☆ THE ART OF COMPUTER PROGRAMMING ☆

☆☆☆☆ ERRATA TO VOLUME 4 FASCICLE 1 ☆☆☆☆

This document is a transcript of the notes that I have been making in my personal copy of *The Art of Computer Programming*, Volume 4, Fascicle 1, since it was first printed in 2009.

Four levels of updates — "errors," "amendments," "plans," and "improvements" — appear, indicated by four different typographic conventions:

▸**Page 666** line 1 ———— 04 Jul 1776

Technical or typographical errors (aka bugs) are the most critical items, so they are flagged with a '▸' preceding the page number. The date on which I first was told about the bug is shown; this is the effective date on which I paid the finder's fee. The necessary corrections are indicated in a straightforward way. If, for example, the book says 'n' where it should have said '$n+1$', the change is shown thus:

$n \rightsquigarrow n+1$

Page 666 line 2 ———— 14 Jul 1789

Amendments to the text appear in the same format as bugs, but without the '▸'. These are things I wish I had known about or thought of when I wrote the original text, so I added them later. The date is the date I drafted the new text.

Page 666 line 3 . . . . 20 Nov 1917

Plans for the future represent a third kind of item. In such notes I sketched my intentions about things that I wasn't ready to flesh out further when I wrote them down. You can identify these items because they're written in slanted type, and preceded by a bunch of dots '. . . . .' leading to the date on which I recorded the plan in my files.

Page 666 line 4 ———— 10 Jan 1938

The fourth and final category — indicated by page and line number in smaller, slanted type — consists of minor corrections or improvements that most readers don't want to know about, because they are so trivial. You wouldn't even be seeing these items if you hadn't specifically chosen to print the complete errata list in all its gory details. Are you sure you wanted to do that?

The ultimate, glorious, future editions of Volumes 1–5 are works in progress. Please let me know of any improvements that you think I ought to make. Send your comments either by snail mail to D. E. Knuth, Computer Science, Gates Building 4B, Stanford University, Stanford CA 94305-9045, or by email to `taocp@cs.stanford.edu`. *(Use email for book suggestions only, please — all other correspondence is returned unread to the sender, or discarded, because I have no time to read ordinary email.) Although I'm working full time on*

Volume 4 these days, I will try to reply to all such messages within a year of receipt. Current news about The Art of Computer Programming is posted on

`http://www-cs-faculty.stanford.edu/~knuth/taocp.html`

and updated regularly.

— Don Knuth, January 2009

He did a lot of editing. That's how he liked to work.
Sometimes he even made alterations between printings.
— P. D. JAMES, *The Lighthouse* (2005)

☆☆ BITWISE TRICKS/TECHNIQUES, BDDS ☆☆

Last updated 1 January 2011

Most of these corrections have already been made in recent printings.

Important note: The changes below include nearly every difference between the paperback fascicle and the first printing of Volume 4A, published in January 2011. All subsequent changes can be found in the errata list for that volume.

▸**Page ∞** back cover line 10 ———— 07 Apr 2009

problems.All ⇝ problems. All

Page ii line −14 ———— 26 Apr 2009

Copyright © 2009 ⇝ Copyright © MMIX

▸**Page iii** line −7 ———— 25 Mar 2009

subection 7.1.3 ⇝ subsection 7.1.3

▸**Page vi** line 5 ———— 22 Jan 2010

by a sharp sign ⇝ by a number sign or hash mark

Page 10 replacement for three lines following (53) ———— 26 May 2010

[This technique was devised in 1967 by Luther Woodrum of IBM's Systems Development Division (unpublished); many other programmers have subsequently discovered it independently.]

Page 14 lines 4–5 ———— 21 Dec 2009

Every setting of the individual crossbars therefore ⇝ To start the recursion when $n = 1$, we let $P(2)$ consist of a single crossbar. Every setting of the individual crossbars clearly

Page 17 a few lines before (81) ———— 01 Jul 2009

the necessary masks ⇝ suitable masks

Page 17 replacement for the paragraph containing (81) ———— 06 May 2009

A "sheep-and-goats" or "grouping" operation has been suggested for computer hardware, extending (78) to produce the general unshuffled word

$$(x_{r-1}\ldots x_1x_0y_{s-1}\ldots y_1y_0)_2 = (z_{i_{r-1}}\ldots z_{i_1}z_{i_0}z_{j_{s-1}}\ldots z_{j_1}z_{j_0})_2; \quad (81)$$

here $i_{r-1} > \cdots > i_1 > i_0$ are the indices where $\chi_i = 0$. But another operation called "gather-flip," which reverses the order of the unmasked bits and gives

$$(x_0x_1\ldots x_{r-1}y_{s-1}\ldots y_1y_0)_2 = (z_{i_0}z_{i_1}\ldots z_{i_{r-1}}z_{j_{s-1}}\ldots z_{j_1}z_{j_0})_2, \qquad (81')$$

turns out to be more useful and easier to implement. Any permutation of 2^d bits is achievable by using either operation, at most d times (see exercises 72 and 73).

