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 7A

CONSTRAINT SATISFACTION
(very preliminary draft)

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY

January 29, 2023

Internet
Stanford GraphBase
`MMIX`

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

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

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

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

Zeroth printing (revision -42), 23 January 2023

BARRY
Internet

PREFACE

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

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

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

To put the material in context, this portion of fascicle 7 previews Section 7.2.2.3 of *The Art of Computer Programming*, entitled "Constraint satisfaction." It will be the first section of Volume 4C. As usual, it covers many topics that are of independent interest and that have close ties to other sections. I haven't had time yet to write a more detailed preface to the subject, but I encourage curious readers to browse the pages and take a look at the illustrations that I've accumulated so far.

* * *

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 52, 66, 87, 105, 108, 121, 131, 150, 187, 200, 319, 366, 372, . . . ; I've also implicitly mentioned or posed additional unsolved questions in the answers to exercises 51, 83, 115, 118, 119, 125(d), 129(e), 150, 180, 197, 211, 290, 313, 321, 365(b), 390(e,f), 404, 457, ... Are those intriguing problems still open? Please inform me if you know of a solution to any of them. And of course if no solution is known today but you do make progress on any of them in the future, I hope you'll let me know.

I urgently need your help also with respect to some exercises that I made up as I was preparing this material. I certainly don't like to receive credit for things that have already been published by others, and most of these results are quite natural "fruits" that were just waiting to be "plucked." Therefore please tell me if you know who deserves to be credited, with respect to the ideas found in exercises 12, 40, 41, 42, 43, 44, 45, 47, 50, 51, 52, 55, 64, 65, 67, 68, 69, 70, 71, 72, 73, 74, 76, 78, 79, 80, 81, 91, 99, 112, 115, 116, 117, 118, 119, 120, 121, 123, 125, 126, 132, 133, 137, 145, 146, 148, 149, 160, 165, 168, 169, 172, 177, 185(c), 187, 194, 199, 211, 214, 221, 222, 224, 229, 235, 255, 259, 260, 274, 283, 291, 297, 298, 304, 306, 311, 333, 341, 352, 353, 354, 356, 371, 380, 381, 382, 383, 384, 390, 400, 401, 402, 403, 404, 451, 452, . . . , and their answers. Furthermore I've credited exercises 36, 148, and . . . to unpublished work of Ira Gessel, Nikolai Beluhov, and . . . . Have any of those results ever appeared in print, to your knowledge?

Can anybody help me identify the source of the crystal maze puzzle? (The answer to exercise 20 tells what I know so far.)

* * *

Special thanks are due to Christian Bessière, Daniel Horsley, Peter Jeavons, Ciaran McCreesh, Patrick Prosser, George Sicherman, Christine Solnon, Filip Stappers, Peter Stuckey, Kokichi Sugihara, James Trimble, Udo Wermuth, and . . . for their detailed comments on my early attempts at exposition, as well as to numerous other correspondents who have contributed crucial corrections. I also thank my wife for help with Fig. 100.

* * *

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:–)

The answers to several of the exercises refer to programs that I wrote while preparing this material. If you want to see a program called FOO, look for FOO on the webpage `http://cs.stanford.edu/~knuth/programs.html`. (Many other example programs can also be found there.)

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

Gessel
Beluhov
Bessière
Horsley
Jeavons
McCreesh
Prosser
Sicherman
Solnon
Stappers
Stuckey
Sugihara
Trimble
Wermuth
Knuth, Jill
internet
downloadable programs

Knuth
TARJAN

Happy reading!

Stanford, California
99 Umbruary 2019

D. E. K.

The field of combinatorial algorithms is too vast
to cover in a single paper or even in a single book.
— ROBERT ENDRE TARJAN, *SIAM Review* (1978)

A foretaste of Section 7.5.1. Section 7.2.2.3 refers forward to the Hopcroft–Karp algorithm, which will be discussed at the beginning of Section 7.5.1 ("Bipartite matching"), according to present plans. That algorithm is copied here for reference. (Further details and exposition can be found in prefascicle 14a, on the Internet at `http://cs.stanford.edu/~knuth/fasc14a.ps.gz`.)

Hopcroft
Karp
Internet
matching
dags
data structures
mate tables
depth-first process
sparse graph representation
edge nodes
maximum cardinality matching

We're given a bipartite graph. The vertices of one part are called "girls" and the vertices of the other part are called "boys," so that we can conveniently use the English language to distinguish the parts. The problem is to find a *maximum matching*, namely a set of disjoint edges that is a large as possible.

Hopcroft and Karp's algorithm constructs dags (directed acyclic graphs) of SAPs (shortest augmenting paths), as explained in that prefascicle. The following implementation uses an interesting combination of data structures. First there are "mate tables" to represent the current matching, with GMATE[g] for $1 \le g \le M$ and BMATE[b] for $1 \le b \le N$ to indicate the partners of girl g and boy b, or 0 if they're currently free.

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

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

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

A similar convention is used to represent the dag, which is constructed dynamically: The first arc from boy b in the dag is BLINK[b], for $1 \le b \le N$, and the next is BNEXT(BLINK[b]), etc. The contents of BLINK and BNEXT are therefore *not* fixed. Every girl g in the dag is the source of exactly one arc, which leads to GMATE[g]. If GMATE[g] $= 0$, that arc leads to $\perp$.

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

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

backtracking
breadth-first
depth-first

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

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

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

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

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

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

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

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

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

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

H12. [Enter level l.] Set $\mathtt{STACK}[l] \leftarrow b$.

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

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

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

H16. [Prepare to augment.] (At this point $l = 1$; $g = g_0$ and $\mathtt{STACK}[1] = b_1$ in a SAP. The other boys are $\mathtt{STACK}[2]$, ..., $\mathtt{STACK}[L]$.) Set $f \leftarrow f-1$, $k \leftarrow \mathtt{IQUEUE}[g]$, $i \leftarrow \mathtt{QUEUE}[f]$, $\mathtt{QUEUE}[k] \leftarrow i$, and $\mathtt{IQUEUE}[i] \leftarrow k$. (Those operations removed g from the list of free girls.) Set $b \leftarrow \mathtt{STACK}[1]$.

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

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

Algorithm H comes with an important free bonus: After it has found a supposedly maximum matching, its data structures contain enough information

certificate of correctness
maximum independent set in bipartite graph

to convince any skeptic that the matching is indeed as large as possible. Indeed, if no girl is free, the matching is perfect and obviously optimum. Otherwise the girls in `QUEUE[k]` for $0 \le k < q$ are adjacent to only t boys in the graph, namely the boys in `MARKED[k]` for $0 \le k < t$. And it's easy to verify that $q = t + f$; hence any matching must leave at least f girls without a partner. Indeed, Algorithm H provides us with a maximum independent set,

