it's a symmetry of the L-twist $\{000, 001, 100, 110\}$ when $c = 1$.

All subgroups of this group are easily found by constructing the BDD for the Boolean function whose 24 variables are the potential symmetries. Indeed, all subsets of any set S that are closed under any given binary operator $\star$ on that set are the solutions to $\bigwedge_{x,y \in S}(\neg x \vee \neg y \vee (x \star y))$. In this case the resulting BDD (found in 2.5 Mμ) has 197 nodes, and it characterizes exactly 30 subgroups.

Two subgroups T and T' are said to be *conjugate* if $T' = t^{-}Tt$ for some $t \in S$. Such subgroups are considered to be equivalent, because they amount to viewing the objects from a different direction. The distinct conjugacy classes of subgroups according to this equivalence relation are called the "symmetry types," and there are 11 of them:

(i) (ii) (iii) (iv) (v) (vi) (vii) (viii) (ix) (x) (xi)



full even 8-fold 6-fold 90° bidiagonal tricentral 120° diagonal axial none

Class (ii) consists of the 12 symmetries whose permutations are even. The smallest polycube which admits these symmetries and no more — and hence it has just two base placements — contains 20 cubies, with 12 surrounding a central core of 8. Class (iv) has one symmetry for each permutation of the three coordinates. Classes (iii), (v), (vi), (vii), (ix), (x), (xi) correspond to the eight symmetry types of a square, with reflections implemented by "turning the square over" to the opposite side. In this interpretation biaxial symmetry becomes "tricentral," because it corresponds to central symmetry about each coordinate axis. The former class called "180°" is now the same as "axial,"

automorphisms
biaxial
triaxial
axial
60°
120°
180°
base placements
signed permutations
rotation
bent tricube
L-twist
subgroups
BDD
Boolean function
binary operator
groupoid, see binop
magma, see binop
conjugate
symmetry types
even
biaxial symmetry
central symmetry

when viewed from either of the two other axes. [Many of these twelve examples have reflective symmetries too; but those don't count. Under the full set of 48 hyperoctahedral symmetries, when reflections are allowed, there are 33 symmetry types(!), nicely presented by W. F. Lunnon in *Graph Theory and Computing* (Academic Press, 1972), 101–108. Lunnon also exhibited the ten symmetry types for polyhexes on pages 87–100.]

388. The directed path of four weak clues in (a) is equivalent to the five strong clues (1, 2, 3, 4, 5). Then there are "hidden singles" in columns 1 and 2, leading to a "naked single" in cell $(4,2)$, etc.; we cruise to victory without branching. Puzzle (b) has a naked single in $(4,2)$ — and we notice, by the way, that the middle cell needn't be 5 even though it is greater than each of its four neighbors. Then $(4,4)$ is naked, and so on; again everything is forced. Puzzle (c) begins with hidden singles, which place the three missing 1s and then the 5 in row 0. After we fix cell $(4,2)$, the rest falls into place.

```
a)  4  5  1  3  2          b)  3 >1  4  5  2          c)  3  5  1  2 <4
                ∧                                         
    5 >4  2 >1  3              2  3  1  4  5              4  1  5  3  2
                                     ∧     ∨
    3  1  5  2  4 ;            5  2 <3 >1  4 ;            2  3  4  5  1 .
          ∨                          ∨
    2  3 <4  5  1              4  5  2  3  1              5  4  2  1  3
       ∨                                   ∧
    1 <2  3  4  5              1  4 <5  2  3              1  2 <3  4  5
```

[*Historical note:* Futoshiki was invented by Yoshihiko Asao, who called it Dainarism ("Greater Than"); see *Puzzle Communication Nikoli* **92** (September 2000).]

389. In general, given a digraph in which each vertex v is supposed to be given an integer label $l(v)$ with $l(v) \geq a(v)$, where the lower bounds $a(v)$ have been specified, we can refine them as follows: For each vertex with $d^+(v) > 0$, push $v \Rightarrow S$, where S is an initially empty stack. Then while S is nonempty, repeatedly do this: Pop $S \Rightarrow v$; for each w with $v \longrightarrow w$ and $a(w) \leq a(v)$, set $a(w) \leftarrow a(v)+1$, and push $w \Rightarrow S$ if $d^+(w) > 0$.

A similar algorithm will refine a given set of upper bounds $b(v)$. For futoshiki, we apply these algorithms with $a(v) = 1$ and $b(v) = n$ initially, except that $a(v) = b(v) = l$ when a strong clue has specified v's label. (*Note:* This method isn't clever enough to prove that the middle element of puzzle (b) must be 3 or more. But it's still very useful.)

390. In both cases we use primary items p_{ij}, r_{ik}, and c_{jk} for $0 \leq i, j < n$ and $1 \leq k \leq n$, as we did for sudoku. There will be one option analogous to (30) for every (i,j) and for every $k \in [a_{ij} \mathinner{.\,.} b_{ij}]$, where the bounds a_{ij} and b_{ij} are calculated as in exercise 389.

(a) Suppose there are w weak clues, where the tth weak clue is $l(i_t j_t) < l(i'_t j'_t)$. Introduce $(n-3)w$ secondary items g_{td} for $1 < d < n-1$ and $1 \leq t \leq w$. Such an item informally means that $l(i_t j_t) > d$ and $d \geq l(i'_t j'_t)$; so we don't want it to appear twice. We include g_{td} in each option for ij with $d < k$, and in each option for $i'j'$ with $d \geq k$.

For example, the options for cells $(0,0)$ and $(0,1)$ in puzzle 388(b) are 'p_{00} r_{02} c_{02} g_{12} g_{13}', 'p_{00} r_{03} c_{03} g_{13}', 'p_{00} r_{04} c_{04}', 'p_{00} r_{05} c_{05}'; 'p_{01} r_{01} c_{11}', 'p_{01} r_{02} c_{12}', 'p_{01} r_{03} c_{13} g_{12}', 'p_{01} r_{04} c_{14} g_{12} g_{13}'. Another option is 'p_{22} r_{23} c_{23} g_{23} g_{33} g_{43} g_{53}'.

(b) Introduce w primary items g_t, and $3n^2$ secondary items P_{ij}, R_{ik}, C_{jk}. The options for p_{ij}, r_{ik}, and c_{jk} are 'p_{ij} r_{ik} c_{jk} $P_{ij}{:}k$ $R_{ik}{:}j$ $C_{jk}{:}i$' for $0 \leq i, j < n$ and $a_{ij} \leq k \leq b_{ij}$. The options for g_t are 'g_t $P_{i_t j_t}{:}k$ $P_{i'_t j'_t}{:}k'$ $R_{i_t k}{:}j_t$ $R_{i'_t k'}{:}j'_t$ $C_{j_t k}{:}i_t$ $C_{j'_t k'}{:}i'_t$' for $k < k'$, where k and k' are within the bounds for $l(i_t j_t)$ and $l(i'_t j'_t)$.

Experience shows that formulation (a) is a clear winner over formulation (b).

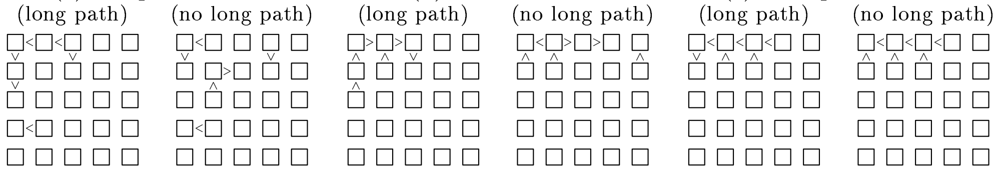
391. Given $5 \cdot 5 \cdot 5$ options 'p_{ij} r_{ik} c_{jk}' as in answer 390, Algorithm X needs just 230 megamems to generate $161280 = 56 \cdot 5! \cdot 4!$ solutions. [Euler enumerated them in his major paper on latin squares [*Verhandelingen Genootschap Wetenschappen Vlissingen*

hyperoctahedral symmetries
Lunnon
polyhexes
hidden singles
naked single
Historical note
Asao
Dainarism
Nikoli
digraph
out-degree $d^+(v)$
stack
Euler

9 (1782), 85–239, §148], though he was nearly blind at the time.] Every 5×5 latin square has 40 pairs of adjacent elements, leading to a string of 40 inequality signs; and we can sort those 161280 strings. Only 115262 distinct strings actually occur; and only 82148 of them occur just once. The other 79132 cannot be identified by weak clues only.

392. Here are the first examples found of each type, and the total number of cases:

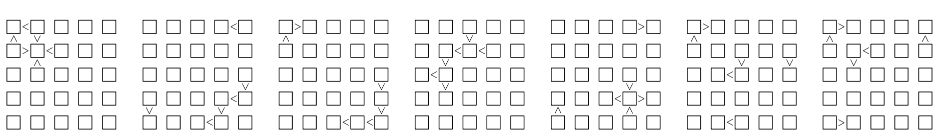
(a) Unique solution — (long path), (no long path); (b) No solutions — (long path), (no long path); (c) Multiple solutions — (long path), (no long path)



2976 4000 369404 405636 1888424 242985880

(More detailed counting shows exactly (369404, 2976, 4216, 3584, …, 80) cases with at least one long path and (0, 1, 2, 3, …, 1344) solutions; (405636, 4000, 4400, 1888, …, 72) cases with no long path and (0, 1, 2, 3, …, 24128) solutions.) Example (i) below is one way to get the maximum number of solutions, using six particularly unhelpful clues.

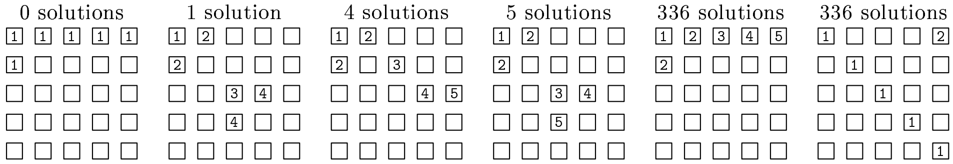
The most interesting cases, of course, are those that make valid puzzles. They fall into equivalence classes under rotation and/or reflection and/or complementation; thus sixteen examples are typically equivalent to any given one. However, there are 46 equivalence classes with only eight members, self-dual under transposition, of which 26 have long paths (as in (ii), (iii), (iv) below) and 18 do not (as in (v), (vi), (vii)). Thus $(173{+}26)+(241{+}18) = 458$ essentially different futoshiki puzzles with six weak clues are valid; however, many of these are really the same, under row-and-column permutations that preserve all clues. The most difficult symmetric instance is probably (vii), because exercise 390 needs a 374-node search tree to solve it. (A clever solver will, however, deduce immediately that all diagonal elements of a symmetric puzzle must be 3!)



(i) (ii) (iii) (iv) (v) (vi) (vii)

393. The $5^6 = 15625$ ways to label six cells can be reduced to $\varpi_6 = 203$, by limiting consideration to restricted growth strings (Section 7.2.1.5), multiplying the results for every such string by $5^{\underline{k}}$ when it has k different labels. (In fact, only 202 such strings are relevant, because the last one (123456) will be multiplied by $5^{\underline{6}} = 0$ and never used.) Running through each subset of five cells, we find respectively (1877807500, 864000, 0, 0, 1296000, 10368000, …, 144000) cases that have (0, 1, 2, 3, 4, 5, …, 336) solutions.

0 solutions 1 solution 4 solutions 5 solutions 336 solutions 336 solutions

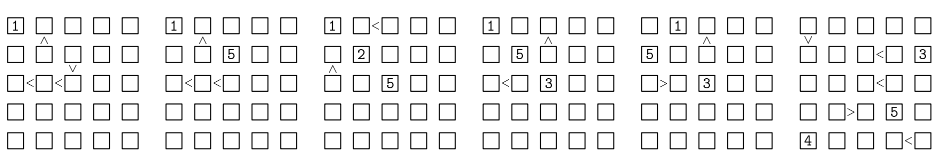


Every case with a unique solution is obtained from the example shown by independently permuting the rows, columns, and labels. (Indeed, $864000 = 5!^3/2$.)

string
sort
self-dual
transposition
Bell numbers
ϖ_n, the nth Bell number
restricted growth strings

Katz
Bean
preprocessing

394. Let there be h strong clues and $k = 5 - h$ weak clues. Four solutions are obtained only in (144, 2016, 2880) cases for $h = (1, 2, 3)$. In every such case, two rows and two columns are completely free from clues; thus the four solutions arise from swapping those two rows and/or those two columns. As in answer 392, most of the cases belong to classes of 16 puzzles that are equivalent under rotation, transposition, and/or complementation. But when $h = 3$ there are 30 classes of size 8, having transposition symmetry (see (iii) and (iv) below); also 6 self-dual classes of size 8 (see (v)). Hence there are $9 + 126 + (36 + 162) = 333$ inequivalent 4-solution 5-clue futoshikis altogether.



