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 9B

A POTPOURRI OF PUZZLES

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY

October 31, 2020

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 -87), 31 Oct 2020

BARRY
Internet

PREFACE

But that is not my point.
I have no point.
— DAVE BARRY (2002)

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 9 previews Section 7.2.2.8 of *The Art of Computer Programming*, entitled "A potpourri of puzzles." It discusses how to apply and extend the techniques of previous sections to a wide variety of classic combinatorial problems that have a recreational flavor.

At present this collection doesn't yet qualify for the nice, fragrant term "potpourri"; it's more of a hodgepodge, mishmash, conglomeration, mélange, pastiche, etc. I'm basically gathering items one by one, as I write other sections, and sticking preliminary writeups into this container. Some day, however, I hope that I'll no longer have to apologize for what is now just a bunch of sketches.

* * *

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 40, 89, . . . ; I've also implicitly mentioned or posed additional unsolved questions in the answers to exercises . . . . 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 38, 39, 42, 51, 57, 65, ... Furthermore I've credited exercises . . . to unpublished work of . . . . Have any of those results ever appeared in print, to your knowledge?

(In particular, I would be surprised if the "tagging algorithm" in answer 39 has not previously been published, although I don't think I've seen it before.)

Kao
Knuth
HAUPTMAN

* * *

Special thanks are due to Regan Murphy Kao for help with Japanese translations, and 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
99 Umbruary 2016

D. E. K.

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

PDI: A perfect digital invariant
Perfect digital invariants-
Dudeney
powers of the digits
radix-b numbers
Rumney
perfect digital invariants
narcissistic numbers
coincidences
Myers
multiset
multicombination
combination
sorting

7.2.2.8. A potpourri of puzzles. Blah blah de blah blah blah. We'll discuss some of the most interesting time-wasters that have captured the attention of computer programmers over the years ... The "obvious" ways to solve them can often be greatly improved by using what we've learned in previous sections ...

* * *

Who knows what I might eventually say here?

* * *

Perfect digital invariants. In 1923, the great puzzlist Henry E. Dudeney observed that

$$370 \;=\; 3^3+7^3+0^3 \qquad \text{and} \qquad 407 \;=\; 4^3+0^3+7^3,$$

and asked his readers to find a similar example that doesn't have a zero in its decimal representation. A month later he gave the solution, $153 = 1^3+5^3+3^3$ [*Strand* **65** (1923), 103, 208]—curiously saying nothing about the obvious answer $371 = 3^3+7^3+1^3$. These examples were rediscovered independently by several other people, and eventually extended to mth powers of the digits for $m>3$, and to radix-b numbers for $b \neq 10$. Max Rumney [*Recreational Math. Magazine* #12 (December 1962), 6–8] mentioned $8208 = 8^4+2^4+0^4+8^4$, $(491)_{13} = 794 = 4^3+9^3+1^3$, ..., and named such numbers *perfect digital invariants* of order m.

Let $\pi_m x$ be the sum of the mth powers of the decimal digits of x. With this notation, the number x is a perfect digital invariant of order m in radix 10 if and only if $\pi_m x = x$. In particular, every order $m>0$ has at least two perfect digital invariants, since the numbers 0 and 1 always qualify. And it turns out that most orders have *more* than two (see exercise 34), because of more-or-less random coincidences. For example, when $m=100$ there's a unique third solution,

$$\begin{aligned} x = {}&26\,56162\,29619\,33010\,98036\,76416\,71003\,29792\,07874\,84348\,54147\\ &71766\,93876\,28693\,32047\,88451\,13744\,80147\,98509\,42958 = \pi_{100}x, \end{aligned} \tag{20}$$

discovered in 2009 by Joseph Myers.

How can such a humongous number be found in a reasonable time? In the first place, we can always write $x = (x_m \ldots x_1 x_0)_{10}$, because exercise 30 shows that every mth-order solution has at most $m+1$ digits. In the second place, we can see that π_m depends only on the multiset $M_m(x) = \{x_m, \ldots, x_1, x_0\}$ of x's digits, not on the actual order of those digits. All we have to do, therefore, is look at each multiset, and see if $M_m(x_m^m + \cdots + x_1^m + x_0^m) = \{x_m, \ldots, x_1, x_0\}$.

A multiset of $m+1$ digits is what Section 7.2.1.3 calls a "multicombination," also known as a *combination of the ten objects* $\{0,1,\ldots,9\}$ *taken* $m+1$ *at a time with repetitions allowed*. If we renumber the subscripts by sorting the digits into order, such a multicombination is nothing more nor less than a solution to

$$9 \geq x_m \geq \cdots \geq x_1 \geq x_0 \geq 0. \tag{21}$$

backtrack algorithm++

Algorithm 7.2.1.3T, together with the correspondence rule 7.2.1.3–(7), is an efficient way to generate them all. Notice that the number of multicombinations is polynomial, only $\binom{m+10}{9}$, while the number of $(m+1)$-digit numbers is 10^{m+1}. For example, when $m = 3$ there are just $\binom{13}{9} = 715$ cases to try. One of them is $\{7, 4, 0, 0\}$; and $7^3+4^3+0^3+0^3 = 407$ happens to have the same multiset of digits.

With these ideas we could find (20) and prove its uniqueness by considering "only" $\binom{110}{9} = 4{,}643{,}330{,}358{,}810$ multicombinations. But that's still a big number, and we can do much better. In fact, there's a nice backtrack algorithm that solves the case $m = 100$ with fewer than 100 gigamems of computation.

Here's how: We generate solutions to (21) by first choosing x_m, then x_{m-1}, then $x_{m-2}, \ldots,$ and we try to rule out bad cases as early as possible. For example, suppose we've tentatively set $x_{100} = \cdots = x_{92} = 9$, and we've already considered all cases with $x_{91} = 9$ or 8; so we've already found (20). Our next task is to decide whether $x_{91} \le 7$ will be viable. Let $\Sigma_l = x_m^m + \cdots + x_{l+1}^m$; this is π_m applied to the digits already chosen. The final power sum will then be at least $a_{91} \leftarrow \Sigma_{91}$, which in this case is $9 \cdot 9^{100} \approx .00002\,39052\,58998\,82873 \times 10^{101}$. And it will be at most $b_{91} \leftarrow a_{91} + 92 \cdot 7^{100}$, which is $\approx .00002\,39052\,59001\,80445 \times 10^{101}$. Wow: These lower and upper bounds have a nice long common prefix. So we know that our multicombination not only contains nine 9s and no 8s, it also must include several known digits that are less than 8, namely $\{0, 0, 0, 0, 2, 3, 0, 5, 2, 5\}$.

We can now refine the lower bound to $a_{91} \leftarrow \Sigma_{91} + 2 \cdot 2^{100} + 3^{100} + 2 \cdot 5^{100} \approx .00002\,39052\,58998\,82873 \times 10^{101}$. And the upper bound can also be improved, namely to $b_{91} \leftarrow a_{91} + 82 \cdot 7^{100} \approx .00002\,39052\,59001\,48100 \times 10^{101}$, because $r_{91} \leftarrow 82$ of the digits are still uncertain. Moving on, if we tentatively set $x_{91} = 7$ we'll find $\Sigma_{90} = \Sigma_{91} + 7^{100}$; and a_{90} will be $a_{91} + 7^{100} \approx .00002\,39052\,58998\,86107\,40 \times 10^{101}$, while $b_{90} \leftarrow b_{91}$ and $r_{90} \leftarrow 81$. It appears that $x_{90} = 7$ is also worth a try.

But after exploring all cases with $x_{90} = 7$ we will eventually want to know if $x_{90} \le 6$ is viable. Then b_{90} will be $a_{90} + 81 \cdot 6^{100} \approx .00002\,39052\,58998\,86107\,45 \times 10^{101}$, and the common prefix will include the impossible digit 8. (In fact, it will include *three* 8s.) So we must backtrack. And if we now test the viability of $x_{91} \le 6$, we find $b_{91} \leftarrow a_{91} + 82 \cdot 6^{100} \approx .00002\,39052\,58998\,82873 \times 10^{101}$; again there's a forbidden 8. We've proved that *a multicombination for* $m = 100$ *that begins with nine* 9*s must be followed by* 9, 8, *or* 7; and if $x_{91} = 7$, then x_{90} must also be 7.

***Fleshing out that "perfect" algorithm.** The algorithm just sketched will explore only 624,434,412 multicombinations when $m = 100$, and its details are instructive. So let's look closer. The notation will be more like our normal conventions if we renumber the subscripts so that (21) becomes

$$9 \ge x_1 \ge \cdots \ge x_m \ge x_{m+1} \ge 0. \tag{22}$$

Now we'll be choosing x_1 first, then x_2, etc., instead of working backward.

The description above was simplified, for expository purposes, but the real algorithm is essentially the same. At each level of the search, beginning with level $l = 1$, we'll try to determine if $x_l \le c$ is feasible, where c is some threshold; initially $c = x_{l-1}$. As above, the search will be guided by $(m + 1)$-digit integers

data structures+
monus
registers

$\Sigma_l = (\Sigma_{lm} \ldots \Sigma_{l0})_{10}$, $a_l = (a_{lm} \ldots a_{l0})_{10}$, and $b_l = (b_{lm} \ldots b_{l0})_{10}$, where a_l and b_l are bounds on any perfect digital invariant whose largest digits include $\{x_1, \ldots, x_{l-1}\}$. We also maintain ten counters $d_l = d_{l9} \ldots d_{l0}$, where d_{lk} is the number of times the digit k appears in $\{x_1, \ldots, x_{l-1}\}$.

There's an index t such that $a_{lm} \ldots a_{l(t+1)} = b_{lm} \ldots b_{l(t+1)}$ is a common prefix of a_l and b_l; here $-1 \le t \le m$. There are ten further counters $e_l = e_{l9} \ldots e_{l0}$ analogous to d_l, where e_{lk} is the number of times k occurs in $\{a_{lm}, \ldots, a_{l(t+1)}\}$. If $k > c$ we must always have $e_{lk} \le d_{lk}$, because all digits greater than c have already been specified. If $k < c$ we always have $d_{lk} = 0$. And if $k = c$ there's no restriction on d_{lk} or e_{lk}; we write $q = q_l = e_{lc} \dot{-} d_{lc} = \max(0, e_{lc} - d_{lc})$.

The number of "unknown" digits is denoted by $r = r_l$, where $m + 1 - r = \sum_{k=0}^{9} \max(d_{lk}, e_{lk})$ is the number of "known" digits. Finally, we write

$$\Sigma_l = \sum_{k=0}^{9} \max(d_{lk}, e_{lk}) \cdot k^m \qquad (23)$$

for the sum of the mth powers of the known digits. (Notice that this differs from the quantity called Σ_l in our former discussion, where e_l wasn't considered.)

Let's use a and b as a convenient shorthand for the digits a_{lt} and b_{lt} that lie just to the right of the current common prefix, assuming that $t \ge 0$. Thus $a \le b$ at all times; and if $a = b$, the current prefix can be lengthened. If $b < c$, one of the unknown digits must lie in the interval $[a\,..\,b]$, hence we must have $r > 0$.

In the algorithm below, unsubscripted variables like a, b, c, q, r, t are regarded as being in a computer's registers. Instructions like $q_l \leftarrow q$ or $q \leftarrow q_l$ mean that register q is to be stored into memory or fetched from memory, respectively.

Algorithm P (*Perfect digital invariants*). Given $m \ge 3$, this algorithm generates all $(m+1)$-digit integers x such that $\pi_m x = x$, by finding all of the appropriate multicombinations $x_1 \ldots x_{m+1}$ that satisfy (22). Its state variables for $1 \le l \le m+2$ are q_l, r_l, t_l, and the $(m+1)$-digit numbers Σ_l, a_l, b_l, as well as the digit counts d_l and e_l. The $(m+1)$-digit constants $j \cdot k^m$ should also be precomputed for $0 \le j \le m+1$ and $0 \le k < 10$.

