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

September 1, 2026

Internet
Stanford GraphBase
MMIX

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

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

See also `https://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 -79), 29 August 2026

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, 4A, and 4B 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 2, 3, 4, 9, 12, 18, 19, 27, 28, 30, 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
Li
Knuth
HAUPTMAN

* * *

Special thanks are due to Regan Murphy Kao for help with Japanese translations, and to Shisheng Li, . . . for their detailed comments on my early attempts at exposition, as well as to numerous other correspondents who have contributed crucial corrections.

* * *

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

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

Happy reading!

Stanford, California D. E. K.
99 Umbruary 2016

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

7.2.2.8. A potpourri of puzzles. By now we've seen lots of powerful techniques for dealing with combinatorial problems, and we've applied them to many intriguing problems. But we're certainly not at the end of the story! Plenty of other instructive challenges haven't entered our discussions yet, because they cut across topics and use several ideas at once. Some of them, which are particularly significant because they've captured the attention of numerous programmers over the years, have therefore been gathered together here. They're mind-stretching puzzles that are worthy of study even though they might be purely recreational and driven entirely by curiosity. The "obvious" ways to solve them can often be greatly improved by using what we've learned in previous sections.

Enigma M3–
cryptographic
Rejewski
Turing
plaintext
ciphertext
permutation–
polyalphabetic substitution cipher
fullswap–
2-cycles
derangement
involution
perfect matchings
semifactorial
rotors–

The Enigma. The first puzzle we shall investigate in this section, however, was deadly serious. We'll study the Enigma M3, a famous cryptographic machine that was introduced in the 1930s and used by the German army to encipher its communications during World War II. And we'll also look at how Marian Rejewski, Alan Turing, and others were able to break its code.

The Enigma had a keyboard like a typewriter, but its keys were restricted to the 26 letters from `A` to `Z`. There also were 26 lights, likewise labeled `A` to `Z`. When the user would type a letter, one of those lights would turn on. Thus, if the user typed a sequence of letters, $x_1x_2\ldots x_n$, another sequence $y_1y_2\ldots y_n$ of letters would be illuminated, and an assistant could read them aloud so that an additional assistant could write them down. We call $x_1x_2\ldots x_n$ the *plaintext* and $y_1y_2\ldots y_n$ the *ciphertext*. Each letter of the ciphertext was produced by a mapping

$$y_k \;=\; x_k\pi_k\,, \tag{1}$$

where π_k was a permutation of $\{\mathtt{A},\ldots,\mathtt{Z}\}$; this permutation was different for each k. (Such a transformation is called a "polyalphabetic substitution cipher.")

In fact, each π_k was a *fullswap*, namely a permutation that consists of thirteen disjoint 2-cycles: No letter was mapped into itself; and if, say, $\mathtt{O}\mapsto\mathtt{H}$, then also $\mathtt{H}\mapsto\mathtt{O}$. (Technically speaking, a permutation is a fullswap if and only if it is a *derangement*, having no fixed points, as well as an *involution*, having order 2. See, for example, Sections 1.3.3 and 5.1.4. Fullswaps also correspond to *perfect matchings*, as in 7.2.1.2–(49). By exercise 1.3.3–21, the number of fullswaps on $\{\mathtt{A},\ldots,\mathtt{Z}\}$ is $26!/(2^{13}13!) = 25\cdot 23\cdot\ldots\cdot 3\cdot 1 = 7{,}905{,}853{,}580{,}625$.)

The fact that each π_k was a fullswap was convenient for Enigma's users, because they could use the same machine for deciphering as for enciphering: If $y_1y_2\ldots y_n$ was entered as plaintext, the corresponding ciphertext would naturally be $x_1x_2\ldots x_n$. But we will see that this restriction to fullswaps was also convenient for the enemy agents who wanted to break Enigma's code.

The enciphering mechanism was based on three "rotors," each of which was a circular disk having 26 equally spaced contact points on both of its surfaces, near the rim. A maze of wires connected the contact points on one side of a rotor to the contact points on the other side. Each disk could be rotated independently into 26 positions; so the three could be hooked together in $26^3 = 17576$ ways.

reflector
Yankees versus Red Sox
ring settings
inverse permutation

There also was a built-in “reflector disk,” which was permanently wired to carry out the following more-or-less random fullswap:

$$\begin{aligned}\rho \;&=\; [\texttt{YRUHQSLDPXNGOKMIEBFZCWVJAT}]\\ &=\; (\texttt{AY})(\texttt{BR})(\texttt{CU})(\texttt{DH})(\texttt{EQ})(\texttt{FS})(\texttt{GL})(\texttt{IP})(\texttt{JX})(\texttt{KN})(\texttt{MO})(\texttt{TZ})(\texttt{VW}).\end{aligned} \qquad (2)$$

For example, Fig. 295 shows two of the ways in which Enigma could be configured with rotors I, II, and III inserted at the right of the reflector. In the right-hand diagram, rotor III has advanced one step from the position that it had at the left.

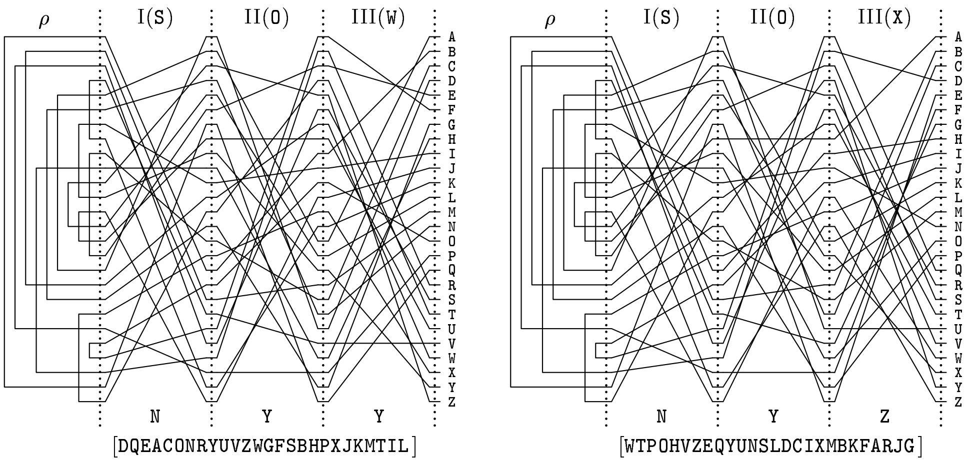


Fig. 295. Two consecutive configurations of the Enigma’s internal network of wires. (Crossing wires don’t actually touch each other.) The “ring settings” are VKC = NYY − SOW.

Five rotors, named I, II, III, IV, and V, were available to be inserted into the machine; the user would choose three of them, and their left-to-right order. Thus there were $5 \cdot 4 \cdot 3 = 60$ ways to mount the disks, and $60 \cdot 26^3 = 1{,}054{,}560$ possible wiring networks such as Fig. 295.

Notice that the endpoints of the wires in Fig. 295 have been labeled A to Z from top to bottom, and that the inter-rotor contacts within the network actually partition the wires into 13 disjoint paths. Each of those paths has, of course, two endpoints. In the left-hand diagram, for example, two of the paths are

$$\texttt{P} \overset{\mathrm{III}}{\text{——}} \texttt{X} \overset{\mathrm{II}}{\text{——}} \texttt{X} \overset{\mathrm{I}}{\text{——}} \texttt{U} \overset{\rho}{\text{——}} \texttt{C} \overset{\mathrm{I}^-}{\text{——}} \texttt{V} \overset{\mathrm{II}^-}{\text{——}} \texttt{F} \overset{\mathrm{III}^-}{\text{——}} \texttt{S}; \qquad (3)$$

$$\texttt{S} \overset{\mathrm{III}}{\text{——}} \texttt{F} \overset{\mathrm{II}}{\text{——}} \texttt{V} \overset{\mathrm{I}}{\text{——}} \texttt{C} \overset{\rho}{\text{——}} \texttt{U} \overset{\mathrm{I}^-}{\text{——}} \texttt{X} \overset{\mathrm{II}^-}{\text{——}} \texttt{X} \overset{\mathrm{III}^-}{\text{——}} \texttt{P}. \qquad (4)$$

(Rotor III maps P to X; rotor II maps X to itself; rotor I maps X to U; the reflector ρ maps U to C; rotor I^-, the inverse permutation of rotor I, maps C to V; and so on.) In the right-hand diagram rotor III has a new permutation III', and we have

$$\texttt{P} \overset{\mathrm{III}'}{\text{——}} \texttt{Y} \overset{\mathrm{II}}{\text{——}} \texttt{N} \overset{\mathrm{I}}{\text{——}} \texttt{S} \overset{\rho}{\text{——}} \texttt{F} \overset{\mathrm{I}^-}{\text{——}} \texttt{D} \overset{\mathrm{II}^-}{\text{——}} \texttt{T} \overset{\mathrm{III}'^-}{\text{——}} \texttt{C}; \qquad (5)$$

$$\texttt{C} \overset{\mathrm{III}'}{\text{——}} \texttt{T} \overset{\mathrm{II}}{\text{——}} \texttt{D} \overset{\mathrm{I}}{\text{——}} \texttt{F} \overset{\rho}{\text{——}} \texttt{S} \overset{\mathrm{I}^-}{\text{——}} \texttt{N} \overset{\mathrm{II}^-}{\text{——}} \texttt{Y} \overset{\mathrm{III}'^-}{\text{——}} \texttt{P}. \qquad (6)$$

letters as numbers
alphabetical arithmetic
cyclic left shift
base permutation
ring settings
plugboard permutation

Let's simplify the notation so that we can see better what's going on. It's convenient to regard letters as numbers, using the equivalences

$$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\ \mathtt{A} & \mathtt{B} & \mathtt{C} & \mathtt{D} & \mathtt{E} & \mathtt{F} & \mathtt{G} & \mathtt{H} & \mathtt{I} & \mathtt{J} & \mathtt{K} & \mathtt{L} & \mathtt{M} & \mathtt{N} & \mathtt{O} & \mathtt{P} & \mathtt{Q} & \mathtt{R} & \mathtt{S} & \mathtt{T} & \mathtt{U} & \mathtt{V} & \mathtt{W} & \mathtt{X} & \mathtt{Y} & \mathtt{Z} \end{matrix}. \tag{7}$$

Then we can do arithmetic with letters, working modulo 26:

$$\mathtt{S}+\mathtt{E}=\mathtt{W};\quad \mathtt{N}+\mathtt{N}=\mathtt{A};\quad \mathtt{N}+\mathtt{O}=\mathtt{B};\quad \mathtt{Z}+\mathtt{Z}=\mathtt{Y};\quad \mathtt{P}-\mathtt{Q}=\mathtt{Z}; \tag{8}$$

and so on. The 26 possible permutations that a rotor R can contribute to an Enigma configuration can be denoted by R(A), R(B), ..., R(Z), and the actual permutations that appear in Fig. 295 are I(S), II(O), III(W), and III(X). Thus, for example, the permutations of the rightmost rotor, which we called III and III′ in (3) and (5), are actually

$$\mathrm{III}(\mathtt{W}) = [\mathtt{OISUCAYWJLNPRTKXZBFDHVGMQE}] \tag{9}$$

$$\begin{aligned} \text{and}\quad \mathrm{III}(\mathtt{X}) &= [\mathtt{ISUCAYWJLNPRTKXZBFDHVGMQEO}]-1 \\ &= [\mathtt{HRTBZXVIKMOQSJWYAECGUFLPDN}]. \end{aligned} \tag{10}$$

To get from the first of these to the second, we've done a cyclic left shift, then subtracted 1 from each letter; this corresponds to rotating the disk by 1 step.

Hence alphabetical arithmetic yields a simple formula for each permutation,

$$\mathrm{R}(x) \;=\; (\mathtt{AB}\ldots\mathtt{Z})^x\, \Pi_{\mathrm{R}}\, (\mathtt{Z}\ldots\mathtt{BA})^x, \tag{11}$$

where Π_{R} is the "base permutation" for rotor disk R:

$$\begin{aligned} \Pi_{\mathrm{I}} &= [\mathtt{DBYJRKALSNTVOUPMZEIWCXFHQG}]; \\ \Pi_{\mathrm{II}} &= [\mathtt{DMPSWGCROHXLBUIKTAQJZVEYFN}]; \\ \Pi_{\mathrm{III}} &= [\mathtt{YWUSFHJLNPGTVXBZDRCIMAKEOQ}]; \\ \Pi_{\mathrm{IV}} &= [\mathtt{KYHXNBDVJWATSCMRUIELFPZQOG}]; \\ \Pi_{\mathrm{V}} &= [\mathtt{VZBRGITYUPSDNHLXAWMJQOFECK}]. \end{aligned} \tag{12}$$

To get $\mathrm{R}(x)$, we shift the base permutation circularly by x steps, then subtract x.

Figure 295 illustrates another important feature, an arbitrary string of three letters called the *ring settings*. To enhance cryptographic security, Enigma's users would configure the machine by specifying the current position $p_0p_1p_2$ (NYY in our example) as well as the ring settings $q_0q_1q_2$ (VKC); then the current permutations for rotors I, II, III would be respectively $\mathrm{I}(o_0)$, $\mathrm{II}(o_1)$, $\mathrm{III}(o_2)$, where

$$o_0o_1o_2 \;=\; p_0p_1p_2 \;-\; q_0q_1q_2, \tag{13}$$

(namely SOW). There also was a *plugboard permutation*, τ, for even more security; we'll discuss the plugboard later. Generalizing (3), (4), (5), and (6), the actual permutation π_k used in the polyalphabetic cipher (1) was then

$$\pi_k \;=\; \tau\, \mathrm{III}(o_2)\, \mathrm{II}(o_1)\, \mathrm{I}(o_0)\, \rho\, \mathrm{I}(o_0)^-\, \mathrm{II}(o_1)^-\, \mathrm{III}(o_2)^-\, \tau \tag{14}$$

for the kth setting of $o_0o_1o_2$, assuming that the rotors were I, II, III in that order.

fast rotor
slow rotor
valid sequence+

The rightmost rotor always advanced by one step, just before enciphering a letter. Therefore it was called the "fast" rotor. The middle rotor would occasionally advance too, so as not to repeat a permutation π_k already used; but it advanced much more slowly. And the remaining rotor, next to the reflector, was called the "slow" rotor because it hardly ever moved.