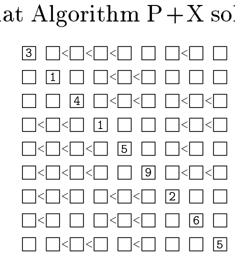
(i) (ii) (iii) (iv) (v) (vi)

[This exercise was inspired by a talk that Dan Katz gave at the Joint Mathematics Meetings in January 2012. He observed, among other things, that valid puzzles exist with $h + k = 6$ for all values $0 \le h \le 6$. Indeed, we can start with example (iv) in answer 392, and repeatedly insert a clue (5, 1, 5, 1, 4, 2) while removing an inequality.]

[The minimum number of strong clues needed to specify an $n \times n$ latin square is known to be $\lfloor n^2/4 \rfloor$ for $n \le 8$. See R. Bean, arXiv:math/0403005 [math.CO] (2004).]

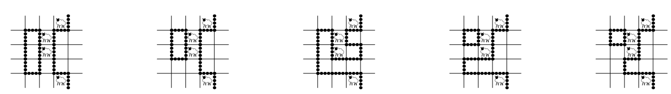
395. Let L solve (vi) in answer 394. [See Appendix E if you're stuck.] The only way to distinguish L from fifteen other latin squares that have the same string of 40 inequality signs is to give at least one clue 2 or 3 in a boundary row or column, at least one clue 4 or 5 in a boundary row or column, and at least one 4 or 5 in cells $\{(1,1), (1,3), (3,1), (3,3)\}$.

396. For example, here's one that Algorithm P+X solves in 90 Mμ. (See Appendix E.)



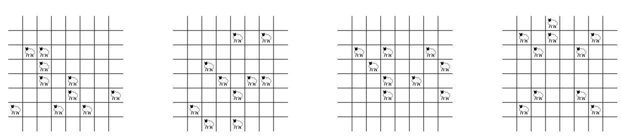
397. (a) Assuming an $m \times n$ grid, let there be $(m{+}1)(n{+}1) - 4$ primary "endpoint" items ij for $0 \le i \le m$, $0 \le j \le n$, and $[i=0] + [i=m] + [j=0] + [j=n] \le 1$; also "sheep" items s_{ij} when a sheep is in cell ij; also "start-stop" items + and −. Let there be mn secondary items x_{ij} for $0 \le i < m$ and $0 \le j < n$, one for each cell. Three kinds of options are used: (i) There are $14(m{-}1)(n{-}1)$ "junction" options 'ij $x_{(i-1)(j-1)}{:}a$ $x_{(i-1)j}{:}b$ $x_{ij}{:}c$ $x_{i(j-1)}{:}d$', for $0 < i < m$ and $0 < j < n$ and $0 \le a, b, c, d \le 1$ and $(a = b$ or $b = c$ or $c = d)$. (ii) There are $2m + 2n - 4$ sets of four "boundary" options typified by '02 $x_{01}{:}0$ $x_{02}{:}0$', '02 $x_{01}{:}0$ $x_{02}{:}1$ −', '02 $x_{01}{:}1$ $x_{02}{:}0$ +', '02 $x_{01}{:}1$ $x_{02}{:}1$', for $0 \le i \le m$, $0 \le j \le n$, and $[i=0] + [i=m] + [j=0] + [j=n] = 1$; adjacent boundary cells, like x_{01} and x_{02} in this example, are listed in clockwise order. (For example, one of the options at the right boundary when $n = 5$ is '35 $x_{24}{:}0$ $x_{34}{:}1$ −'; one of the options at the left is '20 $x_{20}{:}1$ $x_{10}{:}0$ +'.) (iii) Each sheep has up to six "sheep" options, 's_{ij} $x_{ij}{:}1$ $x_{(i-1)j}{:}a$ $x_{i(j+1)}{:}b$ $x_{(i+1)j}{:}c$ $x_{i(j-1)}{:}d$', where $a + b + c + d = 2$; the x items are omitted if the corresponding cells lie outside of the grid, in which case their values are assumed to be 1. For example, the topmost sheep has only three options in the example puzzles, namely 's_{03} x_{03}:1 x_{04}:b x_{13}:c x_{02}:d', where $b + c + d = 1$.

This XCC problem for the rightmost example puzzle has five solutions:



To eliminate the spurious ones, we traverse the fence from '+' to '−', accepting a solution only if that path contains all of the color transitions between adjacent cells.

(b) There's a unique solution if we put k sheep into a diagonal of length k; but that puzzle is trivial, not "interesting." Random trials show that about one configuration in every 10,000 makes a suitable puzzle; the author found the first three examples below in that way. The fourth example was contrived by hand. All are solvable by hand:



[E. Olson invented this game; see J. Henle, *Math. Intelligencer* **40**, 1 (2018), 69–70.]

398. The blanks in rows 0 and 4 of (c) can be filled with 3 and 5 in two ways.

a)

3− 1	4	14+ 2	15× 3	5
9× 3	1	5	2	2÷ 4
6+ 4	3	5+ 1	5	2
2	3− 5	4	5+ 1	3
5 5	2	7+ 3	4	1

;

b)

34560× 4	2	3	1	3− 5
3	5÷ 1	5	4	2
5	3	4	2	10+ 1
3+ 3	9+ 4	1	5	3
1	5	2 2	1− 3	4

;

c)

3− 4	1	14+ 2	15×	
9× 1	3	5	2÷ 2	6+ 4
3	5 5	4	1	2
3− 5	2	3	5+ 4	1
8× 2	4	9+ 1		

.

[Tetsuya Miyamoto invented KenKen® in 2004, as an aid to education. The special case where all operations are multiplication, and all cages are rectangular, had been published by Tatsuo Yano in *Puzzle Communication Nikoli* **92** (September 2000).]

399. Set up an XCC problem with $3n^2$ primary items p_{ij}, r_{ik}, c_{jk} and $3n^2$ secondary items P_{ij}, R_{ik}, C_{jk}, and with n^2 options 'p_{ij} r_{ik} c_{jk} P_{ij}:k R_{ik}:j C_{jk}:i' for $0 \le i, j < n$ and $1 \le k \le n$, as in answer 390(b). Also, if there are w cages, introduce primary items g_t for $1 \le t \le w$. Let C_t be the cells of the tth cage, and let there be an option

$$\text{`}g_t \bigcup \{ \{ P_{i_t j_t}{:}l(i_t,j_t),\ R_{i_t l(i_t,j_t)}{:}j_t,\ C_{j_t l(i_t,j_t)}{:}i_t \} \mid (i_t,j_t) \in C_t \}\text{'}$$

for every feasible way to assign labels $l(i_t,j_t)$ to the cells of C_t. For example, there are two labelings that satisfy the clue '15×' in the third cage in puzzle 398(a), namely either $l(0,3) = 3$ and $l(0,4) = 5$ or $l(0,3) = 5$ and $l(0,4) = 3$; the two options for g_3 are therefore 'g_3 P_{03}:3 R_{03}:3 C_{33}:0 P_{04}:5 R_{05}:4 C_{45}:0' and 'g_3 P_{03}:5 R_{05}:3 C_{35}:0 P_{04}:3 R_{03}:4 C_{43}:0'. The cage of that puzzle whose clue is '9×' has just one option: 'g_4 P_{10}:3 R_{13}:0 C_{03}:1 P_{11}:1 R_{11}:1 C_{11}:1 P_{21}:3 R_{23}:1 C_{13}:2'.

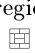
The option for a one-cell cage is trivial, and the options for two-cell cages are also easy. The options for larger cages are readily listed by a straightforward backtrack algorithm: We can represent unchosen labels in each row and column by bit vectors, just as unchosen values in the queens problem were represented in Algorithm 7.2.2B*. Simple upper and lower bounds on the final sum or product, given a partial labeling, yield satisfactory cutoffs in the analog to step B3* of that algorithm, based on the λ and ρ functions of Section 7.1.3. The ten-cell '34560×' cage of puzzle 398(b) turns out to have 288 options, with 31 items each; the links will dance merrily around them all.

author
Olson
Henle
historical note
Miyamoto
Yano
Nikoli
historical note
XCC problem
backtrack
bit vectors
queens problem
λ
ρ

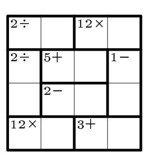
(Incidentally, this formulation doesn't require the cells of a cage to be connected.)

400. The formulation in answer 399 makes it easy to omit the options for any cage. Thus Algorithm C almost instantaneously breezes through those 2048 problems, and finds that exactly 499 of them are uniquely solvable. The number of such puzzles with (5, 6, ..., 11) given clues is (14, 103, 184, 134, 52, 11, 1); for example, one can solve it when given only the clues '15×', '6+', '3−', '5+', '5', in five cages! Exactly (14, 41, 6) of those 499 puzzles have (5, 6, 7) *minimal* clues; minimal-clue puzzles correspond to the prime implicants of the associated monotone Boolean function.

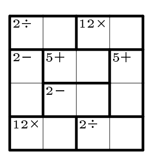
Similar remarks apply to puzzle 398(b), which can be solved uniquely *without* knowing either of the clues '34560×' or '2' — although the reader probably made heavy use of those clues when solving it. (On the other hand, its clue '9+' cannot be omitted.)

401. There are 36 ways to cover a 4×4 board with dominoes, but nearly all of them are unsuitable. For example,  can't define the cages of a valid kenken problem, because the middle rows of any solution could be swapped to give another solution. And no two dominoes can cover a 2 × 2 region whose solution has the form $\substack{ab\\ba}$. Therefore we're left with only two cage patterns,  and its transpose.

A given cage pattern can be filled with two clues of each type in $8!/2!^4 = 2520$ ways. Most of those ways are obviously impossible, because ÷ cannot be applied to the pairs $\{2,3\}$ or $\{3,4\}$. It turns out that (1620, 847, 52, 1) of the cases give a kenken puzzle with respectively (0, 1, 2, 4) solutions. Notable examples are



and



,

where the first is the "most difficult," in the sense that its search tree via the construction of exercise 399 has the most nodes (134). The second is the one with four solutions.

402. The solution follows answer 403. The author constructed this puzzle by first designing the cages, then generating a dozen or so random latin squares via exercise 86 until finding one that had a unique solution. Then the domino clues were permuted at random, ten times; the most difficult of those ten puzzles (77 meganodes) was selected.

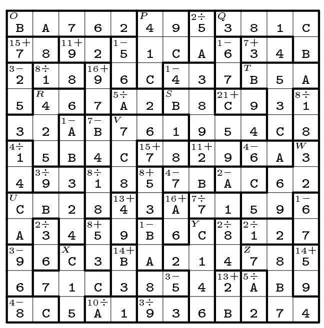
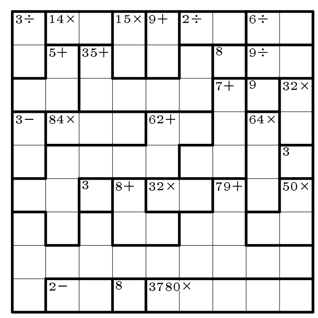
The construction of answer 399 gives an XCC problem with 10914 options, 486 + 432 items, and 163288 total entries. There are respectively (720, 684, 744, 1310, 990, 360, 792, 708, 568, 1200, 606, 30) options for the pentominoes (O, P, ..., Z); preprocessing with Algorithm P reduces those counts to (600, 565, 96, 1122, 852, 248, 744, 656, 568, 1144, 606, 26). Overall, the reduced problem has 8927 options, 484 + 432 items, and 134530 total entries. The total time to find the solution and prove its uniqueness was 9 Gμ for Algorithm P and 293 Gμ for Algorithm C. (Without preprocessing, Algorithm C would have taken 6.4 Tμ, and its search tree would have had 2 giganodes. Could a human being solve this puzzle by hand?)

403. The author's best attempt, shown below, manages to match 35 digits before deviating in the final cage. The construction of answer 399 fails spectacularly on this particular instance, because the monster cage for '79+' has 3,978,616,320 options! We can, however, work around that problem by simply making row 7 unconstrained and subtracting $1+2+\cdots+9 = 45$ from the cage total. (A latin square is determined by any $n-1$ of its rows.) Then Algorithm C solves the problem handily, with a cost of 2 Gμ (from 184422 options), and with a search tree of only 252 nodes. (See Appendix E.

prime implicants
monotone Boolean function
dominoes
author
preprocessing
author
latin square

Surprisingly, the non-π clue '3780×' in the bottom row affects row 1 of the solution.)

Hamiltonian *cycle*
simple path
Benedek
$P_m \boxtimes P_n$

404. Such puzzles can be defined on any N-vertex graph G, some of whose vertices are labeled with elements of $\{1, 2, \ldots, N\}$; the problem is to extend such a labeling to a full Hamiltonian path, in all possible ways. We imagine an additional vertex ∞, which is adjacent to all the others. A Hamiltonian path in G is then equivalent to a Hamiltonian *cycle* in $G \cup \infty$, with ∞ interposed between the first and last vertices of the path.