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

which is certified by the maximum matching and vice versa!*

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

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

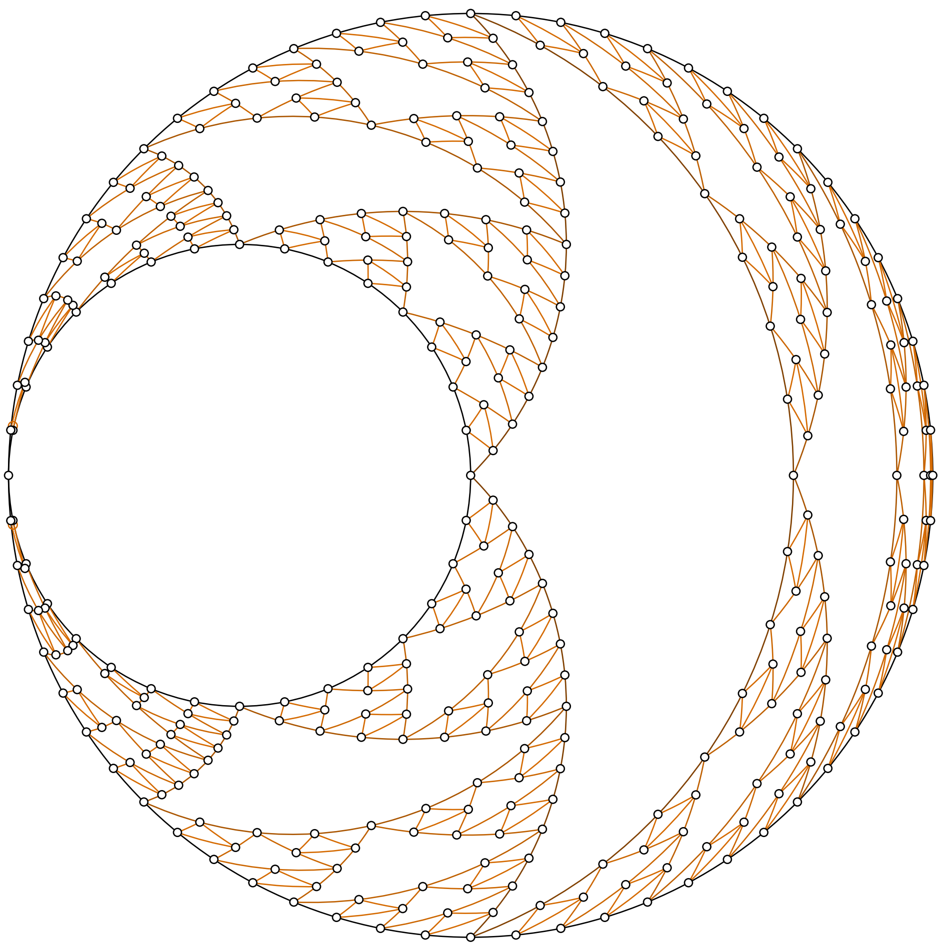
Proof. If the current matching has s' edges, we've observed that at least $s - s'$ vertex-disjoint augmenting paths exist. We also know that each of those paths contains at least $L+1$ edges of a maximum matching. So $s \ge (L+1)(s-s')$. ▮

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

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

* The complement of I is a vertex cover containing C vertices, where C is the size of the matching found. No vertex cover can contain fewer than C vertices; hence I is maximum.

pinched gasket



January 29, 2023

His Lady sad to see his sore constraint,
Cride out, Now now Sir knight, shew what ye bee.
— EDMUND SPENSER, *The Faerie Queene* (1590)

The work under our labour grows, luxurious by restraint.
— JOHN MILTON, *Paradise Lost* (1667)

Liberty exists in proportion to wholesome restraint.
— DANIEL WEBSTER (1847)

It is impossible to be an artist and not care for laws and limits.
Art is limitation; the essence of every picture is the frame.
— GILBERT K. CHESTERTON, *Orthodoxy* (1908)

I surround myself with obstacles.
Whatever diminishes my discomfort diminishes my strength.
The more constraints one imposes, the more one
frees one's self of the chains that shackle the spirit.
— IGOR STRAVINSKY, *Poétique musicale sous forme de six leçons* (1939)

Knight of Holinesse
SPENSER
MILTON
WEBSTER
CHESTERTON
STRAVINSKY
constraint satisfaction problem–
CSP: The constraint sat prob
XCC problems
exact covering with colors
items
options
SAT problems
Boolean satisfiability
satisfiability
literals
clauses
variables
domains
constraints
nogood

7.2.2.3. Constraint satisfaction. In Section 7.2.2.1 we solved numerous examples of XCC problems — exact covering with colors — which featured "items" and "options." Then in Section 7.2.2.2 we resolved lots of SAT problems — Boolean satisfiability — which featured "literals" and "clauses." All of these, and more, are instances of a combinatorial challenge that's more general yet, the *constraint satisfaction problem* — often called the CSP for short — which we will see is based on "variables," "domains," and "constraints."

The idea is simple: We're given a finite list of *variables* $(x_1, x_2, \ldots, x_n)$, to which we can assign values that belong to given finite *domains* $(D_1, D_2, \ldots, D_n)$. And we're also given a set of *constraints* $\{R_1, R_2, \ldots, R_m\}$, each of which specifies that a certain subset of the values $(x_1, x_2, \ldots, x_n)$ must be mutually compatible. Some combinations of values are "good"; the others are "nogood."

For example, let $n = 5$, and suppose that each domain is a set of letters:

$$D_1 = \{\mathtt{B},\mathtt{S}\},\;\; D_2 = \{\mathtt{C},\mathtt{L}\},\;\; D_3 = \{\mathtt{A},\mathtt{I},\mathtt{U}\},\;\; D_4 = \{\mathtt{E},\mathtt{O}\},\;\; D_5 = \{\mathtt{D},\mathtt{N}\}. \qquad (1)$$

Thus there are $2\times2\times3\times2\times2 = 48$ possible settings of $x_1x_2x_3x_4x_5$, from BCAED to SLUON. Let's also impose three constraints:

$$\begin{aligned} R_1(x_1,x_3,x_5) &= `x_1x_3x_5 \in \{\mathtt{BAN},\mathtt{BUD},\mathtt{SIN}\}\text{'};\\ R_2(x_1,x_4) &= `x_1x_4 \in \{\mathtt{BE},\mathtt{SE},\mathtt{SO}\}\text{'};\\ R_3(x_2,x_4,x_5) &= `x_2x_4x_5 \in \{\mathtt{COD},\mathtt{CON},\mathtt{LED}\}\text{'}. \end{aligned} \qquad (2)$$

This CSP has two solutions, easily found by hand (see exercise 1).

Every SAT problem is obviously a CSP in which all the domains are $\{0,1\}$. For example, problem $F = \{1\bar{2}, 23, \bar{1}\bar{3}, \bar{1}\bar{2}3\}$ in 7.2.2.2–(3) has four constraints,

$$x_1x_2 \in \{00,10,11\};\quad x_2x_3 \in \{01,10,11\};\quad x_1x_3 \in \{00,01,10\};$$
$$x_1x_2x_3 \in \{000,001,010,011,100,101,111\}. \qquad (3)$$

Conversely, every CSP can be formalized as an equivalent SAT problem, by using several SAT variables to represent each CSP variable x whose domain size d exceeds 2. For example, if the domain is $\{0, 1, \ldots, d-1\}$, Section 7.2.2.2 discussed the "log encoding," with $l = \lceil \lg d \rceil$ Boolean variables meaning that $x = (x_{l-1} \ldots x_1 x_0)_2$. There's also the "direct encoding," with d variables $x_k = [x = k]$, as well as the "order encoding," which has $d - 1$ variables $x^j = [x \geq j]$. We also discussed a variety of ways to represent arbitrary constraints, in the form of one or more clauses involving such Boolean variables. Each of those encodings has its own virtues and weaknesses, depending on the application.

Every XCC problem can, similarly, be regarded as a CSP. One way is to have a variable x_i for every primary item i, with domain D_i equal to the set of options that contain i. The constraints are that x_i and x_j cannot be options that conflict: If $x_i = o_i$ and $x_j = o_j$, where $o_i \neq o_j$, then o_i and o_j cannot have a common primary item, nor can they have a common secondary item that's colored differently in o_i and o_j. Conversely, exercise 7.2.2.1–100 presented one way to encode any CSP as an XCC problem.

Thus XCC, SAT, and CSP can each be reduced to the other two.

We've already learned how to construct excellent XCC solvers and excellent SAT solvers, so we might be tempted to stop there, regarding CSP as a problem that's already been well solved. But we shall see that careful consideration of the CSP not only clarifies XCC and SAT, it also teaches us important new methods.

Related models. Many groups of researchers have independently adopted conceptual frameworks that are identical to or very similar to the notions of variables, domains, and constraints. For example, a theory of *relational structures* has been developed as part of the branch of mathematics called "universal algebra." A relational structure is a set U together with a set $\{R_1, R_2, \ldots\}$ of relations or "predicates" defined on the elements of U. Each relation R_i depends on k elements, for some $k = k_i$, and it defines the k-tuples of elements for which that predicate is true. [See P. M. Cohn, *Universal Algebra* (1965), Chapter V.]

