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 14A

BIPARTITE MATCHING

(ridiculously preliminary draft)

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY

February 8, 2023

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.

Zeroth printing (revision -91), 08 Feb 2023

BARRY
Internet

PREFACE

But that is not my point.
I have totally forgotten my point.
— DAVE BARRY (2012)

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 14 previews Section 7.5.1 of Chapter 7 of *The Art of Computer Programming*, entitled "Bipartite matching." I haven't had time to write much of it yet — not even this preface!

At present I've only got a few small scraps of copy that I've occasionally put into my computer, on days when possibly relevant material occurred to me. Thus almost everything you see here is more or less a place-holder for better things that hopefully will come later. Some day, however, I hope that I'll no longer have to apologize for what is now just a bunch of crumbs.

* * *

The explosion of research in computer science 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 15, . . . ; I've also implicitly mentioned or posed additional unsolved questions in the answers to exercises 13(d), ... Are those problems still open? Please inform me if you

Stappers
Knuth
HAUPTMAN

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 12, 13, ... Furthermore I've credited exercises 17, . . . to unpublished work of Filip Stappers and . . . . Have any of those results ever appeared in print, to your knowledge?

* * *

Special thanks are due to . . . for their detailed comments on my early attempts at exposition, as well as to 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:–)

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.
99 Umbruary 2021

For all such items, my procedure is the same:
I write them down — and then write them up.
— DON HAUPTMAN (2016)

7.5. GRAPHS AND OPTIMIZATION

HEYWOOD
BURTON
DEFOE
Crusoe
bipartite matching–
bipartite graph
matching, bipartite–
binary relation
matrix of 0s and 1s
maximum cardinality matching
postal codes

WE'VE NOW LEARNED all kinds of important properties of graphs, and their principal characteristics. Now let's focus on questions of *optimality*: Many problems ask for the "best possible" graph or subgraph of a certain type, often after the vertices and arcs have been given weights or other kinds of labels. For example, we studied the shortest path problem in Section 7.3, because it's probably the most important optimization problem of all. But many other challenging and instructive problems await us, and they turn out to be equally interesting.

Thus, lyke to lyke here matched is
What matche may match more mete then this
— JOHN HEYWOOD, *A Balade [. . . for] the Kynges and Queenes highnes* (1554)

Matches are made in heaven.
— ROBERT BURTON, *Anatomy of Melancholy* 3.2.5.5 (1624)

The Calamities of Life were shared among the upper and lower Part of Mankind.
— DANIEL DEFOE, *Robinson Crusoe* (1719)

7.5.1. Bipartite matching

A bipartite graph has vertices of two distinct kinds, say $\{x_1, x_2, \ldots, x_m\}$ and $\{y_1, y_2, \ldots, y_n\}$, and all of its edges run between those two "parts." It's the same concept as a *binary relation* between two sets X and Y, or as an $m \times n$ matrix of 0s and 1s, but expressed in a different way.

A *matching* in a graph is a set of edges that have no vertices in common. Thus, in a bipartite graph, a matching consists of pairs with the property that if x is matched to y and x' is matched to y', then $x = x'$ if and only if $y = y'$. In a 0–1 matrix, a matching is a way to select 1s that don't lie in the same row or column.

We sometimes find it convenient to speak of the participants in a bipartite matching as girls and boys, because matchmaking between the sexes is an ancient ritual for which the English language has developed a rich and useful vocabulary. (Of course when we use such language we don't mean to imply that one gender is more significant than the other.)

Our usual goal with respect to matching is to find a *maximum* matching, more precisely a "maximum cardinality matching," namely a matching that has the largest achievable number of edges. We shall see that there's a beautiful and efficient way to do this; indeed, it's one of the great success stories of combinatorial algorithms.

To set the scene, let's consider a puzzle based on US postal codes. The following 60 standard two-letter codes are often used to identify states or territories:

$$\begin{array}{l}\texttt{AK, AL, AR, AS, AZ, CA, CO, CT, DC, DE, FL, FM, GA, GU, HI, IA, ID, IL, IN, KS,}\\ \texttt{KY, LA, MA, MD, ME, MH, MI, MN, MO, MP, MS, MT, NC, ND, NE, NH, NJ, NM, NV, NY,}\\ \texttt{OH, OK, OR, PA, PR, PW, RI, SC, SD, TN, TX, UM, UT, VA, VI, VT, WA, WI, WV, WY.}\end{array} \qquad (1)$$

How many of these codes can we choose so that they don't "clash," in the sense that their first letters are all different, and so are their last letters? This is a bipartite matching problem, in which there are 19 girls $\{\mathtt{A},\mathtt{C},\ldots,\mathtt{V},\mathtt{W}\}$ (representing the first letters) and 22 boys $\{\mathtt{A},\mathtt{C},\ldots,\mathtt{Y},\mathtt{Z}\}$ (representing the last letters), constrained by 60 edges (representing the allowable pairings), as listed in (1).

maximal versus maximum
Berge
union of subgraphs
connected components
C_2
alternating path
augmenting path

That setup is a bit confusing, however, because we need a way to distinguish between the girl named `A` and the boy named `A`. Let's therefore rename the boys $\{\mathtt{a},\mathtt{c},\ldots,\mathtt{y},\mathtt{z}\}$, and change the postal codes (1) to $\{\mathtt{Ak},\mathtt{Al},\ldots,\mathtt{Wv},\mathtt{Wy}\}$. (Of course we won't try to write those newfangled codes on an actual envelope.)

If we simply pair each girl up with the alphabetically first boy not chosen by alphabetically previous girls, we get a matching of size 16:

$$\mathtt{Ak},\ \mathtt{Ca},\ \mathtt{Dc},\ \mathtt{Fl},\ \mathtt{Gu},\ \mathtt{Hi},\ \mathtt{Id},\ \mathtt{Ks},\ \mathtt{Me},\ \mathtt{Nh},\ \mathtt{Or},\ \mathtt{Pw},\ \mathtt{Tn},\ \mathtt{Um},\ \mathtt{Vt},\ \mathtt{Wv}. \qquad (2)$$

Three of the girls, $\{\mathtt{L},\mathtt{R},\mathtt{S}\}$ are left without a partner, as are six of the boys, namely $\{\mathtt{j},\mathtt{o},\mathtt{p},\mathtt{x},\mathtt{y},\mathtt{z}\}$. So this matching is probably not maximum. It is, however, *maximal*: Every edge *not* in the matching clashes with one of the edges that *is* present. It's always easy to find a *maximal* matching.

To go from maximal to maximum, we can use some nice ideas of Claude Berge [*Proc. National Acad. Sci.* **43** (1957), 842–844], based on the fact that the union $M\cup M'$ of matchings M and M' has a simple structure: Every vertex has degree ≤ 1 in M and degree ≤ 1 in M', so it has degree ≤ 2 in the multigraph $M\cup M'$. Hence the connected components of $M\cup M'$ are either isolated points or cycles or paths.

A cycle of $M\cup M'$ must have $2k$ edges, with k from each matching. (If $k=1$, the "cycle" C_2 comes from a repeated edge of $M\cap M'$.) Similarly, a path of $M\cup M'$ must alternate between edges of opposite matchings; so it has $k+k'$ edges, with k belonging to M and k' belonging to M'. And in that case we must have $|k-k'|\le 1$.