An ingenious, but peculiar,* mechanism governed the rotor movements: The slow rotor advanced if and only if $p_1 = \mathtt{Z}$; the middle rotor advanced if and only if $p_1 = \mathtt{Z}$ *or* $p_2 = \mathtt{Z}$. Thus the actual change at each step was

$$(p_1 = \mathtt{Z}?\ o_0{+}{+},\ o_1{+}{+},\ o_2{+}{+},\ p_0{+}{+},\ p_1{+}{+},\ p_2{+}{+} : \\ p_2 = \mathtt{Z}?\ o_1{+}{+},\ o_2{+}{+},\ p_1{+}{+},\ p_2{+}{+} :\ o_2{+}{+},\ p_2{+}{+}), \qquad (15)$$

where $x{+}{+}$ means increase x by 1, mod 26; the ring settings $q_0q_1q_2$ didn't change.

A surprising consequence of rule (15) is that the middle rotor will sometimes turn twice in succession. For example, the next steps after Fig. 295 go from position `NYZ` to `NZA`, and then to `OAB`. (See exercise 8.) Another consequence, explained in exercise 12, is that the sequence of permutations π_1, π_2, ... in (1) must eventually repeat with a period of length $26{\cdot}25{\cdot}26 = 16900$, not $26^3 = 17576$ as expected! In practice, however, Enigma was not used for messages with more than 250 characters; hence its slow rotor advanced at most once per message.

To fix the ideas, let's try to figure out all ways to configure Enigma so that it will transform `CHAOS` into `ORDER`, without using the plugboard. In other words, we'll try to choose three rotors, together with a starting position $p_0p_1p_2$ and ring settings $q_0q_1q_2$, so that the plaintext `CHAOS` yields the ciphertext `ORDER`.

Any solution to this problem leads immediately to 25 other solutions, because we can add any constant to both p_0 and q_0 without changing the behavior of the machine. Furthermore, we can often add a constant to both p_1 and q_1, and/or to both p_2 and q_2, provided that the rotors will move in the same way. (See exercise 13.)

So let's formulate the problem more carefully: We must choose three distinct rotors, R_0, R_1, R_2, and a sequence of five offsets $o_{k0}o_{k1}o_{k2}$ for $1 \le k \le 5$, such that

$$\begin{aligned} &\text{configuration } \mathrm{R}_0(o_{10})\ \mathrm{R}_1(o_{11})\ \mathrm{R}_2(o_{12}) \text{ maps } \mathtt{C} \mapsto \mathtt{O},\\ &\text{configuration } \mathrm{R}_0(o_{20})\ \mathrm{R}_1(o_{21})\ \mathrm{R}_2(o_{22}) \text{ maps } \mathtt{H} \mapsto \mathtt{R},\\ &\text{configuration } \mathrm{R}_0(o_{30})\ \mathrm{R}_1(o_{31})\ \mathrm{R}_2(o_{32}) \text{ maps } \mathtt{A} \mapsto \mathtt{D},\\ &\text{configuration } \mathrm{R}_0(o_{40})\ \mathrm{R}_1(o_{41})\ \mathrm{R}_2(o_{42}) \text{ maps } \mathtt{O} \mapsto \mathtt{E},\\ &\text{configuration } \mathrm{R}_0(o_{50})\ \mathrm{R}_1(o_{51})\ \mathrm{R}_2(o_{52}) \text{ maps } \mathtt{S} \mapsto \mathtt{R}; \end{aligned} \qquad (16)$$

and furthermore such that

$$o_{10}o_{11}o_{12},\ o_{20}o_{21}o_{22},\ o_{30}o_{31}o_{32},\ o_{40}o_{41}o_{42},\ o_{50}o_{51}o_{52} \text{ is a } \textit{valid sequence},$$

namely a sequence of offsets that can actually occur for some setting of the position $p_0p_1p_2$ and rings $q_0q_1q_2$, using (15) after beginning with initial offsets $o_{00} \leftarrow (p_0 - q_0) \bmod 26$, $o_{01} \leftarrow (p_1 - q_1) \bmod 26$, $o_{02} \leftarrow (p_2 - q_2) \bmod 26$.

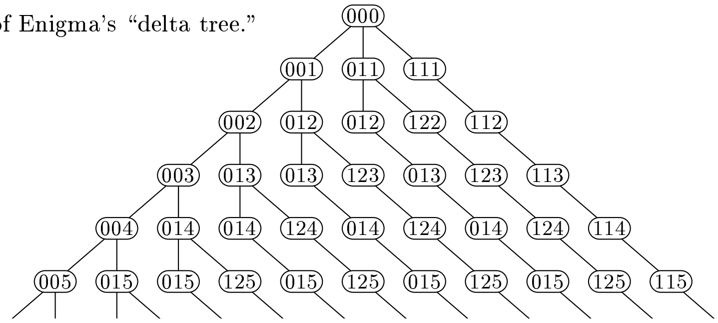
* In fact, Enigma's actual mechanism was even more peculiar. But it was equivalent to the simplified rule described here, except for a change in notation. See exercise 10.

delta tree
brute force
XYZZY

The key concept here is the notion of a valid sequence. In any valid sequence we always have $o_{(k+1)0}\,o_{(k+1)1}\,o_{(k+1)2} - o_{k0}\,o_{k1}\,o_{k2} \equiv (0,0,1)$ or $(0,1,1)$ or $(1,1,1)$, modulo 26. But that necessary condition is far from sufficient. For example, $000, 011, 022$ is not valid. Neither is $111, 122, 123, 234$.

The true nature of valid sequences is captured by an interesting tree structure that we shall call the *delta tree*, illustrated in Fig. 296. The rule for constructing the first 25 levels of this tree, after the three nodes on level 1 have been specified, is quite simple: Each node labeled $00l$ on level l has two children, labeled $00(l{+}1)$ and $01(l{+}1)$. Each node labeled $01l$ has a child labeled $01(l{+}1)$, and also a second child labeled $12(l{+}1)$ if its parent was labeled $00(l{-}1)$. Each node labeled $1xl$ has one child, labeled $1x(l{+}1)$.

Fig. 296. The top levels of Enigma's "delta tree."



Let α_j stand for $o_{j0}\,o_{j1}\,o_{j2}$. By considering (15) carefully, we can readily verify that *a sequence* $\alpha_1, \alpha_2, \ldots, \alpha_k$ *is valid if and only if* $\alpha_1 - \alpha_1$, $\alpha_2 - \alpha_1$, $\ldots$, $\alpha_k - \alpha_1$, *mod 26, is a path in the delta tree.* (See exercise 15.)

Given the delta tree, we can now solve our problem quickly by brute force: For each of the 60 choices of R_0, R_1, and R_2, each of the 26^3 choices of α_1, and each of the nine paths in the delta tree from the root to level 4, we test to see if all five conditions of (16) are satisfied. That's less than 10 million cases to try.

It turns out that there's a unique solution to the problem. We must install rotors IV, V, I (in that order); we must have the offsets $\alpha_1 = $ `ATX` to encode `C` as `O`; and we must follow the path 000, 001, 012, 123, 124 in the delta tree.

OK! In order to follow that path, we can start at position $p_0p_1p_2 = $ `AYX`, because enciphering will then take place at positions `AYY`, `AYZ`, `AZA`, `BAB`, `BAC`, matching the desired path. Then we set the rings to `AFB`, so that enciphering will take place at offsets $\alpha_1 = $ `ATX`, $\alpha_2 = $ `ATY`, $\alpha_3 = $ `AUZ`, $\alpha_4 = $ `BVA`, $\alpha_5 = $ `BVB`. (Alternatively, we could start at `BYX`, with rings at `BFB`, etc.)

Suppose now that we've intercepted a secret message more than 100 characters long, enciphered by Enigma without a plugboard. The first five characters of this ciphertext might be, say, `XYZZY`. If we happen to know that the first five characters of the secret plaintext are `CHAOS`, then we'll know the entire message! Because it takes less than a second to discover, by examining 9.5 million cases, that the only Enigma configuration that maps `CHAOS` $\mapsto$ `XYZZY` is equivalent to one that uses rotors I, II, IV and starts at position `YYY` with rings `ARQ`.

Bombe–
involution
Turing
Welchman

The Bombes. So far we haven't discussed Enigma's plugboard, which was introduced in 1928. The plugboard scrambled the encryptions further by applying an additional permutation τ, making them enormously more difficult to crack. This permutation τ was an *involution*, meaning that it interchanged up to 13 pairs of letters while leaving the others unchanged. (Equivalently, τ was equal to its own inverse.) The user could plug two letters like `E` and `O` together, causing them to be swapped just before being enciphered and then to be swapped again at the end (see Eq. (14)). If Enigma was set up to transform `CHAOS` to `ORDER` without the plugboard, it would transform `CHAES` to `ERDOR` when `E` was plugged to `O`.

The number of possible plugboard settings is the number t_{26} that we computed in Eq. 5.1.4–(40); and it's huge: 532,985,208,200,576. Even if we limit ourselves to 10 plugged pairs, as the Germans eventually did, we'll have $26!/(2^{10}6!\,10!) = 150{,}738{,}274{,}937{,}250$ choices for τ. The number of possible permutation sequences $\pi_1, \pi_2, \ldots$ in (1) has thereby increased from millions to sextillions.

Yet the plugboard was crackable too, with the help of mathematics and a little bit of luck. Extensive computations were needed; but Alan Turing and Gordon Welchman devised a special-purpose electromechanical machine called a *bombe*, which could do the work in parallel.

To illustrate their method, let's work on a toy problem: *Is there an Enigma configuration (with plugboard) that will encipher `POTPOURRI` to `OFPUZZLES`?*

For starters, let's define $\binom{27}{2} = 351$ Boolean variables `AA`, `AB`, `AC`, …, `AZ`, `BB`, `BC`, …, `BZ`, `CC`, …, `XZ`, `YY`, `YZ`, `ZZ`; there's one variable for each pair of letters xy with $\mathtt{A} \le x \le y \le \mathtt{Z}$. Their meaning, as we study the toy problem, is

$$xy \;=\; [x\tau = y] \;=\; [\text{“the plugboard maps } x \mapsto y\text{”}]. \qquad (17)$$

Since τ is an involution, we have $x\tau = y$ if and only if $y\tau = x$. Therefore we can refer to variable xy also by the name yx; for example, `OK` means the same thing as `KO`. It follows that, if x is any letter, exactly one of the 26 Boolean variables $x\mathtt{A}$, $x\mathtt{B}$, …, $x\mathtt{Z}$ will be true, in any solution to our problem.

Suppose now that Enigma has been configured with the rotor permutations I(`S`) II(`O`) III(`W`), as in the left half of Fig. 295. Without a plugboard, we know that it will produce the permutation

$$\pi \;=\; [\mathtt{DQEACONRYUVZWGFSBHPXJKMTIL}]; \qquad (18)$$

in other words, $\mathtt{A}\pi = \mathtt{D}$, …, $\mathtt{Z}\pi = \mathtt{L}$. What plugboard settings will cause `P` to be enciphered as `O`? Algebraically, what involutions τ satisfy the equation $\mathtt{P}\tau\pi\tau = \mathtt{O}$? A moment's thought shows that the 26 conditions

$$\mathtt{PA} \equiv \mathtt{OD},\ \mathtt{PB} \equiv \mathtt{OQ},\ \mathtt{PC} \equiv \mathtt{OE},\ \mathtt{PD} \equiv \mathtt{OA},\ \ldots,\ \mathtt{PO} \equiv \mathtt{OF},\ \ldots,\ \mathtt{PZ} \equiv \mathtt{OL} \qquad (19)$$

are necessary and sufficient. For example, if τ takes $\mathtt{P} \mapsto \mathtt{A}$, then it must also take $\mathtt{O} \mapsto \mathtt{D}$, because the fullswap π takes $\mathtt{A} \mapsto \mathtt{D}$. Conversely, if τ takes $\mathtt{O} \mapsto \mathtt{D}$, then it must also take $\mathtt{P} \mapsto \mathtt{A}$. (We have $\mathtt{P}\tau\pi\tau = \mathtt{O}$ if and only if $\mathtt{P}\tau\pi = \mathtt{O}\tau$.)

One of the conditions in (19) is that $\mathtt{PO} \equiv \mathtt{OF}$, because $\mathtt{O}\pi = \mathtt{F}$. But `PO` and `OF` can't both be true! Therefore they must both be false.

equivalence algorithm
union-find algorithm
viable classes
union-find algorithm
bitmap

Similarly, the configuration I(S) II(O) III(X) at the right of Fig. 295 produces the fullswap

$$\pi' \;=\; [\mathtt{WTPOHVZEQYUNSLDCIXMBKFARJG}]; \tag{20}$$

therefore it will encipher $\mathtt{O}\mapsto\mathtt{F}$ with plugboard τ if and only if

$$\mathtt{OA}\equiv\mathtt{FW},\ \mathtt{OB}\equiv\mathtt{FT},\ \mathtt{OC}\equiv\mathtt{FP},\ \mathtt{OD}\equiv\mathtt{FO},\ \ldots,\ \mathtt{OO}\equiv\mathtt{FD},\ \ldots,\ \mathtt{OZ}\equiv\mathtt{FG}. \tag{21}$$

So we're going to get $\mathtt{POTPOURRI}\mapsto\mathtt{OFPUZZLES}$, starting with $\pi_1\pi_2=\pi\pi'$, only if both (19) and (21) hold. (See exercise 22.) An *equivalence algorithm* such as Algorithm 2.3.3E, also known as a "union-find algorithm," is able to deduce all consequences of a large number of simple equivalences such as (19) and (21).