Let's be a little more precise. The *Cartesian product* of sets $(D_1, \ldots, D_n)$, denoted by $D_1 \times \cdots \times D_n$, is the set of all n-tuples $(x_1, \ldots, x_n)$ such that $x_i \in D_i$ for $1 \leq i \leq n$. Thus, $D_1 \times \cdots \times D_n$ is the set of all solutions to a CSP with domains $(D_1, \ldots, D_n)$, in the case when there are no constraints. An n-tuple such as $(x_1, \ldots, x_n)$ is often written simply as $x_1 \ldots x_n$, when commas aren't necessary. We also write $D \times \cdots \times D = D^n$ when the n domains are all identical.

A k-ary *relation* on sets $(D_1, \ldots, D_k)$ is a subset of $D_1 \times \cdots \times D_k$. We write either $R(x_1, \ldots, x_k)$ or $x_1 \ldots x_k \in R$ when we want to say that the k-tuple $(x_1, \ldots, x_k)$ satisfies relation R. The relation is called *binary* when $k = 2$, *ternary* when $k = 3$, *quaternary* when $k = 4$, and so on; it's *unary* when $k = 1$. (Strictly speaking, there also are *nullary* relations; see exercise 5.)

The simplest nontrivial relational structures arise where there's just a single binary relation. In fact, this case is so simple, we hardly ever think of it as a "relation structure" at all: We call it a *directed graph*. Indeed, we know well that a directed graph is a set V of *vertices*, together with a set $A \subseteq V \times V$ of

log encoding
Boolean variables
direct encoding
order encoding
primary item
options
secondary item
relational structures
universal algebra
predicates
Cohn
Cartesian product
tuples
commas
relation
binary
ternary
quaternary
unary
nullary
directed graph
vertices

arcs; and that's exactly what it means to be a relational structure with a single binary relation. This case is so common, we usually use the special notation $u \longrightarrow v$, instead of writing $A(u,v)$ or $uv \in A$.

Furthermore, if the lone binary relation is symmetrical (meaning that $u \longrightarrow v$ implies $v \longrightarrow u$) and irreflexive (meaning that $v \not\longrightarrow v$), we usually call it E instead of A; and we write $u \,—\, v$ instead of writing $E(u,v)$ or $uv \in E$. In such cases, of course, we have an ordinary (undirected) *graph*, and E is its set of *edges*.

Now let's consider *two* graphs, $G = (V,E)$ and $G' = (V',E')$. Suppose we attach a label $h(v)$ to every vertex $v \in V$, where $h(v)$ belongs to V'. This mapping $h : V \to V'$ is called a *homomorphism* if $E(u,v)$ implies $E'(h(u),h(v))$; in other words, it's a homomorphism if we have

$$h(u) \,—\, h(v) \text{ in } G' \qquad \text{whenever } u \,—\, v \text{ in } G. \tag{4}$$

For example, if G' is the complete graph K_d on vertices $V' = \{1,2,\ldots,d\}$, we have $j \,—\, k$ in G' if and only if $j \neq k$. So *h is a homomorphism from G to K_d if and only if it's a way to color the vertices of G properly with d colors.*

Going the other way, suppose G (not G') is the complete graph K_d. It's easy to see that *h is a homomorphism from K_d to G' if and only if the vertices $\{h(1),\ldots,h(d)\}$ form a d-clique in G'.*

Things get even more interesting when there's more than one relation. If $S = (U, R_1, \ldots, R_t)$ and $S' = (U', R'_1, \ldots, R'_t)$ are relational structures, we say that S and S' are *similar* if R_i and R'_i both have the same "arity," for $1 \le i \le t$. (In other words, R_i and R'_i are both k_i-ary.) In such cases we define a homomorphism h from S to S' to be a mapping from U to U' such that

$$R_i(x_1,\ldots,x_{k_i}) \quad \text{implies} \quad R'_i(h(x_1),\ldots,h(x_{k_i})), \quad \text{for } 1 \le i \le t. \tag{5}$$

For example, consider the augmented graph structure $G^{\neq} = (V,E,\neq)$ whose relations include the nonequality relation '$\neq$' as well as the ordinary edge relation E. A homomorphism from $G^{\neq}$ to $G'^{\neq}$ now has *two* properties:

$$h(u) \,—\, h(v) \text{ in } G' \text{ whenever } u \,—\, v \text{ in } G; \quad h(u) \neq h(v) \text{ whenever } u \neq v. \tag{6}$$

Consequently G is *embedded* in G': The vertices $\{h(v) \mid v \in V\}$ and edges $\{h(u) \,—\, h(v) \mid u \,—\, v \text{ in } G\}$ form a subgraph of G' that's essentially a copy of G. If, for instance, G is the n-cycle C_n, h proves that G' contains an n-cycle.

Sometimes a k-ary relation is *constant*, meaning that it is satisfied by only a single k-tuple. One interesting example is the structure $S = (V, A, \{ab\})$, where (V,A) is a digraph with special vertices a and b. Then a homomorphism h from S to the relational structure $S' = (\{0,1\}, =, \neq)$ will tell us that $u \longrightarrow v$ implies $h(u) = h(v)$, and also that $h(a) \neq h(b)$. Hence every vertex v reachable from a will have $h(v) = h(a)$, and we can conclude that b is unreachable. Conversely, if b isn't reachable from a, such a homomorphism can easily be found.

The evident versatility of homomorphisms has led Peter Jeavons to define the *general combinatorial problem* (GCP) as follows: "Given a pair of similar relational structures S and S', is there a homomorphism from S to S'?" [See *Theo-*

arcs
symmetrical
irreflexive
graph
edges
homomorphism
complete graph
clique
similar
arity
nonequality relation
disequality, see nonequality
embedded
subgraph isomorphism, see embedded graphs
subgraph
copy
constant
reachable
Jeavons
general combinatorial problem
GCP

retical Computer Science **200** (1998), 185–204; see also T. Feder and M. Y. Vardi, *SICOMP* **28** (1998), 57–104.] Exercises 10–12 provide several further examples.

In particular, Jeavons observed that the CSP is indeed a special case of the GCP. To cast (1) and (2) in this framework, for example, we let

$$S = \bigl(\{1,2,3,4,5\},\{1\},\{2\},\{3\},\{4\},\{5\},\{135\},\{14\},\{245\}\bigr); \tag{7}$$
$$S' = \bigl(\{\mathtt{A},\ldots,\mathtt{Z}\},D_1,D_2,D_3,D_4,D_5,R_1,R_2,R_3\bigr); \tag{8}$$

here D_1 through D_5 are the domains in (1), and R_1 through R_3 are the tuples in (2). The general idea is to have only *constant* relations in S, and to put the domains and constraints into S'. A homomorphism h from S to S' will then give us the values $(h(1),\ldots,h(5)) = (x_1,\ldots,x_5)$ that simultaneously belong to the domains and satisfy the constraints.

Conversely, every GCP is readily seen to be a CSP. (See exercise 15.)

The CSP framework is also intimately connected with the theory of *relational databases*. Individual facts in such a database are sets of tuples, involving the values of variables called "attributes." For example, we might have five attributes called 'location', 'employee', 'manager', 'job', 'language', and three relations:

Departments			Layout		Personnel		
location	manager	language	location	job	employee	job	language
basement	Alice	norsk	basement	test	Chris	code	deutsch
basement	Udo	deutsch	solarium	test	Chris	code	norsk
solarium	Iris	norsk	solarium	code	Logan	test	deutsch

What combinations of (location, employee, manager, job, language) exist in this peculiar institution? They correspond precisely to the solutions to the CSP in (1) and (2)! Database theorists call this the *natural join* of the three relations. [See E. F. Codd, *CACM* **13** (1970), 377–387; H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book* (Prentice–Hall, 2002).]