Suppose M is a maximum matching, with $|M|=s$ edges, while M' is an arbitrary matching that has $|M'|=s'\le s$ edges. Let's denote the edges of M by $x \mathbin{—} y$, and the edges of M' by $x \mathbin{=\!=} y$. A path of $M\cup M'$ with $k>k'$ is then

$$x_0 \mathbin{—} y_1 \mathbin{=\!=} x_1 \mathbin{—} y_2 \mathbin{=\!=} \cdots \mathbin{—} y_{k'} \mathbin{=\!=} x_{k'} \mathbin{—} y_k, \qquad (3)$$

with x_0 and y_k unmatched in M'. We call (3) an *augmenting path*, because it allows us to improve M' to a larger matching by changing from (3) to

$$x_0 \mathbin{=\!=} y_1 \mathbin{—} x_1 \mathbin{=\!=} y_2 \mathbin{—} \cdots \mathbin{=\!=} y_{k'} \mathbin{—} x_{k'} \mathbin{=\!=} y_k. \qquad (4)$$

The new M' has k edges in common with M, in place of k' edges it has decoupled.

That's very useful, because *there must exist at least $s-s'$ vertex-disjoint augmenting paths*. Indeed, augmenting paths are the only components of $M\cup M'$ that have $k>k'$. And although we don't know what the maximum matching M is, we do know what every augmenting path looks like, when we know M'.

For example, all of the augmenting paths that begin with an unmatched girl, with respect to the matching M' in (2), must continue as follows:

$$\texttt{L}\text{ — }\texttt{a}=\texttt{C}<\begin{matrix}\texttt{o}\\ \texttt{t}=\texttt{V}\cdots\end{matrix} \quad\text{or}\quad \texttt{R}\text{ — }\texttt{i}=\texttt{H} \quad\text{or}\quad \texttt{S}<\begin{matrix}\texttt{c}=\texttt{D}\text{ — }\texttt{e}=\texttt{M}\cdots\\ \texttt{d}=\texttt{I}<\begin{matrix}\texttt{a}=\texttt{C}\cdots\\ \texttt{l}=\texttt{F}\cdots\\ \texttt{n}=\texttt{T}\cdots\end{matrix}\end{matrix}\ . \qquad (5)$$

Aha! Already we've found an augmenting path $\texttt{L}$ — $\texttt{a}$ = $\texttt{C}$ — $\texttt{o}$ of length 3, and it tells us how to extend (2) to a matching of 17 codes:

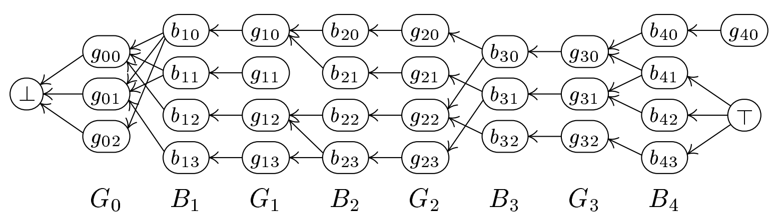
$$\texttt{Ak, Co, Dc, Fl, Gu, Hi, Id, Ks, La, Me, Nh, Or, Pw, Tn, Um, Vt, Wv.} \qquad (6)$$

The middle path of (5), '$\texttt{R}$ — $\texttt{i}$ = $\texttt{H}$', is also helpful, even though it fizzled out after only two steps, because it shows us that the two girls $\texttt{H}$ and $\texttt{R}$ can never be paired with different boys. Thus, $\texttt{H}$ and $\texttt{R}$ can't both be present in any matching; a maximum matching therefore can't contain more than 18 of the 19 girls.

Can we in fact find a matching of size 18? Yes, indeed; it's easy to find an augmenting path for (6). (See exercise 2.)

John Hopcroft and Richard M. Karp noticed [in *SICOMP* **2** (1973), 225–231] that there's a nice way to characterize all of the *shortest* augmenting paths, called SAPs, by constructing a directed acyclic graph. Start with arcs from a girl g to the special node $\perp$, whenever g is free (unmatched). Then proceed breadth-first as follows: After finding all the girls G_{k-1} at distance $2k-1$ from $\perp$, for some $k \ge 1$, find all the boys B_k at distance $2k$ by adding all arcs $g \longleftarrow b$, where b is adjacent to $g \in G_{k-1}$ and hasn't been included before. If none of those newly seen boys are free, add arcs $b \longleftarrow g'$ from their respective mates; these are the girls G_k at distance $2k+1$ from $\perp$, and we can increase k by 1. But if at least one boy at distance $2k$ is free, we're essentially done: We add arcs $b \longleftarrow \top$, one for every free boy in B_k, where $\top$ is another special node. *The SAPs are then precisely the paths from* $\top$ *to* $\perp$ *in this dag*, but with $\top$ and $\perp$ removed.

For example, the Hopcroft–Karp dag might look like this:



(7)

In this illustration the girls of G_k and the boys of B_k are labeled g_{kj} and b_{kj}. All edges of the graph between G_{k-1} and B_k appear here as arcs of the form $g_{(k-1)j} \longleftarrow b_{kj'}$. The other arcs $b_{kj} \longleftarrow g_{kj}$ mean that b_{kj} — g_{kj} is an edge of the current matching. The graph may contain edges that run between G_k and $B_{k'}$ for $k' \le k$; they aren't reflected in the dag. Notice that g_{11} has no edges except to B_1; and b_{40} is matched to g_{40}, hence not free. Therefore vertices g_{11}, b_{11}, b_{40}, and g_{40} don't occur in any paths from $\top$ to $\perp$; but they're (harmlessly) present in the dag anyway, because they arose during the construction process.

Hopcroft
Karp
SAPs
directed acyclic graph
breadth-first
dag: directed acyclic graph

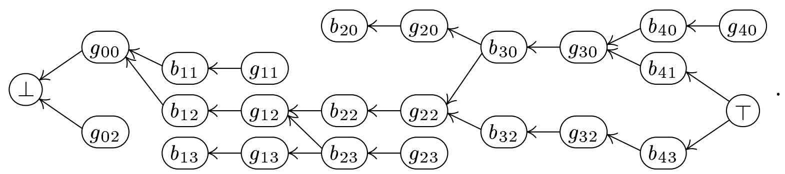
The amount of time needed to construct this dag is at worst proportional to the total number of edges. (And it is typically much less than this, because many additional already-matched girls and boys often exist, as we found in (5).)

Suppose we now *remove* any SAP from the dag, say

$$g_{0j_0} — b_{1j_1} = g_{1j_1} — \cdots = g_{(k-1)j_{k-1}} — b_{kj_k}, \qquad (8)$$

by deleting all its vertices and using it to augment the current matching. It turns out — and this is the key fact, proved in exercise 5 — that *the paths from* $\top$ *to* $\bot$ *in the remaining dag are precisely the augmenting paths of length* $\le 2k-1$ *in the new matching*, but with $\top$ and $\bot$ removed.