P1. [Initialize.] Set $l \leftarrow 1$, $q \leftarrow 0$, $r \leftarrow m+1$, $t \leftarrow m$, $a \leftarrow 0$, $b \leftarrow c \leftarrow 9$, $d_1 \leftarrow e_1 \leftarrow \Sigma_1 \leftarrow 0$, and go to P4.

P2. [Enter level l.] (We've just set $x_{l-1} \leftarrow c$.) Set $d_{lk} \leftarrow d_{(l-1)k} + [k=c]$ and $e_{lk} \leftarrow e_{(l-1)k}$ for $0 \le k < 10$. If $q > 0$, set $\Sigma_l \leftarrow \Sigma_{l-1}$ and $q \leftarrow q - 1$; then go immediately to P5 if q is still positive. Otherwise if $r > 0$, set $\Sigma_l \leftarrow \Sigma_{l-1} + c^m$ and $r \leftarrow r - 1$. Otherwise go to P7.

P3. [Done?] If $l = m+2$, visit the solution Σ_l and go to P7.

P4. [Test feasibility of c.] If there's an easy way to prove that x_l can't be $\le c$, using the current state variables as discussed in exercise 33, go to P7. (This test might update all of the state variables except d_l.)

P5. [Try c.] Set $x_l \leftarrow c$, $q_l \leftarrow q$, $r_l \leftarrow r$, $t_l \leftarrow t$, $l \leftarrow l+1$, and go to P2.

P6. [Try again.] If $c > 0$, set $c \leftarrow c - 1$, $q \leftarrow e_{lc}$, and go to P4. (Now $d_{lc} = 0$.)

P7. [Backtrack.] Terminate if $l = 1$. Otherwise set $l \leftarrow l - 1$, $q \leftarrow q_l$, and repeat this step if $q > 0$. Otherwise set $r \leftarrow r_l$, $t \leftarrow t_l$, $a \leftarrow (t \geq 0?\ a_{lt}\colon 9)$, $b \leftarrow (t \geq 0?\ b_{lt}\colon 9)$, $c \leftarrow x_l$, and go back to P6. ▮

Exercise 33 deals with the most subtle aspects of this algorithm, but two of its simpler features are especially worthy of note. First is the fact that we try decreasing values $x_{l-1}, x_{l-1} - 1, \ldots$ for x_l until finding an infeasible c; then we can immediately backtrack, because our bounds are valid for all $x_l \leq c$. Second is the fact that x_l is given the *forced* value c when $q > 0$. This case arises when c appeared previously in the prefix: Another digit c was supposed to be chosen eventually, and that moment has finally arrived. That's why step P7 repeats itself when encountering $q_l > 0$, and why step P2 goes directly to P5 when $q_{l-1} > 1$.

The running time of Algorithm P is negligible when $m \leq 100$, but it appears to grow roughly as $m^{7.5}$ when m increases.

Historical notes: G. H. Hardy, in *A Mathematician's Apology* (1950), §15, dismissed questions like this as "tiresome"; see D. E. Knuth, *Selected Papers on Computer Science* (1996), 174–175. L. E. Deimel, Jr., and M. T. Jones described their adventures with the computation of perfect digital invariants in *Journal of Recreational Mathematics* **14** (1981–1982), 87–108, 284.

Skeleton multiplication puzzles. Astonishing digital coincidences arise also in quite a different way. Suppose we multiply two numbers by the classical pen-and-paper method, then cover up some of the digits. The hidden quantities can sometimes be reconstructed by knowing only their locations in the remaining "skeleton." For example, consider

$$\begin{array}{r} \Box\Box\Box \\ \times\ \Box\Box\Box\Box \\ \hline \Box\Box\Box\,7 \\ \Box\Box\Box\Box \\ \Box\Box\Box\Box \\ \Box\Box\Box \\ \hline 7\,7\,7\,7\,7\,7 \end{array} \qquad \Box \neq 7. \tag{30}$$

The leftmost digit of each number in the skeleton must be nonzero. Exactly seven 7s appear in the calculation, and all of the other digits have been obscured; yet it's easy to figure out what they must have been (see exercise 50).

Hidden-digit puzzles have a long history, going back at least to eighteenth-century Japan, where puzzles by Yoshisuke Matsunaga were published in his friend Genjun Nakane's book *Kantō Sampō* (1738). They were independently introduced to English-speaking readers by W. P. Workman in *The Tutorial Arithmetic* (London: University Tutorial Press, 1902), Chapter VI, problems 31–34. (See exercises 48 and 49.)

Such puzzles have become especially popular in Japan, where they are called "bug-eaten arithmetic" (*mushikuizan*), and where a special newsletter devoted to their creation and refinement was founded in 1976 by M. Maruo

Hardy
Knuth
Deimel
Jones
Skeleton multiplication
Hidden-digit puzzles
Matsunaga
Nakane
Kantō Sampō
Workman
mushikuizan
Maruo

and Y. Yamamoto. Many classic skeleton puzzles are based on underlying *long division* problems, as in exercise 66; but we shall focus our attention on skeleton *multiplications*, similar to (30).

Yamamoto
long division
Take
Kahan
alphabet
puzzles, invention of

Junya Take introduced a particularly appealing class of bug-eaten multiplications in the *Journal of Recreational Mathematics* **35** (2006), 63, when he submitted the following puzzle in honor of the editor Steven Kahan:

$$\begin{array}{cccccccccccc}
 & & & & & \square&\square&\square&\square&\square&\square&\square\\
 & & & & & & \times&\square&\square&\square&\square&\square\\
\hline
 & & & & & \square&\square&\square&\square&\square&\square&\square\\
 & & & \square&\square&\mathrm{K}&\square&\mathrm{K}&\square&\square&\square& \\
 & & \square&\square&\square&\mathrm{K}&\mathrm{K}&\square&\square&\square& & \\
 & \square&\square&\square&\square&\mathrm{K}&\square&\square&\square& & & \\
\square&\square&\square&\square&\square&\mathrm{K}&\mathrm{K}&\square& & & & \\
\hline
\square&\square&\square&\square&\square&\mathrm{K}&\square&\mathrm{K}&\square&\square&\square&\square
\end{array}
\qquad \square \neq \mathrm{K}. \qquad (31)$$

There's a secret digit, K, all of whose appearances are shown; moreover, the K's just happen to appear in the shape of the letter K! Each $\square$ can be replaced by any of the nine digits *other* than the one reserved for K, except that the most significant digit of a number cannot be 0. The solution — don't peek until you're ready to see it — is unique:

$$\begin{array}{cccccccccccc}
 & & & & & 9&1&7&5&1&4&4\\
 & & & & & & \times&7&2&4&6&1\\
\hline
 & & & & & 9&1&7&5&1&4&4\\
 & & & 5&5&0&5&0&8&6&4& \\
 & & 3&6&7&0&0&5&7&6& & \\
 & 1&8&3&5&0&2&8&8& & & \\
6&4&2&2&6&0&0&8& & & & \\
\hline
6&6&4&8&4&0&1&0&9&3&8&4
\end{array}
\qquad \mathrm{K} = 0. \qquad (32)$$

Who would have guessed that 9175144 and 72461 contain such a surprise?

Subsequent issues of that journal contained a series of similar examples, one for each letter of the alphabet, containing mind-boggling products up to 20 digits long. How on earth was it possible for Take to discover such pairs of numbers, whose digits magically and uniquely form specific geometric designs when they are multiplied?

Rather than trying to *solve* such puzzles, we will consider the more general question of how to *invent* them. As a result, we'll not only learn how to produce amazing numerical patterns, we'll also learn a thing or two about programming and about mathematics.

For concreteness, let's look for puzzles like (31) whose solution has $\mathrm{K} = 0$; other values of K can be investigated in a similar way. We seek decimal numbers

$a = (\ldots a_2a_1a_0)_{10}$ and $b = (b_4b_3b_2b_1b_0)_{10}$, where a is a multiplicand of unknown length, whose multiples $c = ab_0 = (\ldots c_2c_1c_0)_{10}$, $d = ab_1 = (\ldots d_2d_1d_0)_{10}$, $e = ab_2 = (\ldots e_2e_1e_0)_{10}$, $f = ab_3 = (\ldots f_2f_1f_0)_{10}$, $g = ab_4 = (\ldots g_2g_1g_0)_{10}$, and $h = ab = (\ldots h_2h_1h_0)_{10}$ have the property that

$$d_5 = d_3 = e_4 = e_3 = f_3 = g_2 = g_1 = h_6 = h_4 = 0 \qquad (33)$$

while all other digits a_j, b_j, ..., h_j are nonzero. Notice that c, d, e, f, g are distinct, because of (33); hence b_0, b_1, b_2, b_3, b_4 are distinct.

Our strategy will be simply to try all possibilities for a_0, a_1, a_2, ..., in turn, working from right to left and backtracking when we get into trouble. For example, after first setting $a_0 \leftarrow 1$, we find that a_1 cannot be 1, since we need to make $g_1 = 0$. So we try $a_1 \leftarrow 2$; that forces b_4 to be 5. But then there are no options for a_2 that will make $g_2 = 0$. We backtrack and try $a_1 \leftarrow 3$; however, nothing really works well until we try $a_1 \leftarrow 5$. Then b_0, b_1, b_2, b_3 must be odd, to make $c_1d_1e_1f_1 \neq 0$. We also need $a_2 \in \{2,7\}$ and $b_4 \in \{4,8\}$, since $g_1 = g_2 = 0$.

The case $a_2a_1a_0 = 251$ quickly runs out of steam, because we need $d_3 = e_3 = f_3 = 0$. That's possible only when $b_1 = b_2 = b_3$; but the b's must be distinct.

Similarly, $a_2a_1a_0 = 751$ goes nowhere: There are three choices for $\{b_1, b_2, b_3\}$ only if $a_3 \leftarrow 6$; and the multiples of 6751, mod 10000, are

$$6751,\ 3502,\ 0253,\ 7004,\ 3755,\ 0506,\ 7257,\ 4008,\ 0759. \qquad (34)$$

Now the only way to avoid spurious 0s in c, d, e, or f is to choose

$$b_0 \in \{1,5,7\}; \quad b_1, b_2, b_3 \in \{3,9\}; \quad b_4 \in \{4,8\}. \qquad (35)$$

But we can't squeeze three distinct elements into $\{3,9\}$, so we must backtrack.

The next plausible settings of $a_3a_2a_1a_0$ are 5112, 6752, 2572, 7572, 5223. Then we get to a more promising case, $a_3a_2a_1a_0 = 5143$, which turns out to be node number 30 in the backtrack tree so far. Its multiples mod 10^4 are

$$5143,\ 0286,\ 5429,\ 0572,\ 5715,\ 0858,\ 6001,\ 1144,\ 6287; \qquad (36)$$

hence $b_0 \in \{1,3,5,8,9\}$, $b_1, b_2, b_3 \in \{2,4,6\}$, and $b_4 \in \{7\}$. This forced value of b_4 tells us that $a_4 \neq 1$. Then if we try $a_4 \leftarrow 2$, the values of $25143k \bmod 10^5$ are

$$25143,\ 50286,\ 75429,\ 00572,\ 25715,\ 50858,\ 76001,\ 01144,\ 26287, \qquad (37)$$

and our choices are rapidly shrinking: Only 16 possible settings of b remain, with $b_0 \in \{1,3,5,9\}$, $b_1, b_3 \in \{2,6\}$, $b_2 = 4$. Thus we might as well try them all, in order to see how they affect the overall product, h. It turns out that only $76423 \cdot 25143$ makes $h_4 = 0$ without also making $h_1 = 0$. Hence we can conclude that $a_4a_3a_2a_1a_0 = 25143$ implies $b_4b_3b_2b_1b_0 = 76423$.

Again our luck fails us, however, because there's no decent option for a_5. The first time we reach a successful setting of a_5 is at node 44, when $a_5a_4 \ldots a_0 = 175144$. And hurrah: The solution (32) is now found immediately, at node 45.