***Statistical mechanics.** Similar ideas arise also when physicists conceive of the universe as a gigantic collection of discrete particles, each of which has its own "spin." If there are N particles, the overall state is then an N-tuple $\Sigma = \sigma_1 \ldots \sigma_N$ called a *configuration*, where σ_j is the jth spin. Different particles can have different kinds of quantized spins, belonging to a given finite space of possible values, exactly analogous to the domains in a CSP.

Every configuration Σ has an associated *energy* $E(\Sigma)$, which is usually the sum of contributions from particles that interact locally. For example, the "one-dimensional Ising model," formulated by W. Lenz and analyzed by his student E. Ising [*Zeitschrift für Physik* **31** (1925), 253–258], has the energy function

$$E(\Sigma) = -\sum_{j=1}^{N-1} \sigma_j\sigma_{j+1} - B\sum_{j=1}^{N} \sigma_j, \tag{9}$$

where each spin σ_j is ± 1, and where the constant B represents the strength of an external magnetic field. If $\sigma_{j-1} = \sigma_{j+1} = -\sigma_j$ and $B\sigma_j < 2$, particle j will tend to change its spin to match its neighbors, because that would reduce the energy.

Feder
Vardi
Jeavons
relational databases
attributes
natural join
join
Codd
Garcia-Molina
Ullman
Widom
Statistical mechanics
physics–
spin
configuration
energy
Ising model
Lenz
Ising

Any set of k-ary relations between particles can be used to define energy functions. So, in particular, we can cast the CSP of (1) and (2) into this mold, obtaining configurations in $D_1 \times \cdots \times D_5$ whose energy function is

$$E(\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5) = -[R_1(\sigma_1,\sigma_3,\sigma_5)] - [R_2(\sigma_1,\sigma_4)] - [R_3(\sigma_2,\sigma_4,\sigma_5)]. \qquad (10)$$

Here are the 48 possibilities, together with their associated energy levels:

Σ	$E(\Sigma)$	Σ	$E(\Sigma)$	Σ	$E(\Sigma)$	Σ	$E(\Sigma)$	Σ	$E(\Sigma)$	Σ	$E(\Sigma)$
BCAED	−1	BCUED	−2	BLIED	−2	SCAED	−1	SCUED	−1	SLIED	−2
BCAEN	−2	BCUEN	−1	BLIEN	−1	SCAEN	−1	SCUEN	−1	SLIEN	−2
BCAOD	−1	BCUOD	−2	BLIOD	0	SCAOD	−2	SCUOD	−2	SLIOD	−1
BCAON	−2	BCUON	−1	BLION	0	SCAON	−2	SCUON	−2	SLION	−2
BCIED	−1	BLAED	−2	BLUED	−3	SCIED	−1	SLAED	−2	SLUED	−2
BCIEN	−1	BLAEN	−2	BLUEN	−1	SCIEN	−2	SLAEN	−1	SLUEN	−1
BCIOD	−1	BLAOD	0	BLUOD	−1	SCIOD	−2	SLAOD	−1	SLUOD	−1
BCION	−1	BLAON	−1	BLUON	0	SCION	−3	SLAON	−1	SLUON	−1

To analyze such models, physicists essentially calculate the generating function $G(z) = \sum z^{E(\Sigma)}$, summed over all configurations Σ. In our case, for example, $G(z) = 2z^{-3} + 18z^{-2} + 24z^{-1} + 4$. But because physicists understand physics, they do this in an idiosyncratic way by setting $z = e^{-\beta}$, where β is the reciprocal of the "temperature." In other words, they calculate $\sum e^{-\beta E(\Sigma)}$, which they call the *partition function*; and they usually denote that sum by $Z(\beta)$.

Since the partition function is always a sum of positive terms, physicists consider the ratio $e^{-\beta E(\Sigma)}/Z(\beta)$ to be the *probability* of configuration Σ. [Such probability distributions were introduced in the 19th century by Ludwig Boltzmann; see, for example, the *Sitzungsberichte der Mathematisch-Naturwissenschaftlichen Classe der Kaiserlichen Akademie der Wissenschaften* **76** (Wien, 1877), 373–435.]

At high temperatures, β is near 0; hence all configurations are almost equally likely. But at low temperatures, β approaches ∞; then only the configurations with smallest possible energy, the so-called "ground states," are significant, because they are exponentially more probable than any other state. In our 48-state example, each of the configurations with energy -3 occurs with probability $\frac{1}{48} + \frac{13}{384}\beta + O(\beta^2)$ when $\beta \to 0$, but probability $\frac{1}{2} - \frac{9}{2}e^{-\beta} + O(e^{-2\beta})$ when $\beta \to \infty$.

Thus, in general, the solutions to a satisfiable CSP correspond to the ground states of the associated physical problem. And when the CSP is *unsatisfiable*, the ground states satisfy as many of the constraints as possible.

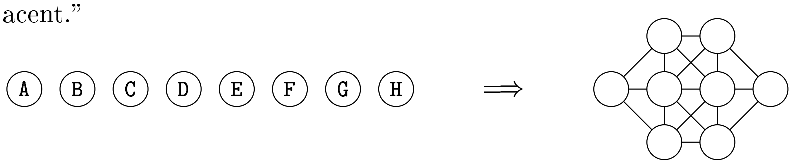
Considerations such as these account for the fact that physicists have contributed significantly to the understanding of random satisfiability problems, in particular by introducing Algorithm 7.2.2.2S. Further discussion of statistical mechanics is, of course, beyond the scope of a book on computer programming; but readers hungry for more may consult *Information, Physics, and Computation* by Marc Mézard and Andrea Montanari (Oxford University Press, 2009).

The takeway message from all these examples is obvious: There has to be something good about the CSP notions of variables, domains, and constraints, when we want to model real-world problems, because so many people have independently come up with essentially the same approach.

generating function
temperature
partition function
probability
Boltzmann
ground states
maxSAT
random satisfiability
Mézard
Montanari

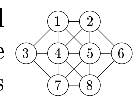
hello world
crystal maze puzzle–global
all-different

A simple example. To warm up, let's look at a little puzzle that appeared on a British TV show called *The Crystal Maze* in 1994. The task is simple—but you've got only two minutes to do it: "Place eight large disks, marked with the letters A through H, onto the eight circles shown; consecutive letters can't be adjacent."



(11)

We're actually facing two challenges here, namely (i) solve the puzzle; and (ii) express it as a constraint satisfaction problem, so that a computer can solve it for us. We'll tackle (ii), so as not to spoil the fun of (i). And we'll allow ourselves ten minutes, say, to accomplish goal (ii).



What are appropriate variables, domains, and constraints? We'd better label the vertices of the graph, so that we can readily describe what we want to define. One approach, based on the labeling shown, is to have eight variables $\{x_1, x_2, \ldots, x_8\}$, one for each vertex, each with domain $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{H}\}$. Then there are seventeen constraints, one for each edge of the graph; for example, the constraint for edge $1 — 2$ is

$$x_1x_2 \in \{\mathtt{AC,AD,AE,AF,AG,AH,BD,BE,BF,BG,BH,CA,CE,CF,CG,CH,DA,DB,DF,DG,DH,EA,}\\ \mathtt{EB,EC,EG,EH,FA,FB,FC,FD,FH,GA,GB,GC,GD,GE,HA,HB,HC,HD,HE,HF}\}, \quad (12)$$

and the same relation is used for all of the other edges. It can be written much more succinctly, if we assume that the letters are represented by integer codes:

$$|x_1 - x_2| \;>\; 1. \qquad (13)$$

OK, that took three minutes. Are we done? Well, no, actually; the seventeen constraints we've specified do not obviously rule out the possibility that $x_1 = x_8$. We're not allowed to put a disk on two different circles.

We could add eleven further constraints, namely $x_i \neq x_j$ for each of the yet-unconstrained pairs. But seasoned CSP solvers generally prefer to append a single *global* constraint instead, involving all of the variables at once:

$$x_1,\, x_2,\, x_3,\, x_4,\, x_5,\, x_6,\, x_7,\, x_8 \text{ are all different.} \qquad (14)$$

Indeed, special methods have been devised for the "all-different" constraint, because it arises in so many different problems. With (14), we've satisfied (ii).