For $1 \le k \le N$, let v_k be the vertex labeled k, or $v_k = \Lambda$ if there's no such vertex. Also let $v_0 = v_{N+1} = \infty$. We define an XCC problem with two kinds of primary items: (i) $-v$ and $+v$ for all unlabeled vertices v; (ii) s_k for $0 \le k \le N$, except when both $v_k \ne \Lambda$ and $v_{k+1} \ne \Lambda$. We also introduce secondary items p_v for all unlabeled v, and q_k for all unused labels k. (Thus the example has 35 primary items $\{-00, +00, -10, +10, -11, \ldots, +33, s_1, \ldots, s_7, s_9, \ldots, s_{16}\}$, and 20 secondary items $\{p_{00}, \ldots, p_{33}, q_2, q_4, \ldots, q_{15}, q_{16}\}$.) The options for s_k are 's_k $-u$ p_u:k q_k:u $+v$ p_v:$k{+}1$ q_{k+1}:v' for all pairs of unlabeled vertices $u \mathrel{—} v$ such that u might be labeled k and v might be labeled $k+1$. However, we omit $-u$ p_u:k q_k:u if $v_k \ne \Lambda$, and we omit $+v$ p_v:$k{+}1$ q_{k+1}:v if $v_{k+1} \ne \Lambda$. For example, four of the options in the 4×4 toy problem are

's_3 $+10$ p_{10}:4 q_4:10', 's_6 -31 p_{31}:6 q_6:31 $+30$ p_{30}:7 q_7:30',
's_4 -11 p_{11}:4 q_4:11', 's_6 -30 p_{30}:6 q_6:30 $+31$ p_{31}:7 q_7:31';

the bottom two appear in the solution, but the top two do not. The secondary items are colored so that interdependent options will always link up properly.

Suppose $l < k < r$ and $v_l \ne \Lambda$, $v_{l+1} = \cdots = v_k = \cdots = v_{r-1} = \Lambda$, $v_r \ne \Lambda$. The statement "u might be labeled k" in the specification above means more precisely that there is a simple path of length $k - l$ from v_l to u and a simple path of length $r - k$ from u to v_r. (This condition is necessary for u to be labeled k, but not sufficient. It is, however, sufficient for our purposes.) A simple path of length 1 is equivalent to adjacency. A simple path of length > 1 can be decided using the algorithm in the following exercise; but if that algorithm is taking too long, we can proceed safely by assuming that a simple path does exist. The value of $\min(k - l, r - k)$ is usually small.

[Gyora Benedek invented Hidato® in 2005 and began to publish examples in 2008. Similar brainteasers sprang up later, based on other kinds of paths; but king moves $P_m \boxtimes P_n$ have a special appeal because they can cross each other.]

405. For $l = 0, 1, \ldots, L$, find the set $\mathcal{S}_l$ of all pairs (S, w) such that at least one simple path from v to w runs through the vertices of $S \cup v$, where S is an l-element set. Clearly $\mathcal{S}_0 = \{(\emptyset, v)\}$; and $\mathcal{S}_{l+1} = \{(S \cup w, w) \mid w \mathrel{—} u \text{ and } w \notin S \text{ for some } (S, u) \in \mathcal{S}_l\}$.

If at most 58 vertices w are reachable from v in $\le l$ steps, we can represent each pair (S, w) in a single octabyte, with 6 bits for w and 58 bits for S. These octabytes can be stored in two stacks, alternately at the low and high ends of a sequential list.

406. The moves from 12 to 19 are forced, as are those on several other diagonals. So everything is quickly filled in, except for blanks between 42 and 51. Aha.

407. Using exercise 404, Algorithm C finds the 52 solutions quickly (1500 kilomems). Only one of them has '18' in row 3, column 3; and that clue makes a puzzle with a nicely symmetric solution (see Appendix E). [We could also put '27' in cell $(2, 4)$; or '18' in $(4, 3)$; or '17' in $(4, 4)$. But that would destroy the smile.]

408. (a)

					6
28					
11			24		33

; (b)

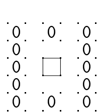
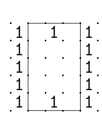
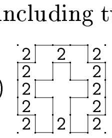
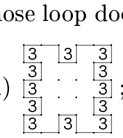
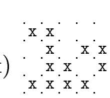
	2		6		10
1		5		9	
22		18		13	
	21		17		14
	25		30		36
26		29		35	

. (See Appendix E for solutions. Are the numbers 5 and 18 best possible for 6×6?)

409. Yes! (This puzzle is fiendishly difficult to solve by hand, although that *has* actually been done. Algorithm C finds the unique solution in 330 Mμ, with a search tree of 161612 nodes. (If you give up, the solution can be found in Appendix E.) A "pidato puzzle" like this is presumably possible only because 10×10 hidato solutions are quite abundant. Indeed, the actual number of 10×10 king paths is 7218332206501318903432956545877450956966; it can be determined with ZDD technology, as explained in Section 7.1.4.)

		31					41		
59			26			53			
	58				97		93		
23		84						50	
	21			100				49	
19			67		81		89		47
18		68				79	3		
									9
				77					

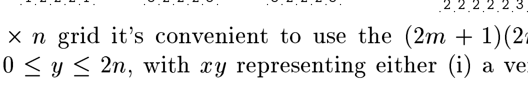
410. Puzzles (a), (b), (d) have unique solutions; remarkably, all 12 of the clues in (b) are essential. But (c) has 40 solutions, including two whose loop doesn't touch a corner.

(a)  ; (b)  ; (c)  ; (d)  ; (x)  .

Pattern (x), incidentally, has unique solutions for $\mathtt{x} = 0, 1, 2$, but none for $\mathtt{x} = 3$.

[*Historical note:* Slitherlink was invented by Nikoli editor Nobuhiko Kanamoto, who combined the puzzle ideas of Ayato Yada and Kazuyuki Yuzawa. See *Puzzle Communication Nikoli* **26** (June 1989).]

411. False; for instance, $\substack{3\,2\\2\,3}$ has two. [But such cases are somewhat mysterious. There are 93 of size 5×5, including three that give two loops despite 8-fold symmetry. A 6×6 example yields *four* loops; can you find them? (See Appendix E.) Are *three* loops possible? If $m+1$ and $n+1$ are relatively prime, N. Beluhov has proved that an $m \times n$ slitherlink diagram with all clues given cannot have more than one solution.]



412. With an $m \times n$ grid it's convenient to use the $(2m + 1)(2n + 1)$ pairs xy for $0 \le x \le 2m$ and $0 \le y \le 2n$, with xy representing either (i) a vertex, if x and y are

octabyte
representation of sets
two stacks
sequential list
smile
pidato puzzle
Hamiltonian king paths
ZDD
Historical note
Nikoli
Kanamoto
Yada
Yuzawa
swastika
8-fold symmetry
Beluhov

planar graph
face
fragment list
dancing slitherlinks
preprocess
cutset

both even; (ii) a cell, if x and y are both odd; or (iii) an edge, if $x+y$ is odd. The edge between two adjacent vertices is their midpoint. The four edges surrounding a cell are obtained by adding $(\pm 1, 0)$ and $(0, \pm 1)$ to the coordinates of the cell.

To obtain the weak solutions for any slitherlink diagram on a planar graph, introduce one primary item for each vertex, one primary item for each face in which the number of edges is specified, and one secondary item for each edge. There are $1+\binom{d}{2}$ options for each vertex v of degree d, namely '$v\ e_1{:}x_1\ \ldots\ e_d{:}x_d$' where $x_j \in \{0,1\}$ and $x_1+\cdots+x_d = 0$ or 2. There are $\binom{d}{k}$ options for each face f of degree d that should have k edges in the path, namely '$f\ e_1{:}x_1\ \ldots\ e_d{:}x_d$' with $x_j \in \{0,1\}$ and $x_1+\cdots+x_d = k$.

For example, the options for vertex 00 in the diagram of exercise 410(i) are '00 01:1 10:1' and '00 01:0 10:0'. The options for cell 11 are '11 01:1 10:1 12:1 21:0', '11 01:1 10:1 12:0 21:1', '11 01:1 10:0 12:1 21:1', '11 01:0 10:1 12:1 21:1'.

This construction yields (2, 2, 104, 2) weak solutions for puzzles 410(a) to 410(d). (In cases (a), (b), (d) we can delete or insert the 4-cycle that surrounds the middle cell.)

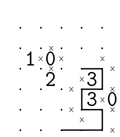
413. Let each record for an item include four new fields `LEFT`, `RIGHT`, `MATE`, `EXMATE`. The `LEFT` and `RIGHT` fields of the secondary item that represents edge u — v will point to the primary items u and v. The `LEFT`, `RIGHT`, and `MATE` fields of the primary item that represents vertex v will represent v's presence or absence in a doubly linked list of endpoints called the "fragment list," which contains the essential information about all edges that currently have been selected.

For example, suppose three edges currently have color 1, say v_1 — v_2, v_3 — v_4, and v_2 — v_5. The fragment list will then contain the endpoints $\{v_1, v_3, v_4, v_5\}$ of the two path fragments v_1 — v_2 — v_5 and v_3 — v_4; we'll also have `MATE`$(v_1) = v_5$, `MATE`$(v_2) = -1$, `MATE`$(v_3) = v_4$, `MATE`$(v_4) = v_3$, `MATE`$(v_5) = v_1$. Other vertices v will have `MATE`$(v) = 0$, indicating their absence from any existing edges. When the 'purify' routine (55) is called to give color 1 to a new edge i, it will know that this edge should *not* be chosen if `LEFT`$(i) = v_1$ and `RIGHT`$(i) = v_5$, say, or in general if `LEFT`(i) and `RIGHT`(i) are mates, because that would close a loop disjoint from the other fragment. Furthermore, 'purify' won't give color 1 to edge i if `MATE(LEFT`(i)`))` $= -1$, or if `MATE(RIGHT`(i)`))` $= -1$, or if the fragment list already contains a completed loop.

In this example, vertex v_2 was deleted from the fragment list when edge v_2 — v_5 was chosen, because v_2 was no longer an endpoint. At that time, 'purify' not only set `MATE`$(v_2) \leftarrow -1$, it also set `EXMATE`$(v_2) \leftarrow v_1$, so that 'unpurify' would be able to undo the operation later. Similarly, a subsequent edge v_1 — v_4 will remove both v_1 and v_4 from the fragment list, by using (1) with `LEFT` and `RIGHT` links, and by setting `EXMATE`$(v_1) \leftarrow$ `MATE`(v_1), `MATE(MATE`(v_1)`))` $\leftarrow$ `MATE`(v_4), `MATE`$(v_1) \leftarrow -1$, etc. Of course the dancing-links trick (2) will eventually be used later, to undo deletion (1).

Caution: Algorithm P must be modified so that it never discards redundant items, when it is used to preprocess a problem for this extension of Algorithm C.

414. After the forced moves have been made as shown, only two edges are undecided between the vertices of rows 1 and 2. A strongest possible algorithm will know that those two edges must either both be present or both absent. (In fact, a truly strongest possible algorithm will force both to be present as soon as *any* edge in or between rows 0 and 1 has been chosen.)

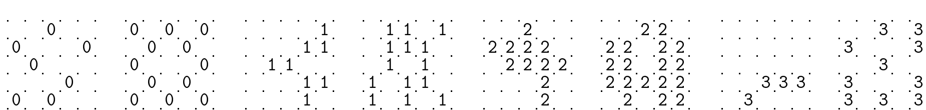


In general, consider the graph G consisting of the original vertices V and all the currently undecided edges. If X is any proper subset of V, connected or not, any loop will contain an even number of edges between X and $V \setminus X$. Thus any cutset of size two will force a relation between two undecided edges. An algorithm that dynamically maintains minimum cutsets of G (see Section 7.5.3) will therefore be helpful.

415. Instead of solving millions of puzzles, we can use the ZDD technology of Section 7.1.4 to list all the loops in $P_6 \square P_6$, of which there are 1222363. Say that the "signature" of a loop is the full sequence of 25 clues — the number of edges around each cell. It turns out that 93 pairs of loops have the same signature (see exercise 411); those 186 loops cannot be the solution to any 5×5 slitherlink puzzle. Let S be the set of 1222270 distinct signatures, and let S' be the subset of 1222177 that give a valid 25-clue puzzle.