[To make room for this new material, several lines of page 16 have moved to page 15, and the index has been changed accordingly.]

Page 19 new sentence for bottom line ——— 06 May 2009

See also R. B. Lee, *IEEE Micro* **16**, 4 (August 1996), 51–59.

Page 22 just above (97) ——— 04 Apr 2009

to storage allocation $\rightsquigarrow$ to finding r contiguous free blocks when allocating storage

▸**Page 27** line 8 ——— 24 Dec 2009

442–451 $\rightsquigarrow$ 442–453

Page 33 line 5 ——— 29 Sep 2010

nodes. $\rightsquigarrow$ nodes. (These are the “inclusive” descendants, including j itself.)

Page 33 line -13 ——— 29 Sep 2010

an *ancestor* $\rightsquigarrow$ an (inclusive) *ancestor*

Page 33 line -2 ——— 29 Sep 2010

vertex u $\rightsquigarrow$ vertex u (always meaning an *inclusive* ancestor)

Page 41 in Fig. 16 ——— 29 Apr 2010

replace the Chinese character at the left by the following more appropriate one:

細

Page 44 line 15 ——— 13 Oct 2009

41–47 $\rightsquigarrow$ 41–48

▸**Page 47** line 20 ——— 03 Jul 2009

$(5,9)$ — $(5,10)$ $\rightsquigarrow$ $(6,9)$ — $(6,10)$

▸**Page 47** replacement for lines 22–25 ——— 03 Jul 2009

Movement to the left in step T8 is conveniently implemented by setting $H(y) \leftarrow H(y) \oplus (1 \ll (x_{\max} - x))$, using the H vectors of (166) and (167). Movement to the right is similar, but we set $x \leftarrow x+1$ first. Step T10 could set

$$H(y) \leftarrow H(y) \oplus ((1 \ll (x_{\max} - \min(x,x'))) - (1 \ll (x_{\max} - \max(x,x')))); \quad (172)$$

▸**Page 51** line 14 ——— 01 Jun 2009

$\bigl(\alpha_{n-1}(x)x^{8(n-1)} + \cdots + \alpha_m(x)x^{8m}\bigr)x^{16} \bmod p(x)$ $\rightsquigarrow$
$\bigl(\alpha_{n-1}(x)x^{8(n-1-m)} + \cdots + \alpha_{m+1}(x)x^{8} + \alpha_m(x)\bigr)x^{16} \bmod p(x)$

Page 52 new copy for last line of exercise 9 ——— 23 Aug 2009

[*Hint:* Use exercise 8.]

▸**Page 55** line 2 of exercise 35 ——— 15 Dec 2009

such that $\nu(n^+) + \nu(n^-)$ is minimized $\leadsto$ such that $n^+ \mathbin{\&} n^- = (n^+ \mid n^-) \mathbin{\&} ((n^+ \mid n^-) \gg 1) = 0$

Page 56 line 5 of exercise 58 ——— 10 Mar 2009

"Omega router." $\leadsto$ "Omega router" or "inverse butterfly."

Page 57 new part for exercise 58 ——— 07 May 2009

f) Prove that the permutation $0\varphi \ldots (N-1)\varphi$ is Omega-routable if and only if it is sorted by Batcher's bitonic sorting network of order N. (See Section 5.3.4.)

Page 57 replacement for exercise 72 ——— 07 May 2009

72. [*25*] (Y. Hilewitz and R. B. Lee.) Prove that the gather-flip operation (81′) is Omega-routable in the sense of exercise 58.

Page 58 replacement for exercise 73 ——— 07 May 2009

73. [*22*] Prove that d well-chosen steps of (a) the sheep-and-goats operation (81) or (b) the gather-flip operation (81′) will implement any desired 2^d-bit permutation.

Page 67 line 2 of exercise 190 ——— 27 Jun 2010

neighbors $\leadsto$ rook-neighbors

Page 70 new exercise ——— 09 Mar 2010

▸**218.** [*M30*] (Hans Petter Selasky, 2009.) For fixed $d \ge 3$, design an algorithm to compute $a \cdot x^y \bmod 2^d$, given integers a, x, and y, where x is odd, using $O(d)$ additions and bitwise operations together with a single multiplication by y.

Page 74 line 21 ——— 18 Oct 2010

list all solutions $\leadsto$ *generate all solutions*

▸**Page 76** line 10 after (12) ——— 05 May 2009

number of paths from node k to $\boxed{\top}$. $\leadsto$ number of ways to go from node k to $\boxed{\top}$ by choosing values for $x_l \ldots x_n$, if l is the label of node k.