Five minutes to go. Is there a better way? Another possibility is to let the variables be $\{\mathtt{A}, \mathtt{B}, \ldots, \mathtt{H}\}$, one for each *disk*, each with domain $\{1, 2, \ldots, 8\}$. Then only *seven* constraints are needed, one for each pair of consecutive letters; e.g.,

$$\mathtt{AB} \in \{16,17,18,23,27,28,32,35,36,38,46,53,61,63,64,67,71,72,76,81,82,83\}. \quad (15)$$

And each of these constraints has only 22 tuples, compared to 42 in (12). It's a win! Of course we also need the global all-different constraint. (See exercise 20.)

If we only had more time, we could have discovered a completely different way to model problem (11) as a CSP, such as the approach in exercise 23.

car sequencing problem
CSPLIB
benchmarks
Gent
Walsh
reflection
mirror images
symmetry breaking

Automating automobiles. We've already seen dozens and dozens of significant examples of constraint-based problems when we studied exact covering and SAT. But we certainly haven't exhausted all of the major applications, and several problems on our yet-unexamined list have been historically associated with the CSP. One of them, known as the *car sequencing problem*, is especially appropriate for us to study next, not only because its initials are "CSP" but also because it is problem 001 in CSPLIB — a noteworthy collection of benchmarks that was launched by I. P. Gent and T. Walsh in 1999 (see *LNCS* **1713** (1999), 480–481).

Consider the portion of an automobile assembly line where optional features are being installed on newly made vehicles. Some of the cars will be made with moonroofs; some will have heated seats; and so on. The assembly line is divided into work areas, one for each special feature. Work area i has space for q_i cars, where q_i is the number of time slots needed to install feature i as the conveyor belt moves the cars along. If at most p_i/q_i of the cars need that feature, p_i installers are on duty, one of whom will commence work when a car enters the area and walk with it until the installation is done. The car sequencing problem is the task of arranging a given set of cars into a sequence so that *no subsequence of q_i consecutive cars will include more than p_i that need feature i.*

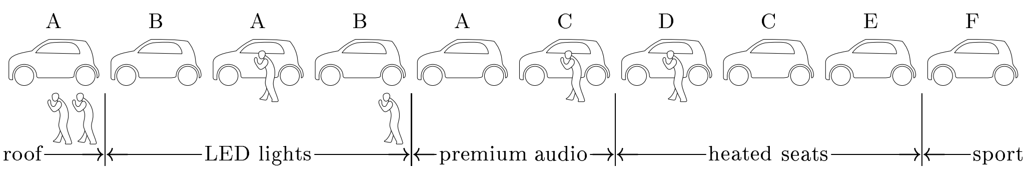


Fig. 100. Cars of models A, B, ... enter this assembly line at the far right, receiving optional features when they're in an appropriate work area. If this sequence has specifications (16), the final car (F) will be delayed in the LED area, because three cars in a row want that feature. The car sequencing problem tries to avoid such delays.

For example, there might be six models using the following subsets of five features:

Model	A	B	C	D	E	F	i	p_i	q_i
premium audio?	☐	☑	☑	☐	☐	☑	0	1	2
LED lights?	☑	☐	☑	☐	☑	☑	1	2	3
heated seats?	☐	☐	☐	☑	☐	☑	2	1	3
moonroof?	☐	☑	☐	☑	☑	☐	3	2	5
sport suspension?	☐	☐	☐	☐	☑	☐	4	1	5

(16)

Suppose ten cars of models $\{A, A, A, B, B, C, C, D, E, F\}$ are to be made. The sequence ABABACDCEF is almost correct, but it fails on the final car (see Fig. 100). Can you find a delay-free sequence? Notice that the left-right reflection of any solution is also a solution; we can rule out mirror images by requiring that model F, say, appears among the first five cars. Exercise 26 has the (unique) answer.

The car sequencing problem has boundary effects at the left and right that make it somewhat unrealistic. (Industrial assembly lines don't really start out empty every day!) Still, it's a nice clean problem, instructive to chew on.

One way to formulate the car sequencing problem in terms of variables, domains, and constraints is to have one variable x_i for every "time slot" in the assembly line sequence. The domain of each x_i is the set of model types, for $0 \le i < t$, where t is the total number of cars to be produced. We can also introduce t inverse variables, one for each vehicle, telling which slot it occupies; those variables have the domain $\{0, 1, \ldots, t-1\}$.

Our example of the 10 cars in Fig. 100 and (16) would therefore have 10 variables $\{x_0, \ldots, x_9\}$ with the 6-element domain $\{\mathrm{A}, \ldots, \mathrm{F}\}$, plus 10 variables $\{a_0, a_1, a_2, b_0, b_1, c_0, c_1, d, e, f\}$ with the 10-element domain $\{0, \ldots, 9\}$. These variables are related to each other by so-called "channelling constraints": For example, we can't have $a_1 = j$ unless $x_j = \mathrm{A}$; and in general the slot occupied by each vehicle must have the corresponding model type. We also constrain $a_0 < a_1 < a_2$, $b_0 < b_1$, and $c_0 < c_1$, so that vehicles of the same type are properly ordered in the overall sequence. (Notice that the number of ways to satisfy the stated constraints between these 20 variables is exactly $10!/(3!\,2!\,2!\,1!\,1!\,1!) = 151200$, which is the number of permutations of the multiset $\{\mathrm{A}, \mathrm{A}, \mathrm{A}, \mathrm{B}, \mathrm{B}, \mathrm{C}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}$. We could cut that number in half by requiring $f < 5$; see exercise 27.)

We also need constraints to rule out bad situations, like the subsequence $x_7x_8x_9 = \mathrm{CEF}$ that delays the lineup in Fig. 100. For this purpose it's convenient to introduce Boolean variables f_{ik} for $0 \le i < t$ and $0 \le k < m$, where m is the number of optional features and $f_{ik} = 1$ if and only if the car in slot i has feature k. There are channelling constraints between x_i and f_{ik}; for example, $x_i = \mathrm{B}$ implies that $f_{i0}f_{i1}f_{i2}f_{i3}f_{i4} = 10010$. The assembly-line constraints are then

$$f_{ik} + f_{(i+1)k} + \cdots + f_{(i+q_k-1)k} \le p_k, \quad \text{for } 0 \le i \le t - q_k \text{ and } 0 \le k < m. \tag{17}$$

For example, $x_7x_8x_9 = \mathrm{CEF}$ causes $f_{71}f_{81}f_{91} = 111$, violating $f_{71} + f_{81} + f_{91} \le 2$.

OK, it looks like we're done. Given any car sequencing problem with t cars and m features, we've now defined $t(2+m)$ variables, and devised sufficient constraints to characterize all the solutions. It turns out, however, that we could actually find those solutions much faster by adding *additional* constraints: If r_k is the total number of cars that will receive feature k, we must also have

$$f_{0k} + f_{1k} + \cdots + f_{(t-lq_k-1)k} \ge r_k - lp_k, \quad \text{for } 0 < l < \lceil r_k/p_k \rceil \text{ and } 0 \le k < m. \tag{18}$$

The reason is that the final lq_k cars in the sequence cannot account for more than lp_k of the total. (In our example, $r_1 = 7$; hence (18) gives f_{01} when $l = 3$; the first car cannot therefore be of type B or D.) The constraints in (18) are redundant, yet a computer might not be able to think of them, and they can significantly reduce the size of the search tree. (See exercise 32.)

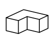
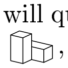
Of course the car sequencing problem can also be formulated as a CSP in many other ways, which will suggest themselves as we gain further experience.