Once we've found that solution, we could try to extend it by setting a_7, a_8, ... in such a way that no new 0s will mess up the desired pattern. Indeed, the number 19175144 does yield a "K of zeros" when multiplied by 72461. But it

sort
data structures

doesn't provide a new puzzle, because 19175144×72461, 19675144×72461, and 14783376×83692 all have the same skeleton. (Remarkably, so does 20324856×72461, this time with $\mathrm{K} = 9$ instead of $\mathrm{K} = 0$!)

We do get a "K of zeros" from $*9175144 \times 72461$ also when $* = 3$, 4, 5, 7, 8, and 9. But only one of these, 39175144×72461, has a unique skeleton; and it doesn't make an especially good puzzle, because it involves more digits than (31). Therefore we are well advised to *stop* trying to extend a, after a solution has been found, and to concentrate on the shortest solutions.

When we limit this method to multiplicands a that have at most 9 digits, we obtain 31 different solutions, while traversing a backtrack tree of 1407 nodes. (The total computation time, about 1.8 megamems, is negligible.) Then we sort the solutions by the shapes of their skeletons; and it turns out that 25 of the solutions cannot be used as puzzles, because their skeletons aren't unique. Three of the remaining six do make rather nice puzzles, namely (i) 9175144×72461 and $\mathrm{K} = 0$, which is (31); (ii) 9783376×83692 and $\mathrm{K} = 0$, whose skeleton is only one digit larger than (31); and the surprising (iii) 324856×72461 and $\mathrm{K} = 9$, whose skeleton is eight digits *shorter* than any of the others. Check it out!

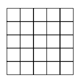
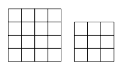
Further possibilities arise when we allow zeros in the multiplier, as discussed in exercise 54. Exercise 56 is devoted to the interesting question of how to design efficient data structures for this computation.

* * *

Discrete dissections
pixels
dissection puzzle
Loyd

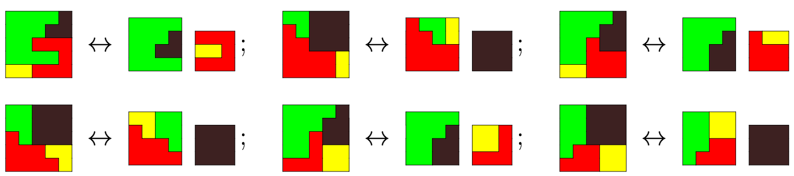
Discrete dissections. It's often convenient to think of the Euclidean plane as an infinite grid of unit squares, also called "cells" or "pixels." A *discrete dissection puzzle* consists of two shapes, A and B, each consisting of N pixels. The problem is to color them with the smallest number of colors, in such a way that A's cells of any given color are congruent to B's cells of that same color.

For example, suppose

$A =$ , $B =$ , $N = 25.$ (50)

Sam Loyd once asked [in the Sunday color section of the *Philadelphia Inquirer*, 14 April 1901] for a way to cut A into four pieces that could be reassembled to make B. He also sought a solution in which none of the pieces had to be rotated.

Loyd's problem turns out to have six essentially different solutions:


(51)

In general, a dissection tends to be particularly nice when its individual pieces have roughly the same size; so we will attempt to minimize the sum of the squares of the sizes. By this criterion the scores of the six solutions are respectively $2^2+3^2+7^2+13^2=231$, $2^2+3^2+9^2+11^2=215$, $2^2+5^2+7^2+11^2=199$, $3^2+5^2+8^2+9^2=179$, $4^2+5^2+5^2+11^2=187$, $4^2+5^2+7^2+9^2=171$, so we prefer the last one. (The others are interesting too, however.)

If we're allowed to rotate the pieces after cutting, there's an even better solution, of score $4^2+6^2+6^2+9^2=169$. And the best conceivable score, $5^2+5^2+6^2+9^2=167$, is attainable if we're also allowed to flip the pieces over:


(52)

What's a good way to solve general problems of this kind by computer? We will assume for convenience that A is always an $n\times n$ square, so that $N=n^2$; the same methods will apply to other shapes, with obvious changes. Formally speaking, when there are d colors, we seek one-to-one correspondences ϕ_k between A's pixels of color k and B's pixels of color k, for $1\le k\le d$.

For example, in (50) we can assume that the pixels of A have coordinates (x,y) for $0\le x,y<n=5$, and that those of B are $[x,y]$ for certain $0\le x<8$, $0\le y<4$. (This particular shape B has no pixels $[x,y]$ with $x=4$, because it's disconnected.) Let τ be the transposition $(x,y)\tau=(y,x)$; let ρ be the rotation $(x,y)\rho=(y,n-1-x)$; and let $\sigma_{a,b}$ stand for shifting by (a,b), so that $(x,y)\sigma_{a,b}=[x+a,y+b]$. Then the transformations in the right-hand solution of (52) are

$$\phi_1=\sigma_{0,0},\quad \phi_2=\sigma_{-1,2},\quad \phi_3=\tau\rho^2\sigma_{1,-2},\quad \phi_4=\sigma_{3,-2}, \qquad (53)$$

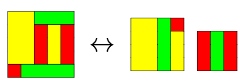
if 1 is the lightest color and 4 is the darkest. For example, pixel $(1,2)$ of A is the tail of a 'P'; it is mapped into $(1,2)\phi_3=(2,3)\sigma_{1,-2}=[3,1]$ within B.

connected
Loyd
bipartite matching

The key point is that relatively few possibilities exist for each ϕ_k; hence we can try them all. Every ϕ_k has the form $\alpha_k\beta_k$, where α_k is one of the eight transformations $\tau^i\rho^j$ that take A into itself, and β_k is a shift. We need only consider shifts $\sigma_{a,b}$ that map at least one pixel of A into a pixel of B. For instance, in problem (50) this occurs for $-4 \le a \le 3$ and $-4 \le b \le 3$, or for $4 \le a \le 7$ and $-4 \le b \le 2$, thus 92 cases altogether, making $8 \times 92 = 736$ possibilities for ϕ_k. We can make a list of all possible shifts, and assign an arbitrary ordering to the elements of that list. Then we can save a factor of about $d!$, by assuming that the shifts we choose satisfy $\beta_1 \le \beta_2 \le \cdots \le \beta_d$. In many problems we can also assume that $\beta_k = \beta_{k+1}$ implies $\alpha_k < \alpha_{k+1}$, because $\phi_k = \phi_{k+1}$ would imply that colors k and $k+1$ could be merged. Furthermore we save another factor of 8, by assuming that α_1 is the identity transformation.

In other words, the problem breaks down into lots of subcases, one for each choice of the mappings $(\phi_1, \ldots, \phi_d)$, but the number of subcases isn't disastrous.

So far we haven't specified that the pixels of a given color must be *connected*, although Loyd's problem referred to "four pieces." Problem (50) can actually be solved with only *three* colors, when connectedness is ignored; for instance thus:

 (54)

Here $\phi_1 = \sigma_{0,-1}$, $\phi_2 = \rho\sigma_{2,0}$, and $\phi_3 = \tau\rho\sigma_{3,-1}$. *We will discuss how to find all of the solutions, connected or not, to a given dissection problem.* Unwanted solutions can be discarded later, by imposing extra conditions such as connectedness.

A closer look at this three-coloring problem reveals that we don't really have to try all $\binom{94}{3} = 134{,}044$ of the ways to choose $\beta_1 \le \beta_2 \le \beta_3$, because most of them fail to cover all the pixels of B. Only 4250 sequences of shifts — about 3% of the total — pass this test, which is independent of the α's.

Consider now a typical sequence of shifts that *does* cover B: $\beta_1 = \sigma_{0,0}$, $\beta_2 = \sigma_{3,0}$, $\beta_3 = \sigma_{2,-1}$. Some cells of B are covered thrice: For example, $[3,2] = (3,2)\beta_1 = (0,2)\beta_2 = (1,3)\beta_3$. Others are covered twice: For example, $[6,1]$ isn't covered by β_1, but it equals $(3,1)\beta_2$ and $(4,2)\beta_3$. Still others, like $[0,0]$ and $[7,1]$, are covered only once.

When β_1, β_2, and β_3 do cover B, we must consider $8^2 = 64$ possibilities for α_2 and α_3. Most of these typically fail to cover A; that is, at least one pixel of A is not the inverse image $[x,y]\phi_k^-$ of any $[x,y] \in B$. For example, when $(\beta_1, \beta_2, \beta_3) = (\sigma_{0,0}, \sigma_{3,0}, \sigma_{2,-1})$, it turns out that only four settings of $(\alpha_1, \alpha_2, \alpha_3)$ pass this test, namely $(1, \rho, 1)$, $(1, \rho^2, \tau)$, $(1, \tau\rho, \tau)$, and $(1, \tau\rho^2, 1)$. In fact, among all $4250 \times 64 = 272{,}000$ candidates for $(\alpha_1\beta_1, \alpha_2\beta_2, \alpha_3\beta_3)$, all but 582 choices are ruled out, at this stage of the search for valid dissections.

Let's take a closeup look at one of those successful cases:

$$\phi_1 = \sigma_{0,0} = 1, \qquad \phi_2 = \tau\rho\sigma_{3,0}, \qquad \phi_3 = \tau\sigma_{2,-1}. \tag{55}$$

We're left with a *bipartite matching* problem on $2N$ vertices, with the pixels of A to be matched to the pixels of B. The "edges" of this matching problem are specified by the mappings ϕ_k. Table 1 summarizes this problem by showing the dancing links

Table 1

THE MATCHING PROBLEM THAT UNDERLIES DISSECTION OF (50) VIA (55)

30	60	50 61	60 62	70
03 31	13 50	23 51 51	33 61 52	71
02 32	12	22 52	32 62	72
01 33	11 30	21 31	31 32	33
00	10 20	20 21	30 22	23

03	13	23 40	33 01 41
02	12	22 30	32 02 31
01	11	21 20	31 03 21
00	10	20 10	30 04 11

22 33	32 34	42
23 23	33 24	43
24 13	34 14	44

possible mates $(x,y)\phi_k$ of A's pixels (x,y) at the left, also showing the possible mates $[x,y]\phi_k^-$ of B's pixels $[x,y]$ at the right. For example, pixel $(2,4)$ of A, which is in the center of the top row, has the entry '50 61'; it means that ϕ_1 does *not* map $(2,4)$ into a pixel of B, but ϕ_2 takes $(2,4)\mapsto[5,0]$ and ϕ_3 takes $(2,4)\mapsto[6,1]$. Similarly, the entry '23 40' means that the only ways to reach pixel $[2,3]$ of B are via ϕ_1^- and ϕ_3^-, which take $[2,3]\mapsto(2,3)$ and $[2,3]\mapsto(4,0)$.