Let's push on and assume that our plugboard works properly also for $\pi_3=[\mathtt{URYIMXOZDQTWESGVJBNKAPLFCH}]$ and $\pi_4=[\mathtt{YKQTNRXPLMBIJEZHCFUDSWVGAO}]$, the two permutations that come next after Fig. 295. (See exercise 8.) We'll need both

$$\mathtt{TA}\equiv\mathtt{PU},\ \mathtt{TB}\equiv\mathtt{PR},\ \mathtt{TC}\equiv\mathtt{PY},\ \mathtt{TD}\equiv\mathtt{PI},\ \ldots,\ \mathtt{TO}\equiv\mathtt{PG},\ \ldots,\ \mathtt{TZ}\equiv\mathtt{PH} \tag{22}$$

$$\text{and } \mathtt{PA}\equiv\mathtt{UY},\ \mathtt{PB}\equiv\mathtt{UK},\ \mathtt{PC}\equiv\mathtt{UQ},\ \mathtt{PD}\equiv\mathtt{UT},\ \ldots,\ \mathtt{PO}\equiv\mathtt{UZ},\ \ldots,\ \mathtt{PZ}\equiv\mathtt{UO} \tag{23}$$

if they're going to take $\mathtt{T}\mapsto\mathtt{P}$ and $\mathtt{P}\mapsto\mathtt{U}$. That gives us lots of nice equivalences in addition to (19) and (21); and the union-find algorithm now tells us that only ten viable classes of equivalent variables satisfy all of the necessary conditions:

$$\begin{gathered}\{\mathtt{AF,ET,JU,MP,OW}\},\{\mathtt{AU,CT,FQ,IO,PY}\},\{\mathtt{BP,FI,KU,OQ,RT}\},\{\mathtt{BU,FF,KP,OV,TT}\},\{\mathtt{CP,EO,FH,QU,TY}\},\\ \{\mathtt{DT,FJ,IP,LU,OY}\},\{\mathtt{EU,FZ,GO,NP,ST}\},\{\mathtt{FG,IU,LP,OZ,TW}\},\{\mathtt{FM,HU,OS,PP,TV}\},\{\mathtt{FS,LT,MO,PW,UV}\}.\end{gathered} \tag{24}$$

Every other equivalence class either (i) contains two variables with a common letter, hence its elements must all be false; or (ii) is a singleton, whose lone variable involves only the letters $\{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{E},\mathtt{G},\mathtt{H},\mathtt{I},\mathtt{J},\mathtt{K},\mathtt{L},\mathtt{M},\mathtt{N},\mathtt{Q},\mathtt{R},\mathtt{S},\mathtt{V},\mathtt{W},\mathtt{X},\mathtt{Y},\mathtt{Z}\}$ that don't appear in either POTP or OFPU. Classes of type (i) are called "unviable." Classes of type (ii) are unimportant (see exercise 23).

Any decent union-find algorithm can readily be extended so that it knows which of the current equivalence classes are viable. We simply maintain a bitmap for each class, representing all of the letters of the variables in that class. The union of two viable classes becomes unviable if and only if their bitmaps intersect.

The next permutation after π_4 is $\pi_5=[\mathtt{JLYGRUDTOAXBQVIZMEWHFNSKCP}]$. It implies, among other things, that $\mathtt{OQ}\equiv\mathtt{ZM}$; consequently $\{\mathtt{BP},\mathtt{FI},\mathtt{KU},\mathtt{OQ},\mathtt{RT}\}$, the third class of (24), joins the singleton $\{\mathtt{MZ}\}$ and becomes $\{\mathtt{BP},\mathtt{FI},\mathtt{KU},\mathtt{MZ},\mathtt{OQ},\mathtt{RT}\}$. But $\mathtt{OZ}\equiv\mathtt{ZP}$ causes $\{\mathtt{FG},\mathtt{IU},\mathtt{LP},\mathtt{OZ},\mathtt{TW}\}$ to become unviable. Exercise 24 shows that two other classes of (24) also die because of π_5; furthermore, all ten classes are wiped out when we proceed to the next permutation, π_6.

Thus we cannot even set the plugs so that POTPOU becomes OFPUZZ, when those six letters are enciphered with the fullswaps π_1, π_2, …, π_6 that begin in Fig. 295. On the other hand there are plenty of complete solutions when we start in a different state. Indeed, three independent solutions are discovered already when we insist that rotors I, II, III are used, and that all nine π_k have the first rotor permutations fixed at I(A) and II(A). One of them starts Enigma at position AAG, with ring settings AAA and plugboard (AT)(BI)(CO)(EM)(FG)(LY)(QR)(SW)(UZ).

secret message–
clues
ENIGMATICALLY
cryptographers
crib
delta tree
viable equivalence classes
equivalence classes

A secret message. Now that we've solved the toy problem, we're ready to tackle a *real* challenge:

```
WMGQR YVGTM UJVJP ABVLX RPMLD RLSGV HIIHX ECSWY ZVAPI
LJWDG LSHKI SUZQW ZIHOG GCCFT ZQTBK OOOIG DPDZD SCUJB
WXFXK JRENU FFRIP YKTTA DQZLH GCLDZ QAEGI                (25)
```

What is the plaintext that lies behind this puzzling 125-character ciphertext?

We're given only three clues: (1) The message is in English. (2) It contains the word ENIGMATICALLY. (3) It was enciphered with Enigma, after deleting spaces and punctuation marks (although each '.' was replaced with 'X').

Professional cryptographers typically need to exploit partial information such as clue (2); a substring of the plaintext that is known or guessed to be present is called a "crib." Our crib is unfortunately quite short — only 13 letters long — but we'll see what we can do. Our approach will be to break the full problem into subproblems analogous to the toy problem, with one subproblem for each of the $126 - 13 = 113$ places where the crib might appear. Subproblem 0 is to find all ways that Enigma might encipher ENIGMATICALLY $\mapsto$ WMGQRYVGTMUJV; and subproblem 112, the last one, is to find all ways that it might take ENIGMATICALLY $\mapsto$ ZLHGCLDZQAEGI. For each solution to a subproblem, we'll find the plaintext that produces the entire 125-character ciphertext. And finally, we'll decide which of those plaintexts is in English.

Some of the subproblems are trivial. For example, subproblem 2 asks us to map ENIGMATICALLY $\mapsto$ GQRYVGTMUJVJP; but that's impossible, because a fullswap cannot take T $\mapsto$ T. Altogether 50 of the 113 subproblems are immediately ruled out in this way.

Exercise 25 discusses an important improvement to the way that we proved the impossibility of suitable plugboard settings in the toy problem. Even with that improvement, however, a large number of possibilities remain plausible and require further investigation.

Indeed, we're faced with $113-50 = 63$ nontrivial subproblems, each of which must consider 60 choices of rotors and 26^3 wheel rotations, as well as 25 different paths from the root of the delta tree to level 12; that's 1,660,932,000 possible scenarios for which we must test whether or not a correct plugboard exists! The improved filter of exercise 25 whittles this number down to "only" 110,860 cases, but we've still got plenty of work ahead of us after that 15000-fold reduction.

Consider, for example, the very first case that happens to get through the filter. It enciphers the E of ENIGMATICALLY with the rotor permutations I(A), II(A), III(E), after which the machine follows the delta tree path 000, 001, 012, 013, ..., 01(12). Furthermore, this polyalphabetic cipher is part of subproblem 69, which maps ENIGMATICALLY $\mapsto$ TZQTBKOOOIGDP. When the bombe method is applied to that case, it finds the following 13 viable equivalence classes:

$$\{\mathtt{AA,CK,DU,EO,GW,IY,JT,LZ,NV,PP,QS}\};$$
$$\{\mathtt{BR,MX}\},\{\mathtt{BX,MR}\}; \qquad (26)$$
$$\{\mathtt{FF}\},\{\mathtt{FH}\},\{\mathtt{FR}\},\{\mathtt{FX}\},\{\mathtt{HH}\},\{\mathtt{HR}\},\{\mathtt{HX}\},\{\mathtt{RR}\},\{\mathtt{RX}\},\{\mathtt{XX}\}.$$

exact cover problem
XC problem
SAT solver

(Remember that each of these pairs, `AA`, `CK`, ..., `XX` is a Boolean variable, meaning that the plugboard fixes `A`, swaps `C` $\leftrightarrow$ `K`, ..., fixes `X`.) Therefore a plugboard setting is valid if and only if it corresponds to the solution of an exact cover problem, in which the 26 items $\{$`A`,`B`,...,`Z`$\}$ must be covered by the options

‘`A C K D ... S`’, ‘`B R M X`’, ‘`B X M R`’, ..., ‘`R X`’, ‘`X`’. (27)

(We obtain these options from the bitmaps of the equivalence classes. The second and third options are actually identical here, because they cover the same letters; but they correspond to different classes, and we must choose one of them.)

That XC problem obviously has just four solutions: We must choose the first option, whose letters don't appear elsewhere. Then we must choose either the second option or the third, thus either swapping `B` $\leftrightarrow$ `R` and `M` $\leftrightarrow$ `X` or `B` $\leftrightarrow$ `X` and `M` $\leftrightarrow$ `R`, with pairs of plugs. And finally, we must choose either ‘`F H`’ (and swapping `F` $\leftrightarrow$ `H`) or both ‘`F`’ and ‘`H`’ (and keeping those letters fixed).

Some of the 110,860 exact cover problems that arise in this way are unsolvable, of course. The best way to find all of their solutions may actually be to use the simple SAT solver of Algorithm 7.2.2.2B, together with exercise 7.2.2.2–125, instead of using an XC solver. (See exercise 27.) It turns out that 15,392 of those problems have no solution, and we needn't consider them further. On the other hand, two of the cases actually yield 2620 different plugboard solutions — each of which might have been used to produce (25), for all we know.

Knowledge of the plugboard is clearly critical; yet our study of the first feasible case is still only half done. We must also determine the set of starting positions and ring settings that will cause the Enigma's rotor permutations to follow the precise sequence I(`A`) II(`A`) III(`E`), I(`A`) II(`A`) III(`F`), I(`A`) II(`B`) III(`G`), ..., I(`A`) II(`B`) III(`Q`), at times $69+1$, $69+2$, $69+3$, ..., $69+13$. Exercise 28 explains how to do that. It turns out that exactly five of the start-and-ring configurations have the desired behavior, namely

`AAG` and `ACU`; `AYG` and `BBU`; `AXG` and `BAU`; `AVG` and `AXU`; `AUG` and `AWU`. (28)

(We may assume without loss of generality that the slow rotor always starts at `A`.)

Combining each of the setups in (28) with each of the plugboards obtained from (26) and (27) gives us 20 distinct possibilities for a plaintext that might have led to the secret message (25). For example, the first of these is obtained by configuring Enigma with parameters

I II III `AAG ACU CK DU EO GW IY JT LZ NV QS BR MX FH`, (29)

where this notation names the rotors first, then the start position and ring settings, and then the plugboard pairs. And we can recover that plaintext by feeding the 125 characters of (25) into the machine. The result, ta da, is

`EAMJDTMPXHRUWRJMMKXJMNYDRQNZPPRPGUDFAHSPGPQIQVCKFUGMBGEYASHLDTAIP`
`BIJOENIGMATICALLYHHXYBDYCRSVDENWRZQURFEEVDVOEEBRKKCMFBFIXZCU.` (30)

Hmmm. This doesn't look at all like the ultrasecret plaintext that we're seeking. But it certainly does have ‘`ENIGMATICALLY`’ in the correct position, following 69 characters of gibberish. Our theories are working!

Let's summarize where we stand: We began with 1,660,932,000 possible scenarios, and reduced that to 110,860 cases that yield decent equivalence classes. We've shown that exactly 20 potential plaintexts belong to case 1; and we know how to generate them quickly. These are all of the plaintexts that satisfy clues (2) and (3) and belong to case 1. So we're on our way, with 110,859 cases to go.

And we're in luck, because the other cases can all be finished in a reasonable time. Most of them generate fewer than 20 potential plaintexts (although one of them actually produces 35,728). All told, the methods we've discussed will give us a complete collection of all plaintexts that satisfy clues (2) and (3); it turns out that there are exactly 3,331,188 of them.

That leaves us with clue (1). How can we determine the one message that happens to be in English, out of more than three million candidates? There isn't time to look at all of them with human eyes.

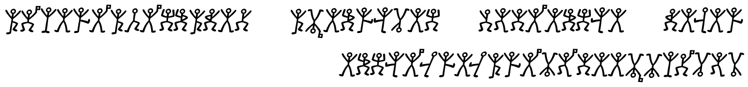
The answer is to use statistics. Message (30) is clearly not English, because it doesn't have the right statistics. The simplest statistics used to solve traditional ciphers are based on single-letter "monogram" frequency counts (see exercise 1); (30) has too many `Q`'s and not enough `T`'s. But we can do a lot better with *five-gram statistics*, namely the $26^5 = 11{,}881{,}376$ relative frequencies with which each possible five-letter substring is typically present.

The Internet contains excellent collections of n-gram statistics. But those stats don't ignore spaces and punctuation between words, as specified in clue (3) (or at least they didn't when this section was first drafted in 2023). So the author went to his electronic source files for Volume 1 of *The Art of Computer Programming* and changed all letters to uppercase, removing not only spaces and punctuation but also all of the numerals, math formulas, and computer programs, etc. From the resulting string of nearly a million characters, in the online file `VOL1TEXT`, it's now easy to obtain the 26^5 counts that are needed for clue (1). (For example, the tallies for `CHAOS`, `ORDER`, and `XYZZY` are respectively 0, 739, and 0.)

Each of our 3,331,188 candidates can therefore be given a score, namely the sum of the tallies of its 121 5-letter substrings. The score of (30) is a measly 538; it picks up most of that from substrings of `ENIGMATICALLY`, then gets 8 more from `ALLYH` and another from `SVDEN`. The minimum possible score, 529, occurs when `ENIGMATICALLY` accounts for *all* of the nonzero tallies. (Exactly 209,729 of the candidates are maximally far from English in this sense.)