▸**Page 80** line 4 ——— 13 Oct 2009

connectivity function $\leadsto$ connectedness function

Page 81 line 3 after (26) ——— 02 Jul 2010

$\binom{n+2}{2} \leadsto \binom{n+1}{2} + 2$

Page 99 beginning at line 21 ——— 01 Dec 2009

[My original presentation of this material could be simplified because the function $a(x)$ wasn't needed. Therefore I've replaced everything from the former (67) through the first three lines of page 100 by the following new copy.]

$$\mathrm{IND}(x) = \neg \bigvee_{u \,-\, v} (x_u \wedge x_v); \tag{67}$$

$$\mathrm{KER}(x) = \mathrm{IND}(x) \wedge \bigwedge_{v} \Bigl(x_v \vee \bigvee_{u \,-\, v} x_u\Bigr). \tag{68}$$

We can form a new graph $\mathcal{G}$ whose vertices are the kernels of G, namely the vectors x such that $\mathrm{KER}(x) = 1$. Let's say that two kernels x and y are *adjacent* in $\mathcal{G}$ if they differ in just the two entries for u and v, where $(x_u, x_v) = (1, 0)$ and $(y_u, y_v) = (0, 1)$, in which case we'll also have $u \longrightarrow v$. Kernels can be considered as certain ways to place markers on vertices of G; moving a marker from one vertex to a neighboring vertex produces an adjacent kernel. Formally we define

$$\mathrm{ADJ}(x, y) = [\nu(x \oplus y) = 2] \wedge \mathrm{KER}(x) \wedge \mathrm{KER}(y). \qquad (69)$$

Then $x \longrightarrow y$ in $\mathcal{G}$ if and only if $\mathrm{ADJ}(x, y) = 1$.

Notice that, if $x = x_1 \ldots x_n$, the function $[\nu(x) = 2]$ is the symmetric function $S_2(x_1, \ldots, x_n)$. Furthermore $f(x \oplus y)$ has at most 3 times as many nodes as $f(x)$, if we interleave the variables zipperwise so that the branching order is $(x_1, y_1, \ldots, x_n, y_n)$. So $B(\mathrm{ADJ})$ won't be extremely large unless $B(\mathrm{KER})$ is large.

Quantification now makes it easy to express the condition that x is an *isolated vertex* of $\mathcal{G}$ (a vertex of degree 0, a kernel without neighbors):

$$\mathrm{ISO}(x) \;=\; \mathrm{KER}(x) \wedge \neg\exists y\, \mathrm{ADJ}(x, y). \qquad (70)$$

For example, suppose G is the graph of contiguous states in the USA, as in (18). Then each kernel vector x has 49 entries x_v for $v \in \{\mathtt{ME}, \mathtt{NH}, \ldots, \mathtt{CA}\}$. The graph $\mathcal{G}$ has 266,137 vertices, and we have observed earlier that the BDD sizes for $\mathrm{IND}(x)$ and $\mathrm{KER}(x)$ are respectively 428 and 780 (see (17)). In this case $\mathrm{ADJ}(x, y)$ in (69) has a BDD of only 7260 nodes, even though it's a function of 98 Boolean variables.

Page 100 bottom line ———— 01 Dec 2009

$a(x \oplus y) \rightsquigarrow [\nu(x \oplus y) = 2]$

▸**Page 106** line −13 ———— 27 Dec 2009

$f(x'') = 1 \rightsquigarrow P_m^\pi(x'') = 1$

▸**Page 106** replacement for (99) ———— 27 Dec 2009

$$\{i_1(x), i_2(x), \ldots, i_{k-1}(x)\} = \{i_1(x'), i_2(x'), \ldots, i_{k-1}(x')\}$$
$$\text{and } \{j_1(x), j_2(x), \ldots, j_{k-1}(x)\} = \{j_1(x'), j_2(x'), \ldots, j_{k-1}(x')\}. \qquad (99)$$

Page 108 line −6 ———— 22 Aug 2010

transmogrified $\rightsquigarrow$ transmogrified into 2s

Page 108 line −2 ———— 22 Aug 2010

appear at the right $\rightsquigarrow$ appear above t_2 at the right

Page 110 line 19 ———— 31 May 2010

new size $\rightsquigarrow$ new size of those levels

Page 110 line −14 ———— 31 May 2010

but every such spawns at most two beads of half the size in $\rightsquigarrow$ but every bead on level $j - 1$ of f spawns at most two beads, of half the size, in

▸**Page 112** line 8 ——— 31 Jun 2010

the variable that is currently called x_j is the original variable x_k $\leadsto$ the original variable x_k is currently called either x_{j-1} or x_j

▸**Page 113** line 11 after (107) ——— 05 Aug 2009

been entirely $\leadsto$ has been entirely

▸**Page 115** line −9 ——— 16 Jun 2010

$Z_{n,y}$ $\leadsto$ $Z_{n,a}$

▸**Page 118** replacement for lines −11 through −6 ——— 04 Apr 2010

also for z-profiles, but with q'_k counting only *nonzero* subtables of order $n-k$:

$$q'_k \;\ge\; z_k, \qquad \text{for } 0 \le k < n; \tag{123}$$

$$q'_k \le 1 + z_0 + \cdots + z_{k-1} \;\text{ and }\; q'_k \le z_k + \cdots + z_n, \quad \text{for } 0 \le k \le n; \tag{124}$$

$$Z(f) \;\ge\; 2q'_k - 1, \qquad \text{for } 0 \le k \le n. \tag{125}$$

Consequently the BDD size and the ZDD size can never be wildly different:

$$Z(f) \;\le\; \frac{n}{2}\bigl(B(f)+1\bigr)+1 \qquad \text{and} \qquad B(f) \;\le\; \frac{n}{2}\bigl(Z(f)+1\bigr)+2. \tag{126}$$