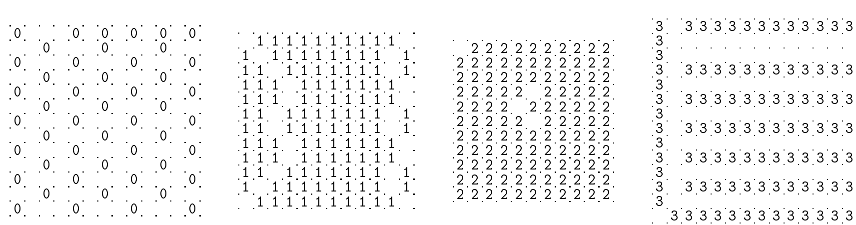
Suppose $s' \in S'$ has $t > 0$ entries equal to digit d; and for $s \in S$ let $p(s, s')$ be the binary "projection vector" $x_1 \ldots x_t$, where $x_k = 1$ if and only if s has d in the kth cell where s' has d. $s = \begin{smallmatrix} 2&1&2&3&1\\ 2&1&3&1&1\\ 2&1&2&3&3\\ 1&1&0&1&1\\ 1&2&1&1&2 \end{smallmatrix}$, $s' = \begin{smallmatrix} 2&2&2&3&1\\ 2&2&1&1&1\\ 2&1&1&2&3\\ 2&3&2&1&1\\ 2&2&1&1&2 \end{smallmatrix}$ For example, if $d = 1$, the signatures s and s' shown above have $t = 10$ and $p(s, s') = 1011101111$. Form the set $P(s') = \{p(s, s') \mid s \neq s'\}$. Then s', with all clues restricted to digit d, is a valid puzzle if and only if $11 \ldots 1 \notin P(s')$. Moreover, the valid puzzles contained in that one are precisely those whose projections aren't contained in any element of $P(s')$. (If we regard $P(s')$ as a family of sets, such projections are the elements of $\wp \nearrow P(s')$, in the notation of exercise 7.1.4–236.) We can find those vectors, and the minimal ones, with a reachability algorithm such as Algorithm 7.1.3R.

In this way we discover exactly (9310695, 833269, 242772, 35940, 25) valid puzzles for $d = (0, 1, 2, 3, 4)$, of which exactly (27335, 227152, 11740, 17427, 25) have no redundant clues. The minimum number of clues, in such irredundant homogeneous puzzles, is respectively (7, 8, 11, 4, 1); and the maximum number is respectively (12, 14, 18, 10, 1). Many of the extreme cases make pleasant little puzzles:



(See Appendix E. This minimum-1s puzzle is one of two based on signature s' above.)

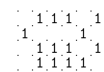
416. Of course $d = 4$ is trivial. So is $d = 0$; but that case has an amusing sparse construction. The following puzzles generalize to all n with $(n + d) \bmod 4 = 1$:



[See the solutions in Appendix E. N. Beluhov, who found these patterns for $d = 2$ and 3, has raised interesting problems of optimum density: Let $\underline{\beta(d)} = \liminf_{n\to\infty} \|S\|/n^2$ and $\overline{\beta(d)} = \limsup_{n\to\infty} \|S\|/n^2$, where S ranges over all valid $n \times n$ slitherlink puzzles that are d-homogeneous, and where $\|S\|$ denotes the number of clues. Clearly $\|S\| \leq n^2/2$ when $d = 3$, because no 2×2 subsquare can contain more than two 3s. Furthermore $\|S\| \geq n^2/4 - O(n)$ when $d = 0$. For we must eliminate at least $n^2 + 2n$ of the $2n(n+1)$ edges if all but one cycle is to be cut; each 0 eliminates at most four. If $n > 5$ we obtain a valid puzzle with only fourteen 1s, by placing a suitable 4×6 pattern in the upper left corner. Similarly, there's a valid puzzle with only four 3s, if $n > 3$.

minimum cutsets
max-flow, min-cut
ZDD
simple cycles, enumerating
signature
projection vector
family of sets
$\wp$, the power set
nonsubsets $f \nearrow g$
minimal elements of a family of sets
reachability algorithm
Beluhov

Therefore these constructions prove that $\overline{\beta(0)} = \overline{\beta(1)} = \overline{\beta(2)} = 1$; $\overline{\beta(3)} = 1/2$; $\underline{\beta(1)} = \underline{\beta(3)} = \underline{\beta(4)} = \overline{\beta(4)} = 0$; $\underline{\beta(0)} = 1/4$. But $\underline{\beta(2)}$ remains unknown.]

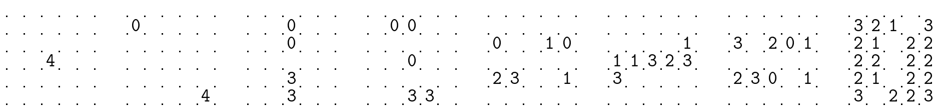


SAT solvers
projection
author
preprocessing
Knuth
pi, as source
Beluhov

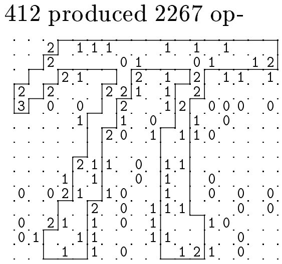
417. The pattern for $d = 3$ in answer 416 works also for $d = 0$, if we remove one clue from the top row. Fascinating diagrams arise when such patterns are attempted for $d = 1$; Beluhov's largest example so far is the 30×30 puzzle obtained when removing the 1 in column 26 of row 0. (Such puzzles are extremely difficult for the algorithm of answer 413 to handle; but SAT solvers have no trouble with them.)

418. (a) $6 \cdot 26^{12} \approx 5.7 \times 10^{17}$, from the central cell and 12 complementary pairs.

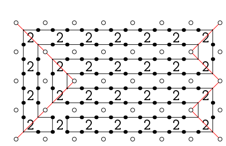
(b, c, d, e) As in answer 415, we define the projection $p(s, s') = x_1 \ldots x_{13}$, where $x_k = 1$ if and only if s and s' agree in the kth pair (or in the center, when $k = 13$). We obtain altogether 2,692,250,947 puzzles, of which 199,470,026 are minimal. The minimal ones include (1, 24, 0, 7, 42, 1648, 13428, 257105, ..., 184, 8) that have respectively (1, 2, 3, 4, 5, 6, 7, 8, ..., 19, 20) clues; here are some choice specimens:



419. To design this puzzle, the author began with the signature of the desired loop (see answer 415), then removed pairs of centrally opposite clues, more or less at random, until no redundant pairs remained. The construction of exercise 412 produced 2267 options on 404+573 items from the final clue set; and Algorithm P needed just 17 Mμ to remove 1246 of those options. Then the algorithm of exercise 413 discovered the solution, and proved it unique, with 5.5 Gμ of computation and a search tree of 15 meganodes. (It's another big win for preprocessing: Otherwise that algorithm would have taken 37 Tμ, with a 78-giganode search tree!) *Reference:* D. E. Knuth, *Computer Modern Typefaces* (Addison–Wesley, 1986), 158–159.



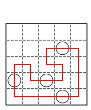
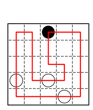
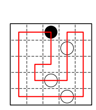
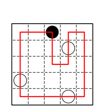
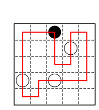
420. Suppose S is a solution (or even a weak solution). Mark with a black dot all vertices whose adjacent 2 includes exactly one edge of S touching that vertex, as in the 7×13 example illustrated. Let G be the graph whose vertices form an $(m+3)/2 \times (n+3)/2$ grid like the white dots in this example, and whose edges connect adjacent white dots that are *not* separated by black dots. Then the corner vertices of G have degree 1; but all other white dots have even degree (possibly degree 4).



Consider the connected component C of G that contains the northwest corner. It also contains another corner, because every graph has an even number of vertices of odd degree. When $n \bmod 4 = 1$ it can't be the northeast corner; when $m \bmod 4 = 1$ it can't be the southwest corner. So C must contain the southeast corner, if $m \bmod 4 = n \bmod 4 = 1$. Similarly, there's a connected component C' that contains the northeast and southwest corners. But that's impossible! All white dots of C have even parity; all white dots of C' have odd parity; and they cannot cross.

[This exercise and proof are due to N. Beluhov. Conversely, the 7×13 example given above generalizes to an $m \times n$ solution whenever $m \bmod 4 = 3$ and n is odd.]

421.

(1 of 44)  (1 of 7)  (1 of 1)  (1 of 7)  (1 of 5) 

[*Historical note:* Masyu was invented by Tatsuo Yano, who developed a white-circles-only version, together with Mitsuhiro Ase, who contributed the black circles. See *Puzzle Communication Nikoli* **84** (April 1999); **89** (March 2000).]

Historical note
Yano
Ase
equivalence algorithm
union-find algorithm
signature

422. Now we use the $(2m-1)(2n-1)$ pairs xy for $0 \le x \le 2m-2$ and $0 \le y \le 2n-2$. Cell (i,j) corresponds to $x = 2i$ and $y = 2j$ (a "vertex"); clue (i,j) corresponds to $x = 2i+1$ and $y = 2j+1$. Edges are as before, and we use the same options to ensure that either 0 or 2 edges touch every vertex in a solution. The only essential change from answer 412 is the treatment of clues, since masyu clues are different from slitherlink clues.

A black masyu clue in (i,j) has four options, corresponding to north-west, north-east, south-west, and south-east legs; for example, the north-west option is '$C(i,j)$ $N(i,j)$:1 $NN(i,j)$:1 $W(i,j)$:1 $WW(i,j)$:1', where $C(i,j) = (2i{+}1)(2j{+}1)$, $N(i,j) = C(i,j) - 10$, $NN(i,j) = C(i,j) - 30$, $W(i,j) = C(i,j) - 01$, $WW(i,j) = C(i,j) - 03$. Edges off the grid have "color" 0, so this option is omitted when $i \le 1$ or $j \le 1$.

A white masyu clue in (i,j) has six options, three for north-south orientation and three for east-west. The three for east-west are '$C(i,j)$ $E(i,j)$:1 $EE(i,j)$:0 $W(i,j)$:1 $WW(i,j)$:0', '$C(i,j)$ $E(i,j)$:1 $EE(i,j)$:0 $W(i,j)$:1 $WW(i,j)$:1', and '$C(i,j)$ $E(i,j)$:1 $EE(i,j)$:1 $W(i,j)$:1 $WW(i,j)$:0'. Again we omit an option that would set an off-board edge to 1. An off-board edge item that sets color 0 is silently dropped.

For example, the options for the black clue in exercise 421's puzzle are '15 14:1 34:1 03:1 01:1', '15 14:1 34:1 05:1 07:1'. The options for the white clue in the bottom row are '97 87:1 85:1 83:0', '97 87:1 85:1 83:1'. That puzzle has 15 clue options altogether, and 119 vertex options '00 01:1 10:1', '00 01:0 10:0', '02 01:1 03:1 12:0', …, '88 78:0 87:0'.

423. Obtain a representative of each class of equivalent variables, for example by adapting Algorithm 2.3.3E. This calculation may show that certain variables are constant. A contradiction might also arise — for example, if there's a white clue in a corner; in such cases the masyu puzzle has no solution.

The vertex options of answer 422 can now be eliminated, at all vertices for which a clue was given. The clue options can also be consolidated, so that equivalent variables don't appear together, and so that constants are suppressed. Every option that tries to set a variable both true and false is, of course, eliminated.

For example, variables 14, 50, 70, 85, and 87 in the puzzle of exercise 421 are forced to be true; variables 61 and 76 are forced to be false. We can eliminate variables 05, 16, 27, 36, 54, 65, and 74 because $05 = {\sim}03$, $16 = 36 = {\sim}25$, $27 = 25$, $54 = 74 = {\sim}63$, $65 = 63$. The options for the black clue become '15 01:1 03:1 34:1', '15 03:0 07:1 34:1'. The options for the white clue in the bottom row become '97 83:0', '97 83:1'.

Caveat: These simplifications are very nice, but they mess up the single-loop-detection mechanism of answer 413 — because that answer uses several fields of item nodes as key elements of its data structure! To keep that algorithm happy, we must append a special option that covers all of the supposedly eliminated vertex items and constant-edge items; this option is '04 26 60 64 86 87:1 85:1 76:0 50:1 70:1 61:0 14:1' in the example. We also need pairs of options such as '#25 16:1 36:1 27:0 25:0' and '#25 16:0 36:0 27:1 25:1', to keep all variables of an equivalence class in sync.

A tenfold speedup is achieved even on small puzzles like the 8×10 in exercise 426.

424. As in answer 415, we can begin with the 1222363 loops that are potential solutions. But this time the "signature" of a loop is the maximum set of clues that it supports. Such a signature turns out to have at most 24 clues; indeed, only puzzle (i) in Fig. A–5, along with its rotations or reflections, attains this maximum. (At the other extreme, 64 loops have an entirely *empty* signature, despite having lengths up to 28.)