The five-gram strategy works like a charm in this application: Only 867 of the candidates have a score exceeding 2000; only 9 of those 867 also score more than 6000; and the top two scores are 7062 and 8000, well separated.

Thus there's a clear winner, which is revealed in exercise 29.



— ARTHUR CONAN DOYLE, *The Adventure of the Dancing Men* (1903)

— WINSTON CHURCHILL, BBC Home Service Broadcast (1 October 1939)

statistics
monogram
five-gram statistics
substring
Internet
n-gram statistics
author
Volume 1
The Art of Computer Programming
online
`VOL1TEXT`
DOYLE
Dancing Men
CHURCHILL

The key is that a good puzzle is a gem, a thing of beauty.
I would argue that the ability to appreciate *a puzzle,*
and the willingness to put effort into solving it, tells us a great deal.

— PETER WINKLER, Preface to *Mathematical Puzzles* (2021)

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 revealed his 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 $b \ne 10$. 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 35), because of more-or-less random coincidences. For example, when $m = 100$ there's a unique third solution,

$$\begin{aligned}x = 26\,56162\,29619\,33010\,98036\,76416\,71003\,29792\,07874\,84348\,54147&\\ 71766\,93876\,28693\,32047\,88451\,13744\,80147\,98509\,42958 = \pi_{100}\,x,&\qquad(31)\end{aligned}$$

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 31 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 permitted.* If we renumber the subscripts by sorting the digits into order, such a multicombination is nothing more nor less than a solution to

$$9 \ge x_m \ge \cdots \ge x_1 \ge x_0 \ge 0. \qquad (32)$$

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.

WINKLER
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

backtrack algorithm++
data structures+

With these ideas we could find (31) 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 (32) 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 (31). 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* 9s *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 (32) becomes

$$9 \ge x_1 \ge \cdots \ge x_m \ge x_{m+1} \ge 0. \tag{33}$$

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 $\Sigma_l = (\Sigma_{lm} \ldots \Sigma_{l0})_{10}$, $a_l = (a_{lm} \ldots a_{l0})_{10}$, 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 $q = q_l = e_{lc} \dot{-} d_{lc} = \max(0, e_{lc} - d_{lc})$ to track the number of yet-unseen digits of the common prefix.

monus
registers

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

$$\Sigma_l = \sum_{k=0}^{9} \max(d_{lk}, e_{lk}) \cdot k^m \tag{34}$$

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 $[a \mathinner{..} b]$, 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 $m \ge 2$, 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 (33). 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 $l \leftarrow 1$, $q \leftarrow 0$, $r \leftarrow m+1$, $t \leftarrow m$, $a \leftarrow 0$, $b \leftarrow c \leftarrow 9$, $\Sigma_1 \leftarrow 0$, $d_1 \leftarrow e_1 \leftarrow (0, \ldots, 0)$, 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 34, 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 step P7 if $q > 0$. Otherwise set $r \leftarrow r_l$, $t \leftarrow t_l$, $a \leftarrow (t \ge 0?\ a_{lt}\colon 9)$, $b \leftarrow (t \ge 0?\ b_{lt}\colon 9)$, $c \leftarrow x_l$, and go back to P6. ▌

Exercise 34 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 $x_l \le c$. 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 $m \le 100$, but it appears to grow roughly as $m^{7.5}$ when m increases.

Historical notes: G. H. Hardy, in *A Mathematician's Apology* (1940), §15, dismissed questions like this as "tiresome"; see the later apology by 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

$$\begin{array}{ccccccc}
 & & & & \square & \square & \square \\
 & \times & & \square & \square & \square & \square \\
\cline{3-7}
 & & & \square & \square & \square & 7 \\
 & & \square & \square & \square & \square & \\
 & \square & \square & \square & \square & & \\
 & \square & \square & \square & & & \\
\hline
 & 7 & 7 & 7 & 7 & 7 & 7
\end{array}
\qquad\qquad \square \ne 7. \qquad\qquad (35)$$

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 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 (35).

Junya Take introduced a particularly appealing class of bug-eaten multiplications in the *Journal of Recreational Mathematics* **35** (2006), 63, when he

Historical notes
Hardy
Knuth
Deimel
Jones
Skeleton
multiplication
Hidden-digit puzzles
Matsunaga
Nakane
Kantō Sampō
Workman
mushikuizan
Maruo
Yamamoto
long division
Take

submitted the following puzzle in honor of the editor Steven Kahan:

Kahan
alphabet
Take
puzzles, invention of

$$\begin{array}{cccccccccccc}
 & & & & & \Box & \Box & \Box & \Box & \Box & \Box & \Box \\
 & & & & & & \times & \Box & \Box & \Box & \Box & \Box \\
\hline
 & & & & & \Box & \Box & \Box & \Box & \Box & \Box & \Box \\
 & & & \Box & \Box & \mathrm{K} & \Box & \mathrm{K} & \Box & \Box & \Box & \\
 & & \Box & \Box & \Box & \mathrm{K} & \mathrm{K} & \Box & \Box & \Box & & \\
 & \Box & \Box & \Box & \Box & \mathrm{K} & \Box & \Box & \Box & & & \\
\Box & \Box & \Box & \Box & \Box & \mathrm{K} & \mathrm{K} & \Box & & & & \\
\hline
\Box & \Box & \Box & \Box & \Box & \mathrm{K} & \Box & \mathrm{K} & \Box & \Box & \Box & \Box
\end{array} \qquad \Box \neq \mathrm{K}. \tag{36}$$

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 $\Box$ 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:

$$\begin{array}{cccccccccccc}
 & & & & & 9 & 1 & 7 & 5 & 1 & 4 & 4 \\
 & & & & & & \times & 7 & 2 & 4 & 6 & 1 \\
\hline
 & & & & & 9 & 1 & 7 & 5 & 1 & 4 & 4 \\
 & & & 5 & 5 & 0 & 5 & 0 & 8 & 6 & 4 & \\
 & & 3 & 6 & 7 & 0 & 0 & 5 & 7 & 6 & & \\
 & 1 & 8 & 3 & 5 & 0 & 2 & 8 & 8 & & & \\
6 & 4 & 2 & 2 & 6 & 0 & 0 & 8 & & & & \\
\hline
6 & 6 & 4 & 8 & 4 & 0 & 1 & 0 & 9 & 3 & 8 & 4
\end{array} \qquad \mathrm{K} = 0. \tag{37}$$

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 (36) 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

$$d_5 = d_3 = e_4 = e_3 = f_3 = g_2 = g_1 = h_6 = h_4 = 0 \tag{38}$$

while all other digits a_j, b_j, $\ldots$, h_j are nonzero. Notice that c, d, e, f, g are distinct, because of (38); 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, $\ldots$, 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

$$6751,\ 3502,\ 0253,\ 7004,\ 3755,\ 0506,\ 7257,\ 4008,\ 0759. \tag{39}$$

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

$$b_0 \in \{1,5,7\}; \quad b_1, b_2, b_3 \in \{3,9\}; \quad b_4 \in \{4,8\}. \tag{40}$$

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

$$5143,\ 0286,\ 5429,\ 0572,\ 5715,\ 0858,\ 6001,\ 1144,\ 6287; \tag{41}$$

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

$$25143,\ 50286,\ 75429,\ 00572,\ 25715,\ 50858,\ 76001,\ 01144,\ 26287, \tag{42}$$

and our choices are rapidly shrinking: Only eight possible settings of b remain, with $b_0 \in \{1,3,5,9\}$, $b_1, b_3 \in \{2,6\}$, $b_2 = 4$, and $b_1 \neq b_3$. 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 $h_4 = 0$. 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 (37) 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, $\ldots$ 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 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 $* = 1$, 3, 4, 5, 7, 8, and 9. But none of them have a unique skeleton. There *is* a further extension, 159175144×72461, whose skeleton is unique; but it doesn't make an especially good puzzle, because it involves more digits than (36). Therefore we are well advised to *stop* trying to extend a, after a solution has been found, and to concentrate on the shortest solutions.

sort
data structures

When we limit this method to multiplicands a that eschew extensions and 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 are also unusable, because they share a shape with a non-shortest solution. But the other three do make rather nice puzzles, namely (i) 9175144×72461 and $\mathrm{K} = 0$, which is (36); (ii) 9783376×83692 and $\mathrm{K} = 0$, whose skeleton is only one digit larger than (36); and the surprising (iii) 324856×72461 and $\mathrm{K} = 9$, whose skeleton is eight digits *shorter* than any of the others. Check it out!

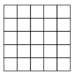
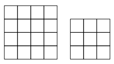
Further possibilities arise when we allow zeros in the multiplier, as discussed in exercise 54. Exercise 56 is devoted to the interesting question of how to design efficient data structures for this computation.

* * *

Discrete dissections. 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.

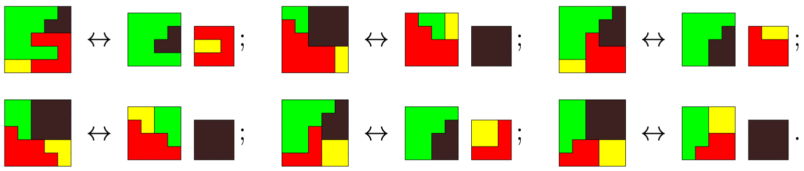
Discrete dissections
pixels
dissection puzzle
Loyd

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

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

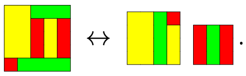
if 1 is the lightest color and 4 is the darkest. For example, pixel $(1,2)$ of A is the tail of a 'P'; it is mapped into $(1,2)\phi_3=(2,3)\sigma_{1,-2}=[3,1]$ within B.

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 '$\boxed{\ \ 50\,61}$'; 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 '$\boxed{23\ \ \ 40}$' 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

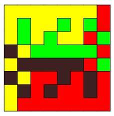
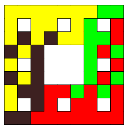
 (56)

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 184,939 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 "Sierpiński carpet" [W. Sierpinski, *Comptes Rendus Acad. Sci.* **162** (Paris, 1916), 629–632], using just *four* kingwise connected pieces(!):

 $\leftrightarrow$ . (57)

Exercise 83 discusses data structures for the algorithm just sketched. Heuristic methods that work successfully on several problems that are too large for those 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.)

rookwise connected
flips
kingwise connectivity
Sierpiński carpet
Sierpinski
data structures
Zhou
Wang
Frederickson

EXERCISES

fullswap
5-letter words of English
English
substitution cipher
Alphabetical arithmetic
fixed points
lexicographically
Enigma's history
simple path
delta tree
animal names

1. [*18*] Consider the fullswap (ET)(AO)(IN)(SH)(RD)(LU)(CM)(FW)(YP)(VB)(GK)(QJ)(XZ). What 5-letter words of English does it map into other words (like COUTH $\leftrightarrow$ MALES)?

2. [*M22*] Let φ and ψ be fullswaps on $2m$ letters. Prove or disprove:
a) $\varphi\psi\varphi$ is a fullswap.
b) There is an involution τ such that $\psi = \varphi\tau$.
c) There is an involution τ such that $\psi = \tau\varphi\tau$.
d) There is a fullswap τ such that $\psi = \tau\varphi\tau$.

▶ **3.** [*22*] Design an algorithm that visits all fullswaps on an alphabet of size $2n$.

4. [*20*] Alphabetical arithmetic as in (7) and (8) yields surprising formulas such as LARGE + TRAIN = ERROR. Do other common 5-letter words have noteworthy sums?

▶ **5.** [*M20*] What are the fixed points of the rotor permutations $\mathrm{R}(x)$ defined in (11)?

6. [*05*] What are the two fullswaps defined by the networks of wires in Fig. 295?

▶ **7.** [*21*] Specify in detail how to emulate the Enigma encoding (14) on a modern computer, using arrays to represent the internal permutations.

8. [*24*] Show the sequel to Fig. 295: What should be the next two configurations?

9. [*20*] Compute all $5 \cdot 4 \cdot 3 \cdot 26^3 = 1054560$ fullswaps that an Enigma M3 could possibly produce without using its plugboard. Are they all distinct? Which of them is lexicographically smallest? Which of them is lexicographically largest?

10. [*24*] Study Enigma's history, in which (15) was replaced by a more complex rule for advancing the rotors. Explain how to translate any setting of the machine described here into a setting of the historic Enigma M3 that would have had identical behavior.

11. [*21*] All of the participants in a network of Enigma machines would typically use the same three rotors for an entire day, as well as the same ring and plugboard settings, based on preprinted instructions. But they independently chose random starting positions $p_0p_1p_2$ for each message, so that no two messages would be enciphered with the same sequence of permutations π_k. How could they use Enigma to communicate those secret starting positions safely?

▶ **12.** [*M22*] Let D be the digraph with 26^3 vertices $V = \{p_0p_1p_2 \mid \mathtt{A} \le p_0, p_1, p_2 \le \mathtt{Z}\}$ and with 26^3 arcs, where $u \longrightarrow v$ means that rule (15) changes u to v. Characterize all vertices v that have no predecessors (in-degree 0). What is the longest simple path in D?

▶ **13.** [*20*] Encipher the message $\mathtt{A}^{50} = \mathtt{AA}\ldots\mathtt{A}$ of length 50 (see exercise 7). Use the rotors, ring settings, and starting position at the left of Fig. 295, and set the plugboard to the fullswap in exercise 1. (a) What's the result? (b) Now encipher the same message with a slight change: The starting position and ring settings should be NYX and VKB, not NYY and VKC. (c) Now try $p_0p_1p_2 = \mathtt{NYZ}$, $q_0q_1q_2 = \mathtt{VKD}$. (d) Explain what happened.

14. [*17*] How many times does the label (a) $01l$ (b) $12l$ appear in the delta tree?