▸**Page 121** line 26 ——— 22 Aug 2010

(131). $\leadsto$ (131)

Page 124 line 12 ——— 13 Sep 2010

function is *sweet* $\leadsto$ function f is *sweet*

▸**Page 126** line 8 ——— 14 Oct 2009

5–44 $\leadsto$ 5–36

▸**Page 126** line 20 ——— 14 Aug 2009

weights satisfying $w_i \ge 0$ $\leadsto$ weights are nonnegative

▸**Page 126** line 23 ——— 14 Oct 2009

(1999) $\leadsto$ (1997)

Page 129 line 1 of exercise 37 ——— 24 Dec 2009

Vuillemin $\leadsto$ Vuillemin, 1976

Page 129 line 1 of exercise 44 ——— 13 Sep 2010

S_n $\leadsto$ Σ_n

▸**Page 134** line 1 of exercise 110 ——— 14 Feb 2010

$f_n = U_n$ $\leadsto$ $B(f_n) = U_n$

Page 134 line 1 of exercise 113 ——— 28 Oct 2009

having sink nodes $\leadsto$ having two sink nodes

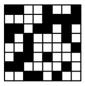
Page 137 line 2 of exercise 155 ——— 01 Dec 2009

see (67) $\leadsto$ see (68)

Page 137 line 5 of exercise 160 ———— 30 Jan 2010

$L_{ij} = 0 \rightsquigarrow L_{ij}(X) = 0$

Page 138 replacement for pattern (4) in line 2 ———— 18 Nov 2010



▸**Page 146** line 1 of exercise 256 ———— 06 Mar 2009

as family of $\rightsquigarrow$ as a family of

▸**Page 146** line 1 of exercise 259 ———— 17 May 2009

can be can be $\rightsquigarrow$ can be

Page 148 new exercise ———— 01 Oct 2009

▸**265.** [*21*] Devise an algorithm that finds the mth smallest solution to $f(x) = 1$ in lexicographic order of $x_1 \ldots x_n$, given m and the BDD for a Boolean function f of n variables. Your algorithm should take $O(nB(f) + n^2)$ steps.

Page 149 in answer 5 ———— 04 Feb 2010

digit $\rightsquigarrow$ bit (four times)

▸**Page 149** line 3 of answer 8 ———— 29 Nov 2009

$|\alpha| = |\beta| = |\gamma| \rightsquigarrow |\alpha'| = |\beta'| = |\gamma'|$

▸**Page 149** line 5 of answer 10 ———— 13 May 2009

$xy' + x'y + x'y' < xy$ when $x' < x$ and $y' < y \rightsquigarrow xy' + x'y + x'y' \neq xy$ when $0 \le x' < x$ and $0 \le y' < y$

▸**Page 153** last line of answer 19 ———— 16 May 2009

263–279 $\rightsquigarrow$ 265–279

Page 156 last line of answer 35 ———— 24 Dec 2009

paper a8, 14 pp. $\rightsquigarrow$ A08:1–A08:14.

▸**Page 156** replacement for last line of answer 36 ———— 12 Dec 2009

(See exercises 66 and 70 for applications of $x^{\oplus}$; see also exercises 128 and 209.)

▸**Page 156** line 2 of answer 38 ———— 04 Oct 2010

`SRU y,y,1; CMPU t,x,0; ADDU y,y,t`' takes $15v \rightsquigarrow$ `SRU t,y,1; SUBU y,y,t`' takes $14v$

Page 157 replacement for lines 9–12 of answer 44 ———— 04 Oct 2010

But on `MMIX`, which has sideways addition built in, there's a better solution by J. Dallos:

```
SADD  nu2,x,m5     SADD  t,x,m3       2ADDU nu2,nu2,t    SADD  t,x,m0
SADD  t,x,m4       2ADDU nu2,nu2,t    SADD  t,x,m1       2ADDU nu2,nu2,t  ▮
2ADDU nu2,nu2,t    SADD  t,x,m2       2ADDU nu2,nu2,t
```

▸**Page 158** line 4 of answer 53 ———— 12 Dec 2009

$(u_{d-1}v_{d-1}) \rightsquigarrow (u_{d-t}v_{d-t})$

Page 159 line 3 of answer 58(d) ——— 05 May 2009

$\phi \leadsto \varphi$ (twice)

then τ.] $\leadsto$ *then* τ. The Sylow group T includes many interesting and important permutations, including bit reversal and cyclic shifts. It corresponds to settings of the Omega network where crossbars of length 2^j that are congruent mod 2^{j+1} all switch or all pass, as a unit.]