That's good news, because bipartite matching problems are relatively easy to solve. Moreover, the problems that arise from dissection scenarios are *especially* easy, because each vertex has at most d neighbors. Indeed, the problem in Table 1 is almost immediately solvable by hand, because nearly all of the moves are forced: Looking only at cases where there's one choice from A, we must match

$$(0,0)\overset{1}{\leftrightarrow}[0,0],\ (0,4)\overset{2}{\leftrightarrow}[3,0],\ (1,2)\overset{1}{\leftrightarrow}[1,2],\ (1,4)\overset{3}{\leftrightarrow}[6,0],$$
$$(4,0)\overset{3}{\leftrightarrow}[2,3],\ (4,1)\overset{3}{\leftrightarrow}[3,3],\ (4,2)\overset{2}{\leftrightarrow}[7,2],\ (4,3)\overset{2}{\leftrightarrow}[7,1],\ (4,4)\overset{2}{\leftrightarrow}[7,0];$$

and in cases where there's just one choice to B, also

$$(0,1)\overset{1}{\leftrightarrow}[0,1],\ (0,2)\overset{1}{\leftrightarrow}[0,2],\ (0,3)\overset{1}{\leftrightarrow}[0,3],$$
$$(1,0)\overset{1}{\leftrightarrow}[1,0],\ (1,1)\overset{1}{\leftrightarrow}[1,1],\ (1,3)\overset{1}{\leftrightarrow}[1,3];$$

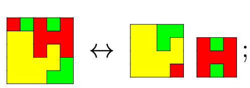
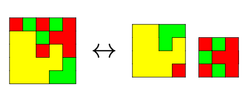
and then, working back and forth with the remaining possibilities, also

$$(3,0)\overset{3}{\leftrightarrow}[2,2],\ (3,4)\overset{3}{\leftrightarrow}[6,2],\ (3,2)\overset{1}{\leftrightarrow}[3,2],\ (3,1)\overset{1}{\leftrightarrow}[3,1],\ (2,1)\overset{1}{\leftrightarrow}[2,1],$$
$$(2,0)\overset{1}{\leftrightarrow}[2,0],\ (2,4)\overset{2}{\leftrightarrow}[5,0],\ (3,3)\overset{2}{\leftrightarrow}[6,1],\ (2,2)\overset{2}{\leftrightarrow}[5,2].$$

Now the entire matching is determined, except for two final choices

$$(2,3)\overset{2}{\leftrightarrow}[5,1] \qquad \text{or} \qquad (2,3)\overset{3}{\leftrightarrow}[5,1];$$

we have discovered *two more* three-color dissections, namely

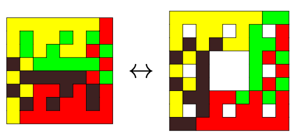


(56)

The 582 successful choices for (α_1,β_1), (α_2,β_2), (α_3,β_3) don't always yield solvable matching problems. In fact, all but 14 of them quickly turn out to be self-contradictory, when forced moves are propagated. And 9 of those 14 reduce to a trivial problem, in which *all* moves are forced. Thus only five cases require the use of a rudimentary bipartite matching algorithm, such as dancing links; one of those five has five unforced vertices and leads to 16 dissections. Altogether

39 three-color dissections of (50) are found, some of which are equivalent to each other because of local symmetries.

Considerably more work is involved when we ask for all *four*-color dissections of (50), but the computations still need only a few gigamems. In this case 230,497 of the $\binom{95}{4} = 3{,}183{,}545$ shift sequences $\beta_1 \le \beta_2 \le \beta_3 \le \beta_4$ cover B, and 4,608,039 subsequent transformations $(\alpha_1\beta_1, \ldots, \alpha_4\beta_4)$ successfully cover A. Noncontradictory matching problems arise after 416,872 of those cases; all but 116,725 of those problems are trivial. The nontrivial ones submit to dancing links, giving a grand total of 1,042,383 dissections — of which 51,472 are rookwise connected as in (52). (If "flips" are disallowed, so that each α_k is simply a rotation ρ^j, the number of dissections goes down to 106,641, of which 6874 are rookwise connected.)

We know from Section 7.1.3 that *kingwise* connectivity is an interesting and important alternative to rookwise connectivity. Discrete dissection problems that ask for kingwise connectivity are largely unexplored and potentially quite interesting. For example, there's a beautiful dissection of the 8×8 square into the 9×9 "Sierpinski carpet" [W. Sierpinski, *Comptes Rendus Acad. Sci.* **162** (Paris, 1916), 629–632], using just *four* kingwise connected pieces(!):


(57)

Exercise 83 discusses data structures for the algorithm just sketched. Heuristic methods that work successfully on several problems that are too large for these exact methods have been introduced by Y. Zhou and R. Wang, *Proc. of Bridges 2012: Mathematics, Music, Art, Architecture, Culture* (July 2012), 49–56. Greg N. Frederickson's book *Dissections: Plane & Fancy* (1997) is a standard reference for all kinds of dissection puzzles, discrete or otherwise.

* * *

(Please stay tuned for further adventures in puzzledom.)

flips
kingwise connectivity
Sierpinski carpet
Sierpinski
data structures
Zhou
Wang
Frederickson

EXERCISES

30. [*M21*] If $b > 2$ and $x = (x_n \ldots x_1 x_0)_b \ge b^{m+1}$, prove that $x_n^m + \cdots + x_1^m + x_0^m < x$.

31. [*10*] What's the difference between $M_3(x)$ and $M_4(x)$, as defined in the text?

▶ **32.** [*22*] Algorithm P requires frequent examination of the individual decimal digits of multiprecision numbers. What's a good way to do that on a binary computer?

33. [*M28*] Complete the description of Algorithm P by specifying step P4.

a) First show how to obtain a refined lower bound a_l and a refined upper bound b_l based on the current values of a, b, and Σ_l, without changing t.

b) Then explain how to decrease t, when $a = b$ and $t \ge 0$.

34. [*24*] Implement Algorithm P. For which $m < 100$ are all solutions trivial?

35. [*20*] For which m are there nontrivial perfect digital invariants with no 9s?

▶ **37.** [*M27*] Find an efficient way to compute the largest integer x such that $\pi_m x \ge x$.

38. [*M22*] A "perfect number" is equal to the sum of its divisors, excluding itself; for example, $6 = 1+2+3$ and $28 = 1+2+4+7+14$. "Amicable numbers" are equal to the sums of each other's divisors in a similar way; for example, $220 = 1+2+4+71+142$ and $284 = 1+2+4+5+10+11+20+22+44+55+110$.

This classic definition, going back to followers of Pythagoras in ancient Greece, suggests that "perfect digital invariants" are akin to "amicable digital pairs" such as

$$136 = 2^3 + 4^3 + 4^3 \qquad \text{and} \qquad 244 = 1^3 + 3^3 + 6^3.$$

a) What's another amicable digital pair of order 3?

b) Devise a good way to find all such pairs of order m when m isn't extremely large.

▶ **39.** [*M25*] If $x = x_0 \ge 0$, repeated application of the mapping $x_{j+1} \leftarrow \pi_m x_j$ will eventually reach a cycle of period length $\lambda > 0$, where $x_{j+\lambda} = x_j$ for all $j \ge \mu$. (See exercise 37 and exercise 3.1–6.) What's a good way to discover all such periods, given $m \ge 2$?

40. [*HM46*] Are there infinitely many m for which $x = \pi_m x$ has (a) 2 (b) >2 solutions?

41. [*M25*] Prove that the number of radix-b numbers with $x = (x_n \ldots x_1 x_0)_b = x_n^2 + \cdots + x_1^2 + x_0^2$ is $s(b^2+1)$ when b is even and $2s(b^2+1)$ when $b > 1$ is odd, where $s(r) = [z^r]\,(1 + z + z^4 + \cdots)^2$ is the number of ways to write r as a sum of two squares.

▶ **42.** [*M23*] Explain how to find solutions to $(d_1 d_2 \ldots d_m)_{10} = d_1^1 + d_2^2 + \cdots + d_m^m$ in a reasonable amount of time, if m is reasonably small. (Here $0 \le d_j < 10$ for $1 \le j \le m$; the leading digit d_1 is allowed to be zero.) For example, the solutions when $m = 7$ are 0000000, 0000001, 0063760, 0063761, 0542186, and (amazingly) 2646798. You should be able to find all solutions for $m = 16$ in less than a minute.

48. [*M17*] (Y. Matsunaga, 1738.) Once upon a time, a certain amount was paid to each of 37 people. Unfortunately, most of the records of that transaction were eaten away by moths; existing accounts show only that the individual amounts were ▭23, and that ▭23□□ was paid altogether. Can the original amounts be reconstructed?

49. [*M19*] (W. P. Workman and R. H. Chope, 1902.) Find the missing digits:

```
      3 □ □
    × 6 □
    -------
    2 4 □ □
  □ □ 8 □
  ---------
  □ □ 2 □ □
```

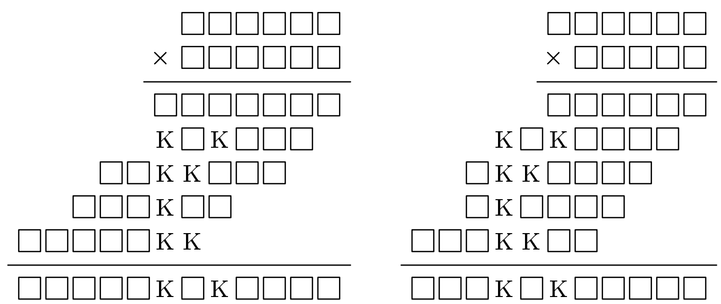
multiprecision numbers
digit extraction
extraction of digits
perfect digital invariants
perfect number
Amicable numbers
Pythagoras
amicable digital pairs
Digital pairs, amicable
mapping
cycle
radix-b numbers
sum of two squares
narcissistic numbers
Matsunaga
Workman
Chope

50. [*M19*] Solve the skeleton multiplication puzzle (30) quickly by hand.

51. [*M21*] The solution to exercise 50 begins with the fact that the product in (30) is completely known. Devise a similar puzzle in which all seven 7s appear only within the *partial* products — not in the multiplicand, or in the multiplier, or in the product.

▸ **52.** [*M22*] Extend exercise 51 to a complete set of nine puzzles, having respectively one 1, two 2s, …, eight 8s, and nine 9s.

▸ **54.** [*21*] Consider the following variants of puzzle (31), where □ ≠ K:



On the left, one of the multiplier digits is implicitly forced to be zero.

On the right, the Ks are positioned in such a way that at least two □s appear at the right of each line. The right-hand puzzle is therefore said to have "slack 2," while the left-hand one has "slack 0" and (31) has "slack 1."

These puzzles were discovered by an algorithm like that of the text, having specified "offsets" $O = (o_0, o_1, \ldots, o_m)$ and "patterns" $P = (p_0, p_1, \ldots, p_m)$. In the left example, $O = (0, 1, 2, 4, 5, 0)$ and $P = (0, 101000, 11000, 100, 11, 1010000)$; in the right example, $O = (0, 1, 2, 3, 4, 0)$ and $P = (0, 1010000, 110000, 10000, 1100, 10100000)$.

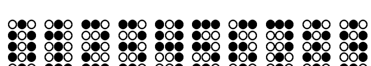
a) Explain how to run through all offsets O, for multipliers that have m nonzero digits and at most z zero digits.

b) Given offsets O, a pixel pattern such as , and a slack s, explain how to compute the patterns P.

55. [*M21*] Show that the K shape in (31) and exercise 54 can be embedded in a skeleton puzzle that has a 5-digit multiplicand and a 4-digit multiplier (and slack 0).

▸ **56.** [*30*] Choose appropriate data structures for an algorithm that looks for skeleton multiplications as in the text, given offsets O and patterns P as in exercise 54. Also sketch the details of that algorithm.

57. [*24*] Design a series of ten puzzles, one for each digit d from 0 to 9, in which all occurrences of d appear in the shape of a d. Exactly five digits of each multiplier should be nonzero; an example appears in Fig. 300. Use the following pixel patterns for the shapes:



58. [*24*] Design a series of twenty-six puzzles, one for each letter of the alphabet, in which all occurrences of some digit appear in the shape of a particular letter. Your puzzles should be "optimum" in the sense that (i) exactly four digits of each multiplier should be nonzero; (ii) the total number of digits should be as small as possible. An

skeleton multiplication
slack
pixel pattern
data structures
pixel patterns
alphabet

Schuh
Feynman
division skeleton problem
exact cover problem

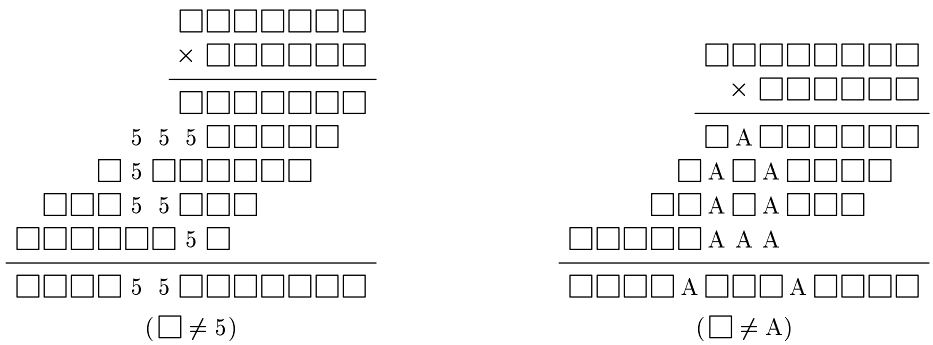
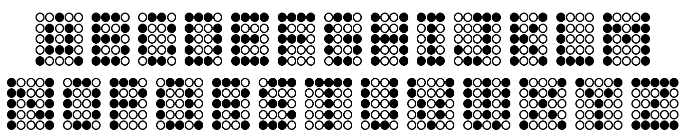


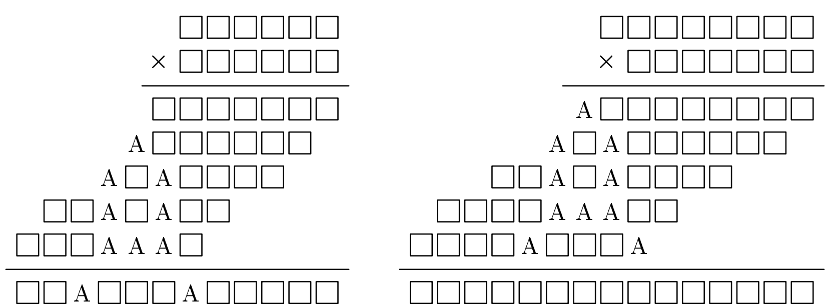
Fig. 300. Prototypes for two series of puzzles (see exercises 57 and 58).

example appears in Fig. 300; however, the puzzle shown there is *not* optimum, because a smaller skeleton is possible using slack 1! Use the following pixel patterns:



(Note that this K is *wider* than the Ks in (31) and exercise 54.)