For example, removing $g_{01} — b_{10} = g_{10} — b_{21} = g_{21} — b_{31} = g_{31} — b_{42}$ in this way from (7) yields



. (9)

Hopcroft and Karp's algorithm therefore proceeds to dismantle the dag, SAP by SAP, until no more remain. Then a new dag can be made, for longer SAPs.

The following implementation of their algorithm uses an interesting combination of data structures. First there are "mate tables" to represent the current matching, with GMATE[g] for $1 \le g \le M$ and BMATE[b] for $1 \le b \le N$ to indicate the partners of g and b, or 0 if they're currently free.

The breadth-first construction of a dag is controlled by an array QUEUE[k] for $0 \le k < M$, which records the girls currently present. If f girls are free, they appear in the first f positions of QUEUE. There's also a partial inverse, IQUEUE[g] for $1 \le g \le M$: If $0 \le k < f$ and QUEUE[k] $= g$, then IQUEUE[g] $= k$. Yet another array, MARK[b] for $1 \le b \le N$, equals l if $b \in B_l$; otherwise MARK[b] $= 0$. There's also MARKED[t], for $0 \le t < N$; it lists the boys for which MARK[b] $\ne 0$.

The algorithm also involves a depth-first process, to remove SAPs after the dag has been built. Those steps use the array STACK[l], for $0 \le l < M$, to remember the boy of B_l who is currently being visited.

The bipartite graph that underlies everything is represented sparsely as a collection of *edge nodes*, each of which contains four fields GTIP, BTIP, GNEXT, BNEXT. An edge between girl g and boy b is represented by an edge node e for which GTIP(e) $= g$ and BTIP(e) $= b$; here $1 \le e \le E$, where E is the total number of edges. The first edge involving g, for $1 \le g \le M$, is GLINK[g]; the next one is GNEXT(GLINK[g]); and so on, until 0 terminates the list. The values of GTIP, BTIP, and GNEXT remain fixed throughout the computation.

A similar convention is used to represent the dag, which is constructed dynamically: The first arc from boy b in the dag is BLINK[b], for $1 \le b \le N$, and the next is BNEXT(BLINK[b]), etc. The contents of BLINK and BNEXT are therefore *not* fixed.

Hopcroft
Karp
data structures
mate tables
depth-first process
sparse graph representation
edge nodes

maximum cardinality matching
backtracking

Every girl g in the dag is the source of exactly one arc, which leads to `GMATE[`g`]`. If `GMATE[`g`]` $= 0$, that arc leads to $\perp$.

Algorithm H (*Maximum bipartite matching*). Given a bipartite graph with M girls, N boys, and E edges, represented as explained above, this algorithm computes a maximum cardinality matching, which will appear in the `GMATE` and `BMATE` arrays. It also uses the auxiliary arrays `QUEUE`, `IQUEUE`, `MARK`, `MARKED`, and `STACK`, defined above. The `MARK` array must be initially zero.

H1. [Prime the pump.] Set `GMATE` and `BMATE` to a maximal (not necessarily maximum) matching; also set f to the number of unmatched girls, and list them in the first f slots of `QUEUE`. (See exercise 7.)

H2. [Start building the dag.] Set $t \leftarrow i \leftarrow l \leftarrow r \leftarrow 0$, $q \leftarrow f$, and $L \leftarrow 0$.

H3. [Begin level $l+1$.] (At this point the girls of G_l are listed in `QUEUE[`k`]` for $i \le k < q$, and the dag contains t boys.) Set $q' \leftarrow q$.

H4. [Process a $g \in G_l$.] Go to H10 if $i = q'$. Otherwise set $g \leftarrow$ `QUEUE[`i`]`, $i \leftarrow i+1$, and $e \leftarrow$ `GLINK[`g`]`.

H5. [Let b be a suitor for g.] If $e = 0$, return to H4; otherwise set $b \leftarrow$ `BTIP(`e`)`.

H6. [Is b new?] If `MARK[`b`]` $= 0$, go to H8. Otherwise if `MARK[`b`]` $> l$, set `BNEXT(`e`)` $\leftarrow$ `BLINK[`b`]`, `BLINK[`b`]` $\leftarrow e$.

H7. [Loop on b.] Set $e \leftarrow$ `GNEXT(`e`)` and return to H5.

H8. [Enter b into B_{l+1}.] If $L > 0$ and `BMATE[`b`]` $\neq 0$, go to H7. Otherwise set `MARK[`b`]` $\leftarrow l+1$, `MARKED[`t`]` $\leftarrow b$, $t \leftarrow t+1$, `BLINK[`b`]` $\leftarrow e$, `BNEXT(`e`)` $\leftarrow 0$.

H9. [Is b free?] If `BMATE[`b`]` $\neq 0$, set `QUEUE[`q`]` $\leftarrow$ `BMATE[`b`]`, $q \leftarrow q+1$. Otherwise if $L = 0$, set $L \leftarrow l+1$, $r \leftarrow 1$, $q \leftarrow q'$ (we've reached the final level). Otherwise set $r \leftarrow r+1$ (there are r free boys on level L). Go to H7.

H10. [Is the dag complete?] If $q \neq q'$, set $l \leftarrow l+1$ and return to H3. (Otherwise the dag is complete, and the last r elements of `MARKED` are the free boys in B_L.) Terminate the algorithm if $L = 0$ (there are no augmenting paths).

H11. [Start to find a SAP.] If $r = 0$, set `MARK[MARKED[`k`]]` $\leftarrow 0$ for $0 \le k < t$ and return to H2. Otherwise set $b \leftarrow$ `MARKED[`$t-r$`]`, $r \leftarrow r-1$, $l \leftarrow L$.

H12. [Enter level l.] Set `STACK[`l`]` $\leftarrow b$.

H13. [Advance.] Set $e \leftarrow$ `BLINK[`b`]`, and go to H15 if $e = 0$. Otherwise set `BLINK[`b`]` $\leftarrow$ `BNEXT(`e`)`, $g \leftarrow$ `GTIP(`e`)`. If `MARK[GMATE[`g`]]` < 0, repeat this step (g has been deleted). Otherwise set $b \leftarrow$ `GMATE[`g`]`.

H14. [SAP complete?] If $b = 0$ (g is free), go to H16. Otherwise set $l \leftarrow l-1$ and return to H12.

H15. [Resume higher level.] Set $l \leftarrow l+1$. Then go to H11 if $l > L$; otherwise set $b \leftarrow$ `STACK[`l`]` and go back to H13. (This is like "backtracking," except that we never retrace a step because we're destroying the dag as we go.)

H16. [Prepare to augment.] (At this point $l = 1$; $g = g_0$ and `STACK[1]` $= b_1$ in a SAP. The other boys are `STACK[2]`, ..., `STACK[`L`]`.) Set $f \leftarrow f-1$,

$k \leftarrow$ `IQUEUE[`g`]`, $i \leftarrow$ `QUEUE[`f`]`, `QUEUE[`k`]` $\leftarrow i$, and `IQUEUE[`i`]` $\leftarrow k$. (Those operations removed g from the list of free girls.) Set $b \leftarrow$ `STACK[1]`.