Let S be the set of 905472 distinct signatures; and let S' be the subset of 93659 signatures that aren't contained in any other. These are the signatures of loops that can solve a valid 6×6 puzzle. If $s' \in S'$ has t clues, we define the projection vector $p(s, s') = x_1 \ldots x_t$ for $s \in S$ by setting $x_j = 1$ when s agrees with s' in the jth cell where s' has a clue. For example, when s' is puzzle (i) and s is its transpose, the projection $p(s, s')$ is 111010111101011110111011.

Form the set $P(s') = \{p(s, s') \mid s \neq s'\}$. We know that $11 \ldots 1 \notin P(s')$, because s' isn't dominated by any other signature. Moreover, the valid puzzles having the loop of s' as their solution are precisely those whose clues are not contained in any element of $P(s')$. We can find such puzzles, and the minimal ones, with a reachability computation like Algorithm 7.1.3R, whose running time is $O(2^t)$. For example, the loop of (i) turns out to be the solution to 8924555 puzzles(!). Four of them, such as (ii) and (iii), are minimal with only four clues; three of them, such as (iv), are minimal with eleven.

Most elements of S' have far fewer than 24 clues. Hence it isn't difficult to determine that there are exactly 1,166,086,477 valid 6×6 masyu puzzles altogether, of which 4,366,185 are minimal. (There are (80, 1212, 26188, 207570, ..., 106) minimal puzzles with (3, 4, 5, 6, ..., 12) clues. One of the 3s is puzzle (v); it also has the shortest loop. One of the 12s is puzzle (vi); it also has the *longest* loop — a Hamiltonian cycle. (A Hamiltonian cycle can actually be forced by only four clues; see puzzle (xvii).)

The valid puzzles include 5571407 that are pure white, 4820 that are pure black. The white clues can take on 22,032,015 different patterns; the black clues can assume only 39140. A surprisingly large number of 6×6 puzzles, 37472, can be "inverted," remaining valid when white and black are swapped. If we restrict consideration to minimal puzzles, these figures become: 574815 pure white, 1914 pure black, 2522171 white patterns, 22494 black patterns, 712 invertible. The latter include many amusing and amazing pairs, such as (vii)–(viii), (ix)–(x), (xi)–(xii), as well as self-dual examples such as (xiii), (xiv), (xv), (xvi); there are 49 essentially distinct invertible puzzles of size 6×6. [Considerably larger invertible puzzles have been published in the anonymous blog `uramasyu.blog80.fc2.com/`, every few days since 2006.]

The author thinks puzzle (vi) may well be the hardest 6×6, although its search tree via exercise 423 has only 212 nodes. (That tree has 1001 nodes with exercise 422.)

425. A "balanced" $n \times n$ masyu solution of order k clearly requires $2 \le k \le \lfloor n^2/4 \rfloor$. All such k turn out to be achievable, for $n \le 6$, except that the upper limit $\lfloor n^2/4 \rfloor$ is not; see (xviii)–(xxiv) in Fig. A–5. (Solutions for $k = 2$ exist for all $n \ge 3$; solutions for $k = 3$ exist for all $n \ge 4$. The situation for larger k and n has yet to be investigated; for example, perhaps $k = 4$ is impossible for all large n.)

426. The clue in the corner must obviously be '●'. That leaves us with 2^{28} other possibilities to consider, many of which can be rejected immediately because certain local patterns are impossible. (For example, there cannot be three consecutive '●', nor five '○'s that form an X or T.) Consider the Boolean function of $x_0 x_1 \ldots x_{27}$ that's true if and only if the diagram has at least one solution, with '●' when $x_j = 1$ and '○' when $x_j = 0$. One can easily verify that there's no solution when $x_0 x_1$ or $x_1 x_3$ or $x_0 \bar{x}_1 x_4$ or ... or $\bar{x}_3 \bar{x}_4 \bar{x}_5$ or $x_6 x_7$ or $\bar{x}_7 \bar{x}_8 \bar{x}_{10} \bar{x}_{11}$, etc.; also when we replace x_j by x_{j+12}. We can also rule out extreme cases such as $\bar{x}_1 x_{26}$.

	24	0	1	2	12	13	14	27	
	25		3			15			
26			4			16			
			5			17			
		9	6		21	18			
		10	7		22	19			
		11	8		23	20			

projection vector
minimal elements of a family of sets
reachability
Hamiltonian cycle
inverted
anonymous
author
balanced masyu solutions

author
BDD
focus

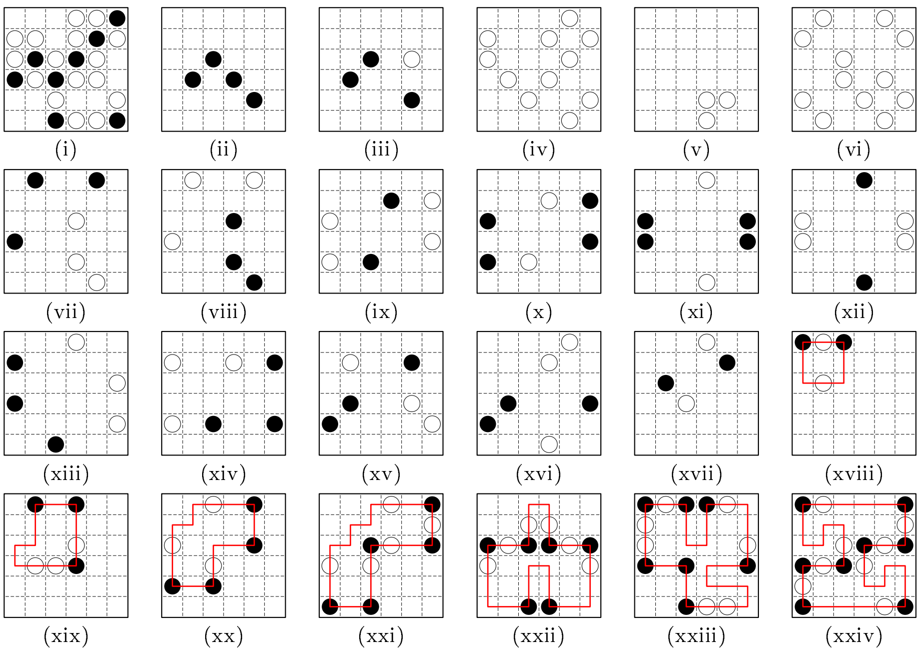
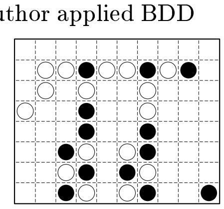
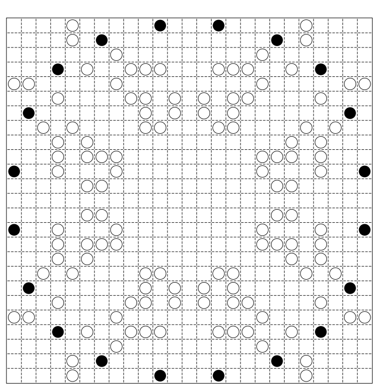


Fig. A–5. A gallery of interesting 6×6 masyu puzzles.

After compiling several dozen such "bad" configurations, the author applied BDD technology: Less than a megamem sufficed to generate a BDD of size 715, which showed that exactly 10239 vectors $x_0x_1 \ldots x_{27}$ were not yet ruled out. The masyu solver of exercise 423 tossed off those cases with search trees of 3 nodes per problem, on average; and it turned out that exactly (10232, 1, 1, 1, 4) vectors had (0, 1, 2, 3, 4) solutions. The unique winning puzzle is shown here (and solved in Appendix E).

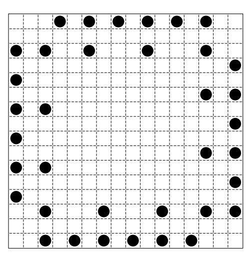
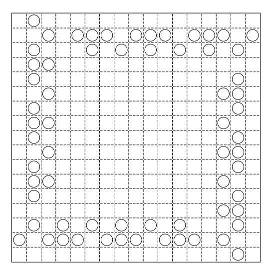


427. Here's an example with $8 \cdot 15$ white clues (solved in Appendix E):

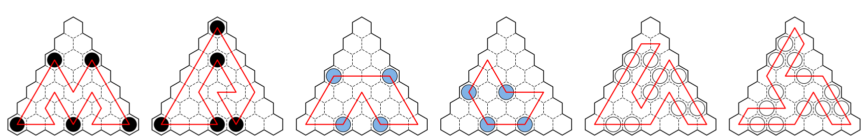


It turns out to be problematic for the method of exercise 423, which severely loses focus and takes forever to prove that there's only one solution. One can, however, exploit symmetry by modifying Algorithm C as follows: Whenever a color setting is made on the rightmost branch of the search tree, all settings that are equivalent to it by symmetry can be forced. Then uniqueness is proved in about 36 Mμ, provided that the primary items are suitably ordered. [This exercise was inspired by Nikoli's *Giant Logic Puzzles for Geniuses* (Puzzlewright Press, 2016), #53.]

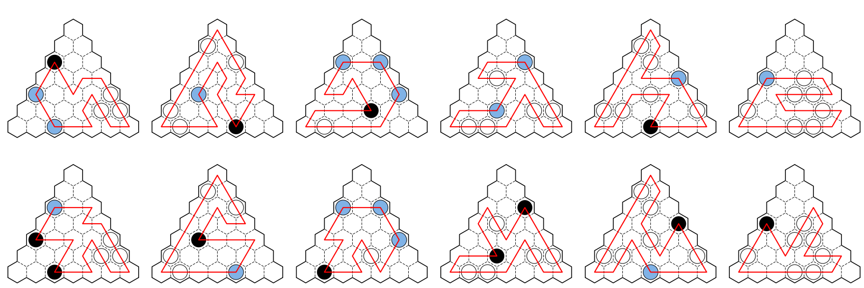
428. (Solution by N. Beluhov.) $3n-12$ black clues suffice when $n \bmod 4 = 0$; $5n-21$ white clues suffice when $n \bmod 4 = 1$. (Are these constants 3 and 5 the best possible?)

(a)  ; (b)  .

429. (a) Incidentally, each of these puzzles is minimal (all clues important):



(b) In fact, *two* permutations of the colors are possible in each case:



430. (a) The lower right corner must contain 5. See Appendix E for the other cells.

(b) Set $c_{nk} \leftarrow 0$ for all n and k. Now do this for $3 \le x < 512$: Set $k \leftarrow n \leftarrow 0$; for $0 \le t < 9$ set $k \leftarrow k+1$, $n \leftarrow n+t+1$ if $x \mathbin{\&} (1 \ll t) \ne 0$; finally if $k > 1$, set $C_{nkc_{nk}} \leftarrow x$ and $c_{nk} \leftarrow c_{nk}+1$. The n-in-k combinations are now C_{nkj} for $0 \le j < c_{nk}$.

The maximum c_{nk}, 12, is obtained for $(n,k) = (20,4)$ or $(25,5)$. Notice that $c_{nk} = c_{(45-n)(9-k)}$ when $1 < k < 8$. Cases with $c_{nk} = 1$ are called "restricted" or "magic blocks"; they're extremely helpful when present (but our example doesn't have any).

(c) The middle must be 7 9 8 (an odd digit < 9, 9, then an even digit).

(d) The tables from (b) convert kakuro to generalized kakuro. Introduce a primary item ij for each cell to be filled. Let there be H horizontal blocks, and assume that horizontal block h has c_h combinations X_{hp} of length k_h, for $1 \le h \le H$ and $1 \le p \le c_h$. Introduce $c_h k_h$ primary items H_{hpx}, for $x \in X_{hp}$, to represent the elements of block h's pth combination. (For example, the primary items for the first horizontal block of our example are H_{111}, H_{114}, H_{122}, H_{123} because the two combinations are $\{1,4\}$ and

symmetry
modifying Algorithm C
Nikoli
Beluhov
minimal
invertible masyu
bitwise tricks
magic blocks

color
preprocessing
focus
copyright
restricted growth strings
permutations
dual

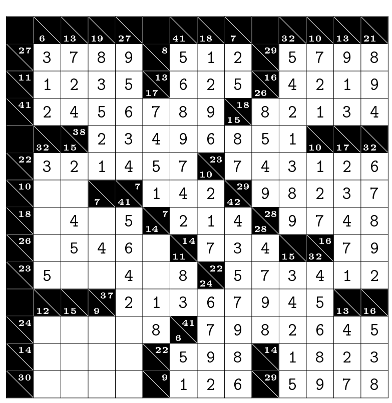
$\{2,3\}$.) Similarly, introduce primary items V_{vqy} for the elements of the qth combination Y_{vq} of vertical block v, for $1 \le v \le V$ and $1 \le q \le d_v$.