Page 160 new paragraph for end of answer 58 ——— 07 May 2009

(f) Any setting of the crossbars corresponds to a permutation that makes Batcher's comparator modules do the equivalent switching.

Page 161 replacement for lines 6–∞ of answer 70 ——— 01 Jul 2009

The computation ends with $k = \lambda\nu\bar{\chi}+1$; the remaining masks $\theta_k, \ldots, \theta_{d-1}$, if any, are zero and those steps can be omitted from (80). "Minimal" masks, for which $*^{2^{k+1}} = 0^{2^{k+1}}$ in answer 69, are obtained if the operations '$\theta_k \leftarrow x$, $\psi \leftarrow (\psi \mathbin{\&} \bar{x}) \gg 2^k$' are replaced by '$\psi \leftarrow (\psi \mathbin{\&} \bar{x}) \gg 2^k$, $\theta_k \leftarrow x \mathbin{\&} (x+\psi)$' in the loop above.

[See *compress* in H. S. Warren, Jr., *Hacker's Delight* (Addison–Wesley, 2002), §7–4; also G. L. Steele Jr., *U.S. Patent 6715066* (30 March 2004). The BESM-6 computer, designed in 1965, implemented *compress* under the name «сборка» ("gather" or "pack"). Its «разборка» command ("scatter" or "unpack") went the other way.]

Page 161 replacement for answers 72 and 73 ——— 07 May 2009

72. Assume that the leftmost mask bit, χ_{N-1}, is zero, since it is immaterial. Then the result $(z_{(N-1)\varphi} \ldots z_{1\varphi} z_{0\varphi})_2$ of any gather-flip corresponds to a permutation with $0\varphi < \cdots < k\varphi > \cdots > (N-1)\varphi$, where $k = \nu\chi$. For example, if $N = 8$ and $\chi = (00101100)_2$, the result is $(z_0 z_1 z_4 z_6 z_7 z_5 z_3 z_2)_2$. So φ is a cyclic shift of a bitonic permutation, and $\varphi \in \Omega$ by exercises 58(d) and 58(f).

Moreover, the masks $\theta_0, \theta_1, \ldots, \theta_{d-1}$ for the 1-swap, 2-swap, …, 2^{d-1}-swap can be computed as follows: The permutation $\psi = \varphi^-$ satisfies $j\psi = (N-1-j)\bar{\chi}_j + s_j$, where $s_j = \chi_{j-1} + \cdots + \chi_1 + \chi_0$ counts the 1s following mask bit χ_j. Let $\psi_0 = \psi$ and $\theta_k = (\lfloor \psi_k/2^k \rfloor \bmod 2) \mathbin{\&} \mu_k$, where ψ_{k+1} is the 2^k-swap of ψ_k with mask θ_k. (In our example, $s_7 \ldots s_1 s_0 = 33221000$ and $(0\bar{\chi}_7)\ldots(6\bar{\chi}_1)(7\bar{\chi}_0) = 01030067$; hence $\psi_0 = (7\psi)\ldots(1\psi)(0\psi) = 33221000 + 01030067 = 34251067$. Then $\theta_0 = (10011001)_2 \mathbin{\&} \mu_0 = (00010001)_2$; $\psi_1 = 34521076$; $\theta_1 = (10010010)_2 \mathbin{\&} \mu_1 = (00010011)_2$; $\psi_2 = 32547610$; $\theta_2 = (00111100)_2 \mathbin{\&} \mu_2 = (00001100)_2$. In general $j\psi_k \equiv j$ (modulo 2^k).) Represent each permutation ψ_k as a set of d bit vectors, namely as the "bit slices" $\psi_k \bmod 2$, $\lfloor \psi_k/2 \rfloor \bmod 2$, etc. Then $O(d^2)$ bitwise operations suffice for this computation.

The *scatter-flip* operation, which undoes the effect of gather-flip, is obtained via the same crossbar network but from right to left (first a 2^{d-1}-swap, ending with a 1-swap).

[See *Journal of Signal Processing Systems* **53** (2008), 145–169.]

73. (a) Equivalently, d sheep-and-goats operations must be able to transform the word $x^\pi = (x_{(2^d-1)\pi} \ldots x_{1\pi} x_{0\pi})_2$ into $(x_{2^d-1} \ldots x_1 x_0)_2$, for any permutation π of $\{0, 1, \ldots, 2^d-1\}$. And this can be done by radix-2 sorting (Algorithm 5.2.5R): First bring the odd numbered bits to the left, then bring the bits j for odd $\lfloor j/2 \rfloor$ left, and so on. For example, when $d = 3$ and $x^\pi = (x_3 x_1 x_0 x_7 x_5 x_2 x_6 x_4)_2$, the three operations yield successively $(x_3 x_1 x_7 x_5 x_0 x_2 x_6 x_4)_2$, $(x_3 x_7 x_2 x_6 x_1 x_5 x_0 x_4)_2$, $(x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0)_2$. [See Z. Shi and R. Lee, *Proc. IEEE Conf. ASAP'00* (IEEE CS Press, 2000), 138–148.]