H17. [Augment.] Set `MARK[`b`]` $\leftarrow -1$, $g' \leftarrow$ `BMATE[`b`]`, `BMATE[`b`]` $\leftarrow g$, and `GMATE[`g`]` $\leftarrow b$. Then if $g' \neq 0$, set $g \leftarrow g'$, $l \leftarrow l+1$, $b \leftarrow$ `STACK[`l`]`, and repeat this step. Otherwise go back to H11. ▮

This algorithm has many steps, but it's not frighteningly complicated. It essentially consists of two separate-but-cooperating subalgorithms, namely the breadth-first dag construction in H2–H10 and the depth-first dag deconstruction in H11–H17.

Algorithm H comes with an important free bonus: After it has found a supposedly maximum matching, its data structures contain enough information to convince any skeptic that the matching is indeed as large as possible. Indeed, if no girl is free, the matching is perfect and obviously optimum. Otherwise the girls in `QUEUE[`k`]` for $0 \le k < q$ are adjacent to only t boys in the graph, namely the boys in `MARKED[`k`]` for $0 \le k < t$. And it's easy to verify that $q = t + f$; hence any matching must leave at least f girls without a partner. (See exercise 9.) Indeed, Algorithm H provides us with a maximum independent set,

$$I = \{g \mid g \text{ is a girl in the final dag}\} \cup \{b \mid b \text{ is a boy not in the final dag}\}, \quad (10)$$

which is certified by the maximum matching and vice versa! (See exercise 10.)

Algorithm H's main claim to fame, however, is that it runs remarkably fast. Give it a graph, and it churns out a maximum matching, lickety-split. The reason is that SAPs are extremely good augmenters:

Theorem H. *Let s be the size of a maximum matching. When $r = 0$ in step H11, the size of the current matching is at least $\frac{L}{L+1}s$.*

Proof. If the current matching has s' edges, we've observed that at least $s - s'$ vertex-disjoint augmenting paths exist. We also know that each of those paths contains at least $L+1$ edges of a maximum matching. So $s \ge (L+1)(s-s')$. ▮

Corollary K. *The running time for Algorithm H to find a maximum matching of size s is $O\big((M+N+E)\sqrt{s}\,\big)$.*

Proof. Every time a dag is constructed, the value of L increases. Each round of construction and deconstruction clearly involves $O(M+N+E)$ steps. If the algorithm hasn't terminated before the value of L exceeds $\sqrt{s}$, a matching of size $\ge \frac{\sqrt{s}-1}{\sqrt{s}}s = s - \sqrt{s}$ has been found, and $\sqrt{s}$ more rounds will complete the task. ▮

It's not easy to come up with bipartite graphs for which Algorithm H might take anywhere near $\sqrt{s}$ rounds of computation. Exercise 14 investigates some hypothetical examples, which might be as bad as possible; but even when such a graph is given, its edges have to appear in a particularly perverse order if the algorithm is to be repeatedly fooled.

Thus it makes sense to act in practice as if the running time of Algorithm H were just $O(E)$. Rajeev Motwani has proved, for example, that long SAPs are extremely rare, in many models of random graphs [*JACM* **41** (1994), 1329–1356].

breadth-first
depth-first
certificate of correctness
maximum independent set in bipartite graph
Motwani
random graphs

If the input to Algorithm H is transposed, so that $M \leftrightarrow N$ and the girls and boys switch roles, the size of the final matching won't change; but the computations leading up to it might be quite different. Empirical tests by Filip Stappers show that it's usually better to have fewer girls than boys, if you have a choice.

Stappers
edge cover
Lovász
Plummer
Berge
marriage theorem
latin square

* * *

Who knows what I might eventually say next? [Among many important potential directions, I should probably turn at this point to minimum edge covers: If there are n nonisolated points, the edge cover number plus the matching number is n. Lovász and Plummer explain this nicely near the beginning of their book Matching Theory (1986). See also C. Berge, Graphs and Hypergraphs (1973), §7.2.]

Certainly I'll be mentioning the "marriage theorem," which other sections refer to as Theorem 7.5.1H: A perfect matching exists if and only if no subset of k boys has fewer than k potential mates. Also its corollary, Theorem 7.5.1L: Every $m \times n$ latin rectangle with $m < n$ can be completed to an $n \times n$ latin square.

* * *

Historical remarks.

historical remarks–
Dinitz
maximum flow in networks
shortest augmenting paths
Even
Itai

* * *

Yefim Dinitz devised an algorithm for maximum flow in networks [*Doklady Akademii Nauk SSSR* **194** (1970), 754–757; English translation in *Soviet Math. Doklady* **11** (1970), 1277–1280], based on shortest augmenting paths. Subsequent improvements by S. Even and A. Itai led to a procedure that is essentially equivalent to Algorithm H, in the special case where a maximum flow is also a maximum bipartite matching. (See *LNCS* **3895** (2006), 218–240.)

* * *

EXERCISES

1. [*21*] Find a maximal matching of the postal codes (1) that's as small as possible.

2. [*16*] Extend (6) to a matching of size 18, via an augmenting path of length 5.

▸ **3.** [*20*] Notice that only one of the codes in (1) ends with `Z`. Prove that, without loss of generality, `AZ` can be assumed to be part of *any* maximum matching of those codes. What further reductions of this sort can be used to simplify that matching problem?

▸ **4.** [*21*] If the matching that underlies (7) were augmented, by pairing $g_{01} = b_{10}$, $g_{10} = b_{21}$, $g_{21} = b_{31}$, and $g_{31} = b_{42}$ instead of $b_{10} = g_{10}$, $b_{21} = g_{21}$, and $b_{31} = g_{31}$, what dag would the Hopcroft–Karp procedure construct for the new matching?

5. [*M21*] Prove that if we remove any augmenting path of length $2k-1$ from a Hopcroft–Karp dag, and use it to augment the current matching, we don't get any new augmenting paths of length $\le 2k-1$. *Hint:* The set of all boys adjacent to $G_0 \cup \cdots \cup G_{j-1}$ is $B_1 \cup \cdots \cup B_j$, for $1 \le j \le k$.

6. [*18*] Given pairs (g_e, b_e) for $1 \le e \le E$, representing the edges of a bipartite graph, construct the sparse representation of that graph in the format required by Hopcroft and Karp's Algorithm H.

▸ **7.** [*20*] Explain how to implement step H1, performing at most $O(M+N+E)$ steps.

8. [*20*] Why would it be wrong to say '$l = 1$' instead of '$b = 0$' in step H14?

9. [*16*] Show that $q \ne t + f$ in Algorithm H only when $L > 0$.

▸ **10.** [*M20*] Let the matching found by Algorithm H consist of C couples.

a) Show that no independent set can have more than $M + N - C$ elements.

b) Furthermore the set I in (10) has $M + N - C$ elements.