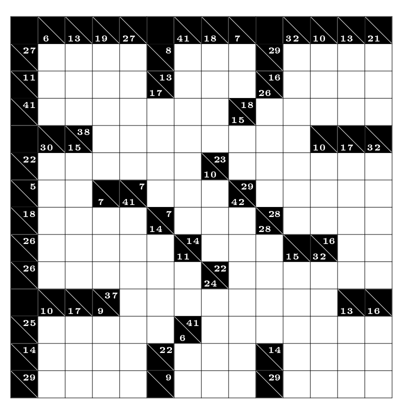
Also introduce secondary items H_h and V_v for $1 \le h \le H$ and $1 \le v \le V$, one for each block. The "color" of such an item represents the choice of combination to be used.

The options for cell ij are 'ij H_{hpx} $\mathrm{H}_h{:}p$ V_{vqx} $\mathrm{V}_v{:}q$', where h and v indicate the horizontal and vertical blocks through ij, for $1 \le p \le c_h$ and $1 \le q \le d_v$ and $x \in X_{hp} \cap Y_{vq}$. (Thus, the options for the upper left blank cell in our example are '11 H_{111} $\mathrm{H}_1{:}1$ V_{111} $\mathrm{V}_1{:}1$', '11 H_{114} $\mathrm{H}_1{:}1$ V_{124} $\mathrm{V}_1{:}2$', '11 H_{122} $\mathrm{H}_1{:}2$ V_{122} $\mathrm{V}_1{:}2$'. Set intersections are easily computed from the bitmaps X_{hp} and V_{vq}.)

Additional options are also necessary to "absorb" the combinations not used. These are '$\bigcup\{\mathrm{H}_{hpx} \mid x \in X_{hp}\}$ $\mathrm{H}_h{:}p'$' for $1 \le p, p' \le c_h$ and $p \ne p'$; '$\bigcup\{\mathrm{V}_{vqy} \mid y \in Y_{vq}\}$ $\mathrm{V}_v{:}q'$' for $1 \le q, q' \le d_v$ and $q \ne q'$. (Thus the options for $h = 1$ in our example are 'H_{111} H_{114} $\mathrm{H}_1{:}2$', 'H_{122} H_{123} $\mathrm{H}_1{:}1$'.) This instructive construction deserves careful study.

431. There are 18 solutions, because of two ways to complete the middle left portion and (independently) nine ways to complete the lower left corner. (The digits that *are* uniquely determined by his conditions are shown below.) We can freeze most of those digits, and extract two much smaller problems, then insert a few wildcards as in exercise 433 until obtaining uniqueness. One suitable patch, shown below, changes seven clues and has the solution found in Appendix E. (In this problem, preprocessing greatly improves the focus, reducing the search tree size from 115 million to just 343!)





[Funk had copyrighted a Cross Sums Puzzle already in September 1935; see *Canadian Patent Office Record and Register of Copyrights and Trade Marks* **63** (1935), 2253.]

432. (a) We save a lot of time by considering only "restricted growth strings" as solutions (see Section 7.2.1.5). That is, we can assume that the top row is '12'; then the second row is either '213' or '234' or '312' or '314' or '34x' for $1 \le x \le 5$; etc. Altogether there are (5, 28, 33, 11, 1) such strings with maximum element (3, 4, 5, 6, 7). Thus we know that the blanks can be filled in $5\cdot 9^{\underline{3}} + 28\cdot 9^{\underline{4}} + 33\cdot 9^{\underline{5}} + 11\cdot 9^{\underline{6}} + 9^{\underline{7}} = 1432872$ ways. And we can quickly compute the 1432872 sequences of block sums from those restricted growth strings, using a table of 9! permutations built by Algorithm 7.2.1.2L. Exactly 78690 of those sequences, about 5.5%, occur uniquely and define a kakuro puzzle.

Every kakuro puzzle has a *dual*, obtained by replacing all clue-sums s for blocks of length k by $10k - s$; the dual is solved by changing each digit d to $10 - d$. Thus, if a puzzle of type (a) is defined by horizontal and vertical sums $s_1s_2s_3/t_1t_2t_3$, its dual is defined by $(20-s_1)(30-s_2)(20-s_3)/(20-t_1)(30-t_2)(20-t_3)$. Diagonal symmetry also makes $s_1s_2s_3/t_1t_2t_3$ equivalent to $s_3s_2s_1/t_3t_2t_1$ and $t_1t_2t_3/s_1s_2s_3$; so we get up to eight equivalent puzzles from each sequence. There are 9932 essentially distinct puzzles, only one of which has four symmetries, namely 6 15 14/14 15 6; 190 have one symmetry, and the remaining 9741 are asymmetric. (The asymmetric ones are, of course, more difficult to solve, because a symmetric puzzle will have a symmetric solution.) The example 5 19 6/6 10 14 in exercise 430 is asymmetrical; but it's relatively easy because it has a forced move in the lower right corner. The easiest puzzles, with *four* forced moves, are 4 15 12/12 15 4 and 4 15 16/12 15 8, both symmetric. Altogether 4011 of the asymmetric puzzles have no forced moves. And of those, 570 have no "magic blocks." And of those, puzzle 6 19 6/8 11 10 is the hardest, in the sense that it maximizes the number of nodes (79) in Algorithm C's search tree, using the construction of answer 430(d).

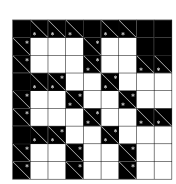
(b) Similarly, this shape has $2\cdot 9^{\underline{3}}+42\cdot 9^{\underline{4}}+186\cdot 9^{\underline{5}}+234\cdot 9^{\underline{6}}+105\cdot 9^{\underline{7}}+18\cdot 9^{\underline{8}}+9^{\underline{9}}=43038576$ sequences of block sums, of which $6840\approx 0.016\%$ are unique. Those 6840 yield 49 equivalence classes under the symmetries $s_1s_2s_3/t_1t_2t_3\mapsto s_2s_1s_3/t_1t_2t_3$, $s_3s_2s_1/t_1t_2t_3$, $s_1s_2s_3/t_2t_1t_3$, $s_1s_2s_3/t_3t_2t_1$, $t_1t_2t_3/s_1s_2s_3$, $(30-s_1)(30-s_2)(30-s_3)/(30-t_1)(30-t_2)(30-t_3)$. All but 3 of those 49 puzzles are asymmetric; 7 11 20/7 11 20 and 7 19 20/7 19 20 are self-transpose, and 7 15 23/10 15 20 is self-dual. They aren't great, because they all have at least one forced move from 7 opposite 20 or from its dual.

[It's extremely difficult to find a kakuro puzzle whose spaces make a 4×4 grid. But Johan de Ruiter discovered in 2010 that there are five essentially different ways. For example, 11 15 23 29/12 15 23 28 has a 488-node search tree, so it's a nice little challenge.]

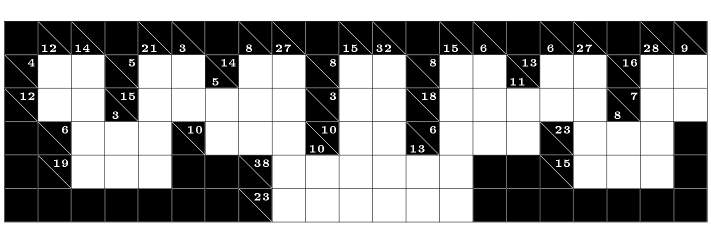
433. A slight extension to the construction of answer 430(d) allows "wildcard" blocks, with unspecified length and with the universal combination $\{1,\ldots,9\}$ as their X or Y. The items H_{h1x} or V_{v1y} for such wildcards are secondary, not primary. Algorithm C now pumps out 89638 solutions (in 150 Mμ); and 12071 of the corresponding sum sequences $s_1\ldots s_7/t_1\ldots t_7$ occur once only and yield valid puzzles. (The easiest ones, 16 4 18 $(d{+}14)$ 16 16 16/9 34 24 6 d 12 15 for $7\le d\le 9$, have a search tree of only 47 nodes. A median puzzle such as 16 4 20 18 16 16 15/9 22 24 6 17 12 15 needs 247 nodes. And the hardest, 16 4 23 19 16 16 13/9 25 24 6 17 11 15, needs 1994.)

[The author tried 10000 experiments in which all 21 cells of this diagram were simply filled at random, and their block sums recorded. Those 10000 problems had ≈ 75 solutions, on average, with standard deviation ≈ 1200. Only five of them led to valid puzzles; the most difficult one, 15 3 21 16 27 8 10/9 22 28 11 21 5 4, needed 1168 nodes.]

434. In 700 Mμ, a BDD with 64 variables and 124487 nodes characterizes 93,158,227,648 solutions. N. Beluhov proved in 2018 that there are at most 38 blocks, achieved for example as shown, by listing all cases with 38 or more. He also observed that the maximum for $n\times n$ kakuro is $n^2/3-O(n)$, using a similar construction with (i,j) black $\iff (i+j) \bmod 3=0$ except near the boundary.



435. The search tree for this one has 566 nodes. (See Appendix E.)



magic blocks
de Ruiter
wildcard
author
BDD
Beluhov

436. (a) Any solution with a black seed works also with that cell white.

(b) A solution with a non-articulation point would work also with that cell black.

437. Introduce a primary item `*` to make the seeds white; also primary items Ric and Cjc for each character c that occurs more than once in row i or column j. Introduce secondary items ij for $0 \le i < m$ and $0 \le j < n$, representing cell (i,j). For example, the first option for puzzle 436(α) is '`*` 01:0 02:0 10:0 13:0 14:0 21:0 31:0 32:0'.

Suppose row i contains character c in columns $j_1, \ldots, j_t$, where $t > 1$. Then Ric normally has $t+1$ options 'Ric ij_1:e_1 … ij_t:e_t u_1:0 … u_s:0' for $e_1 + \cdots + e_t \ge t-1$, where $\{u_1, \ldots, u_s\}$ are the non-seed neighbors of the cells being colored 1. However, this option is suppressed if it would assign two colors to the same item. For example, if $i = 1$, $t = 3$, and $j_1 j_2 j_3 = 123$, there is only one option 'R1c 11:1 12:0 13:1 01:0 03:0 10:0 14:0 21:0 23:0' (but with entries deleted that color a seed with 0), because the other three options are contradictory.

Of course the options for Cjc are similar. For example, the options for `C3L` in puzzle 436(α) are '`C3L` 23:0 33:1 34:0' and '`C3L` 33:0 23:1 22:0 24:0'.

[Notice, incidentally, that this XCC problem is a special case of 2SAT. Therefore it can be solved in linear time. Furthermore, by Theorem 7.1.1S, the median of any three solutions is also a solution — a curious fact!]

438. The basic idea is to abandon partial solutions that cut off any white cells from the first seed. Connectedness can be assured by maintaining a triply linked spanning tree, rooted at that seed, with the help of new fields in each item record. Changes to the spanning tree need not be undone when unblackening a cell while backtracking; *any* spanning tree on the currently nonblack cells is satisfactory.

[This method can be patched to handle the rare instances that have *no* seeds. To ensure uniqueness, as in exercise 436(b), each solution should also be tested for articulation points. Hopcroft and Tarjan's algorithm for bicomponents does that efficiently. See Section 7.4.1; also *The Stanford GraphBase*, pages 90–99.]

439. (a) Property (ii) states that U is a vertex cover (or equivalently that $V \setminus U$ is independent). Thus (i) and (ii) together state that U is a connected vertex cover. Adding property (iii) gives us a *minimal* connected vertex cover. [Minimal connected vertex covers were introduced by M. R. Garey and D. S. Johnson in *SIAM J. Applied Math.* **32** (1977), 826–834, who proved that it is NP-complete to decide if a planar graph with maximum degree 4 has a connected vertex cover of a given size.]

(b) This is the thrust of exercise 436(b). [N. Beluhov has proved constructively that every $m \times n$ hitori cover for $m, n > 1$ solves at least one valid puzzle, using an alphabet of at most $\max(m,n)$ letters.]

440. False (if neither `A` is alone in its column). Consider $\begin{array}{|c|c|c|}\hline \mathtt{A}&\mathtt{B}&\mathtt{A}\\\hline \mathtt{B}&\mathtt{C}&\mathtt{A}\\\hline \mathtt{A}&\mathtt{C}&\mathtt{C}\\\hline\end{array}$ or $\begin{array}{|c|c|c|c|c|c|c|}\hline \mathtt{0}&\mathtt{A}&\mathtt{B}&\mathtt{C}&\mathtt{D}&\mathtt{A}&\mathtt{1}\\\hline \mathtt{B}&\mathtt{2}&\mathtt{3}&\mathtt{4}&\mathtt{5}&\mathtt{6}&\mathtt{D}\\\hline \mathtt{B}&\mathtt{A}&\mathtt{B}&\mathtt{C}&\mathtt{D}&\mathtt{A}&\mathtt{D}\\\hline\end{array}$.