▶ **15.** [*21*] If Enigma begins at position IZY, it moves first to position JAZ, then JBA. So why isn't there a path from 000 to 111 to 122 in the delta tree?

16. [*24*] To assess the versatility of Enigma's enciphering mechanism, study the number of ways in which it can transform each of the following animal names into each of the others, without a plugboard: BISON, BURRO, HORSE, HYENA, LLAMA, MOOSE, MOUSE, OTTER, SHEEP, SKUNK, SLOTH, SWINE, TIGER, ZEBRA.

17. [*24*] For each word $\alpha \in \mathtt{WORDS}(5757)$, determine the number of ways in which Enigma can transform CHAOS $\mapsto \alpha$, without using its plugboard.

▸ **18.** [*25*] Find the maximum n for which Enigma can map $\mathtt{A}^n \mapsto \mathtt{B}^n$, *with* a plugboard.

19. [*27*] Find the maximum n for which Enigma (and its plugboard) can map $\mathtt{A}^n$ into a sequence of (i) B's and C's; (ii) B's, C's, and D's; ...; (v) B's, C's, D's, E's, F's, and G's.

20. [*16*] Suppose Enigma is set up to map CHAOS ↦ ORDER without a plugboard. If the plugboard actually swaps the thirteen pairs in exercise 1, what ciphertext corresponds to the plaintext ADOTD?

21. [*17*] The text observes that (19) forces both PO and OF to be false. What other variables must be false as a consequence of (19)?

▸ **22.** [*18*] Do (19) and (21) together imply that OO is false? What about PP and FF?

23. [*17*] The toy problem POTPOURRI ↦ OFPUZZLES uses 351 Boolean variables. Explain why more than a third of them will never become equivalent to any other.

24. [*22*] Using the fact that $\pi_6 =$ [WMJEDTPSLCVIBZYGUXHFQKARON], complete the text's proof that no plugboard is able to extend the 4-step solutions (24) of POTP ↦ OFPU to a 6-step solution of POTPOU ↦ OFPUZZ.

▸ **25.** [*26*] One of the very first Enigma configurations that arises when trying to map POTPOURRI ↦ OFPUZZLES leads to the following viable equivalence classes:

{AF,GO,JT,MP,UX,VZ},{AL,EX,QR},{BI,EQ,LS,RX},{BL,DR,EM},{CS,II},
{DE,JL,MR},{EN,IR,KS,LV},{ES,IJ,LL,RW},{EW,IK,LX,RS},{EY,JR,LM},
{GI,SS},{HI,MS},{HS,IM},{IN,SV},{IQ,SX},{IV,NS},{IX,QS}.

(There also are singleton classes whose variables contain only unused letters.)

a) Prove that a plugboard setting is therefore impossible.
b) Devise an improvement to the "union-find" strategy described in the text, with which cases such as this could immediately be ruled out.

26. [*20*] Why are *singleton* equivalence classes shown in (26), but not in (24) or in the previous exercise?

27. [*21*] Given a set of viable equivalence classes (such as (26)), formulate a SAT problem whose solutions precisely match the solutions to the corresponding exact cover problem (such as (27)), as well as the corresponding feasible plugboards.

▸ **28.** [*M29*] To solve a setup problem such as (28), we're given five things: A crib length, $c < 26$; a message length, $n \le 625$; a crib position $m \ge 0$, where $m + c \le n$; ring offsets $o_0 o_1 o_2$; and a path in the delta tree from the root to level $c-1$. (In (28), for example, we have $c = 13$, $n = 125$, $m = 69$, $o_0 o_1 o_2 =$ AAE, and the delta path begins with 000, 001, 012.) Explain how to find all of the inequivalent start positions $s_0 s_1 s_2$ and ring settings $q_0 q_1 q_2$ such that, when Enigma begins in position $p_{00} p_{10} p_{20} = s_0 s_1 s_2$ at time 0, its position $p_{0t} p_{1t} p_{2t}$ at time $t = m+1$ will satisfy $p_{0t} p_{1t} p_{2t} - q_0 q_1 q_2 \equiv o_0 o_1 o_2$ (modulo 26); and the positions for $t = m+1, \ldots, m+c$ will follow the given delta path.

29. [*20*] Using exercise 7, encipher the secret message (25) with the parameters

III I IV FRE NCH MD CL IT HE AR OF SN UV WX YZ.

30. [*21*] Generalize (14) by changing it to

$$\pi_k \;=\; \tau\,\mathrm{III}(o_2)\mathrm{II}(o_1)\mathrm{I}(o_0)\,\rho\,\mathrm{I}(o_0)^-\,\mathrm{II}(o_1)^-\,\mathrm{III}(o_2)^-\,\tau^-,$$

so that the "plugboard" τ is now an *arbitrary* permutation instead of an involution. Extend the text's union-find technique, so that it will now be able to discover an arbitrary permutation τ that has enciphered a sufficiently long crib.

plugboard
viable equivalence classes
equivalence classes
singleton classes
union-find
SAT problem
exact cover problem
crib
delta tree
start positions
ring settings
secret message
parameters
involution
union-find

31. [*M22*] If $b > 2$, $n > m$, and $x = (x_n \ldots x_0)_b \geq b^{m+1}$, prove that $x_n^m + \cdots + x_0^m < x$.

32. [*10*] What's the difference between $M_3(x)$ and $M_4(x)$, as defined in the text?

▶ **33.** [*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?

34. [*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 $t \geq 0$.

35. [*24*] Implement Algorithm P. For which $m < 100$ are all solutions trivial?

36. [*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 $\pi_m x \geq x$.

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 $x_0 \geq 0$, 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 $j \geq \mu$. (See exercise 37 and exercise 3.1–6.) What's a good way to discover all such periods, given $m \geq 2$?

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 $0 \leq d_j < 10$ for $1 \leq j \leq m$; 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 $\boxed{}\,23$, and that $\boxed{}\,23\square\square$ was paid altogether. Can the original amounts be reconstructed?

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

$$\begin{array}{r} 3\,\square\,\square \\ \times\; 6\,\square \\ \hline 2\;4\,\square\,\square \\ \square\,\square\,8\,\square\;\; \\ \hline \square\,\square\,2\,\square\,\square \end{array}$$

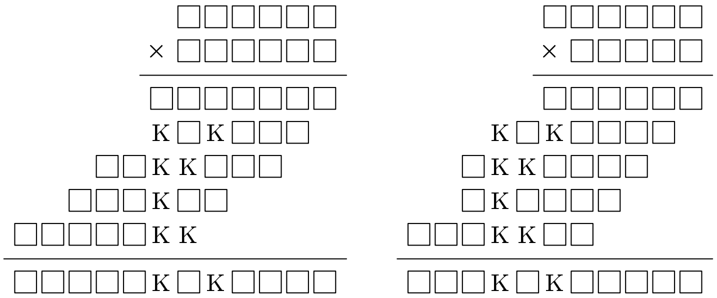
50. [*M19*] Solve the skeleton multiplication puzzle (35) quickly by hand.

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
skeleton multiplication

51. [*M21*] The solution to exercise 50 begins with the fact that the product in (35) 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 (36), 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 (36) has "slack 1."

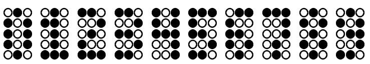
These puzzles were discovered by an algorithm like that of the text, having specified "offsets" $O = (o_0, o_1, \ldots, o_m)$ and "patterns" $P = (p_0, p_1, \ldots, p_m)$. In the left example, $O = (0, 1, 2, 4, 5, 0)$ and $P = (0, 101000, 11000, 100, 11, 1010000)$; in the right example, $O = (0, 1, 2, 3, 4, 0)$ 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 (36) 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 below. 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 example appears in Fig. 300; however, the puzzle shown there is *not* optimum, because

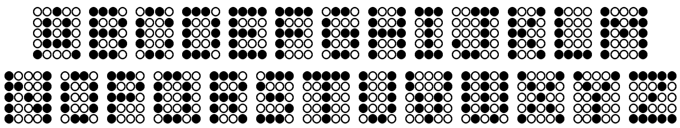
slack
pixel pattern
data structures
pixel patterns
alphabet

```
           □□□□□□□                 □□□□□□□□
         × □□□□□□                × □□□□□□
          □□□□□□□                □ A □□□□□□
        5 5 5 □□□□□            □ A □ A □□□□
        □ 5 □□□□□□           □□ A □ A □□□
    □□□ 5 5 □□□          □□□□□ A A A
   □□□□□□ 5 □
  □□□□ 5 5 □□□□□□□        □□□□ A □□□ A □□□□
       (□ ≠ 5)                 (□ ≠ A)
```

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

Schuh
Feynman
division skeleton problem
exact cover problem

a smaller skeleton is possible using slack 1! Use the following pixel patterns:



(Note that this K is *wider* than the Ks in (36) 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 (36), with no special digits in the first partial product. The second series should have no special digits in the *total* product. Examples (□ ≠ A):

```
          □□□□□□              □□□□□□□□
        × □□□□□□             × □□□□□□□
         □□□□□□□             A □□□□□□□□
        A □□□□□□           A □ A □□□□□□
      A □ A □□□□        □□ A □ A □□□□
   □□ A □ A □□        □□□□ A A A □□
  □□□ A A A □        □□□□ A □□□ A
 □□ A □□□ 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).

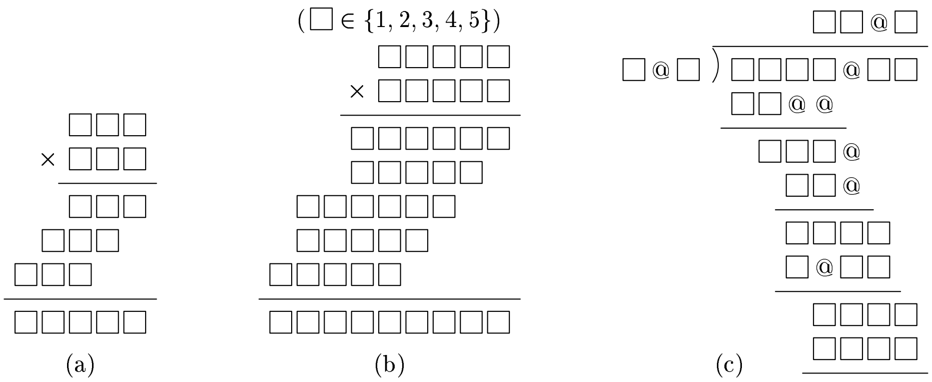


Fig. 301. Special skeleton puzzles, discussed in exercises 64–66.

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.

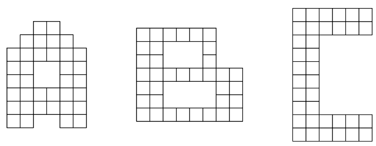
▸ **87.** [*23*] Continuing exercise 86, find all of the discrete four-piece unrotated dissections — connected or not — of (a) 5^2 into $4^2 + 3^2$; (b) 13^2 into $12^2 + 5^2$; (c) 17^2 into $15^2 + 8^2$. (*Don't* allow pieces to jump between squares as they do in (54) and (56).)

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*] For which triples (u, v, w) with $u^2 + v^2 = w^2$ are such dissections possible?

▸ **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*] Can the 8×8 square be dissected into the 9×9 Sierpiński carpet using at most six *rookwise* connected pieces?

Loyd
data structures
discrete dissection
rookwise connected
Frederickson
font
alphabet
Sierpiński carpet

disconnected solutions

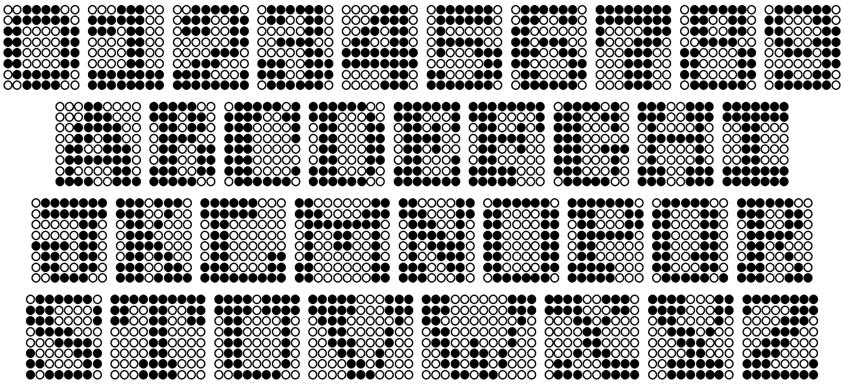


Fig. 302. ‘**FONT36**’, a special font designed for dissection puzzles.

96. [*40*] Experiment with algorithms for dissection that rule out disconnected solutions early in the process, instead of first generating the complete set of dissections.

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

SECTION 7.2.2.8

1. There are 10 pairs in WORDS(5757), including WORTH ↔ FADES, HUNCH ↔ SLIMS, CONCH ↔ MAIMS. [The given fullswap comes from the order of keys in a Linotype machine, which approximates the frequency of letters in English.]

2. (a) Indeed, any conjugate of any fullswap is a fullswap. (See exercise 1.3.3–38.)

(b) Such τ exists $\iff \varphi\psi = \tau$ is an involution. So it fails if, for example, $m > 2$ and $\varphi = (0\,1)(2\,3)(4\,5)(6\,7)\ldots$, $\psi = (0\,5)(1\,2)(3\,4)(6\,7)\ldots$.

(c) Consider the multigraph G whose vertices are the letters, and whose edges are $\{u \longrightarrow v \mid u\varphi = v\} \cup \{u \longrightarrow v \mid u\psi = v\}$. This is the union of two perfect matchings; so every vertex of G has degree 2, and the connected components of G are cycles. Let $a_0 \longrightarrow a_1 \longrightarrow \cdots \longrightarrow a_{l-1} \longrightarrow a_0$ be one of those cycles, with $a_0\varphi = a_1$ and $a_0\psi = a_{l-1}$. Then $a_1\psi = a_2$ and $a_{l-1}\varphi = a_{l-2}$. So we see that l is even. Hence we can put $(a_0)(a_1a_{l-1})\ldots(a_{l/2-1}a_{l/2+1})(a_{l/2})$ into the cycle decomposition of τ.

(d) If $l/2 = q$ is odd in (c), we can put $(a_0a_q)(a_1a_{q+1})\ldots(a_{q-1}a_{l-1})$ into τ. Or if G has two cycles $a_0 \longrightarrow \cdots \longrightarrow a_{l-1} \longrightarrow a_0$ and $b_0 \longrightarrow \cdots \longrightarrow b_{l-1} \longrightarrow b_0$ of the same length, we can use $(a_0b_1)(a_1b_2)\ldots(a_{l-1}b_0)$. But we can avoid fixed points in τ if and only if the cycles of G whose length is a multiple of 4 can be paired up in this fashion.

3. It's convenient to represent the 2-cycles $(a_0a_1)(a_2a_3)\ldots(a_{2n-2}a_{2n-1})$ by an array with $a_0 < a_2 < \cdots < a_{2n-2}$ and $a_{2j} < a_{2j+1}$ for $0 \le j < n$; see 7.2.1.2–(49). We also compute the fullswap permutations $p_0p_1\ldots p_{2n-1}$, which are visited in lexicographic order. Auxiliary bitmaps w_k for $0 \le k < n$ represent the elements not in $\{a_0, \ldots, a_{2k}\}$.

F1. [Initialize.] Set $k \leftarrow t \leftarrow 0$, $b \leftarrow 1$, $s \leftarrow (1 \ll 2n) - 1$, and go to F6.

F2. [Finish swap k.] (At this point $\nu s = 2n - 1 - 2k$ and $b = 2^t$.) While $b \mathbin{\&} s = 0$, set $t \leftarrow t + 1$ and $b \leftarrow 2b$. Then set $a_{2k+1} \leftarrow t$, $s \leftarrow s - b$, $p_t \leftarrow a_{2k}$, $p_{p_t} \leftarrow t$. Go to F5 if $s > 0$.

F3. [Visit.] (Now $k = n - 1$.) Visit the fullswap permutation p.

F4. [Move left.] Terminate if $k = 0$. Otherwise set $k \leftarrow k-1$, $t \leftarrow a_{2k+1}+1$, $b \leftarrow 1 \ll t$, $s \leftarrow w_k$. If $b \le s$, go to F2. Otherwise repeat this step.

F5. [Move right.] Set $t \leftarrow a_{2k} + 1$, $b \leftarrow 1 \ll t$, $k \leftarrow k + 1$.

F6. [Start swap k.] (At this point $\nu s = 2n - 2k$ and $b = 2^t$.) While $b \mathbin{\&} s = 0$, set $t \leftarrow t + 1$ and $b \leftarrow 2b$. Then set $a_{2k} \leftarrow t$, $s \leftarrow s - b$, $t \leftarrow t + 1$, $b \leftarrow 2b$, $w_k \leftarrow s$, and go to F2. ▌

4. From WORDS(1000) we get 39 valid sums, including ROYAL+WAVES = NOTED, WAGON+GRIEF = CROSS, DWELL+DEPTH = GATES, PLANS+DREAM = SCENE, ROBOT+PANEL = GOOSE. (And WORDS(500) yields SHELL+GAMES+THERE = ROUGH; STUDY+HEAVY+GRAPH = FOUND; LARGE + PAPER + ENDED = ENJOY; IDEAS + LAKES + TREES = MUSIC.)

5. I(x) has a fixed point at B $- x$; II(x) has fixed points at L $- x$ and V $- x$; III(x) has a fixed point at R $- x$; IV(x) and V(x) have none.

6. [DQEACONRYUVZWGFSBHPXJKMTIL] = (AD)(BQ)(CE)(FO)(GN)(HR)(IY)(JU)(KV)(LZ)(MW)(PS)(TX) and [WTPOHVZEQYUNSLDCIXMBKFARJG] = (AW)(BT)(CP)(DO)(EH)(FV)(GZ)(IQ)(JY)(KU)(LN)(MS)(RX).

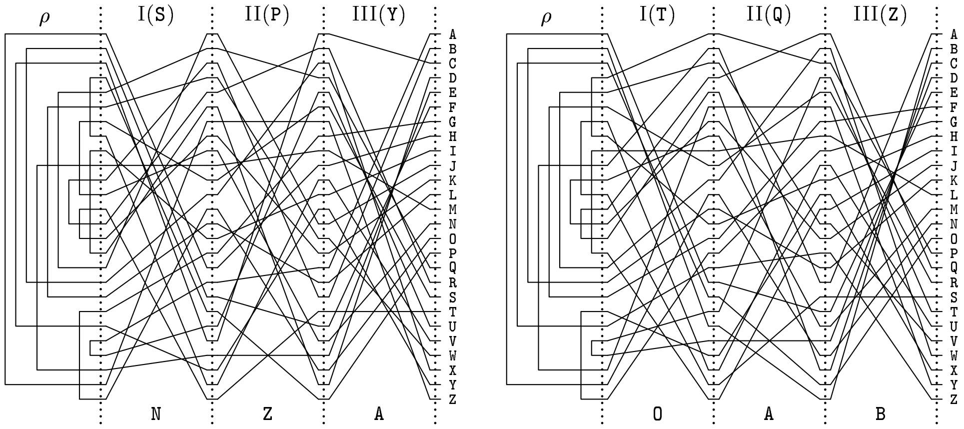
7. Store the relevant base permutations in arrays *perm*[0], *perm*[1], *perm*[2]; also store the reflector and plugboard permutations in *refl* and *plug*. (For example, if the slow rotor is IV, we'd have *perm*[0][A] = K, *perm*[0][B] = Y, . . . , by (7) and (12); also *refl*[A] = Y, *refl*[B] = R, . . . , by (2).) Precompute the inverse permutations *iperm*[r][*perm*[r][c]] $\leftarrow c$ for $0 \le r < 3$ and A $\le c \le$ Z. Define $o_0o_1o_2$ by (13). Then, to encipher the letter c, set $c \leftarrow plug[c]$, $c \leftarrow perm[2][c \oplus o_2] \ominus o_2$; $c \leftarrow perm[1][c \oplus o_1] \ominus o_1$; $c \leftarrow perm[0][c \oplus o_0] \ominus o_0$;

WORDS(5757)
etaoin shrdlu
Linotype machine
frequency of letters in English
conjugate
multigraph
perfect matchings
components
cycles
cycle decomposition
lexicographic order
base permutations
slow rotor
inverse permutations

$c \leftarrow \mathit{refl}[c]$; $c \leftarrow \mathit{iperm}[0][c \oplus o_0] \ominus o_0$; $c \leftarrow \mathit{iperm}[1][c \oplus o_1] \ominus o_1$; $c \leftarrow \mathit{iperm}[2][c \oplus o_2] \ominus o_2$; $c \leftarrow \mathit{plug}[c]$. Here $\oplus$ and $\ominus$ denote addition and subtraction mod 26. (See (3)–(6).)

joke
sorting
carries
base permutations
lexicographic order

8. Rule (15) is used twice. As before, the ring settings are VKC.



[URYIMXOZDQTWESGVJBNKAPLFCH] [YKQTNRXPLMBIJEZHCFUDSWVGAO]

9. By sorting them we can verify that there are no duplicates. The smallest and largest, [BADCFEHGNOPSXIJKUYLZQWVMRT] and [ZYXWNSIVGRPTOEMKUJFLQHDCBA], are fairly close to the globally smallest and largest fullswaps, [BADCFEHGJI...] and [ZYXWV...].

10. Rule (15) causes "carries" when stepping from Z to A on any rotor. But that was true in the historic machine only for rotor V. The other four rotors were manufactured differently, so that the carry-triggering notch positions for I, II, III, and IV were encountered when stepping respectively from Q to R, E to F, V to W, and J to K.

To emulate the old behavior on the machine described here, keep the ring settings unchanged; but add (R, F, W, K, A) to the position coordinates for rings (I, II, III, IV, V). For example, Fig. 295 shows positions EDU and EDV of the historic M3 (again with ring settings VKC). The historic base permutations were not (12) but rather

$$\begin{aligned}
\Pi_{\overline{\mathrm{I}}} &= \mathrm{I}(-\mathtt{R}) = [\mathtt{EKMFLGDQVZNTOWYHXUSPAIBRCJ}];\\
\Pi_{\overline{\mathrm{II}}} &= \mathrm{II}(-\mathtt{F}) = [\mathtt{AJDKSIRUXBLHWTMCQGZNPYFVOE}];\\
\Pi_{\overline{\mathrm{III}}} &= \mathrm{III}(-\mathtt{W}) = [\mathtt{BDFHJLCPRTXVZNYEIWGAKMUSQO}];\\
\Pi_{\overline{\mathrm{IV}}} &= \mathrm{IV}(-\mathtt{K}) = [\mathtt{ESOVPZJAYQUIRHXLNFTGKDCMWB}].
\end{aligned}$$

11. The sender would do this: (S1) Choose *two* random starting positions, say FOO and BAR. (S2) Set the starting position to FOO and encipher BAR, obtaining SMH (say). (S3) Set the starting position to BAR and encipher PLAIN, obtaining CIFER. (S4) Transmit the message FOOSMHCIFER. Then the receiver of that message would do this: (R1) Set the starting position to FOO and encipher SMH, obtaining BAR. (R2) Set the starting position to BAR and encipher CIFER, obtaining PLAIN.

12. Let U be the 26^2 vertices $\{p_0\mathtt{AA}, p_0\mathtt{ZB}, p_0\mathtt{ZC}, \ldots, p_0\mathtt{ZZ}\}$, for $\mathtt{A} \le p_0 \le \mathtt{Z}$. There's a path through the remaining $N = 26^2 \cdot 25$ vertices $V \setminus U$, when they are listed in lexicographic order from AAB to ZZA; and this path becomes a cycle of length N when we add $\mathtt{ZZA} \longrightarrow \mathtt{AAB}$. Let $U_0 = \{p_0\mathtt{AA}\}$ and $U_1 = \{p_0\mathtt{ZZ}\}$. Each vertex of U leads directly into the N-cycle, except for the 26 vertices of U_1, whose successors are in U_0.

Hence the source vertices are $U \setminus U_0$. The longest simple paths are the 26 paths of length $N + 1$ that begin in U_1.

source vertex
delta tree
bitmap

13. (a) `RKXNP SGSJJ QBUYS XDYSN NBXWU TMGSJ KMTLW QVIGO RRHDG HGPEV`.

(b) `RHZNP SGSJJ QBUYS XDYSN NBXWU TMMSJ KMTLW QVIGO RRHDG HGPEV`.

(c) `SMXNP SGSJJ QBUYS XDYSN NBXWU TDGSJ KMTLW QVIGO RRHDG HGPEV`.

(d) The first enciphered `A` is `R` in both (a) and (b), because the rotor permutations I(`S`), II(`O`), III(`X`) are the same in each case (`NYZ` − `VKC` = `NYY` − `VKB`). The second enciphered `A`, however, uses I(`S`), II(`P`), III(`Y`) in (a) but I(`S`), II(`O`), III(`Y`) in (b). Similarly, the third one also uses different rotor permutations.

The fourth one is again the same (`OAC` − `VKC` = `OAB` − `VKB`); and equality continues until step 28. In general, the encipherment of any plaintext will be the same for (a) and (b) except perhaps at the kth characters, where $k \bmod 26 = 2$ or $k \bmod 650 = 3$. (Sometimes, for instance when $k = 1016$, the encipherments of `A` agree by chance.)

Similarly, (a) matches (c) except perhaps when $k \bmod 26 = 1$ or $k \bmod 650 = 2$.

14. (a) l; (b) $l - [l > 0]$. (And $00l$ once; $11l$ once, for $l > 0$.)

15. Enigma always advances the rotors *before* enciphering a letter. Therefore the position at enciphering time is never a source vertex in the digraph of exercise 12. But `IZY` *is* a source vertex.

16. This chart has '×' in places like `BISON` versus `TIGER` where there clearly is no solution, because the plaintext and ciphertext have a matching letter. Otherwise it shows the number of solutions. For example, there are three ways to convert `BISON` to `SKUNK`, obtained as follows: (i) Rotors I, V, III; start at `FJW` with rings `AAG`. (ii) Rotors IV, V, III; start at `TYZ` with rings `ATN`. (iii) Rotors V, III, II; start at `RYY` with rings `AGE`. (Solution (iii) follows the path 000, 011, 122, 123, 124 in the delta tree, beginning at α_1 = `RSV`.) Eight of the solutions take the leftmost path, to 004; for example, `HORSE` becomes `ZEBRA` with rotors IV, I, III, starting at `DUQ` with rings `AAA`. Here there's great flexibility, since no "carries" occur. We could, for example, start at `AYA` with rings `XEK`.