59. [*24*] Use the pixel patterns of exercise 58 to design two more series of alphabetic puzzles, this time with multipliers that have exactly *five* nonzero digits. The first series should be like (31), with no special digits in the first partial product. The second series should have no special digits in the *total* product. Examples (□ ≠ A):



64. [*M24*] (F. Schuh, 1943.) Prove that the skeleton multiplication in Fig. 301(a) has exactly one solution in which each of the digits $\{0, 1, \ldots, 9\}$ occurs exactly twice.

65. [*M22*] What's the unique way to insert digits 1, 2, 3, 4, or 5 into Fig. 301(b)?

66. [*M20*] In 1939, Richard Feynman (age 21) was intrigued by the long-division skeleton problem of Fig. 301(c), in which all occurrences of a secret digit '@' have been specified (□ ≠ @).

a) What skeleton *multiplication* corresponds to this division?
b) Does that skeleton multiplication have a unique solution?

68. [*M26*] Show that any *exact cover problem* can be converted in a natural way to a skeleton multiplication problem that has the same number of solutions. Demonstrate your construction by applying it to example 7.2.2.1–(5).

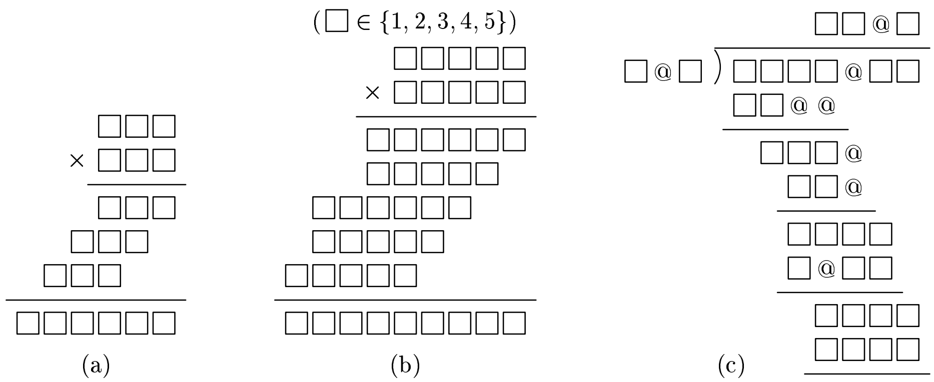


Fig. 301. Special skeleton puzzles, discussed in exercises 64–66.

Loyd
data structures
discrete dissection
rookwise connected
Frederickson
font
alphabet
disconnected solutions

80. [*M20*] Why does the text say that the "best conceivable score" for a solution to Loyd's problem (50) is $5^2+5^2+6^2+9^2$?

81. [*M20*] What are the four transformations ϕ_k in the *left-hand* dissection of (52)?

▶ **83.** [*23*] Design good data structures for the text's discrete dissection algorithm.

85. [*21*] The dissection in (54) has the smallest score, $7^2+7^2+11^2$, among all 3-colorings of (50). What is the *largest* attainable score?

86. [*22*] Extending (51), find all of the discrete four-piece rookwise connected dissections of a 13×13 square into 12×12 and 5×5 squares, with no pieces rotated or flipped.

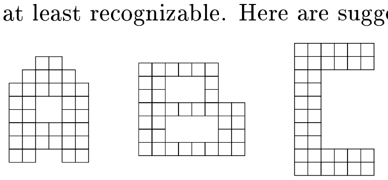
▶ **87.** [*23*] Continuing exercise 86, find all of the discrete four-piece unrotated dissections — connected or not — of (a) 5^2 into 4^2+3^2; (b) 13^2 into 12^2+5^2; (c) 17^2 into 15^2+8^2. (*Don't* allow pieces to jump between squares as they do in (54) and (56).)

88. [*M30*] (G. N. Frederickson.) Prove that there's a discrete four-piece rookwise connected dissection of a $w\times w$ square into squares of sizes $u\times u$ and $v\times v$ whenever (a) $u=2p^2+2p$, $v=2p+1$, $w=2p^2+2p+1$; (b) $u=4p^2-1$, $v=4p$, $w=4p^2+1$.

89. [*M46*] Are such dissections possible for *all* (u,v,w) with $u^2+v^2=w^2$? (The smallest unsolved cases occur for $(u,v,w)=(20,21,29)$; $(28,45,53)$; $(33,56,65)$; $(48,55,73)$.)

▶ **91.** [*24*] Each of the 36 characters in '**FONT36**' (see Fig. 302) can be obtained by dissecting a 6×6 square into at most four pieces. Find "best possible" dissections, giving preference to well-connected pieces of near-equal size.

▶ **92.** [*29*] Design a 26-character font in which each letter from A to Z is obtainable by dissecting a 6×6 square into at most *three* pieces, each of which is *rookwise connected*. *Note:* Your letters won't be as consistent with each other as they are in **FONT36**. But strive to make them at least recognizable. Here are suggestions for A, B, and C:



93. [*29*] And can a decent alphabet be made with *two*-piece dissections?

95. [*40*] Experiment with algorithms for dissection that rule out disconnected solutions early in the process, instead of first generating the complete set of dissections.

Sierpinski carpet

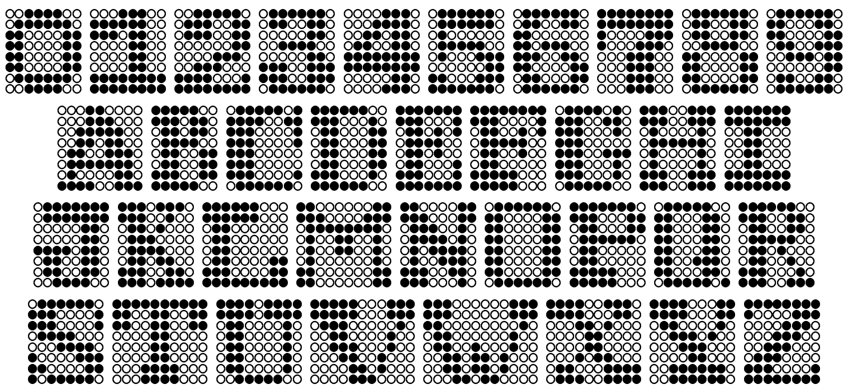


Fig. 302. '**FONT36**', a special font designed for dissection puzzles.

96. [*40*] Can the 8×8 square be dissected into the 9×9 Sierpinski carpet using at most six *rookwise* connected pieces?

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

SECTION 7.2.2.8

binary-coded decimal
bitwise operations
Oliveira
bootstrap algorithm
Stewart
Iséki
multicombination

30. Let $t_m = (m+2)(b-1)^m/b^{m+1}$. Then $t_{m+1}/t_m = (m+3)(b-1)/((m+2)b)$; so $t_0 < \cdots < t_{b-3} = t_{b-2} > t_{b-1} > \cdots$, and the maximum is $t_{b-2} = (1-1/b)^{b-2} < 1$.

Similarly, if $n \geq m+1$ we have $b^n/(n+1) \geq b^{m+1}/(m+2)$. Hence $x_n^m + \cdots + x_0^m \leq (n+1)(b-1)^m \leq b^{n-m-1}(m+2)(b-1)^m < b^n \leq x$.

31. $M_3(0) = \{0,0,0,0\}$, $M_4(0) = \{0,0,0,0,0\}$; $M_m(x)$ is defined only for $x < 10^{m+1}$.

32. Algorithm P needs multiprecise arithmetic *only* for addition. Therefore Σ_l, a_l, b_l, and the basic constants $j \cdot k^m$ can be represented conveniently as binary-coded decimal integers, with 15 digits per octabyte. For example, the number x in (20) would appear in memory as seven octabytes `#2656162`, `#296193301098036`, . . . , `#801479850942958`. Binary-coded addition is easy with bitwise operations as in exercise 7.1.3–100.

33. (a) If $t < 0$, go to P7. Otherwise if $b \geq c$, set $a_l \leftarrow \Sigma_l$ and $b_l \leftarrow \Sigma_l + r \cdot c^m$. Otherwise if $r = 0$ go to P7. Otherwise set $a_l \leftarrow \Sigma_l + a^m$ and $b_l \leftarrow \Sigma_l + b^m + (r-1) \cdot c^m$. Then set $a \leftarrow a_{lt}$ and $b \leftarrow b_{lt}$. Repeat until a_l and b_l don't change further.

(b) This is the most delicate part, because we may have learned a new digit. Do the following steps, while $a = b$ and $t \geq 0$; then go back to (a): (i) If $b < c$, go to (v). (ii) If $e_{lb} < d_{lb}$, go to (vii). (iii) If $b > c$, go to P7 (we've already saturated digit b). (iv) Set $q \leftarrow e_{lb} + 1 - d_{lb}$ (which is positive). (v) Set $r \leftarrow r - 1$, and go to P7 if $r < 0$. (vi) Set $\Sigma_l \leftarrow \Sigma_l + b^m$ (because b is a newly known digit less than c). (vii) Set $e_{lb} \leftarrow e_{lb} + 1$ and $t \leftarrow t - 1$. If $t \geq 0$, also set $a \leftarrow a_{lt}$ and $b \leftarrow b_{lt}$.

[Tomás Oliveira e Silva has observed that better bounds are possible. For example, in the text's discussion we could raise the lower bound a_{91} to the exact value $.00002\,39052\,58999 \times 10^{101}$, because 8s are forbidden; and a similar idea applies to upper bounds. Further exploration of such techniques should prove to be interesting.]

34. Only m = 2, 12, 15, 18, 22, 26, 28, 30, 40, 41, 48, 50, 52, 58, 80, 82, 88, 98.

35. 3, 4, 5, 6, 7, 8, 9, 13, 17, 25, 27, 29, 47, and no other $m \leq 1000$. (Just change 9 to 8 in step P1.) Incidentally, there's a solution for $m = 73$ with only *one* 9!

37. Let $\alpha = a_m a_{m-1} \ldots a_t$ be a string of decimal digits, of length $m+1-t$, where $a_t > 0$; and let $\alpha - 1$ be the same string but with a_t decreased by 1. Let x_α be the decimal number obtained by appending t zeros to the right of α, and let y_α be $x_\alpha - 1$. Hence y_α is the decimal number obtained by appending t 9s to the right of $\alpha - 1$.