▸ **11.** [*HM34*] (*Mancala solitaire.*) An ancient mancala-type game called Tchouka has a one-person variant called *Tchoukaillon* that leads to fascinating mathematical patterns: There are s stones arranged in "pits," with p_k stones in pit k. A legal move consists of finding a $k > 0$ with $p_k = k$, removing all k stones from that pit, and "sowing" them into pits $k-1, \ldots, 1, 0$. The object of the game is to end with all stones in pit 0, so that $p_0 = s$. For example, a 10-stone winning strategy begins with $p_5p_4p_3p_2p_1 = 53110$ and makes the following 10 moves: 04221, 04220, 04201, 04200, 00311, 00310, 00021, 00020, 00001, 00000 (with $p_0 = j$ after j moves).

a) Show that there's exactly one way to win s-stone Tchoukaillon. What is it?

b) Let that winning strategy have $p_{k,s}$ stones in pit k, and let $m_{k,s}$ be the total number of moves it makes from that pit. For example, $p_{5,10}p_{4,10}\ldots p_{1,10} = 53110$ and $m_{5,10}m_{4,10}\ldots m_{1,10} = 11125$. Also let $q_{k,s} = p_{k+1,s} + p_{k+2,s} + \cdots$ and $l_{k,s} = m_{k+1,s} + m_{k+2,s} + \cdots$, so that $q_{0,s} = l_{0,s} = s$ and $q_{\infty,s} = l_{\infty,s} = 0$. Prove that these four arrays of numbers are highly interconnected: If $k > 0$ and $s \ge 0$ we have

i) $m_{k,s} = \lceil l_{k-1,s}/(k+1) \rceil$; iv) $q_{k,s} = (k+1)l_{k,s}$;

ii) $l_{k,s} = \lfloor \frac{k}{k+1} l_{k-1,s} \rfloor$; v) $p_{k,s} = q_{k-1,s} \bmod (k+1)$;

iii) $p_{k,s} = km_{k,s} - l_{k,s}$; vi) $p_{k+1,s} - p_{k,s} = (k+2)m_{k+1,s} - km_{k,s}$.

c) Let $Ч_n$ be the least s that needs pit n. (This sequence begins $Ч_1, Ч_2, Ч_3, \ldots =$ 1, 2, 4, 6, 10, 12, 18, 22, 30, 34, 42, 48, 58, 60, 78, 82, 102, 108, ...) Prove that

$$Ч_n = \left\lceil \frac{2}{1} \left\lceil \frac{3}{2} \left\lceil \frac{4}{3} \cdots \left\lceil \frac{n}{n-1} \right\rceil \cdots \right\rceil \right\rceil \right\rceil \qquad \text{(with } n-1 \text{ ceiling brackets).}$$

d) Asymptotically, $Ч_n = n^2/\pi + O(n^{4/3})$. *Hint:* Let $f_{m,s} = \min\{k \mid m_{k,s} \le m\}$, and show that $4^{-m}\binom{2m}{m} f_{0,s} - 1 \le f_{m,s} \le 4^{-m}\binom{2m}{m} f_{0,s} + 1$ for all $m \ge 0$.

postal codes
augmenting path
Hopcroft
Karp
sparse representation
Mancala solitaire
solitaire
game
Tchoukaillon
$Ч_n$, Tchoukaillon numbers
Asymptotically

12. [*20*] The "sieve of Tchoukaillon" is like the famous sieve of Eratosthenes, but simpler. Begin with a list of all positive integers. Then accept the first, and strike out every second entry that follows. Accept the first remaining element, and strike out every third entry of the current survivors. Again accept the first remaining element; this time strike out every fourth survivor. And so on:

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	32	33	34	35	36
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~~	32	~~33~~	34	~~35~~	36
1	**2**		4		6		~~8~~		10		12		~~14~~		16		18		~~20~~		22		24		~~26~~		28		30		~~32~~		34		36
1	**2**		**4**		6				10		12				~~16~~		18				22		24				~~28~~		30				34		36
1	**2**		**4**		**6**				10		12						18				22		~~24~~						30				34		36
1	**2**		**4**		**6**				**10**		12						18				22								30				34		~~36~~

What numbers are accepted? (See the previous exercise.)

▸ **13.** [*M25*] The *Tchoukaillon array* $\text{Ч}^{(n)}$ *of order* n is an $\infty \times n$ permutation of the positive integers defined by the following recursive rule:

$$\text{Ч}^{(1)}_{q,1} = q+1; \quad \text{Ч}^{(n+1)}_{qn+r,j} = \begin{cases} \text{Ч}^{(n)}_{q(n+1)+r,j}, & \text{if } j+r<n; \\ \text{Ч}^{(n)}_{q(n+1)+r+1,j-1}, & \text{if } j+r \ge n; \end{cases} \quad \text{for } n \ge 1,\ q \ge 0,\ 0 \le r < q.$$

Here, for example, are rows 0, 1, …, 5 of the first seven arrays:

$\text{Ч}^{(1)}$	$\text{Ч}^{(2)}$		$\text{Ч}^{(3)}$			$\text{Ч}^{(4)}$				$\text{Ч}^{(5)}$					$\text{Ч}^{(6)}$						$\text{Ч}^{(7)}$						
1	1	2	1	2	4	1	2	4	6	1	2	4	6	10	1	2	4	6	10	12	1	2	4	6	10	12	18
2	3	4	3	5	6	3	5	8	10	3	5	8	11	12	3	5	8	11	16	18	3	5	8	11	16	20	22
3	5	6	7	8	10	7	9	11	12	7	9	14	16	18	7	9	14	17	20	22	7	9	14	17	23	24	30
4	7	8	9	11	12	13	14	16	18	13	15	17	20	22	13	15	21	23	24	30	13	15	21	26	28	32	34
5	9	10	13	14	16	15	17	20	22	19	21	23	24	30	19	25	26	28	32	34	19	25	29	33	35	36	42
6	11	12	15	17	18	19	21	23	24	25	26	28	32	34	27	29	33	35	36	42	27	31	37	38	40	46	48

a) Where do the numbers Ч_n appear in these arrays?

b) Prove that the rows and columns of $\text{Ч}^{(n)}$ are monotonically increasing.

c) Prove that $\text{Ч}^{(n)}_{i,j} \le \text{Ч}^{(n+1)}_{i,j}$, for all integers $i \ge 0$, $j \ge 0$, and $n > j$.

d) What's the smallest n for which $\text{Ч}^{(n)}_{i,j}$ reaches its largest value, $\text{Ч}^{(\infty)}_{i,j}$?

▸ **14.** [*M34*] One way to try to make Algorithm H run slowly is to find a graph for which L increases by only 1 in each round, and the minimum number of SAPs is removed. Suppose the graph input to Algorithm H has a perfect matching of size $s = M = N$.

a) If every augmenting path of the current matching has the same length, show that the dag constructed in steps H2–H10 has $f = |G_0| = |B_1| = \cdots = |G_{L-1}| = |B_L|$.

b) Consequently steps H11–H17 will find at least $\lceil f/(L+1) \rceil$ SAPs.

c) Say that G is a Tchoukaillon graph if it might cause Algorithm H to continually satisfy assumption (a) as it builds dags for $L = 2, 3, \ldots$, starting with $f = \lfloor s/2 \rfloor$ free girls after step H1 and always removing only $\lceil f/(L+1) \rceil$ thereafter. How many rounds would the algorithm then perform? (See exercise 11.)

d) Construct a Tchoukaillon graph with $s = 16$.

15. [*46*] Does a Tchoukaillon graph exist for arbitrary s?