441. When $n = 1$ any single letter `a` is trivially a valid puzzle. When $n > 1$ the possibilities are (i) `a`α`a` for every string α of $n-2$ distinct letters containing an `a` (thus $(n-2)d^{\underline{n-2}}$ puzzles); (ii) `a`α`b` for `a` $\ne$ `b` and every string α of $n-2$ distinct letters containing `a` and `b` (thus $(n-2)^2 d^{\underline{n-2}}$ puzzles); altogether $(n-2)^2 d^{\underline{n-2}}$ valid puzzles.

442. A "frontier-based" algorithm analogous to those of answers 7.1.4–55 and 7.1.4–225 will produce an unreduced ZDD for the family f of all *complements* $V \setminus U$ of connected vertex covers, from which a variant of Algorithm 7.1.4R will give a ZDD. Then the NONSUB subroutine of answer 7.1.4–237 will produce a ZDD for $f^{\uparrow}$, the complements of hitori covers (the black cells of potential solutions). In the most complicated case, $m =$

2SAT
median
triply linked
spanning tree
articulation points
Hopcroft
Tarjan
bicomponents
vertex cover
Garey
Johnson
NP-complete
Beluhov
frontier
ZDD
NONSUB subroutine
$f^{\uparrow}$

$n = 9$, an unreduced ZDD of size 203402 is reduced quickly to 55038 nodes; then 550 Gμ of computation produces a ZDD of size 1145647 for the family of maximal black cells.

Those ZDDs make it easy to count and generate hitori covers; we obtain the totals

$$\begin{pmatrix} 1 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 4 & 6 & 12 & 20 & 36 & 64 & 112 & 200 \\ 1 & 6 & 11 & 30 & 75 & 173 & 434 & 1054 & 2558 \\ 1 & 12 & 30 & 110 & 382 & 1270 & 4298 & 14560 & 49204 \\ 1 & 20 & 75 & 382 & 1804 & 7888 & 36627 & 166217 & 755680 \\ 1 & 36 & 173 & 1270 & 7888 & 46416 & 287685 & 1751154 & 10656814 \\ 1 & 64 & 434 & 4298 & 36627 & 287685 & 2393422 & 19366411 & 157557218 \\ 1 & 112 & 1054 & 14560 & 166217 & 1751154 & 19366411 & 208975042 & 2255742067 \\ 1 & 200 & 2558 & 49204 & 755680 & 10656814 & 157557218 & 2255742067 & 32411910059 \end{pmatrix}.$$

Further statistics about these fascinating patterns are also of interest:

$$\begin{pmatrix} [1\,.\,.\,1] & [1\,.\,.\,1] & [2\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,2] \\ [1\,.\,.\,1] & [1\,.\,.\,1] & [1\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,3] & [2\,.\,.\,3] & [3\,.\,.\,4] & [3\,.\,.\,4] & [3\,.\,.\,5] \\ [2\,.\,.\,2] & [1\,.\,.\,2] & [2\,.\,.\,4] & [2\,.\,.\,4] & [3\,.\,.\,6] & [4\,.\,.\,6] & [4\,.\,.\,8] & [5\,.\,.\,8] & [5\,.\,.\,10] \\ [2\,.\,.\,2] & [2\,.\,.\,2] & [2\,.\,.\,4] & [4\,.\,.\,5] & [4\,.\,.\,7] & [5\,.\,.\,8] & [6\,.\,.\,9] & [7\,.\,.\,10] & [8\,.\,.\,12] \\ [2\,.\,.\,2] & [2\,.\,.\,3] & [3\,.\,.\,6] & [4\,.\,.\,7] & [5\,.\,.\,9] & [6\,.\,.\,10] & [8\,.\,.\,12] & [9\,.\,.\,14] & [10\,.\,.\,15] \\ [2\,.\,.\,2] & [2\,.\,.\,3] & [4\,.\,.\,6] & [5\,.\,.\,8] & [6\,.\,.\,10] & [8\,.\,.\,12] & [9\,.\,.\,14] & [11\,.\,.\,16] & [12\,.\,.\,18] \\ [2\,.\,.\,2] & [3\,.\,.\,4] & [4\,.\,.\,8] & [6\,.\,.\,9] & [8\,.\,.\,12] & [9\,.\,.\,14] & [11\,.\,.\,17] & [12\,.\,.\,19] & [14\,.\,.\,21] \\ [2\,.\,.\,2] & [3\,.\,.\,4] & [5\,.\,.\,8] & [7\,.\,.\,10] & [9\,.\,.\,14] & [11\,.\,.\,16] & [12\,.\,.\,19] & [14\,.\,.\,21] & [16\,.\,.\,24] \\ [2\,.\,.\,2] & [3\,.\,.\,5] & [5\,.\,.\,10] & [8\,.\,.\,12] & [10\,.\,.\,15] & [12\,.\,.\,18] & [14\,.\,.\,21] & [16\,.\,.\,24] & [18\,.\,.\,27] \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 3 & 2 & 5 & 1 & 6 & 2 & 10 \\ 1 & 0 & 2 & 0 & 2 & 0 & 2 & 0 & 2 \\ 1 & 0 & 5 & 2 & 10 & 2 & 21 & 1 & 46 \\ 1 & 0 & 1 & 0 & 2 & 0 & 1 & 0 & 2 \\ 1 & 0 & 6 & 2 & 21 & 1 & 48 & 1 & 150 \\ 1 & 0 & 2 & 0 & 1 & 0 & 1 & 0 & 3 \\ 1 & 0 & 10 & 2 & 46 & 2 & 150 & 3 & 649 \end{pmatrix}$$

The left-hand matrix shows how many black cells can occur in hitori covers. The right-hand matrix shows how many hitori covers have both horizontal and vertical symmetry; when $m \neq n$, such covers are counted just once in the previous totals, while the unsymmetrical covers are counted twice or four times. When $m = n$, such covers are counted either once (if there's 8-fold symmetry) or twice (otherwise); there are respectively (1, 0, 1, 0, 2, 0, 2, 0, 11) $n \times n$ hitori covers with 8-fold symmetry. Further types of 4-fold symmetry are possible when $m = n$: There's 90° rotational symmetry (but not 8-fold) in (0, 0, 0, 1, 1, 3, 11, 30, 106) pairs of cases; there's symmetry about both diagonals (but not 8-fold) in (0, 0, 0, 0, 0, 1, 4, 9, 49) pairs of cases. Figure A–6 shows some of the winners in this beauty contest for symmetrical hitori covers.

horizontal and vertical symmetry
8-fold symmetry
4-fold symmetry
90° rotational symmetry
symmetry about both diagonals

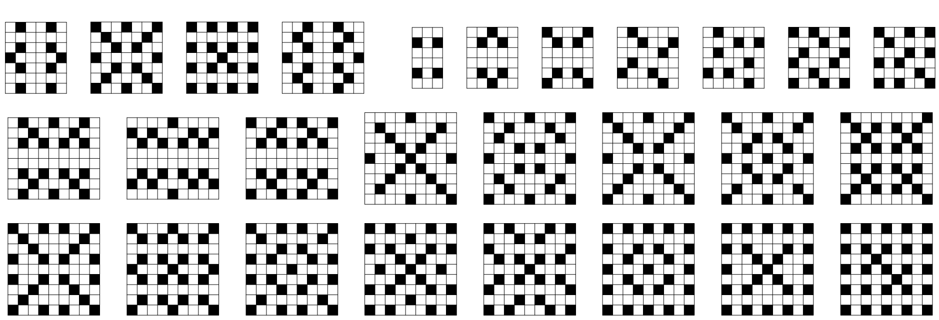


Fig. A–6. A gallery of interesting hitori covers.

Fourfold horizontal and vertical symmetry is impossible when m and n are both even, because it forces at least 12 white cells near the center. The number of $2 \times n$

hitori covers can readily be shown to satisfy the recurrence $X_n = 2X_{n-2} + 2X_{n-3}$, growing as $\Theta(r^n)$ where $r \approx 1.76929$.

443. (Solution by N. Beluhov.) Let there be s black cells, of which a lie in the interior, b on the boundary but not in a corner, and c in a corner. One can show that $b+2c \le m+n+2-[m\,\text{even}]-[n\,\text{even}]-[mn\,\text{odd}]$. Therefore the number of edges in $P_m \mathbin{\square} P_n \,|\, U$ is $m(n-1)+(m-1)n-4a-3b-2c = 2mn-m-n-4s+b+2c \le 2mn-4s+1$. But $P_m \mathbin{\square} P_n \,|\, U$ is connected, so it has at least $mn-s-1$ edges.

[Beluhov has also proved that the number of black cells is always at least $mn/5 - O(m+n)$. One can obtain a small hitori cover by blackening (i,j) when $i+2j$ is a multiple of 5, and possibly a few more cells; this cover has at most $mn/5+2$ black cells.]

444. No. By exercise 443, the solution has at most $\lfloor (n^2/3+2)/n \rfloor$ black cells in some row. This is at most $n/3$, when $n > 5$; hence $2n/3$ elements of that row are white. Conversely, the puzzle illustrated here for $n = 9$ can be generalized to $3k \times 3k$ for all $k > 1$. (It's a simplification of a construction by N. Beluhov. Notice that every nonzero element is a seed!)

0	0	1	0	2	3	0	4	5
0	1	0	2	3	0	4	5	0
1	6	2	3	0	4	5	0	0
0	2	3	6	4	5	0	0	1
2	3	0	4	5	0	0	1	0
3	0	4	5	6	0	1	0	2
0	4	5	0	0	1	6	2	3
4	5	0	0	1	0	2	3	0
5	0	0	1	0	2	3	6	4

445. Array (α) below is a seedless puzzle that corresponds to (ii), if you change its lowercase letters to uppercase. (The lowercase letters are convenient for our purposes in understanding seedlessness, because they indicate the cells that we'll want to darken.) When every black cell has a different letter to be hidden, a seedless puzzle must fill each white cell (i,j) with a hidden letter from either row i or column j.

Given a hitori cover, its "RC problem" is to put either R or C into each white cell so that the number of Rs in each row is at most the number of black cells in that row, and the number of Cs is similar but for columns. Array (β) shows the RC solution that corresponds to (α); this is one of four ways to solve the RC problem for (ii).

Suppose a hitori cover has s black cells. Every solution to its RC problem has at most s white cells marked R and at most s marked C; so we must have $s \ge n^2/3$ in an $n \times n$ cover. Consequently s must be 12 when $n = 6$, by exercise 443. In particular, pattern (i) can't lead to a seedless puzzle. Also, equality must hold when we said "at most."

It's easy to formulate the RC problem as an MCC problem, by introducing a primary item ij for each white cell (i,j), also primary items R_i and C_j for each nonwhite row i and column j. In the problem for pattern (ii) we have, for example, two options '23 R_2' and '23 C_3' for item 23. The multiplicity of C_3 is 2. (This is actually a bipartite matching problem; we use Algorithm M only because of the multiplicities.)

Array (γ) shows a seedless puzzle different from (α) that comes from the same RC solution (β). Indeed, (β) yields $3!1!2!2!1!3! \cdot 3!1!2!2!1!3! = 20736$ different seedless puzzles, because the letters chosen in each row and column can be permuted arbitrarily.

All such permutations yield valid puzzles. *Proof:* Each of the 12 letters occurs thrice. To solve the puzzle we must blacken each letter at least once, preserving white connectedness. One successful solution is to kill two birds with each stone; any other way would blacken 13 or more. But no 6×6 hitori cover has more than 12 black cells.

Pattern (iii) has eight RC solutions, and 20736 seedless puzzles for each of them.

Pattern (iv) has no RC solutions. But pattern (v) has the unique solution (δ), and one of its $3!0!3!2!1!3! \cdot 2!2!1!3!1!3! = 62208$ seedless puzzles is (ϵ).

(α)

a	A	P	P	B	b
A	W	Y	w	X	B
S	z	Q	R	Z	q
s	X	P	S	x	Q
D	Z	y	W	Y	C
d	D	R	r	C	c

(β)

■	R	■	R	R	■
C	R	C	■	C	C
C	■	R	C	R	■
■	R	C	R	■	C
C	C	■	C	R	C
■	R	R	■	R	■

(γ)

a	P	P	A	B	b
S	W	P	w	X	Q
A	z	Z	R	Q	q
s	X	Y	S	x	B
D	Z	y	W	Y	C
d	C	D	r	R	c

(δ)

R	■	R	■	R	■
C	C	C	C	C	C
R	■	R	■	R	■
■	R	R	C	■	C
C	C	■	C	R	C
■	R	R	■	R	■

(ϵ)

A	A	B	B	C	C
G	A	I	B	H	C
D	D	E	E	F	F
G	G	H	E	H	F
J	D	I	K	I	L
J	J	K	K	L	L

recurrence
Beluhov
Beluhov
seed
RC problem
bipartite matching problem

[N. Beluhov has proved that valid $n \times n$ seedless puzzles exist $\iff n \bmod 6 = 0$.]