BISON	×	1	1	2	1	0	1	0	3	1	1	×	2
BURRO		×	1	3	1	1	1	2	1	0	2	2	×
HORSE			×	1	×	×	2	0	1	0	×	2	1
HYENA				×	2	2	1	×	×	2	×	1	×
LLAMA					0	0	0	1	1	×	0	1	×
MOOSE						×	2	0	3	×	×	0	3
MOUSE							4	3	×	0	×	1	0
OTTER								×	0	1	0	×	1
SHEEP									×	×	×	×	0
SKUNK										×	×	0	0
SLOTH											×	2	1
SWINE												2	0
TIGER													0
ZEBRA													

17. The "scores" as in exercise 16 are respectively (×, 0, 1, 2, 3, 4, 5, 6, 7) for exactly (2742, 1205, 1068, 483, 164, 62, 20, 9, 4) of the 5757 words. The most common words having those respective scores turn out to be (`WHICH`, `OTHER`, `ABOUT`, `BEGAN`, `WATER`, `BEGIN`, `WORLD`, `HABIT`, `FAIRY`).

18. It's easy to see that we seek the largest n for which there exist letters x and y such that Enigma can map $x^n \mapsto y^n$, *without* a plugboard. To find it, we can try all $60 \cdot 26^3$ wiring networks and all $2n + 1$ paths to level n of the delta tree.

A bitmap can be stored in each node of the tree, showing which letters x have at most one image in that node and its ancestors. This bitmap is usually zero; hence we needn't explore the subtree rooted there. But remarkable coincidences do show up. With rotors I, II, and IV, for example, the permutations for offsets `OWH`, `OWI` are [`UFTLIBQYERSDXZVWGJKCAOPMHN`], [`UOTSIWQYEPRVXZBJGKDCALFMHN`]: a 14-fold agreement!

The maximum n is 6, obtainable in 18 ways. One of them uses rotors II, III, IV, starting at `XGV` with rings `AAE` and plugboard (`AP`)(`BT`). [But $n = 7$ *would* be achievable if Enigma had advanced the rotors *after* enciphering, not before! With rotors I, IV, III, starting at α_1 = `SCH` and following the path 000, 111, 112, 113, 124, 125, 126, the string `AAAAAAA` maps to `WWWWWWW`. That path, however, isn't in the delta tree; see exercise 15.]

forced
at-most-one constraints
at-least-one constraints
big carry
carry

19. Generalize the previous answer, so that the bitmap shows letters x with at most (i) 2 … (v) 6 images in that node and its ancestors. Champion results are:

	Rotors	Start	Rings	Plugboard	n	Ciphertext	Solutions
(i)	IV, II, V	UIX	AAL	(AX)(BJ)(CU)	10	BCCCBCBCCB	2
(ii)	II, IV, III	MBO	AAX	(AM)(DH)	13	DBDDDDBCDCBDC	4
(iii)	I, III, II	OCO	AAR	(BX)(CY)(DN)	16	BEEECCBCDEBBCEDB	1
(iv)	IV, II, III	XUU	AAM	(AY)(CX)(DV)(EW)(FP)	20	BCDDBDCEEDFBBDBDCDCD	2
(v)	I, V, II	MYM	AEK	(AM)(BJ)(CT)(DR)(EU)	23	BFFBCGDEFFBCBDFBEFDBFCB	1

20. The plugboard changes ADOTD to ORAER, which gets mapped to CHDOS, then MSRAH.

21. Since PS $\equiv$ OP, PS must also be false. (But no other variables are falsified by (19).)

22. OO $\equiv$ PF by (19) and PF $\equiv$ OC by (21); hence OO and OC (and PF) are false. On the other hand the equivalence classes {PP, FM, OS} and {FF, KP, OV} are viable.

23. The 15 letters {A, B, C, D, G, H, J, K, M, N, Q, V, W, X, Y} don't appear in either the plaintext or the ciphertext. But every equivalence in constraints like (19) involves a plaintext letter on the left and a ciphertext letter on the right. Hence $\binom{16}{2} = 120$ of the variables are unimportant. Plugs between them don't affect the encipherment.

24. From π_5 we deduce that OI $\equiv$ ZO and OU $\equiv$ ZF; hence all four variables are false by (24). To rule out the remaining seven classes of (24), we have ZM $\equiv$ OQ $\equiv$ UK $\equiv$ ZV; ZR $\equiv$ OE $\equiv$ UQ $\equiv$ ZU; ZC $\equiv$ OY $\equiv$ UL $\equiv$ ZI; ZN $\equiv$ OV $\equiv$ UB $\equiv$ ZM; ZS $\equiv$ OW $\equiv$ UJ $\equiv$ ZC; ZW $\equiv$ OS $\equiv$ UH $\equiv$ ZS; ZQ $\equiv$ OM $\equiv$ UV $\equiv$ ZK.

25. (a) The letters F, O, P, T, U, and Z appear only in the first class. Therefore all variables of that class must be true. But then all of the other nonsingleton classes become *unviable*, except for {CS, II}, because they have a letter in common with that first class. This leaves us with no viable classes for E, L, and R.

(b) Say that a class is "forced" if some letter appears in the variables of no other viable class. If more than one class is forced, we can make them all equivalent, because all of their variables must be true. Thus we can assume that there's at most one forced class. Any class with a letter in common with that forced class is unviable; and then further classes may become forced. So we should continue forcing until either (i) some letter belongs to no viable class or (ii) the forced class coexists peacefully with all others.

26. Singletons are irrelevant when mapping only POTP $\mapsto$ OFPU or POTPOURRI $\mapsto$ OFPUZZLES, because their letters aren't used. But in a problem like (25), *all* letters are used, because the plugboard is applied to the entire 125-character-long ciphertext.

27. If the classes are $C_1, \ldots, C_n$, let there be n Boolean variables $x_1, \ldots, x_n$. Also let L_j be the set of letters involved in C_j. There are at-most-one constraints $(\bar{x}_j \vee \bar{x}_k)$ for all pairs of classes with $L_j \cap L_k \neq \emptyset$. And there are 26 at-least-one constraints $(\bigvee_{l \in L_j} x_j)$, one for each letter l. The plugboard corresponding to a solution should swap l with $l' > l$ if and only if the pair ll' is in a class C_j with x_j true.

28. More precisely, when $t = m + j$ for $1 \le j \le c$ we must have $p_{0t}p_{1t}p_{2t} - q_0q_1q_2 \equiv o_0o_1o_2 + \Delta_j$, where Δ_j is the label of the jth path node. There are two basic tasks:

a) Given $b < n$, find $s_0s_1s_2$ and $q_0q_1q_2$ so that the "big carry" — when all three rotors move — occurs between times b and $b + 1$.

b) Given a "carry" value r with $0 \le r < 26$, find the smallest $s_0s_1s_2$ and $q_0q_1q_2$ such that there are no big carries at times $t < n$, and the middle rotor moves between t and $t + 1$ if and only if $t \bmod 26 = r$.

alchemy
roman numerals
FRENCH
author
online
unviable
viable
binary-coded decimal
bitwise operations

In both tasks we can set $s_0 \leftarrow 0$. Then the unique solution to (a) is found by setting $r \leftarrow (b-1) \bmod 26$, $s_2 \leftarrow 25-r$, $s_1 \leftarrow 24-\lfloor (b-1)/26 \rfloor$, $x_0 \leftarrow [b \le m]$, $x_2 \leftarrow s_2+m+1$, $x_1 \leftarrow s_1 + \lfloor (s_2+m+1)/26 \rfloor + x_0$, and $q_j \leftarrow (x_j - o_j) \bmod 26$ for $0 \le j < 3$.

In task (b), we set $s_1 \leftarrow 0$, $s_2 \leftarrow 25-r$, $x_0 \leftarrow 0$, $x_1 \leftarrow \lfloor (s_2+m+1)/26 \rfloor$, $x_2 \leftarrow (s_2+m+1) \bmod 26$, then q_j as in (a).

The given problem has three cases. The easiest, when the delta path ends with $\Delta_{c0} = 1$, uses (a) with $b = m+j$, where j is maximal with $\Delta_{j0} = 0$. The medium case, when the delta path ends with $\Delta_{c0} = 0$ and $\Delta_{c1} = 1$, uses (b) with $r = (m+j) \bmod 26$, where j is maximal with $\Delta_{j1} = 0$; then it also uses (a), for all $1 \le b < n$ such that $b \equiv r+1$ (modulo 26) and either $b \le m$ or $b > m+c$. And the hard case, when $\Delta_{c0} = \Delta_{c1} = 0$, uses the medium case for $r = (m+k) \bmod 26$ and $c \le k \le 26$. (These are the setups for which no carry or big carry occurs while encrypting the crib.) This reasoning is delicate, very easy to get wrong!

29. `PHILO SOPHE RSWHE NTHEY WROTE ANYTH INGTO OEXCE LLENT FORTH EVULG ARTOK NOWEX PRESS EDITE NIGMA TICAL LYTHA TTHES ONSOF ARTON LYMIG HTUND ERSTA NDITX`. That's a quotation from a work about alchemy written in MDCLI:

> *Philosophers when they wrote any thing too excellent*
> *for the vulgar to know, expressed it enigmatically,*
> *that the sons of Art only might understand it.*
> — JOHN FRENCH, *The Art of Distillation* (1653)

(The original 1651 printing said 'sonnes of Art'; but the spelling was soon modernized.) For comparison, here are the 2nd-best and 9th-best candidates (scores 7062 and 6435):

`PHILO SOPHE RSWHE NTHEY WROTE ANYTH INGTO OEXCE LLENT FORTH EVULG ARTOK NOWEX PRESS EDITE NIGMA TICAL LYTHA TTHES ONSOF DKLBB NXVGF IEDQY UHGMC KCOZJ`;

`PHILO SOPHE RSWHE NTHEN GROTE ANYTH INGTO OEXCE LLENR HARTH EVULG ARTOK NOWEX PRESS FJSTE NIGMA TICAL LYTHA TTHES OZFFF ARTON LYMIG HTUND ERSTA NDLOO`.

[The program ENIGMATIC-PUZZLE with which the author solved (25) is online.]

30. Define the Boolean variables xy by (17) as before; but now there are 26^2 of them, and we do *not* regard yx as being identical to xy. Consider (18); if we want `P` to be enciphered as `O`, we now have the equation $\mathtt{P}\tau\pi\tau^- = \mathtt{O}$, that is, $\mathtt{P}\tau\pi = \mathtt{O}\tau$. Hence (19) and (21) remain correct. A class now becomes unviable in this extended sense if and only if it contains distinct variables xy and $x'y'$ with either $x = x'$ or $y = y'$.

31. 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 $t_{b-2} = (1-1/b)^{b-2} < 1$.

Since $n > m$ we have $b^n/(n+1) \ge b^{m+1}/(m+2)$. We can assume that $x_n > 0$. Hence $x_n^m + \cdots + x_0^m \le (n+1)(b-1)^m \le b^{n-m-1}(m+2)(b-1)^m < b^n \le x$.

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

33. 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 (31) 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.

34. (a) If $t < 0$, go to P5. Otherwise if $b \ge c$, 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. Finally, go to P5 if $a \ne b$.

(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 or equal to 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 $.00002\,39052\,59 \times 10^{101}$, because 8s are forbidden; and a similar idea applies to upper bounds. Further exploration of such techniques should prove to be interesting.]

Oliveira
bootstrap algorithm
Stewart
Iséki
multicombination
birthday paradox
coincidence
binary-coded
binary search
bucket-sorting

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

36. 1, 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 smallest 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), 578–583.]

(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, y).

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 $d_3 d_2 d_1 d_0 = 8740$ 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 $T_k \leftarrow 1$ and $k \leftarrow g(k)$. Then set $i \leftarrow j$; while $i \ne g(k)$, set $T_i \leftarrow 2$ and $i \leftarrow g(i)$. Then if $T_i < 2$, we've found a new cycle, beginning at i; set $l \leftarrow 1$, $T_i \leftarrow 2$, 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 7.2.1.3T with $s = 9$, $t = m+1$, and $n = m+10$. When visiting $c_t \ldots c_1$ in step T2, we compute $(c_t + 1 - t)^m + \cdots + (c_2 - 1)^m + c_1^m = (a_m \ldots a_1 a_0)_{10}$, bucket-sort to get $\{e_t, \ldots, e_2, e_1\} = \{a_m, \ldots, a_1, a_0\}$ with $e_t \ge \cdots \ge e_1 \ge 0$, set $g(p) \leftarrow \binom{e_t+t-1}{t} + \cdots + \binom{e_2+1}{2} + \binom{e_1}{1}$, 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 that, for even b, $s(b^2+1) = 2$ 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 product $A \Cap B$. (See answer 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 $m = 16$ the reduced multiset B still has 99,795,483 elements, but A reduces to only 20,846,476. After sorting those multisets, the product 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. Kozniuk 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.]