16. [*16*] How large can L become, in Algorithm H?

▸ **17.** [*22*] Prove that there exists a maximum matching with girl g unmatched if and only if Algorithm H terminates with $g \in \{\texttt{QUEUE[0]}, \ldots, \texttt{QUEUE[}q-1\texttt{]}\}$.

18. [*M25*] (F. Stappers, 2021.) A bipartite graph with M girls and N boys can be defined by an $M \times N$ matrix (a_{ij}) of 0s and 1s, where $a_{ij} = [g_i \mathrel{—} b_j]$ for $1 \le i \le M$

sieve of Tchoukaillon
Eratosthenes
Tchoukaillon array
recursive
monotonically increasing
Tchoukaillon graph
unmatched in maximum matching
Stappers
0-1 matrix

and $1 \le j \le N$. Consider two ways to find a partial matching in this graph, starting with `GMATE[`i`]` $=$ `BMATE[`j`]` $= 0$ for all i and j:

Method G: For $i = 1, 2, \ldots, M$ do this: For $j = 1, 2, \ldots, N$, if $a_{ij} = 1$ and `BMATE[`j`]` $= 0$, set `GMATE[`i`]` $\leftarrow j$ and `BMATE[`j`]` $\leftarrow i$.

Method B: For $j = 1, 2, \ldots, N$ do this: For $i = 1, 2, \ldots, M$, if $a_{ij} = 1$ and `GMATE[`i`]` $= 0$, set `BMATE[`j`]` $\leftarrow i$ and `GMATE[`i`]` $\leftarrow j$.

a) Do both methods yield a maximal matching?

b) Do both methods yield exactly the same matching?

99. [*00*] this is a temporary dummy exercise

999. [*M00*] this is a temporary exercise (for dummies)

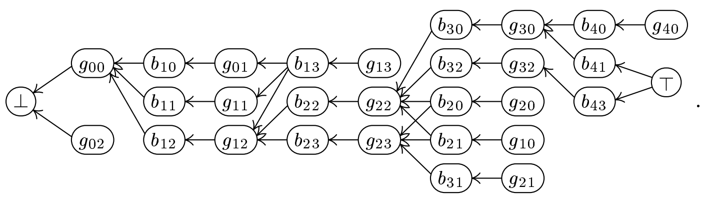
SECTION 7.5.1

1. A maximal matching is a kernel of the line graph of the graph, so it can be characterized by the Boolean function in Eq. 7.1.4–(68). A ZDD for that function has 45971 nodes, and shows that there are exactly 1798286 kernels, with generating function $16z^{11} + 3432z^{12} + \cdots + 68096z^{17} + 1976z^{18}$. One of the sixteen smallest is $\{\mathtt{AL}, \mathtt{CT}, \mathtt{DC}, \mathtt{GA}, \mathtt{KY}, \mathtt{MD}, \mathtt{NM}, \mathtt{OH}, \mathtt{PR}, \mathtt{TN}, \mathtt{WI}\}$; all sixteen include $\{\mathtt{AL}, \mathtt{CT}, \mathtt{DC}, \mathtt{MD}, \mathtt{NM}, \mathtt{TN}, \mathtt{WI}\}$.

2. For example, fans of Texas will prefer the augmenting path $\mathtt{S} — \mathtt{d} = \mathtt{I} — \mathtt{n} = \mathtt{T} — \mathtt{x}$, which yields `Ak`, `Co`, `Dc`, `Fl`, `Gu`, `Hi`, `In`, `Ks`, `La`, `Me`, `Nh`, `Or`, `Pw`, `Sd`, `Tx`, `Um`, `Vt`, `Wv`.

3. If `AZ` isn't in a matching, we can put it in by excluding the edge that contains `A`. And in general whenever a bipartite graph has a vertex of degree 1, we can remove that vertex and its mate, obtaining a graph with an equivalent maximum-matching problem but with one less girl and one less boy. Any maximum matching of (1) can therefore be assumed to contain `AZ`, then `TX`, and successively `PW`, `GU`, `OR`, `MP`, `KS`, `CO`, `IN`, `FL`, `NJ`, `WY`, `UM`, `VT`, `DE`, `SC`, `LA`, `RI` — and the remaining graph has no edges!

4. Well, the answer depends on unknown parts of the graph: We don't know which edges between G_k and $B_{k'}$ are present when $k \ge k'$, except that $g_{kj} — b_{kj}$. If we assume that *all* such edges exist, the dag is



(Although this looks rather different from (9), it defines the same two SAPs.)

5. The hint is true by induction on j. Consequently, if $g_0 — b_1 = g_1 — b_2 = \cdots = g_{l-1} — b_l$ is any augmenting path for the new matching, we have $g_j \in G_0 \cup \cdots \cup G_j$ and $b_j \in B_1 \cup \cdots \cup B_j$. Moreover, if $g_j \notin G_j$ or $b_j \notin B_j$ for any j, then $g_{j+1} \notin G_{j+1}$ and $b_{j+1} \notin B_{j+1}$. Hence $l > k$, for b_k will lie in $B_1 \cup \cdots \cup B_{k-1}$ and will not be free.

6. Start with $\mathtt{GLINK}[g] \leftarrow 0$ for $1 \le g \le M$, and allocate E edge nodes. Then for $1 \le e \le E$, set $\mathtt{GTIP}(e) \leftarrow g_e$, $\mathtt{BTIP}(e) \leftarrow b_e$, $\mathtt{GNEXT}(e) \leftarrow \mathtt{GLINK}[g_e]$, $\mathtt{GLINK}[g_e] \leftarrow e$.

7. Set $f \leftarrow 0$ and $\mathtt{BMATE}[b] \leftarrow 0$ for $1 \le b \le N$. Then do the following for $M \ge g \ge 1$: Set $e \leftarrow \mathtt{GLINK}[g]$; and while $e > 0$ and $\mathtt{BMATE}[\mathtt{BTIP}(e)] > 0$, set $e \leftarrow \mathtt{GNEXT}(e)$. Then if $e > 0$, set $b \leftarrow \mathtt{BTIP}(e)$, $\mathtt{GMATE}[g] \leftarrow b$, $\mathtt{BMATE}[b] \leftarrow g$. But if $e = 0$, set $\mathtt{QUEUE}[f] \leftarrow g$, $\mathtt{IQUEUE}[g] \leftarrow f$, and $f \leftarrow f + 1$. (This matching is maximal, so the first dag will set $L \ge 2$; but if it weren't, the first round would find a maximal matching with $L = 1$.)

8. A girl g at level 0 might no longer be free, because the matching was augmented via a previously found SAP. In that case, her current mate b has $\mathtt{MARK}[b] < 0$.

9. Initially $q = f$, $t = 0$. When t advances in H8, q advances in H9.

10. (a) The complement of an independent set is a vertex cover, so it has $\ge C$ vertices.
(b) If the final dag has k boys, it has $f + k$ girls, where $f = M - C$.