Historical notes: Successful experiments with the car sequencing problem were first carried out by M. Dincbas, H. Simonis, and P. Van Hentenryck [*ECAI* **8** (1988), 290–295]. They were able to solve randomly generated problems with $t = 200$, $m = 5$, $(p_0/q_0, \ldots, p_4/q_4) = (1/2, 2/3, 1/3, 2/5, 1/5)$, and with overall utilization $r_k \approx .9tp_k/q_k$, by introducing the redundant constraints (18).

variables
domains
slot
inverse variables
channelling constraints
permutations of the multiset
multiset
Boolean variables
redundant
Historical notes
Dincbas
Simonis
Van Hentenryck

An international competition was held in 2005, based on actual industrial data. It included additional constraints, such as the colors of paint to be used and the initial contents of the assembly line, and it inspired many creative solutions. [See C. Solnon, V. D. Cung, A. Nguyen, and C. Artigues, *EJOR* **191** (2008), 912–927.] The winning programs were based on local search methods analogous to WalkSAT, using "greedy" heuristics.

Line labeling in computer vision. Speaking of history, let's turn now to some fascinating aspects of computer vision that influenced much of the early work on constraint processing. When a camera photographs a scene, it makes a two-dimensional image of three-dimensional reality; interesting problems arise when we try to reconstruct the reality from the image.

We'll work with an extremely simplified yet powerful model, as the original researchers did: Our "reality" will be a world of special polyhedral objects, where exactly *three faces* meet at each of the vertices. For example, an ordinary cube or tetrahedron or  will qualify. But an octahedron will not, nor will an Egyptian-style pyramid, nor , because a vertex where four faces meet isn't allowed. These three-faced concepts can be generalized, of course, but it's helpful to start with a thorough understanding of the comparatively simple trihedral world.

More precisely, the 3D objects we shall deal with have no curved surfaces. They are defined by vertices, edges, and faces, where the vertices are "corners" at which edges and faces come together. All of the faces are "flat," meaning that their points all lie on some plane. Each face is bounded by an exterior polygon, possibly with one or more interior polygons delimiting "holes" in the face. Each edge runs between two vertices and is part of the (infinite) line where the planes of two adjacent faces meet; it's a segment of the polygonal boundaries of those faces. And significantly, *each vertex is the endpoint of exactly three edges.* We shall call such an object a *three-valent polyhedral object*, or 3VP for short. (See Fig. 101.)

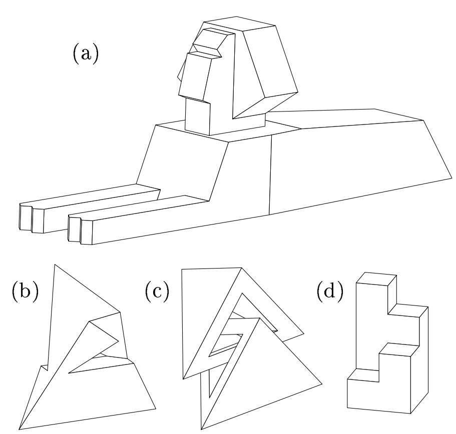
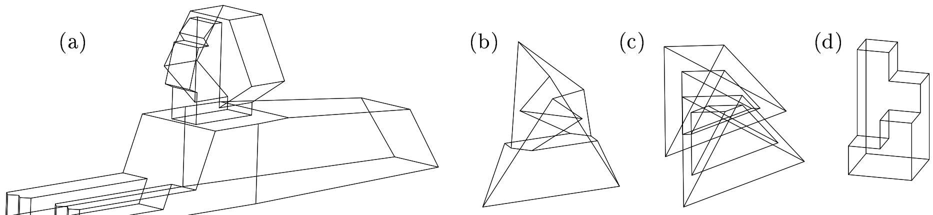


Fig. 101. Examples of 3VPs (three-valent polyhedra): (a) A stylized sphinx. [68 vertices, 102 edges, 38 faces.] (b) The Szilassi polyhedron, defined in exercise 39. Each of its seven faces is adjacent to all of the other six(!). [14 vertices, 21 edges, 7 faces.] (c) A clasp formed from two identical, interlocked objects, each of which is a tetrahedron from which a large triangular wedge has been hollowed out. [20 vertices, 30 edges, 14 faces.] (d) The histoscape for the matrix $\binom{4\,3}{1\,2}$, as defined in exercise 40. [20 vertices, 30 edges, 12 faces.] Many of the vertices, edges, and faces of these examples are invisible because they lie behind the parts that we *can* see.

The two-dimensional images shown here make sense to us, somehow, although significant depth information has been lost. In some mysterious way we've learned to rely on visual cues in order to understand what's really present.

competition
contest
real-world data
Solnon
Cung
Nguyen
Artigues
WalkSAT
greedy
computer vision
vision
photographs
scene
faces
octahedron
pyramid
three-faced
trihedral world
3D objects
vertices
edges
faces
three-valent polyhedral object
Szilassi polyhedron
histoscape

Huffman
Clowes
half edge
valley folds
origami
HC picture
junctions



(all of these 3D images are orthographic projections)

Fig. 102. If the objects in Fig. 101 were transparent, except for the edges, none of the edges would have been hidden. Each edge is a segment of a straight line, on the boundary between two adjacent faces. Exactly three of them meet at each vertex of a 3VP.

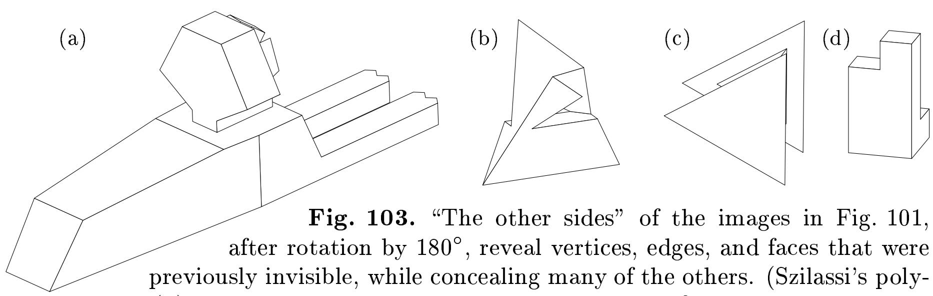


Fig. 103. "The other sides" of the images in Fig. 101, after rotation by 180°, reveal vertices, edges, and faces that were previously invisible, while concealing many of the others. (Szilassi's polyhedron (b) looks the same as before, because it has 180° rotational symmetry: The horizontal face is symmetrical, but the other three were visible only from behind.)

What are those visual cues? Working independently, D. A. Huffman and M. B. Clowes were able to decipher them successfully, in a pair of influential papers that were published at almost the same time [*Machine Intelligence* **6** (1971), 295–323; *Artificial Intelligence* **2** (1971), 79–116]. Given a 2D image that represents a 3VP X in a 3D scene, their first key idea was to classify each line segment by giving it one of four labels, according to its context:

- a *convex* edge (+), where points between the adjacent faces belong to X;
- a *concave* edge (−), where points between adjacent faces aren't part of X;
- a *half* edge (> or <), where only one of its adjacent faces can be seen.

(A half edge in the 2D image is actually a convex edge in X itself. But one of the two faces joined by this edge is invisible, because that face lies behind what we can see.) The label of a half edge is chosen so that the visible adjacent face appears to our *right* as we walk toward the point of the arrow.

For example, Fig. 104 is a marked-up version of Fig. 101, with all lines properly labeled. Convex edges are identified by tick marks, suggesting + signs; concave edges are shown as dashed lines, like the "valley folds" in standard origami diagrams; the half edges are decorated with arrows in the proper directions. Notice that the outer boundary in each case is a polygon that consists entirely of half edges, traversed clockwise.