(b) With gather-flip, the same strategy always yields $(x_{g(2^d-1)}\ldots x_{g(1)}x_{g(0)})_2$, where $g(k)$ is Gray binary code, 7.2.1.1–(9). For instance, the example of (a) is now $(x_5x_7x_1x_3x_0x_2x_6x_4)_2$, $(x_6x_2x_3x_7x_5x_1x_0x_4)_2$, $(x_4x_5x_7x_6x_2x_3x_1x_0)_2$.

[I've actually put this material on page 243, temporarily, since it doesn't fit on page 161.]

Page 162 line 10 of answer 75 —— 16 May 2009

is odd. $\rightsquigarrow$ is odd. For example, if $(i_0,\ldots,i_5) = (j_0,\ldots,j_5) = (0,1,3,5,7,8)$, the subproblems have $i = j = (0,1,3,4)$ and $(0,2,4)$; $x_7\ldots x_0 \mapsto x_6x_7x_5x_4x_2x_3x_1x_0 \mapsto \cdots \mapsto x_5x_7x_5x_3x_1x_3x_1x_0 \mapsto x_7x_5x_5x_3x_1x_1x_0$.

Page 164 last lines of answer 91 —— 16 May 2009

`1 #0101010101010101` $\rightsquigarrow$
after which ... (67v) $\rightsquigarrow$ but omit the final `SLU`, then repeat the first four instructions again. The total time for eight α-blends (66v)

▸**Page 165** last line of answer 101 —— 12 Dec 2009

CACM **6** $\rightsquigarrow$ *CACM* **18**

▸**Page 168** line 5 of answer 124 —— 03 Jul 2010

$V_0 = U_t \setminus V_1 \rightsquigarrow V_0 = U_{t-1} \setminus V_1$

▸**Page 169** in answer 128 —— 30 Dec 2009

36 and 117 $\rightsquigarrow$ 36 and 66

▸**Page 176** line 4 of answer 167 —— 02 Jan 2010

$f_1 \rightsquigarrow S_1$

▸**Page 177** replacement for first lines of answer 173 —— 16 May 2009

173. (a) The hint is readily verified. Notice that if X and Y are closed, $X\,\&\,Y$ is closed; if X and Y are open, $X \mid Y$ is open. Thus X^D is closed and X^L is open; $X^{DD} = X^D$

Page 178 lines 13 and 14 of answer 179 —— 29 Sep 2010

tree of components ... all children $\rightsquigarrow$ tree of current components, where each node has several fields: `CHILD`, `SIB`, and `PARENT` (tree links); `DORMANT` (a circular list, via `SIB` links, of all former children

▸**Page 178** line −3 —— 30 Jun 2009

$\mathtt{R} = \mathtt{R}'$is $\rightsquigarrow$ $\mathtt{R} = \mathtt{R}'$ is

▸**Page 181** line −3 of answer 187 —— 24 Dec 2009

9, 3 $\rightsquigarrow$ **19**, 3

▸**Page 184** line 1 —— 11 Apr 2009

Meerö $\rightsquigarrow$ Mérő

Page 186 line 4 of answer 216 —— 06 Apr 2010

Double precision $\rightsquigarrow$ Double-precision

▸**Page 186** bottom line —— 24 Dec 2009

12 (2008) $\rightsquigarrow$ **13** (2008)

▸**Page 186** new answer (which will actually be on page 244) — 09 Mar 2010

218. Let b be any integer with $b \bmod 8 = 5$. Then $x = b^{L(x)} \bmod 2^d$ for some integer $L(x)$, depending on b, whenever $0 < x < 2^d$ and $x \bmod 4 = 1$ (see Section 3.2.1.2). The following algorithm computes $s = 4L(x)$, given a table of the numbers $t_k = -4L(2^k+1)$ for $1 < k < d$: Set $s \leftarrow 0$, $j \leftarrow 1$; then while $x \neq 1$, set $j \leftarrow j+1$, and if $x \mathbin{\&} (1 \ll j) \neq 0$ also set $x \leftarrow (x + (x \ll j)) \bmod 2^d$, $s \leftarrow (s + t_j) \bmod 2^d$.

Now to compute $a \cdot x^y$ we can proceed as follows (with all arithmetic done mod 2^d): If $x \mathbin{\&} 2 \neq 0$, set $x \leftarrow -x$ and $a \leftarrow (-1)^{y\&1}a$. (Now $x \bmod 4 = 1$.) Set $s \leftarrow 4L(x) \cdot y$, using the algorithm above, and $j \leftarrow 1$; then while $s \neq 0$, set $j \leftarrow j+1$, and if $s \mathbin{\&} (1 \ll j) \neq 0$ also set $s \leftarrow s + t_j$, $a \leftarrow a + (a \ll j)$. The desired answer is then a.