11. (a) Any winning state must have $p_k \le k$ for all k; otherwise the stones in pit k would be stuck forever. Therefore any winning strategy must choose the *smallest* $k > 0$ with $p_k = k$. Hence pit k will be the smallest empty pit after that winning move; and we can reconstruct an s-stone winning state uniquely from the one for $s-1$ stones. (It's

kernel
line graph
Boolean function
ZDD
edge nodes
complement
vertex cover

helpful to study the list of those states for $s \le 24$, say, with leading zeros suppressed: 1, 20, 21, 310, 311, 4200, 4201, 4220, 4221, 53110, 53111, 642000, 642001, 642020, 642021, 642310, 642311, 7531200, 7531201, 7531220, 7531221, 86420110, 86420111, 86424000.)

(b) Let's omit the second subscript when it's equal to s; then (i) becomes simply '$m_k = \lceil l_{k-1}/(k+1) \rceil$', etc. The total number of stones removed from pit k is $p_k + l_k$, and they are removed k at a time; this proves (iii), and (vi) follows from (iii).

If $q_0 > q_1 > \cdots > q_{h-1} = q_h$, we have $p_1 \ne 0, \ldots, p_{h-1} \ne 0$, $p_h = 0$; hence $q_{k,s+1} = q_k + (k+1)[k < h]$, and (iv) follows by induction. The identity $p_k = q_{k-1} - q_k$ now establishes (v), because $p_k \equiv q_{k-1}$ (modulo $k+1$) and $0 \le p_k \le k$.

Finally, (i) and (ii) are equivalent because $l_{k-1} - l_k = m_k$. If $l_0 > 2l_1 > \cdots > hl_{h-1} = (h+1)l_h$, we've shown that $l_{k,s+1} = l_k + [k < h]$, for all $k \ge 0$. Therefore if (ii) holds for s, we have $\lfloor \frac{k}{k+1}(l_{k-1} + [k < h]) \rfloor = \lfloor \frac{k}{k+1} l_{k-1} \rfloor + [k < h]$, and it holds for $s+1$.

(c) The least l_{k-1} satisfying (ii) is $\lceil \frac{k+1}{k} l_k \rceil$; start with $l_{n-1} = 1$.

(d) Write just f_m for $f_{m,s}$. We're interested in the case $s = \text{Ч}_n$ and $f_0 = n+1$. Notice that the coefficients $\phi_m = 4^{-m}\binom{2m}{m} = (\pi m)^{-1/2} + O(m^{-3/2})$ satisfy $\phi_0 + \cdots + \phi_{m-1} = 2m\phi_m$. The hint can therefore be proved by induction if we can prove

$$f_0 + f_1 + \cdots + f_{m-1} - m \;\le\; 2mf_m \;\le\; f_0 + f_1 + \cdots + f_{m-1} + m. \qquad (*)$$

Let $k = f_m$. Then we have $m_{k-1} > m \ge m_k = \lceil l_{k-1}/(k+1) \rceil$. Hence $(f_m + 1)m \ge l_{k-1} = m(f_{m-1} - f_m) + (m-1)(f_{m-2} - f_{m-1}) + \cdots + (1)(f_0 - f_1) = f_0 + f_1 + \cdots + f_{m-1} - mf_m$ (the left half of $(*)$). Furthermore if $m_{k-1} = m+1$, we have $m < l_{k-2}/k$; hence $mf_m \le l_{k-2} - 1 = l_{k-1} + m = f_0 + f_1 + \cdots + f_{m-1} - mf_m + m$ (the right half).

The proof is more delicate when $m_{k-1} > m+1$, because $m+1$ doesn't occur as a value. Suppose $m_{k-1} = m + r + 1$. Then $f_m = f_{m+1} = \cdots = f_{m+r} > f_{m+r+1}$; and we've proved that $S \ge 0$, where $S = f_0 + \cdots + f_{m+r-1} + m + r - 2(m+r)f_{m+r}$. Good: Now $f_0 + \cdots + f_{m-1} + m - 2mf_m = S + \Delta$, where $\Delta = -f_{m+r-1} - \cdots - f_m - r + 2(m+r)f_{m+r} - 2mf_m = rf_m - r$. Either $\Delta \ge 0$ or $f_m < 1$; we've proved the hint!

Our goal is to approximate $s = \sum_{k=1}^{n} p_k = \sum_{k=1}^{f_r - 1} p_k + \sum_{m=1}^{r} S_m$, where $S_m = \sum_{f_m \le k < f_{m-1}} p_k$. Let $r = n^{2/3}$; then the first sum is negligible, because $p_k \le k$ and $\sum_{k=1}^{f_r - 1} k = O(f_r^2) = O(n^{4/3})$. Each remaining sum S_m, over the range $f_m \le k < f_{m-1}$, is the sum of an increasing arithmetic progression with difference $2m$, by (vi). The first term of this progression, assuming that $f_m \ne f_{m-1}$, is $p_{f_m} = p_{f_m - 1} + (f_m + 1)m - (f_m - 1)m_{f_m - 1} \le 2m$. Hence $S_m = m(f_{m-1} - f_m)^2 + E_m$, where $|E_m| \le m(f_{m-1} - f_m)$; and $\sum_{m=1}^{r} E_m = O(l_r) = O(n^{4/3})$. The fact that $\phi_{m-1} - \phi_m = \phi_{m-1}/(2m)$ tells us that $m(f_{m-1} - f_m)^2 = n^2\phi_{m-1}^2/(4m) + O(n\phi_{m-1})$, another negligible error. The coefficient of n^2 in the final sum therefore comes to $\sum_{m=1}^{\infty} \phi_{m-1}^2/(4m) = \frac{1}{4}F\left({1/2,1/2 \atop 2} \,\middle|\, 1\right) = 1/\pi$.

[See D. M. Broline and D. E. Loeb, *UMAP* **16** (1995), 21–36, who stated erroneously that they had proved a better bound, $O(n)$, on the error. It turns out that the only number $n \le 100000$ with $|\text{Ч}_n - n^2/\pi| > 2n$ is $n = 63313$, when we have $\text{Ч}_n = 1276095874 \approx n^2/\pi + 2.2n$. However, such examples are not uncommon for larger n; for example, $\text{Ч}_n = 12430256394 \approx n^2/\pi + 2.4n$ when $n = 197609$. See also B. Jones, L. Taalman, and A. Tongen, *AMM* **120** (2013), 706–724; and OEIS A002491.]

It is a pleasant surprise to see π arise from such a simple game.
— NEIL J. A. SLOANE, *My favorite integer sequences* (1998)

12. Let h_s be the pit selected in the winning strategy for s. Pit k is selected every $(k+1)$st time we haven't selected a prior one, because its contents decrease steadily as s increases. When h_s first equals n, we accept $s = \text{Ч}_n$.

[P. Erdös and E. Jabotinski, *Indagationes Mathematicæ* **20** (1958), 115–128.]

arithmetic progression
Broline
Loeb
UMAP: The Journal of Undergraduate Mathematics and Applications, published by the Consortium for Mathematics and its Applications since 1980.
Jones
Taalman
Tongen
OEIS
SLOANE
Ч_n
Erdös
Jabotinski

13. (a) The top row of $\text{Ч}^{(n)}$ contains Ч_1, …, Ч_n. The other Tchoukaillon numbers appear in the rightmost column, where they are sieved out exactly as in exercise 12: The rightmost elements in rows $n+1$, $2(n+1)$, $3(n+1)$, … of $\text{Ч}^{(n)}$ are deleted.