Let's say that an *HC picture* is a list of distinct 2D points $j = (x, y)$, called "junctions," together with lines j — j' between designated junctions, for which (i) every junction has degree 2 or 3; (ii) two lines intersect only at junctions;

planar
Huffman
Clowes
orthographically
general position
projection

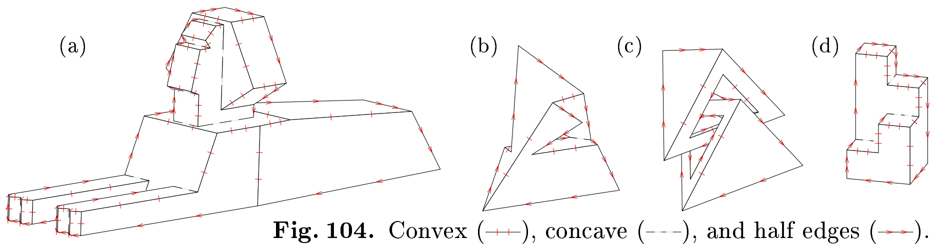


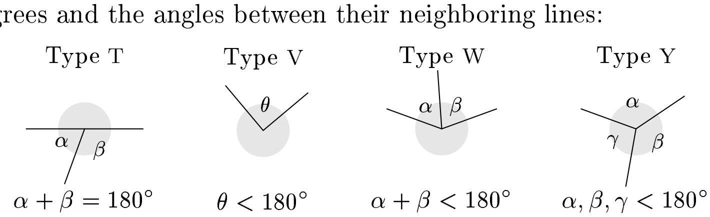
Fig. 104. Convex (—+—), concave (- - -), and half edges (—>—).

(iii) the two lines at a junction of degree 2 aren't collinear. Property (ii) means that the associated graph is planar. Property (iii) means that we can "see" all the junctions just by looking at the lines. (The "HC" in this definition stands for Huffman and Clowes.)

Given any 3VP X, suppose we project its vertices $v = (x, y, z)$ and edges v —— v' orthographically onto the (x, y) plane, eliminating hidden points and lines by assuming that (x, y, z) is in front of (x, y, z') whenever $z < z'$. We shall also assume that X is in *general position*, meaning that a slight rotation of X won't change the number of lines we see or the ways they relate to each other. (This assumption rules out exceptional cases that might occur accidentally, but with probability zero; exercise 57 has a formal definition.)

The resulting projection is always an HC picture, to which labels might be attached. For example, Figs. 101 and 103 are HC pictures, and Fig. 104 is a labeled HC picture. Every visible vertex of X appears as a junction in the HC picture. Furthermore, additional junctions are often present at the left of half edges, as artifacts of the projection process: We see them wherever an edge of X is partly hidden, but they aren't really intrinsic to X itself. (One such junction is below the middle of Fig. 104(d); Fig. 104(c) has 15 of them.)

The junctions of an HC picture can be classified into four types, based on their degrees and the angles between their neighboring lines:



(19)

(Type T junctions are the artifacts of projection, mentioned above.)

And now we get to the punch line, noticed independently by Huffman and Clowes: *When the lines of an HC picture are labeled with* + *or* - *or* > *or* <, *in order to distinguish between convex edges, concave edges, and half edges, only a small number of cases are actually possible, for each type of junction.* In fact,

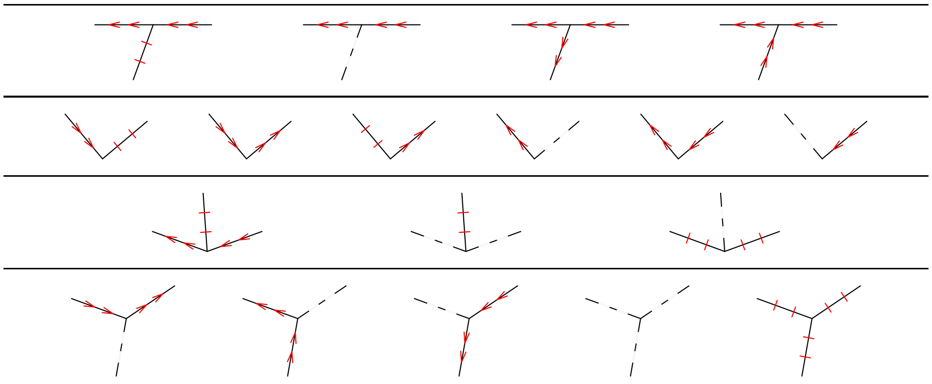
- A T junction can be labeled in only four ways (not $4^3 = 64$);
- A V junction can be labeled in only six ways (not $4^2 = 16$);
- A W junction can be labeled in only three ways (not $4^3 = 64$);
- A Y junction can be labeled in only five ways (not $4^3 = 64$).

That's part of the reason why we're able to perceive depth rather easily.

HC network
line labeling problem

Table 1

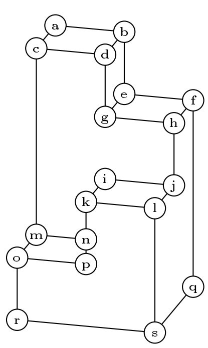
LEGAL LABELS FOR EACH JUNCTION TYPE



Exercise 58 works out the complete list of possibilities, exhibited in Table 1. And there's also more good news, a second punch line: Every line appears in two junctions, but has only one label; hence it's constrained at both ends!

Let's convert these geometric concepts to a purely combinatorial problem, by abstracting away the coordinates and considering only the underlying graph. We shall say that an *HC network* is a list of named junctions, where each junction is either $T(l,m,r)$, $V(l,r)$, $W(l,m,r)$, or $Y(a,b,c)$; here l, m, r, a, b, and c are the names of other junctions, and junction j' appears in the definition of j if and only if j appears in the definition of j'.

For example, here's the HC network that corresponds to Fig. 101(d):



$$\begin{array}{ll} \mathrm{a} = V(\mathrm{b},\mathrm{c}); & \\ \mathrm{b} = W(\mathrm{e},\mathrm{d},\mathrm{a}); & \mathrm{k} = W(\mathrm{i},\mathrm{l},\mathrm{n}); \\ \mathrm{c} = W(\mathrm{a},\mathrm{d},\mathrm{m}); & \mathrm{l} = Y(\mathrm{j},\mathrm{s},\mathrm{k}); \\ \mathrm{d} = Y(\mathrm{b},\mathrm{g},\mathrm{c}); & \mathrm{m} = Y(\mathrm{c},\mathrm{n},\mathrm{o}); \\ \mathrm{e} = Y(\mathrm{b},\mathrm{f},\mathrm{g}); & \mathrm{n} = T(\mathrm{k},\mathrm{m},\mathrm{p}); \\ \mathrm{f} = W(\mathrm{q},\mathrm{h},\mathrm{e}); & \mathrm{o} = W(\mathrm{m},\mathrm{p},\mathrm{r}); \\ \mathrm{g} = W(\mathrm{d},\mathrm{e},\mathrm{h}); & \mathrm{p} = V(\mathrm{o},\mathrm{n}); \\ \mathrm{h} = Y(\mathrm{f},\mathrm{j},\mathrm{g}); & \mathrm{q} = V(\mathrm{s},\mathrm{f}); \\ \mathrm{i} = V(\mathrm{j},\mathrm{k}); & \mathrm{r} = V(\mathrm{o},\mathrm{s}); \\ \mathrm{j} = W(\mathrm{l},\mathrm{i},\mathrm{h}); & \mathrm{s} = W(\mathrm{r},\mathrm{l},\mathrm{q}). \end{array} \qquad (20)$$

(Every HC picture has a unique HC network, except that the parameters of Y junctions can be permuted cyclically. For example, we could have written '$\mathrm{d} = Y(\mathrm{g},\mathrm{c},\mathrm{b})$' or '$\mathrm{d} = Y(\mathrm{c},\mathrm{b},\mathrm{g})$' instead of '$\mathrm{d} = Y(\mathrm{b},\mathrm{g},\mathrm{c})$' in (20); and there also are three equivalent ways to define each of the other Y junctions $\{\mathrm{e},\mathrm{h},\mathrm{l},\mathrm{m}\}$. But '$\mathrm{d} = Y(\mathrm{b},\mathrm{c},\mathrm{g})$' would be incorrect, because it doesn't match the HC picture. The branches of a Y must be listed in clockwise order.)