446. There are only 1804 hitori covers, according to answer 442; but the exact probability appears to be difficult to compute. Experiments with millions of random numbers show convincingly, however, that the probability is $\approx .0133$. It drops to $\approx .0105$ with radix 8, and even further to $\approx .0060$ with radix 16; the "sweet spot" appears to be radix 10(!). [Also, the probability for decimal 4×4 is $\approx .0344$; for 6×6 only $\approx .0020$.]

447. Yes, when $2 \le m \le 4$ and $n = 6$! (Johan discovered the 4×6, and the 5×5 for e, in 2017. The cases 2×6, 3×2, and 4×5 also work for e. By exercise 443, we can assume that $m, n \le 15$.)

448. There are just two answers. (Also a nice 6×6 with only one not-so-common word.)

T	A	S	T	E
U	P	P	E	R
F	R	I	A	R
T	O	R	S	O
S	N	E	E	R

I	D	L	E	D
S	W	E	A	R
L	E	A	S	E
E	L	V	E	S
S	L	E	D	S

S	C	H	E	M	A
H	A	U	L	E	D
I	S	S	U	E	D
R	E	T	A	K	E
T	I	L	T	E	R
S	N	E	E	R	S

449. A few more nuggets: Johan noticed (i) in the (appropriately named) 1990 movie *Home Alone*; and he found (ii) in the King James Bible, Luke 9 : 56. George Sicherman hit on Falstaff's famous repartee (iii) in *1 Henry IV*, Act V, Scene 4, Line 119. The author found (iv) within the graffiti on page 278 of *CMath*; also (v), an inspiring remark by Francis Sullivan, on page 2 of *Computing in Science and Engineering* **2**, 1 (January/February 2000). Example (vi) appears in the front matter to Volume 1. And example (vii), also 11×3, shows that a nice hitori can involve lowercase letters, spaces, and punctuation; it's a quote from Samuel Rogers's poem *Human Life* (1819).

(i)

M	E	R	R
Y	C	H	R
I	S	T	M
A	S	S	W
E	E	T	H
E	A	R	T

(ii)

F	O	R	T	H	E
S	O	N	O	F	M
A	N	I	S	N	O
T	C	O	M	E	T
O	D	E	S	T	R
O	Y	M	E	N	S
L	I	V	E	S	B
U	T	T	O	S	A
V	E	T	H	E	M

(iii)

T	H	E	B
E	T	T	E
R	P	A	R
T	O	F	V
A	L	O	R
I	S	D	I
S	C	R	E
T	I	O	N

(iv)

W	E	L	L
T	H	E	Y
C	A	N	T
R	E	A	L
L	Y	G	O
A	T	I	T
T	H	I	S
L	O	N	G

(v)

G	R	E	A
T	A	L	G
O	R	I	T
H	M	S	A
R	E	T	H
E	P	O	E
T	R	Y	O
F	C	O	M
P	U	T	A
T	I	O	N

(vi)

D	O	Z
E	N	S
O	F	N
E	W	E
X	E	R
C	I	S
E	S	H
A	V	E
B	E	E
N	A	D
D	E	D

(vii)

N	e	v
e	r	
l	e	s
s		a
l	o	n
e		t
h	a	n
	w	h
e	n	
a	l	o
n	e	.

450. The solutions are characterized by 25 items {`tot`, `tibi`, ..., `caelo`, 1a, 1b, 1c, ..., 5a, 5b, 5c, 6a, 6b} and 80 options '`tot` 1a', '`tot` 1b 1c', ..., '`tot` 4b 4c', '`tot` 5a', '`tot` 6a', '`tot` 6b'; '`tibi` 1b 1c', '`tibi` 1c 2a', ..., '`tibi` 5c 6a'; ..., '`sidera` 1a 1b 1c', ..., '`sidera` 5a 5b 5c'; '`caelo` 1a 1b 1c', '`caelo` 1b 1c 2a', ..., '`caelo` 4b 4c 5a', '`caelo` 6a 6b'.

Beluhov
Bible
Sicherman
Falstaff
author
graffiti
Sullivan
Rogers
poem

APPENDIX E

ANSWERS TO PUZZLES IN THE ANSWERS

1	2	8	3	7	9	4	6	5
4	6	7	1	5	2	9	8	3
9	3	5	8	6	4	7	2	1
3	9	4	6	2	8	5	1	7
8	7	6	5	9	1	2	3	4
2	5	1	7	4	3	6	9	8
5	4	3	2	1	6	8	7	9
6	1	9	4	8	7	3	5	2
7	8	2	9	3	5	1	4	6

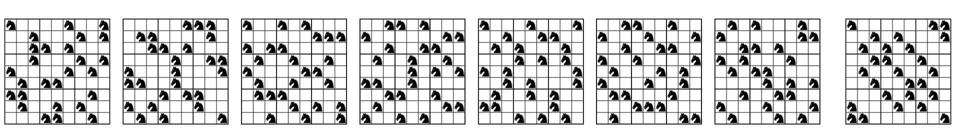
(see answer 52)

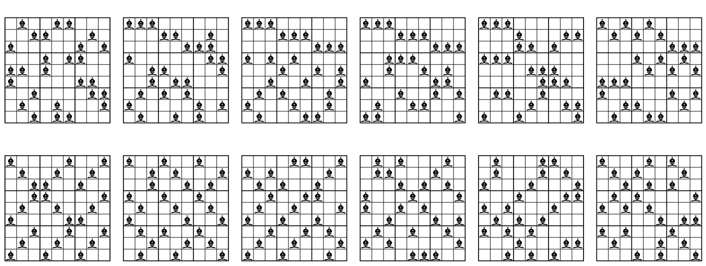
8	1	2	3	4	5	6	9	7
3	9	6	1	7	2	4	5	8
5	4	7			6	3	2	1
7	2		4	5		1	6	3
6	3		2	1	7		4	5
4	5	1		6	3		7	2
1	7	4	5			2	3	6
9	6	3	7	2	1	5	8	4
2	8	5	6	3	4	7	1	9

(see answer 58)

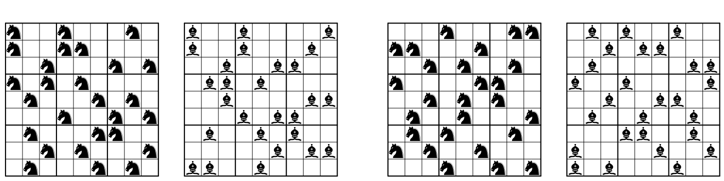
`SEVENTH`, `FOURTEEN`, `FIGHTER`, `REINVENT`, `VENTURES`;
`NONE`, `FORGIVEN`, `FORGIVES`, `UNTHRONE`;
`UNDOERS`, `FOUNDERS`, `CONDORS`, `TRIODES`, `ROUNDEST`,
`SECONDO`, `CERTIFY`, `FORTIFY`, `EXTRUDES`.

(see answer 112)

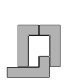




(see answer 173)



(see answer 174)

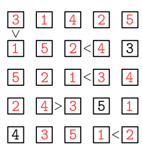


(see answer 282)

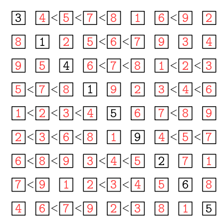
3	1	4	2	6	9	7	5	8
7	4	1	5	9	2	6	8	3
6	9	2	7	8	1	3	4	5
8	3	5	9	2	7	4	6	1
5	8	6	1	3	4	2	7	9
4	6	7	8	1	5	9	3	2
1	5	9	6	4	3	8	2	7
9	2	3	4	7	8	5	1	6
2	7	8	3	5	6	1	9	4

(see answer 302(c))

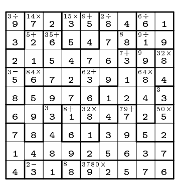
red bot $= {}^{188}_{185}{}_{335}$, mid $= {}^{179}_{279}{}_{435}$, top $= {}^{279}_{266}{}_{446}$; green bot $= {}^{599}_{869}{}_{866}$, mid $= {}^{557}_{877}{}_{443}$, top $= {}^{112}_{122}{}_{433}$ (see answer 337)



(see answer 395)



(see answer 396)



(see answer 403)

23	24	31	32	6	5	14	13
22	30	25	17	7	15	4	12
21	29	18	26	16	8	3	11
20	19	28	27	1	2	9	10

(see answer 407)

1	2	3	4	5	6
28	29	16	17	7	19
27	15	30	8	18	20
14	26	9	31	21	36
13	10	25	22	32	35
11	12	23	24	34	33

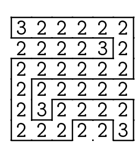
;

3	2	7	6	11	10
1	4	5	8	9	12
22	23	18	19	13	15
24	21	20	17	16	14
27	25	31	30	33	36
26	28	29	32	35	34

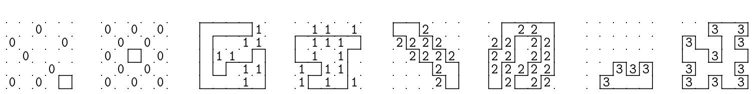
. (see answer 408)

60	30	31	32	33	34	35	41	40	39
59	61	29	26	27	54	53	36	42	38
62	58	25	28	55	97	52	93	37	43
63	24	57	56	98	96	94	51	92	44
23	64	84	85	86	99	95	91	50	45
22	21	65	83	100	87	88	90	49	46
19	20	66	67	82	81	80	89	48	47
18	69	68	74	75	1	79	3	5	6
70	17	73	76	14	78	2	4	7	9
71	72	16	15	77	13	12	11	10	8

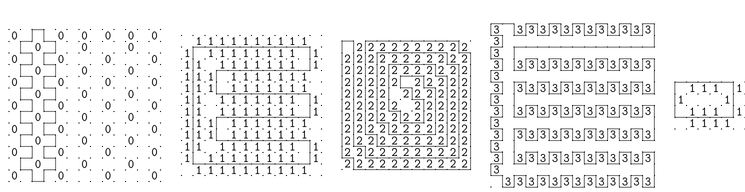
(see answer 409)



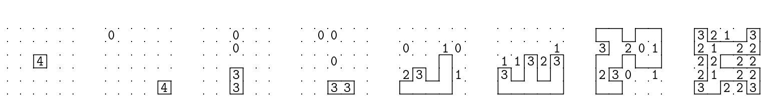
(see answer 411)



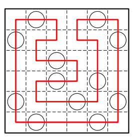
(see answer 415)



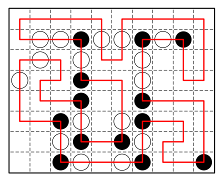
(see answer 416)



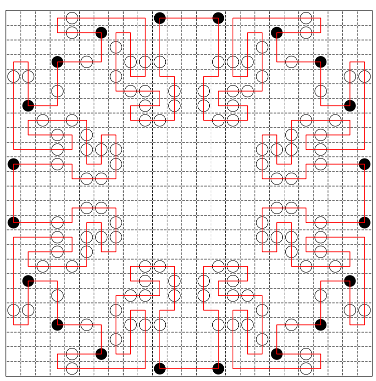
(see answer 418)



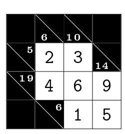
(see answer 424)



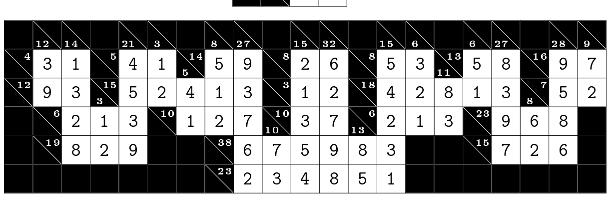
(see answer 426)



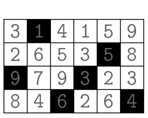
(see answer 427)



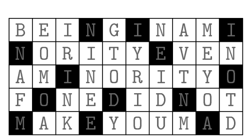
(see answer 430(a))



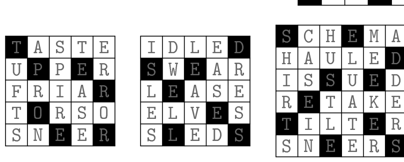
(see answer 435)



(see answer 447)



(see answer 449)



(see answer 448)

By my troth, we that have good wits, have much to answer for.
— SHAKESPEARE (*As You Like It*, Act V, Scene 1, Line 11)

INDEX AND GLOSSARY

Pope
Homer
WHEATLEY

There is a curious poetical index to the Iliad in Pope's Homer,
referring to all the places in which similes are used.
— HENRY B. WHEATLEY, *What is an Index?* (1878)

When an index entry refers to a page containing a relevant exercise, see also the *answer* to that exercise for further information. An answer page is not indexed here unless it refers to a topic not included in the statement of the exercise.