(b) Notice that we get $\text{Ч}^{(n+1)}$ by partitioning $\text{Ч}^{(n)}$ into $(n+1)\times n$ blocks and permuting each of those blocks into an $n\times(n+1)$ block. The rows stay monotone, by induction, because elements (i,j) and $(i,j+1)$ of $\text{Ч}^{(n+1)}$ are either in the same row of $\text{Ч}^{(n)}$ or in the same column. Monotonicity in column 0 is also easy.

But a *stronger* induction hypothesis is needed to verify that the other columns are monotone. The trick is to prove inductively that element (i,j) is less than element $(i+1,j-1)$, when $j>0$.

(c) In $\text{Ч}^{(n)}$, "squash" rows n, $2n+1$, $3n+2$, … one by one, applying (b).

(d) $n=i+j+1$. [Column 0 becomes the Flavius Josephus sieve, OEIS A00960.]

(The limiting array $\text{Ч}^{(\infty)}$ suggests a host of fascinating questions, yet unresolved. For example, is there a "simple" way to understand the elements {3, 5, 8, 11, 16, 20, 24, 32, 36, 46, 54, 59, 72, 80, 90, …} of row 1? The elements {2, 5, 9, 15, 25, 31, 43, 61, 67, 87, 103, 123, 139, 169, 183, …} of column 1 [A100287] are known to be related to Josephus's sieve; but what about column 2? Is there a rapid way to determine which row and column contains a given integer? When the elements are colored modulo m, do they make interesting pixel patterns? What's the asymptotic value of $\text{Ч}_{i,j}^{(\infty)}$? What's the asymptotic shape of the region occupied by numbers $\le n$? Numerical evidence for small i and j has led N. Beluhov to conjecture that we will have $\text{Ч}_{i,j}^{(\infty)} \approx (\pi i+2j)^2/(4\pi)$ in general. His formula is known to be correct when $i=0$, by exercise 11(d); it is also known to be correct when $j=0$, because M. E. Andersson [*Acta Arithmetica* **85** (1998), 301–307] showed that the nth element of the Flavius Joseph sieve is $\frac{\pi}{4}n^2+O(n^{4/3})$.)

14. (a) There are f "basic" vertex-disjoint augmenting paths of the form $g_{0j} — b_{1j} = g_{1j} — \cdots = g_{(L-1)j} — b_{Lj}$, where the edges $g_{(i-1)j} — b_{ij}$ belong to the unknown perfect matching. Therefore $|G_{i-1}| \le |B_i|$ for $1 \le i \le L$. Furthermore $|B_L| \le f$; there can't be more free boys than free girls when $M=N$. And we always have $|B_i| = |G_i|$.

(b) No augmenting path intersects with more than $L+1$ of the basic ones.

(c) The rule $f \leftarrow f - \lceil f/(L+1)\rceil = \lfloor \frac{L}{L+1} f\rfloor$ matches the formula for $l_{k,s}$ in exercise 11(b). So the number of rounds (the final value of L) is $n \approx \sqrt{\pi s}$ when $\text{Ч}_n \le s < \text{Ч}_{n+1}$.

(d) Let $g_k — b_k$ for $1 \le k \le s$ be the edges of the perfect matching. It's convenient to represent each dag of a Tchoukaillon graph by an $f \times L$ matrix whose rows contain the indices of the boy-girl pairs in the basic augmenting paths; for example, the row '1 2 4' stands for the basic path $g_1 — b_1 = g_2 — b_2 = g_4 — b_4$. To go from L to $L+1$, we must find $\lfloor \frac{L}{L+1} f\rfloor$ suitable disjoint paths from left to right in the $f\times L$ matrix, where each path consists of $L-1$ rightward steps plus a step that stays in the same column (but doesn't necessarily move to an adjacent row). One solution for $s=16$ has matrices

$$\begin{array}{cc} 1&2\\ 3&4\\ 5&6\\ \cdots&\\ 15&16 \end{array}\qquad \begin{array}{ccc} 1&2&4\\ 3&5&6\\ 7&8&10\\ 9&11&12\\ 13&14&16 \end{array}\qquad \begin{array}{cccc} 1&2&4&6\\ 3&5&8&10\\ 7&9&11&12 \end{array}\qquad \begin{array}{ccccc} 1&2&4&6&10\\ 3&7&9&11&12 \end{array}\qquad \begin{array}{cccccc} 1&2&4&6&10&12 \end{array};$$

it has 19 edges besides the perfect ones, namely $b_1 — g_2$, $b_3 — g_4$, …, $b_{15} — g_{16}$, $b_2 — g_4$, $b_3 — g_5$, $b_8 — g_{10}$, $b_9 — g_{11}$, $b_{14} — g_{16}$, $b_4 — g_6$, $b_5 — g_8$, $b_7 — g_9$, $b_6 — g_{10}$, $b_3 — g_7$, $b_{10} — g_{12}$. (This solution almost agrees with the opening rows of the arrays in exercise 13. But unfortunately we cannot have $b_8 — g_{11}$, as in $\text{Ч}^{(5)}$; $b_{\mathrm{even}} — g_{\mathrm{odd}}$ is forbidden because the matching in $\text{Ч}^{(2)}$ is supposed to be *maximal*.)

induction hypothesis
Flavius Josephus sieve
Josephus sieve
sieve
pixel patterns
asymptotic value
Beluhov
Andersson

15. An even harder question is to ask whether any graph exists that forces the algorithm to take more rounds than a hypothetical Tchoukaillon graph would (by violating assumption (a) and/or by finding more SAPs). Matula

16. The augmenting path $g_0 — b_0 = g_1 \cdots b_{M-1} = g_{M-1} — b_M$ makes $L = M$.

17. The partial dag has level sets G_0, B_1, $\ldots$, G_l, with $B_l = \emptyset$. It contains paths $g_0 — b_1 = g_1 — \cdots b_k = g_k$ from which we can make any $g_k \in G_k$ mateless, by changing the current matching to $g_0 = b_1 — g_1 = \cdots b_k — g_k$. Conversely, any maximum matching with g free leads to such a path in the partial dag. (See D. W. Matula, *Annals of Discrete Math.* **2** (1978), 91–106, Theorem 3.4.)

18. (a) Yes. (For example, G matches i unless all her potential partners are matched.)

(b) Yes. Let $b_{ij} = [g_i \text{ is matched to } b_j]$. Then $b_{ij} = a_{ij} \wedge \neg b_{i0} \wedge \cdots \wedge \neg b_{i(j-1)} \wedge \neg b_{0j} \wedge \cdots \wedge \neg b_{(i-1)j}$, and this condition is symmetrical between rows and columns.

99. ...

999. ...

INDEX AND GLOSSARY

Hippocrates
D'ISRAELI

I, for my part, venerate the inventor of indexes;
and I know not to whom to yield the preference,
either to Hippocrates, who was the first great anatomiser of the human body,
or to that unknown labourer in literature,
who first laid open the nerves and arteries of a book.

— ISAAC D'ISRAELI, *Miscellanies* (1796)

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.