The following "bootstrap algorithm" works with strings α such that $x \geq x_\alpha$ implies $\pi_m x < x$; this condition holds initially with the one-digit string $\alpha = \lceil (m+1)9^m/10^m \rceil$. Given such an α we form $z_\alpha = \pi_m y_\alpha = (z_m z_{m-1} \ldots z_0)_{10}$, and find the largest r such that $z_m \ldots z_{r+1} = y_m \ldots y_{r+1}$. If $r < 0$ or $z_r < y_r$, the answer is y_α. Otherwise, if $r \leq t$, we set $t \leftarrow r$ and $\alpha \leftarrow z_m \ldots z_r + 1$. Otherwise we set $t \leftarrow t+1$ and increase t further if necessary until $a_t > 0$.

When $m \leq 150$, this algorithm finds the solution in fewer than 14 iterations and fewer than 115 Kμ. The answers y_α for $1 \leq m \leq 5$ are 09, 099, 1999, 19999, 229999; they can be represented more succinctly by their prefixes $\alpha - 1$, namely 0, 0, 1, 1, and 22. The analogous prefix for $m = 100$ is 000251. [See B. M. Stewart, *Canadian J. Math.* **12** (1960), 374–389.]

38. (a) $919 = 1^3 + 4^3 + 5^3 + 9^3$, $1459 = 9^3 + 1^3 + 9^3$. [K. Iséki also exhibited two 3-cycles, in *Proc. Japan Academy* **36** (1960), 578–587.]

(b) For each multicombination $9 \geq d_m \geq \cdots \geq d_0 \geq 0$, form $x \leftarrow d_m^m + \cdots + d_0^m$ and $y \leftarrow \pi_m x$. If $x < y$, also form $z \leftarrow \pi_m y$; and if $x = z$, report the pair (x, z).

If $m > 33$ we can assume $d_0 = 0$, because $(m+1)9^m < 10^m$. Also $d_1 = 0$ if $m > 61$.

When $m = 3$ this method actually reports $(919, 1459)$ twice, from $d_3d_2d_1d_0 = 8740$ and 9541, because of the "birthday paradox" coincidence $0^3+7^3+8^3 = 1^3+5^3+9^3$(!).

The number of (distinct) amicable pairs for $m = 2, 3, \ldots, 33$ is (0, 2, 1, 2, 1, 2, 0, 2, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 2, 1, 1, 0, 3, 0, 0, 1, 4); the largest pair for $m = 33$ is

$$(95\,26805\,32993\,29396\,93391\,76210\,89100,\;\; 248\,50076\,01437\,39486\,22580\,87152\,05099).$$

No streamlined method analogous to Algorithm P appears to be possible.

39. *Solution 1:* As in answer 38(b), we generate all multicombinations $d_m \ldots d_0$; but this time we store them in memory, as binary-coded hexadecimal numbers $(d_m \ldots d_0)_{16}$. Let them be $D_0 = 00\ldots0$, $D_1 = 10\ldots0$, $\ldots$, $D_{p-1} = 99\ldots9$, where $p = \binom{m+10}{9}$. Notice that the method of Algorithm 7.2.1.3T yields these numbers in increasing order; hence it's easy to do a binary search in this array.

Let $g(j)$ be the function such that $D_{g(j)}$ equals the result of bucket-sorting the digits of $\pi_m D_j$. For example, when $m = 3$ we have $D_{314} = 7755$ and $7^3+7^3+5^3+5^3 = 936$ and $D_{557} = 9630$, so $g(314) = 557$.

There's a simple "tagging algorithm" that finds all cycles of any mapping g from $[0\,.\,.\,p)$ into itself: We explore from an untagged vertex j, tagging every vertex that we see until first encountering a tagged vertex k. Then we double-tag all vertices from j to k; and if k wasn't already double-tagged, it begins a new cycle (which we proceed to double-tag before moving to another j). Formally, assume that we're given an array of two-bit quantities T_j for $0 \le j < p$, initially zero, and do the following steps for $j = 0$, $1, \ldots, p-1$: If $T_j > 0$ do nothing. Otherwise, set $k \leftarrow j$; while $T_{g(k)} = 0$, set $k \leftarrow g(k)$ and $T_k \leftarrow 1$. Then set $i \leftarrow j$; while $i \ne k$, set $i \leftarrow g(i)$ and $T_i \leftarrow 2$. Then if $T_{g(k)} < 2$, we've found a new cycle, beginning at (say) k; set $i \leftarrow g(i)$, $l \leftarrow 0$, and while $T_{g(i)} < 2$ set $i \leftarrow g(i)$, $l \leftarrow l+1$, $T_i \leftarrow 2$. The cycle length is l.

Solution 2: We need not store the multicombinations in memory, nor do binary search, because Theorem 7.2.1.3L tells us exactly where to find any multicombination.

More precisely, we set $p \leftarrow 0$ and perform Algorithm 7.2.1.2T with $s = 9$, $t = m+1$, and $n = m + 10$. When visiting $c_t \ldots c_1$ in step T2, we compute $(c_t+t-1)^m + \cdots + (c_2+1)^m + c_1^m = (e_m \ldots e_1e_0)_{10}$, set $g(p) = \binom{e_t+t-1}{t} + \cdots + \binom{e_2+1}{2} + \binom{e_1}{1}$, and $p \leftarrow p+1$.

Historical notes: A. Porges found the cycles for $m = 2$ by hand [*AMM* **52** (1945), 379–382]. K. Iséki found them for $m = 3$ in 1960, also by hand (see answer 38(a)). Then computers came into the picture, at first with cumbersome methods because of limited memory. In unpublished work communicated to Martin Gardner, R. L. Patton, Sr., R. L. Patton, Jr., and J. S. Madachy reached $m = 17$ by 1975.

40. Empirical results for $m \le 180$ give strong support for (b) and mild support for (a). Both conjectures may well be true, although they are well beyond any known proof techniques. [See B. L. Schwartz, *J. Recreational Mathematics* **3** (1970), 88–92.]

41. The conditions are equivalent to $x_j = 0$ for $j > 1$ and $b^2+1 = (b-2x_1)^2+(2x_0-1)^2$.

Suppose $b^2+1 = u^2+v^2$; the corresponding solutions are $x_1 = (b \pm u)/2$ and $x_0 = (1 \pm v)/2$. Hence v must be odd, and these four cases lead to exactly two in the range $0 \le x_1, x_0 < b$. [N. J. Fine, *AMM* **71** (1964), 1042–1043, noted also that $s(b^2+1) = 2$ if and only if b^2+1 is prime; otherwise there are solutions with $x_1 > 0$.]

42. Let $p_{r,d} = d \cdot 10^{m-r} - d^r$. Set $l \leftarrow \lceil m/2 \rceil$ and form the multisets of 10^l values $A = \{p_{1,d_1} + \cdots + p_{l,d_l}\}$ and 10^{m-l} values $B = \{-p_{l+1,d_{l+1}} - \cdots - p_{m,d_m}\}$. Then the solutions correspond to the elements of the multiset intersection $A \cap B$. (See exercise 4.6.3–19.)

We can gain some efficiency by omitting negative elements of B, and by omitting from A all elements that exceed the largest element of B. For example, when $m = 16$

birthday paradox
coincidence
binary-coded
binary search
bucket-sorting
tagging algorithm
double-tagged
Porges
Iséki
Gardner
Patton, Sr.
Patton, Jr.
Madachy
Schwartz
Fine
multiset intersection

the reduced multiset B still has 99,795,483 elements, but A reduces to only 20,846,476. After sorting those multisets, the intersection is quickly found.

The number of solutions for $m = (2, 3, \ldots, 16)$ is (3, 8, 5, 2, 4, 6, 2, 2, 3, 2, 2, 3, 3, 2, 3), respectively; 0033853790788237 is the surprising solution for $m = 16$.

[Such numbers were introduced by D. Kozniak in *Recreational Mathematics Magazine* #10 (August 1962), 42; he and J. A. H. Hunter found all solutions for $m = 2$ and $m = 3$. J. S. Madachy found the first five solutions for $m = 7$ in 1970; see *Fibonacci Quarterly* **10** (1972), 295–298.]

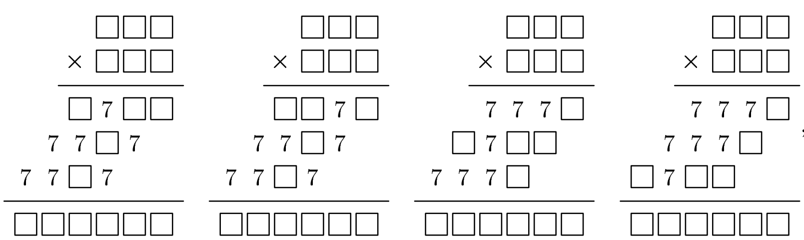
sorting
Kozniak
Hunter
Madachy
backtrack tree
author
Yamamoto
Okoma
Take
Maruo

48. The smallest solution to □23 × 37 = □23□□ is 9523 × 37 = 352351. For each 10000 added to the individual amounts, add 370000 to the total.

49. 347 × 67. (Only ten cases 314, 330, ... work for the product by 6.)

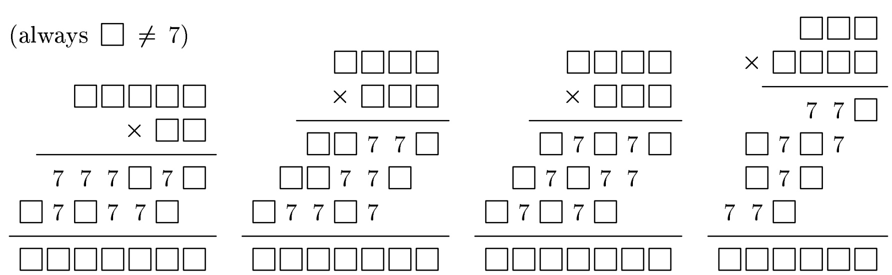
50. The only 3-digit divisors of $777777 = 3 \cdot 7^2 \cdot 11 \cdot 13 \cdot 37$ that contain no 7s and don't end in 1 are (143, 259, 429, 539). Their cofactors are (5439, 3003, 1813, 1443). And only 539 × 1443 gives just seven 7s without introducing a leading zero.

51. A simple backtrack shows that the shortest such puzzles with a unique solution are



But they're even *easier* than (30), because 77∗7 and 777∗ can be a multiple of a one-digit 7-free number only if that divisor is 9. Hence ∗ = 6, and this gives the answers away.

So the best short puzzles of this kind — each findable with a small backtrack tree, given the number of digits in multiplicand and multiplier — are just a bit longer:



52. The author's favorites, among many possibilities, are shown in Fig. A–30. [Such puzzles were pioneered by Yukio Yamamoto in 1975. See S. Okoma, J. Take, and M. Maruo's excellent book *Mushikuizan pazuru 700-sen* (Kyoritsu, 1985), 42, 200.]

54. (a) To run through all sequences with $0 = o_0 < o_1 < \cdots < o_{m-1} < m + z$ and $o_m = 0$, use (say) Algorithm 7.2.1.3T with $s = z$, $t = m - 1$, $o_j = c_j + 1$.

(b) There's a "raw" pixel pattern, independent of offsets, which can be represented as $R = (r_0, r_1, \ldots, r_m)$; the example pattern has $R = (000, 101, 110, 100, 110, 101)$. Suppose r_j ends with t_j zeros. Then $p_j = (r_j \ll t) \gg o_j$, where $t = s + \max_{0 \le j \le m}(o_j - t_j)$.

decimal arithmetic
bignum
stamping+
inverse permutation
deletion from lists

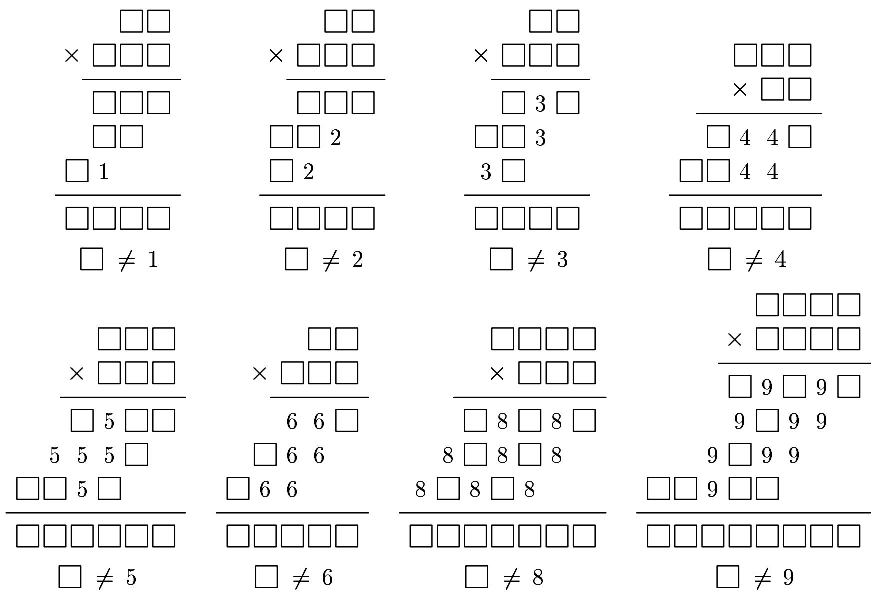


Fig. A–30. Skeleton multiplication puzzles with d ds.