Suitable numbers t_k can be computed by setting $t_{d-1} \leftarrow 1 \ll (d-1)$ and proceeding as follows for $k = d-2, d-3, \ldots, 2$: Set $x \leftarrow 1 + (1 \ll k)$, $x \leftarrow x + (x \ll k)$, $s \leftarrow 0$, $j \leftarrow k$; then while $x \neq 1$ set $j \leftarrow j+1$, and if $x \mathbin{\&} (1 \ll j) \neq 0$ also set $x \leftarrow x + (x \ll j)$, $s \leftarrow s - t_j$; finally $t_k \leftarrow s \gg 1$. For example, when $d = 32$ we get $t_k = 2^k$ for $32 > k \geq 16$, $t_{15} =$ `#20008000`, $t_{14} =$ `#18004000`, $t_{13} =$ `#0e002000`, $t_{12} =$ `#07801000`, $t_{11} =$ `#03e00800`, $t_{10} =$ `#41f80400`, $t_9 =$ `#18fe0200`, $t_8 =$ `#0b7f8100`, $t_7 =$ `#319fe080`, $t_6 =$ `#5e8bf840`, $t_5 =$ `#4a617e20`, $t_4 =$ `#17c26f90`, $t_3 =$ `#6119d1e8`, $t_2 =$ `#2c30267c`. (This procedure finds the L's for *some* integer b, without revealing the actual value of b itself!)

[The methods of this exercise have interesting connections to the algorithms of Briggs and Feynman for *real*-valued logarithm and exponential in exercises 1.2.2–25 and 1.2.2–28. Our broadword procedure for x^y works also for calculating the inverse of x, modulo 2^d, when $y = -1$; but there's a direct algorithm available for that: Set $z \leftarrow 1$, $j \leftarrow 0$; while $x \neq 1$ set $j \leftarrow j+1$, and if $x \mathbin{\&} (1 \ll j) \neq 0$, also set $z \leftarrow (z + (z \ll j)) \bmod 2^d$, $x \leftarrow (x + (x \ll j)) \bmod 2^d$. The final z is the inverse of the original odd number x.]

▸**Page 188** line 1 of answer 11 — 14 Feb 2010

all paths $\rightsquigarrow$ the number of settings of $x_{v_k} \ldots x_n$ that lead

Page 192 lines 3 and −2 of answer 44 — 13 Sep 2010

$S_n \rightsquigarrow \Sigma_n$

▸**Page 197** line −5 — 14 Oct 2009

the properly $\rightsquigarrow$ the property

Page 200 line 1 of answer 87 — 04 Oct 2010

Sort $\rightsquigarrow$ Sort the given pointer values f, g, h

▸**Page 204** in answer 115 — 04 Apr 2010

lines 4 and 5: Every such branch corresponds to some subtable of order $n-k$. $\rightsquigarrow$ Every subtable of order $n-k$ corresponds to some such branch.
last line: 'zead' in these arguments. $\rightsquigarrow$ 'zead', and 'q_k' to 'q'_k'.

▸**Page 205** line 6 — 24 Jan 2010

Conf. Design Automation $\rightsquigarrow$ *Design Automation Conf.*

▸**Page 205** step H2 in answer 124 — 20 Apr 2010

Set $v =$ $\rightsquigarrow$ Set $v \leftarrow$

▸**Page 212** lines 9–10 — 24 Jan 2010

Conf. Design Automation $\rightsquigarrow$ *Design Automation Conf.*

Page 212 line 12 of answer 142 — 24 Jan 2010

$i \in I, j \in J \rightsquigarrow i \in I,\ \ j \in J$

▸**Page 213** line −3 — 30 Jan 2010

$1,\ 2,\ldots,\ n \rightsquigarrow 1,\ 2,\ \ldots,\ n$

Page 214 in answer 152 — 29 Sep 2010

are made. The ... 5 2 1 $\rightsquigarrow$ are made: 1,342,191,700 nodes after one sift reduce eventually to 208,478,228 after 139,245 total swaps. Moreover, Filip Stappers used sifting together with random swapping in September 2010 to get the value of $B(h_{100}^{\pi})$ down to only 198,961,868, with the following "current champion" permutation π:

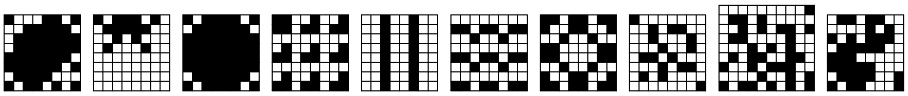
3	4	6	8	10	12	14	16	18	20	22	24	27	28	30	32	35	37	39	41
43	45	47	49	51	53	54	83	85	98	99	100	79	77	81	75	73	95	71	97
69	96	57	91	67	59	65	60	63	62	64	61	66	87	58	68	56	94	93	70
92	72	90	74	76	78	80	89	88	86	84	82	55	52	50	48	46	44	42	40
38	36	34	33	31	29	26	25	23	21	19	17	15	13	11	9	7	5	1	2

▸**Page 215** line 1 — 10 Sep 2009

variables $h_{100} \rightsquigarrow$ variables of h_{100}

Page 216 replacement for last two lines of answer 160 — 18 Nov 2010

has no 8×8 predecessor; but it does, for example, have the 9×10 in (A8):



(A0) (A1) (A2) (A3) (A4) (A5) (A6) (A7) (A8) (A9)

Page 218 reference for line −4 of answer 168 — 06 Apr 2010

[See G. Bennett, *AMM* **117** (2010), 334–351.]