48. The smallest solution to $\boxed{}\,23 \times 37 = \boxed{}\,23\boxed{}\boxed{}$ is $9523 \times 37 = 352351$. For each 10000 added to the individual amounts, add 370000 to the total.

49. 347×67. (Only ten cases 314, 330, $\ldots$ 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.

tagging algorithm
double-tagged
bucket-sort
Historical notes
Porges
Iséki
Gardner
Patton, Sr.
Patton, Jr.
Madachy
Schwartz
Fine
multiset product ($\Cap$)
sorting
Kozniuk
Hunter
Madachy

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

$$\begin{array}{r}\square\square\square\\ \times\ \square\square\square\\ \hline \square7\square\square\\ 77\square7\\ 77\square7\\ \hline \square\square\square\square\square\square\end{array}\qquad
\begin{array}{r}\square\square\square\\ \times\ \square\square\square\\ \hline \square\square7\square\\ 77\square7\\ 77\square7\\ \hline \square\square\square\square\square\square\end{array}\qquad
\begin{array}{r}\square\square\square\\ \times\ \square\square\square\\ \hline 777\square\\ \square7\square\square\\ 777\square\\ \hline \square\square\square\square\square\square\end{array}\qquad
\begin{array}{r}\square\square\square\\ \times\ \square\square\square\\ \hline 777\square\\ 777\square\\ \square7\square\square\\ \hline \square\square\square\square\square\square\end{array},\quad \square\neq7.$$

But they're even *easier* than (35), 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:

(always $\square\neq7$)

$$\begin{array}{r}\square\square\square\square\square\\ \times\ \square\square\\ \hline 777\square7\square\\ \square7\square77\square\\ \hline \square\square\square\square\square\square\square\end{array}\qquad
\begin{array}{r}\square\square\square\square\\ \times\ \square\square\square\\ \hline \square\square77\square\\ \square\square77\square\\ \square77\square7\\ \hline \square\square\square\square\square\square\square\end{array}\qquad
\begin{array}{r}\square\square\square\square\\ \times\ \square\square\square\\ \hline \square7\square7\square\\ \square7\square77\\ \square7\square7\square\\ \hline \square\square\square\square\square\square\square\end{array}\qquad
\begin{array}{r}\square\square\square\\ \times\ \square\square\square\square\\ \hline 77\square\\ \square7\square7\\ \square7\square\\ 77\square\\ \hline \square\square\square\square\square\square\end{array}$$

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.]

$$\begin{array}{r}\square\square\\ \times\ \square\square\square\\ \hline \square\square\square\\ \square\square\\ \square1\\ \hline \square\square\square\square\\ \square\neq1\end{array}\qquad
\begin{array}{r}\square\square\\ \times\ \square\square\square\\ \hline \square\square\square\\ \square\square2\\ \square2\\ \hline \square\square\square\square\\ \square\neq2\end{array}\qquad
\begin{array}{r}\square\square\\ \times\ \square\square\square\\ \hline \square3\square\\ \square\square3\\ 3\square\\ \hline \square\square\square\square\\ \square\neq3\end{array}\qquad
\begin{array}{r}\square\square\square\\ \times\ \square\square\\ \hline \square44\square\\ \square\square44\\ \hline \square\square\square\square\square\\ \square\neq4\end{array}$$

$$\begin{array}{r}\square\square\square\\ \times\ \square\square\square\\ \hline \square5\square\square\\ 555\square\\ \square\square5\square\\ \hline \square\square\square\square\square\square\\ \square\neq5\end{array}\qquad
\begin{array}{r}\square\square\\ \times\ \square\square\square\\ \hline 66\square\\ \square66\\ \square66\\ \hline \square\square\square\square\square\\ \square\neq6\end{array}\qquad
\begin{array}{r}\square\square\square\square\\ \times\ \square\square\square\\ \hline \square8\square8\square\\ 8\square8\square8\\ 8\square8\square8\\ \hline \square\square\square\square\square\square\square\\ \square\neq8\end{array}\qquad
\begin{array}{r}\square\square\square\square\\ \times\ \square\square\square\square\\ \hline \square9\square9\square\\ 9\square99\\ 9\square99\\ \square\square9\square\square\\ \hline \square\square\square\square\square\square\square\square\\ \square\neq9\end{array}$$

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

backtrack tree
author
Yamamoto
Okoma
Take
Maruo

decimal arithmetic
bignum
stamping+
inverse permutation
deletion from lists
sparse-set representation
sort

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 $t = s + \max_{0 \le j \le m}(o_j - t_j)$.

(By the way, the answers to the given puzzles are 237457×720845 and $\mathrm{K} = 9$; 467224×62581 and $\mathrm{K} = 3$. Another nice puzzle with slack 0 is answered by 38522×3597001 and $\mathrm{K} = 6$. There are none with slack 0 and $z = 0$.)

55. Take the skeleton of 38522×3597, with $\mathrm{K} = 6$.

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 $x_0 x_1 \ldots x_{N-1}$, with (say) $N \approx 25$; the value of x is $(x_t \ldots x_1)_{10}$, where $t = x_0$, and $x_t \ne 0$ 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 $\mathtt{JA}[l][j]$ of bignums, representing $j \cdot (a_l \ldots a_0)_{10}$ at level l of the algorithm, for $0 \le j < 10$. Clearly $\mathtt{JA}[l][j] = \mathtt{JA}[l-1][j] + 10^l(j \cdot a_l)$ when $l > 0$. (See (41) and (42); 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 $\mathtt{STAMP}[l][j]$ by which we can tell if $\mathtt{JA}[l][j]$ is valid (see below).

Next there's $\mathtt{CHOICE}[k]$, for $0 \le k < m$, which is a permutation of $\{0, 1, \ldots, 9\}$; also $\mathtt{WHERE}[k]$, which is the inverse permutation. (Thus $\mathtt{CHOICE}[k][i] = j$ if and only if $\mathtt{WHERE}[k][j] = i$.) The multiplier digits that haven't been ruled out by constraint p_k at level l are the first $\mathtt{S}[l][k]$ elements of $\mathtt{CHOICE}[k]$, namely the elements j such that $\mathtt{WHERE}[k][j] < \mathtt{S}[l][k]$. 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 $\mathtt{ID}[k] = \mathtt{ID}[k']$. 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 (36)). 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 $l = 0$.

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

At the beginning of step B2, set $\mathtt{NODES} \leftarrow \mathtt{NODES}+10$. If $\mathtt{S}[l][k] = 1$ 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 $\mathtt{JA}[l-1][\mathtt{CHOICE}[k][0]]$ for $0 \le k < m$.)

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 $s \leftarrow \mathtt{S}[l][k]$. For $0 \le i < s$, set $j \leftarrow \mathtt{CHOICE}[k][i]$ 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 $j' \leftarrow \mathtt{CHOICE}[k][s]$, $\mathtt{CHOICE}[k][i] \leftarrow j'$, $\mathtt{WHERE}[k][j'] \leftarrow i$, $\mathtt{CHOICE}[k][s] \leftarrow j$, $\mathtt{WHERE}[k][j] \leftarrow s$, and $i \leftarrow i - 1$. If there was no exit to B4, set $\mathtt{S}[l+1][k] \leftarrow s$; also, if $s = 1$ and $\mathtt{S}[l][k] > 1$, set $\mathtt{STACK}[p] \leftarrow k$, $p \leftarrow p + 1$.

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

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

After the stack is empty, we also want to test the overall product constraint p_m, if $P = \prod_{k=0}^{m-1} \mathtt{S}[l+1][k]$ 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 $=7$), 9615958×3568 (V $=7$), $43441858589 \times 3209004$ (W $=7$), 6495974×7546 (X $=8$), 128052×3975 (Y $=6$), $10740737877 \times 800300059$ (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.)

Feynman
Cheney
NP-complete
solvable
unique solution
Matsui
Sequential lists
dancing links
Loyd

66. (a, b) The skeleton shown here is solvable for $\square \neq @$ 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.]

$$\begin{array}{r}
\square @ \square \\
\times\ \square\square @ \square \\
\hline
\square\square\square\square \\
\square @ \square\square \\
\square\square @ \\
\square\square @\ @ \\
\hline
\square\square\square\square @ \square\square
\end{array}$$

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 400101001001001011001010010100100001000 1101 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, transform the problem as in exercise 7.2.2.1–34.

(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.)

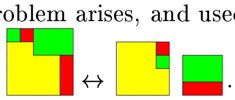
80. If we use k colors for the 4×4 square and $4-k$ for the 3×3, the minimum score is $16^2+3^2+3^2+3^2$ for $k=1$, $8^2+8^2+4^2+5^2$ for $k=2$, and $5^2+5^2+6^2+9^2$ for $k=3$.

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

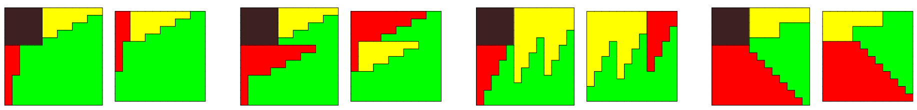
83. Pixels (x,y) and $[x,y]$ can be represented internally as integers such as $32x+y$. Then shifts $\sigma_{a,b}$ can be represented as $32a+b$, and computations are readily done with one-dimensional arrays. To generate valid sequences $\beta_1 \le \cdots \le \beta_d$, it's helpful to precompute the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

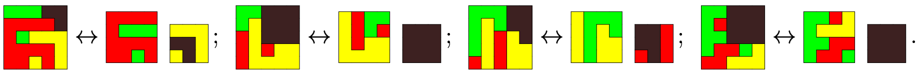


[The rightmost is due to Sam Loyd in the *Philadelphia Inquirer*, 26 February 1899.]

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

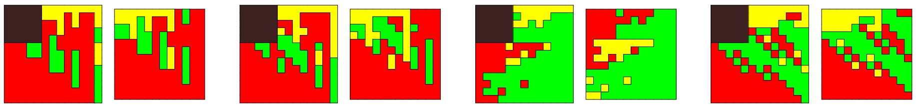
$(x,y)\sigma'_{a,b} = \langle x+a, y+b\rangle$. For example, ϕ_4 becomes $\sigma'_{-2,-2}$ in (53), instead of $\sigma_{3,-2}$. The number of feasible shifts is now $113 = 8^2+7^2 = (w+u-1)^2+(w+v-1)^2$, not 92.

(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).

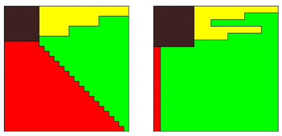
(b) Tens of thousands of new solutions arise, but they introduce only two new triples (ϕ_1, ϕ_2, ϕ_3) of usable shifts. "Random" examples are:



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 $(u,v,w) = (15,8,17)$.



89. Shisheng Li has proved [to appear] that the case (20, 21, 29) is impossible. The next smallest cases not known to have a rookwise 4-piece dissection are (28, 45, 53); (33, 56, 65); (48, 55, 73).

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`, `H`, `K`, `M`, `N`, `Q`, `R`, `S`, `V`, `W`.

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`, `W`, `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 shape and the number of pieces.

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/`.

Loyd
Li
author
Rookwise connectivity
kingwise connectivity
flipped
Demaine
Demaine
Uno

Stappers
rookwise neighbors
nondeterministically

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

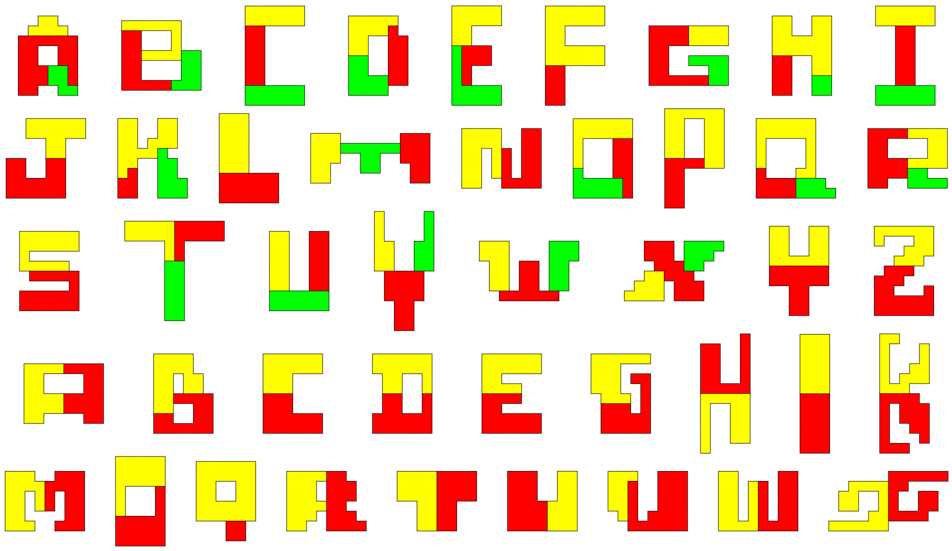
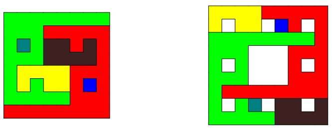


Fig. A–33. Fabricating a less constrained alphabet from a 6×6 square.

95. (Solution by Filip Stappers, 2024.)



96. 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.)

999. . . .

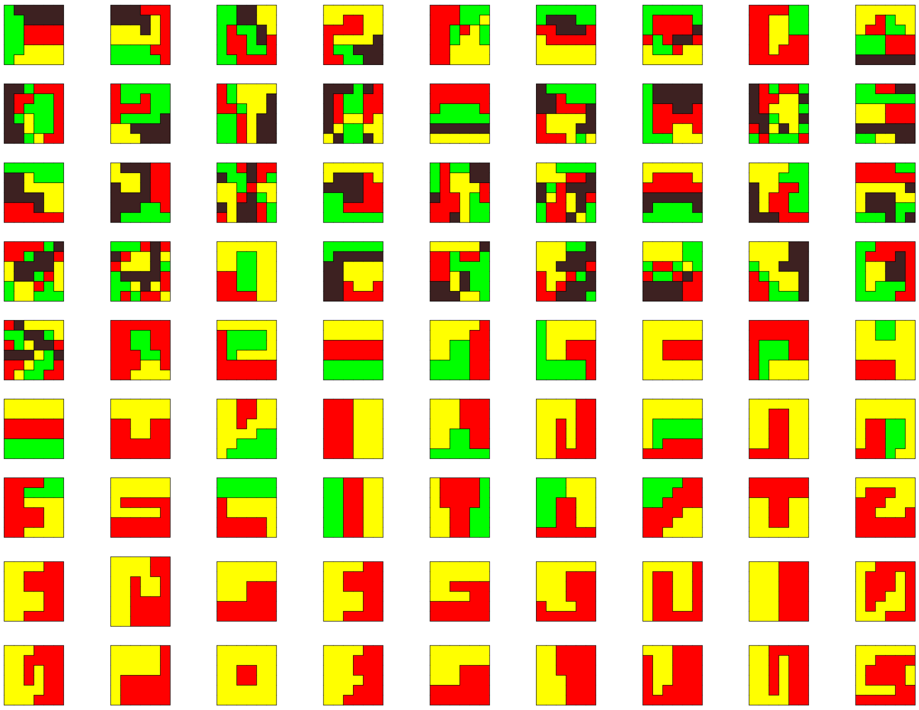


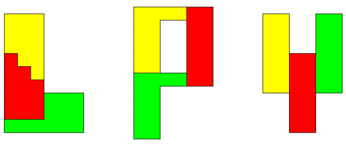
Fig. A–34. How to dissect 6 × 6 squares in order to obtain Figs. A–32 and A–33.

APPENDIX E

ANSWERS TO PUZZLES IN THE ANSWERS

$863 \times 992,\ 863 \times 996,\ 864 \times 929,\ 864 \times 299$
$86386 \times 29,\ 4193 \times 944,\ 5959 \times 838,\ 193 \times 4394$ (see answer 51)

$27 \times 329,\ 18 \times 498,\ 17 \times 298,\ 816 \times 93$
$794 \times 872,\ 83 \times 228,\ 9092 \times 992,\ 3233 \times 4336$ (see answer 52)



(see answer 92)

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.