(By the way, the answers to the given puzzles are 237457×720845 and K $= 9$; 467224×6521 and K $= 3$. Another nice puzzle with slack 0 is answered by 38522×3597001 and K $= 6$. There are none with slack 0 and $z = 0$.)

55. Take the skeleton of 38522×3597, with K $= 6$.

56. Instead of using the computer's built-in multiplication, it's best to implement decimal arithmetic from scratch. Say that a *bignum* is a nonnegative integer x that's represented as a sequence of bytes $x_0x_1\ldots x_{N-1}$, with (say) $N \approx 25$; the value of x is $(x_t\ldots x_1)_{10}$, where $t = x_0$, and $x_t \neq 0$ unless $t = 0$. It's easy to write a routine that computes $x + 10^q y$, given bignums x and y and an offset q, and to prepare the basic multiplication table of bignum constants $a \cdot b$ for $0 \le a, b < 10$.

We maintain an array `JA`$[l][j]$ of bignums, representing $j \cdot (a_l \ldots a_0)_{10}$ at level l of the algorithm, for $0 \le j < 10$. Clearly `JA`$[l][j]$ $=$ `JA`$[l-1][j] + 10^l(j \cdot a_l)$ when $l > 0$. (See (36) and (37); but we don't truncate to l digits as shown there.) These values need to be computed only when j is a potentially useful multiplier digit. So we have another array `STAMP`$[l][j]$ by which we can tell if `JA`$[l][j]$ is valid (see below).

Next there's `CHOICE`$[k]$, for $0 \le k < m$, which is a permutation of $\{0, 1, \ldots, 9\}$; also `WHERE`$[k]$, which is the inverse permutation. (Thus `CHOICE`$[k][i] = j$ if and only if `WHERE`$[k][j] = i$.) The multiplier digits that haven't been ruled out by constraint p_k at level l are the first `S`$[l][k]$ elements of `CHOICE`$[k]$, namely the elements j such that `WHERE`$[k][j] <$ `S`$[l][k]$. This setup permits easy deletion from lists while backtracking, because p_k becomes stronger as l increases; see 7.2.2–(23).

Finally we prepare an array `ID` such that $p_k = p_{k'}$ if and only if `ID`$[k]$ $=$ `ID`$[k']$. A `STACK` is used to propagate forced constraints. And the variable `NODES`, initially 0, holds ten times the serial number of the current node.

The algorithm has an outer loop for $0 \le d < 10$, where d is the special digit of the pattern (called 'K' in (31)). We allow $d = 0$ only if $o_{m-1} = m - 1$. A backtrack scheme like Algorithm 7.2.2B is followed for each d, but starting at level $l = 0$. sort author

To initialize in step B1, first set $i \leftarrow 0$ and do the following for $1 \le j < 10$: If $j \ne d$, set $\texttt{CHOICE[}k\texttt{][}i\texttt{]} \leftarrow j$ and $\texttt{WHERE[}k\texttt{][}j\texttt{]} \leftarrow i$ for $0 \le k < m$, then set $i \leftarrow i + 1$. Then $\texttt{S[0][}k\texttt{]} \leftarrow i$, $\texttt{CHOICE[}k\texttt{][}i\texttt{]} \leftarrow d$, $\texttt{WHERE[}k\texttt{][}d\texttt{]} \leftarrow i$, $\texttt{WHERE[}k\texttt{][0]} \leftarrow 9$, for $0 \le k < m$.

At the beginning of step B2, set $\texttt{NODES} \leftarrow \texttt{NODES}+10$. If $\texttt{S[}l\texttt{][}k\texttt{]} = 1$ for $0 \le k < m$ and if all constraints $p_0, \ldots, p_m$ are totally satisfied under the assumption that $a_j = 0$ for all $j \ge l$, output the current solution and go to B5. (The current solution is represented by the multiplicand a and multiplier b. To sort for unique skeletons, we also want the length of a and the lengths of $\texttt{JA[}l-1\texttt{][CHOICE[}k\texttt{][0]]}$ for $0 \le k \le m$.)

Step B3, which tests if $a_l \leftarrow x$ is viable, is the heart of the algorithm. Reject x if $x = d$. Otherwise set $p \leftarrow 0$, and do the following for $m > k \ge 0$: Set $s \leftarrow \texttt{S[}l\texttt{][}k\texttt{]}$. For $0 \le i < s$, set $j \leftarrow \texttt{CHOICE[}k\texttt{][}i\texttt{]}$ and test if j would remain viable for p_k when $a_l = x$. If not, go to B4 if $s = 1$; otherwise set $s \leftarrow s - 1$; and if $i \ne s$, swap j into position s by setting $j' \leftarrow \texttt{CHOICE[}k\texttt{][}s\texttt{]}$, $\texttt{CHOICE[}k\texttt{][}i\texttt{]} \leftarrow j'$, $\texttt{WHERE[}k\texttt{][}j'\texttt{]} \leftarrow i$, $\texttt{CHOICE[}k\texttt{][}s\texttt{]} \leftarrow j$, $\texttt{WHERE[}k\texttt{][}j\texttt{]} \leftarrow s$, and $i \leftarrow i - 1$. If there was no exit to B4, set $\texttt{S[}l+1\texttt{][}k\texttt{]} \leftarrow s$; also, if $s = 1$ and $\texttt{S[}l\texttt{][}k\texttt{]} > 1$, set $\texttt{STACK[}p\texttt{]} \leftarrow k$, $p \leftarrow p + 1$.

Here is the promised test for viability of j: If $\texttt{STAMP[}l\texttt{][}j\texttt{]} \ne \texttt{NODES} + x$, set $\texttt{STAMP[}l\texttt{][}j\texttt{]} \leftarrow \texttt{NODES} + x$ and compute $\texttt{JA[}l\texttt{][}j\texttt{]}$. Then '$j$ remains viable for p_k' means that digit l of $\texttt{JA[}l\texttt{][}j\texttt{]}$ equals d if and only if digit l of p_k equals 1.

Step B3 is not yet finished. After the stated loop on k, we need to clear the stack: While $p > 0$, set $p \leftarrow p-1$, $k \leftarrow \texttt{STACK[}p\texttt{]}$, and delete $\texttt{CHOICE[}k\texttt{][0]}$ from all constraints $\ne p_k$. That means to set $j \leftarrow \texttt{CHOICE[}k\texttt{][0]}$, and for $0 \le k' < m$ with $\texttt{ID[}k'\texttt{]} \ne \texttt{ID[}k\texttt{]}$ to set $s \leftarrow \texttt{S[}l+1\texttt{][}k'\texttt{]} - 1$, $i \leftarrow \texttt{WHERE[}k'\texttt{][}j\texttt{]}$, and if $i \le s$ to do the following: Go to B4 if $s = 0$; otherwise set $\texttt{S[}l+1\texttt{][}k'\texttt{]} \leftarrow s$; if $s = 1$ set $\texttt{STACK[}p\texttt{]} \leftarrow k'$, $p \leftarrow p + 1$; and if $i \ne s$, swap j down as above.

After the stack is clear, we also want to test the overall product constraint p_m, if $P = \prod_{k=0}^{m-1} \texttt{S[}l+1\texttt{][}k\texttt{]}$ is at most some threshold (like 25). That involves an inner loop over P possibilities, in which we find P' cases that satisfy p_m up to digit l. Then we rule out all choices of multiplier digits that aren't present in any of those P' cases.

The good news is that these data structures require no further updating. Indeed, steps B4 and B5 of Algorithm 7.2.2B need no amendments and do no downdating.

Sometimes l reaches the limiting precision of our bignums. In such cases one can usually verify by inspection that no short solutions are being overlooked.

57. The author's candidates are generated by the respective multiplications 1513378×98621, 965289×98467, 46007×33478, 148669×75896, 1380552×7089305, 7939486×390271, 532207×832057, 15543×99458, 46966×35469, 743713×370841.

(The puzzle for $d = 0$ was the most difficult to find.)

58. Here are the author's favorites, using positive slack only when it helps: 8282223×200956 (A = 4), $95283341007 \times 90020507$ (B = 6), 1205719×6827 (C = 4), 66617057×907085 (D = 3), $22222340739 \times 509070003$ (E = 6), 2592619×275009 (F = 3), 13941943×68904 (G = 5), $16604761497 \times 90007058$ (H = 3), 190567×98067 (I = 3), 4055903×75902 (J = 1), 29885338×250309 (K = 6), 18289×6842 (L = 3), 18983233124×75203 (M = 6), 2855352138×70539 (N = 6), 23879578×400975 (O = 1), 74080955×30078009 (P = 6), 82884224×4030208 (Q = 6), $4851881332 \times 8500079$ (R = 6), 5558467×84076 (S = 3), 7407382×807509 (T = 6), 82195935×54809

(U = 7), 15009232 × 35704 (V = 6), 43441858589 × 3209004 (W = 7), 6495974 × 7546 (X = 8), 128052 × 3975 (Y = 6), 10740737877 × 800300059 (Z = 6).

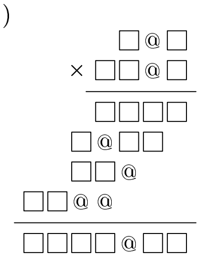
(The puzzles for E and Z were much harder to discover than the others.)

Junya Take introduced alphabet-shaped skeletons in the special puzzle issue VI of *Sūri Kagaku* **19** (1981), 25–26; but those puzzles did not indicate *all* appearances of the special digit. His alphabet puzzles in *Journal of Recreational Mathematics* can be found in **36** (2007), 63–64, 263, 355; **37** (2008), 70–71, 160, 250, 253–254, 347, 350; **38** (2014), 55, 58, 129, 132.

59. (This answer has been omitted so that readers can have the fun of discovery.)

64. Let a, b, c, d, e, f be the numbers involved, so that $a \times b = c + d + e = f$. We also have $a+b+2ab = a+b+c+d+e+f \equiv 0$ (modulo 9), because $0+0+1+1+\cdots+9+9 \equiv 0$. Hence there are six cases, with $(a \bmod 9, b \bmod 9) = (0,0)$, $(2,5)$, $(3,6)$, $(5,2)$, $(6,3)$, or $(8,8)$. Each case is amenable to hand calculation, after which we conclude that $a = 179$ and $b = 224$. [*Wonderlijke Problemen* (1943), §§235–237.]

65. 44511 × 11513. (Only eight multiplications of 5-digit numbers involve only the requested digits; and only three of them have skeletons of a unique shape. The other two such candidates for puzzles are 22431 × 51511 and 41514 × 13331.)



66. The skeleton shown here is solvable for □ ≠ @ only when it computes 484 × 7289 and @ = 8; this is a *stronger* assertion than the statement that the *division* problem has a unique solution. [See R. Feynman, *Perfectly Reasonable Deviations* (2005), 4–5. The division problem is due to W. F. Cheney, Jr., *AMM* **43** (1936), 305.]