Page 220 line 5 — 29 Nov 2010

$q_k^{-1},\ q_k^{-2} \rightsquigarrow q_k^{-1},\ q_{q_k^{-1}}^{-1}$

Page 222 line 3 of answer 176(c) — 30 Jan 2010

$\bmod\ (2^{l-1}) \rightsquigarrow \bmod\ 2^{l-1}$ (twice)

▸**Page 222** line −2 of answer 176 — 16 Jun 2010

$B_{\min}(Z_{n,y}) \rightsquigarrow B_{\min}(Z_{n,a})$

Page 222 new sentence for end of answer 177 — 11 Nov 2010

[M. Sauerhoff, in *Discrete Applied Math.* **158** (2010), 1195–1204, has proved the lower bound $\Omega(2^{6n/5})$ for this ordering.]

▸**Page 224** line 3 of answer 191 — 11 Feb 2010

(1, 2, 6, 1806, 3263442, ...) $\rightsquigarrow$ (1, 2, 6, 42, 1806, 3263442, 10650056950806, ...)

Page 229 last lines of answer 215 — 10 Jun 2009

413–420.] ... $(1 - z^{m-1} - z^{m+1})$. $\rightsquigarrow$ 413–420. The set of all tatami tilings has been characterized by Dean Hickerson; the corresponding generating functions have been obtained by Frank Ruskey and Jennifer Woodcock, *Electronic J. Combinatorics* **16**, 1 (2009), #R126.]

▸**Page 230** last three lines of answer 217 ______ 20 Nov 2010

There are 42,159,777,732 ... this problem.) $\rightsquigarrow$ There are 554,626,216 strongly separated coverings, findable after 245 gigamems with a ZDD of size 4,785,236. (But standard backtracking finds them faster and with neglible memory.)

Page 236 line 1 of answer 240 ______ 01 Dec 2009

of (67) $\rightsquigarrow$ of (68)

▸**Page 236** line 7 of answer 240 ______ 07 Feb 2010

$\bigvee_v (e_v \sqcup \bigsqcup_{u \mathrel{-\!\!-} v} e_u) \rightsquigarrow \bigcup_v (e_v \sqcup \bigsqcup_{u \mathrel{-\!\!-} v} e_u)$

▸**Page 237** line 1 of answer 241 ______ 05 Mar 2009

Let g the $\rightsquigarrow$ Let g be the

▸**Page 238** line −3 of answer 241 ______ 05 Mar 2009

295–200 $\rightsquigarrow$ 295–300

▸**Page 238** line −6 of answer 242 ______ 17 May 2009

including 51 $\rightsquigarrow$ including 57

▸**Page 240** line −1 of answer 255 ______ 07 Feb 2010

3–10 $\rightsquigarrow$ 4–11

Page 241 line 7 of answer 260 ______ 10 Jun 2009

for $n \geq 5 \rightsquigarrow$ for $n \geq 2$

▸**Page 241** line 12 of answer 260 ______ 10 Jun 2009

$n = 10 \rightsquigarrow n = 100$

▸**Page 243** line 6 ______ 24 Dec 2009

2042–2047 $\rightsquigarrow$ 2042–2048

Page 243 new answer ______ 01 Oct 2009

265. Compute counts c_j bottom-up as in Algorithm C, using n-bit arithmetic. Then proceed top-down, by starting with $k \leftarrow s-1$, $j \leftarrow 1$, $m \leftarrow m-1$, and repeating the following steps (during which we will have $0 \leq m < 2^{v_k - j} c_k$): If $v_k > j$, set $x_j \leftarrow \lfloor m/2^{v_k-j-1} c_k \rfloor$, $m \leftarrow m \bmod 2^{v_k-j-1} c_k$, $j \leftarrow j+1$; otherwise if $k = 1$, terminate; otherwise set $l \leftarrow l_k$, $h \leftarrow h_k$, and if $m < 2^{v_l - v_k - 1} c_l$ set $x_j \leftarrow 0$, $k \leftarrow l$, $j \leftarrow j+1$; otherwise set $x_j \leftarrow 1$, $m \leftarrow m - 2^{v_l - v_k - 1} c_l$, $k \leftarrow h$, $j \leftarrow j+1$.

Page 244 and following ______ 03 Jan 2009

Miscellaneous changes to the existing index of Volume 4 Fascicle 1 are collected here, including corrections and amendments to the old entries as well as new entries that are occasioned by the new material. Thus, the lines of the full index that have changed serve also as an index to the present document. However, when a correction or amendment has caused an old index entry to be deleted, the deletion is usually not indicated.