68. Assume first that all column sums c_j of the matrix A are less than 9. Let the multiplicand be the $(mn+1)$-digit number obtained by appending the elements of A to the digit '4'. (Thus it is 4001011010010010110010100100001000010001101 in the example.) The multiplier is a completely hidden $(mn-n+1)$-digit number. Each partial product is a completely hidden number of $mn+1$ digits; the offsets are 0, n, $2n$, $\ldots$, $(m-1)n$, implying many zeros in the multiplier. The total product has $2mn-n+1$ digits, of which the rightmost $(m-1)n$ and the leftmost $(m-1)n+1$ are obscured. The other digits are specified to be $((c_1+1)\ldots(c_n+1))_{10}$; in the example they're 3334334.

The idea is that the '4' and the offsets force the multiplier to have the form $z_m 0^{n-1} z_{m-1} 0^{n-1} \ldots 0^{n-1} z_1$, where each z_j is 1 or 2. A solution to the skeleton occurs if and only if the rows for which $z_j = 2$ exactly cover all columns. Thus, the unique solution for the example has multiplier 1000000200000020000001000000100000002.

With larger column sums, we need more space to do the summing, so the number n in the above is increased. For example, we'd use two adjacent digit positions instead of one, in each block of digits, when $9 \le c_j < 99$. But the same general idea applies.

(Hence it is NP-complete to decide whether a given skeleton multiplication can be solved; also to decide whether or not a given solvable skeleton has a unique solution, by exercise 7.2.2.1–33. See T. Matsui, *J. Information Processing* **21** (2013), 402–404.)

80. If we use k colors for the 4×4 square and $4-k$ for the 3×3, the minimum score is $16^2+3^2+3^2+3^2$ for $k=1$, $8^2+8^2+4^2+5^2$ for $k=2$, and $5^2+5^2+6^2+9^2$ for $k=3$.

81. $\phi_1 = \sigma_{0,-1}$, $\phi_2 = \rho\sigma_{-2,2}$, $\phi_3 = \rho$, $\phi_4 = \sigma_{3,0}$.

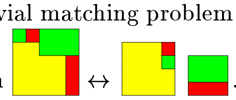
83. Pixels (x,y) and $[x,y]$ can be represented internally as integers such as $32x+y$. Then shifts $\sigma_{a,b}$ can be represented as $32a+b$, and computations are readily done with one-dimensional arrays. To generate valid sequences $\beta_1 \le \cdots \le \beta_d$, it's helpful to

Take
Feynman
Cheney
NP-complete
solvable
unique solution
Matsui

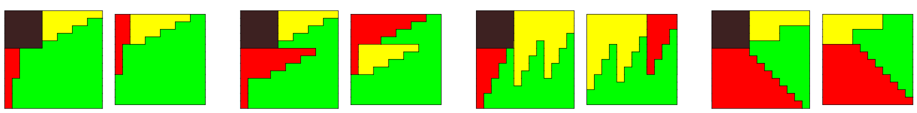
precompute the list of all pixels of B that are covered by a given feasible shift. The total number of such pixels also helps to identify invalid sequences quickly.

As soon as the transformations ϕ_k are known, it's helpful to have lists of all possible mates for each (x, y) and each $[x, y]$, as in Table 1. The sizes of those lists also facilitate the propagation of forced moves. We also want a list of the pixels that haven't yet been matched. Sequential lists are good for this purpose, as they adequately support the deletion operation (see 7.2.2–(18)).

The more elaborate four-way-linked structures of the dancing links algorithm are also useful. But they should be set up only when a nontrivial matching problem arises, and used only for vertices whose mates are unknown.

85. $4^2 + 7^2 + 14^2$ is uniquely attainable by the dissection .

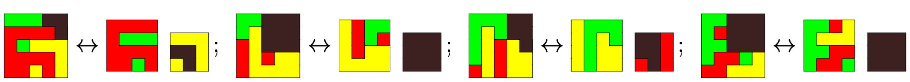
86. There are just four solutions. In order of decreasing "score" they are:



[The rightmost is due to Sam Loyd in the *Philadelphia Inquirer*, 26 February 1899.]

87. To avoid "jumping," in a dissection of w^2 into $u^2 + v^2$, we consider the pixels of B to be either $[x, y]$ for $0 \le x, y < u$ or $\langle x, y\rangle$ for $0 \le x, y < v$; and we let $(x, y)\sigma'_{a,b} = \langle x + a, y + b\rangle$. For example, ϕ_4 becomes $\sigma'_{-2,-2}$ in (53), instead of $\sigma_{3,-2}$. The number of feasible shifts is now $113 = 8^2 + 7^2 = (w + u - 1)^2 + (w + v - 1)^2$, not 92.

(a) Four new solutions arise in addition to (51):



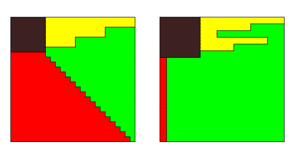
The last of these has $\phi_1 = \sigma_{-1,2}$, $\phi_2 = \sigma_{1,0}$, $\phi_3 = \sigma_{0,-1}$, in common with the last of (51).

(b) Tens of thousands of new solutions arise, but they introduce only two new triples (ϕ_1, ϕ_2, ϕ_3) of usable shifts. "Random" examples are:

The latter two have the same shift-triples as the second and fourth in answer 86.

(c) In this case there are no solutions; all matching problems are self-contradictory.

88. The constructions illustrated here for $p = 3$ generalize to all p. Case (a), which is based on Loyd's construction in answer 86, needs no rotation. [See 'Method 1A' and 'Method 2A' in Frederickson's *Dissections* book.] Answer 87 shows that rotation is necessary for $(u, v, w) = (15, 8, 17)$.



91. The author's favorites appear in Fig. A–32, *after* dissection; also in Fig. A–34, *before* dissection. (These illustrations appear on separate pages, so that readers who like puzzles can have fun figuring out how to go from one form to the other.)

Notice that only *three* pieces are necessary for the `4`, `7`, and `T`. (There also are three-piece dissections for the `1`, but only with *disconnected* pieces.) Rookwise connectivity can be achieved for `0`, `1`, `3`, `5`, `7`, `A`, `D`, `F`, `I`, `J`, `L`, `O`, `P`, `T`, `U`, `Z`; but not even kingwise connectivity is possible for `C`, `E`, `G`, `K`, `M`, `N`, `Q`, `R`, `S`, `V`, `W`, `X`, `Y`.

Sequential lists
dancing links
Loyd
author
Rookwise connectivity
kingwise connectivity

flipped

Fig. A–32. Good ways to fabricate each character of **FONT36** from a 6×6 square.

Given the number of pieces and the level of connectivity, preference has been given to dissections into pieces of nearly uniform size. The number of flipped pieces has also been minimized, if that doesn't reduce uniformity. (For example, there's a no-flip solution to I that has score $7^2 + 9^2 + 9^2 + 11^2$; but it has been superseded by a 2-flip solution with the perfect score $9^2 + 9^2 + 9^2 + 9^2$.) Perfect uniformity has been achieved in cases E, G, I, K, N, R, S, X. The dissections 2, 3, A, and M are *unique*, in the sense that no other four-piece dissection has the same connectivity.

92. See Figs. A–33 and A–34. In this case F, J, L, N, P, S, Y, Z are formed from only *two* pieces. Perfect uniformity is achieved for C, I, L, S, T, U, Y. (Perfectly uniform *three*-piece dissections also exist for L, P, Y; can the reader find them?) The dissections for B, F, J, K, M, N, S, V, X, and Z are unique, given the number of pieces.

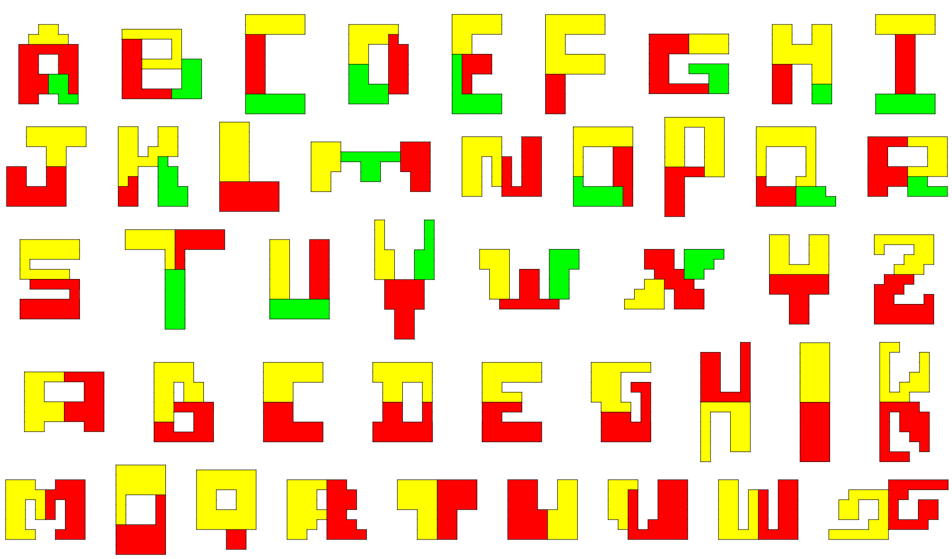


Fig. A–33. Fabricating a less constrained alphabet from a 6×6 square.

93. (Solution by E. Demaine, M. Demaine, and Y. Uno.) All the characters in Figs. A–32, A–33, A–34 can be played with online at `erikdemaine.org/fonts/dissect/`.

Demaine
Demaine
Uno
nondeterministically
Zhou
Wang

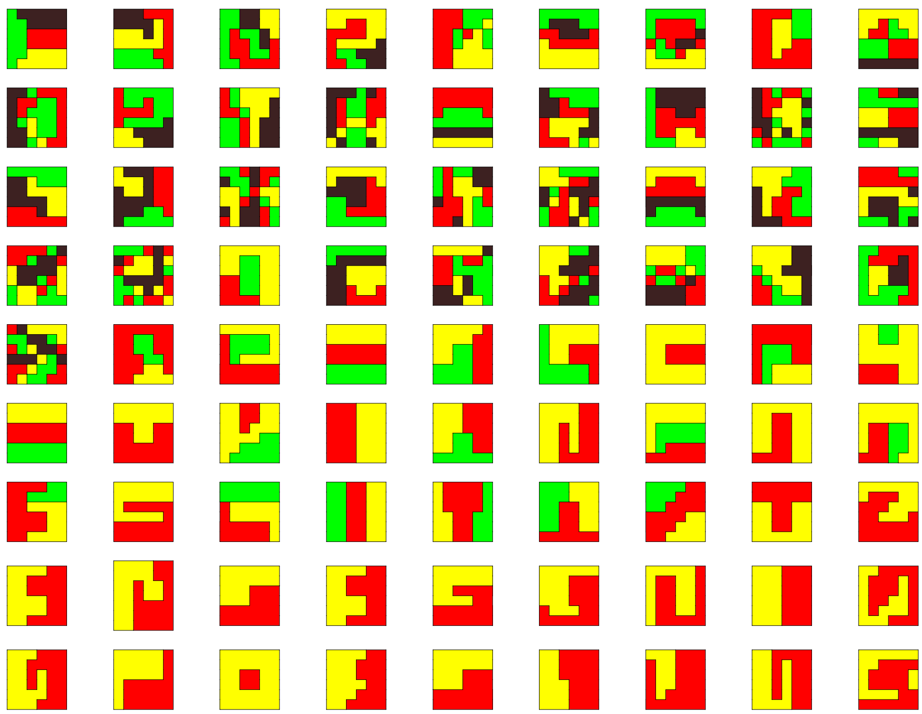


Fig. A–34. How to dissect 6×6 squares in order to obtain Figs. A–32 and A–33.

95. For example, let A_k be the pixels of A that have viable mates of color k, and consider the graph G_k in which two such vertices are adjacent if and only if they are rookwise neighbors. If G_k isn't connected, we must nondeterministically choose one of its components and abandon the others. (In Table 1, G_2 and G_3 aren't connected.)

96. Zhou and Wang have achieved seven, in several ways.

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.