Given an HC network, the *line labeling problem* is to classify each of the lines between adjacent junctions as either convex (+), concave (−), or a properly

boundary cycle
standard

oriented half edge ($\texttt{<}$ or $\texttt{>}$), in such a way that every junction conforms to one of the patterns in Table 1. This is, of course, a constraint satisfaction problem: The variables are the lines; the domains are the symbols $\{\texttt{+},\texttt{-},\texttt{<},\texttt{>}\}$; and the constraints are given by Table 1.

For example, the line labeling problem for (20) has the 26 variables

$$\{\mathrm{ab},\mathrm{ac},\mathrm{bd},\mathrm{be},\mathrm{cd},\mathrm{cm},\mathrm{dg},\mathrm{ef},\mathrm{eg},\mathrm{fh},\mathrm{fq},\mathrm{gh},\mathrm{hj},\\ \mathrm{ij},\mathrm{ik},\mathrm{jl},\mathrm{kl},\mathrm{kn},\mathrm{ls},\mathrm{mn},\mathrm{mo},\mathrm{np},\mathrm{op},\mathrm{or},\mathrm{qs},\mathrm{rs}\} \qquad (21)$$

and the following 19 constraints:

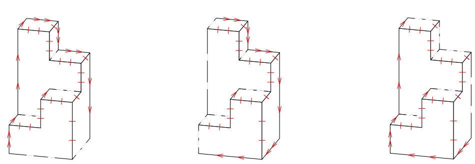
$$\begin{aligned}
(\mathrm{ab},\mathrm{ac}) &\in \{\texttt{<+},\texttt{<>},\texttt{+>},\texttt{>-},\texttt{><},\texttt{-<}\};\\
(\mathrm{be},\mathrm{bd},\mathrm{ab}) &\in \{\texttt{>+>},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{ac},\mathrm{cd},\mathrm{cm}) &\in \{\texttt{<+<},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{bd},\mathrm{dg},\mathrm{cd}) &\in \{\texttt{<->},\texttt{-<<},\texttt{>>-},\texttt{---},\texttt{+++}\};\\
(\mathrm{be},\mathrm{ef},\mathrm{eg}) &\in \{\texttt{<-<},\texttt{-<>},\texttt{>>-},\texttt{---},\texttt{+++}\};\\
(\mathrm{fq},\mathrm{fh},\mathrm{ef}) &\in \{\texttt{>+>},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{dg},\mathrm{eg},\mathrm{gh}) &\in \{\texttt{<+<},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{fh},\mathrm{hj},\mathrm{gh}) &\in \{\texttt{<->},\texttt{-<<},\texttt{>>-},\texttt{---},\texttt{+++}\};\\
(\mathrm{ij},\mathrm{ik}) &\in \{\texttt{<+},\texttt{<>},\texttt{+>},\texttt{>-},\texttt{><},\texttt{-<}\};\\
(\mathrm{jl},\mathrm{ij},\mathrm{hj}) &\in \{\texttt{>+>},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{ik},\mathrm{kl},\mathrm{kn}) &\in \{\texttt{<+<},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{jl},\mathrm{ls},\mathrm{kl}) &\in \{\texttt{<->},\texttt{-<<},\texttt{>>-},\texttt{---},\texttt{+++}\};\\
(\mathrm{cm},\mathrm{mn},\mathrm{mo}) &\in \{\texttt{<-<},\texttt{-<>},\texttt{>>-},\texttt{---},\texttt{+++}\};\\
(\mathrm{kn},\mathrm{np}) &\in \{\texttt{<<}\};\\
(\mathrm{mo},\mathrm{op},\mathrm{or}) &\in \{\texttt{<+<},\texttt{-+-},\texttt{+-+}\};\\
(\mathrm{op},\mathrm{np}) &\in \{\texttt{>+},\texttt{><},\texttt{+<},\texttt{<-},\texttt{<>},\texttt{->}\};\\
(\mathrm{qs},\mathrm{fq}) &\in \{\texttt{<+},\texttt{<<},\texttt{+<},\texttt{>-},\texttt{>>},\texttt{->}\};\\
(\mathrm{or},\mathrm{rs}) &\in \{\texttt{>+},\texttt{>>},\texttt{+>},\texttt{<-},\texttt{<<},\texttt{-<}\};\\
(\mathrm{rs},\mathrm{ls},\mathrm{qs}) &\in \{\texttt{<+>},\texttt{-+-},\texttt{+-+}\}.
\end{aligned} \qquad (22)$$

(Here '$\texttt{<+}$' stands for the ordered pair $(\texttt{<},\texttt{+})$; '$\texttt{>+>}$' stands for $(\texttt{>},\texttt{+},\texttt{>})$; and so on.)

Notice that the constraint for junction b was not written '$(\mathrm{be},\mathrm{bd},\mathrm{ba}) \in \{\texttt{>+<},\texttt{-+-},\texttt{+-+}\}$', because 'ba' isn't one of the variables: The line between junctions b and a is represented by 'ab' in (21). We could have had 52 variables $\{\mathrm{ab},\mathrm{ac},\mathrm{ba},\mathrm{bc},\ldots,\mathrm{sr}\}$ instead of 26, by introducing 26 further constraints such as $(\mathrm{ab},\mathrm{ba}) \in \{\texttt{++},\texttt{--},\texttt{<>},\texttt{><}\}$. But that would have wasted time and space.

Notice also that the constraint for junction n was not written '$(\mathrm{kn},\mathrm{mn},\mathrm{np}) \in \{\texttt{<+<},\texttt{<-<},\texttt{<<<},\texttt{<><}\}$'. The simpler and more direct statement in (22) is more efficient, and in fact it's the best way to understand the top row of Table 1.

The CSP in (22) is readily expressed as an XCC problem (see exercise 61), and it turns out to have just four solutions. The labeled picture in Fig. 104(d) represents the histoscape "floating in air"; the other three solutions



(23)

represent it "attached to the ground," or "attached to a wall" at the left or back.

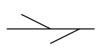
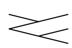
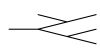
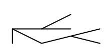
Every connected HC picture has a unique *boundary cycle*, consisting of the junctions that touch the "outside" region, in clockwise order. For example, the boundary cycle of (20) is (abefqsromc). A line labeling is called *standard* if every line between consecutive junctions of the boundary cycle has been labeled as a half edge pointing clockwise. That makes sense, because it means that the object lies entirely inside the boundary — unattached to any unbounded background environment. All four of the labelings in Fig. 104 are standard.

The sphinx of Fig. 101(a) has only two standard labelings, in spite of its numerous junctions and lines. The other possibility, besides Fig. 104(a), simply changes two of the labels so that the head isn't necessarily attached to the body.

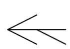
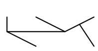
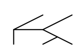
The Szilassi polyhedron, Fig. 101(b), likewise has exactly two standard labelings. (See exercise 62.) But Fig. 101(c) is far more ambiguous: It has 256 standard labelings. Indeed, three of its lines are completely unconstrained, because they're the stems between two T junctions.

A surprising thing happens when we ask for *all* valid labelings of Fig. 101, standard or not: The possibilities for the interior lines — the lines *not* between adjacent junctions of the boundary cycle — remain the same! More precisely, the number of ways to satisfy the constraints only at the boundary junctions turns out to be (720, 3, 6, 4), for Figs. 101(a), (b), (c), (d), respectively, while the total number of valid labelings is $(720\cdot 2, 3\cdot 2, 6\cdot 256, 4\cdot 1)$. In other words, all of the consistent boundary labelings are mutually interchangeable; hence the boundary can essentially be "factored out." When this happens we say that the HC picture has a *free boundary.* Not every picture has a free boundary, but exceptions seem to be rare in practice. Exercises 67–74 explore this curious phenomenon.

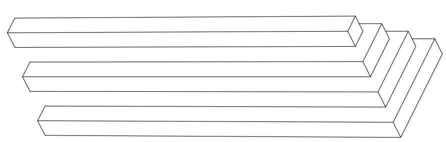
It's not difficult to construct HC pictures that *cannot* be labeled. For example, any picture that contains a subpicture of the forms

 (TT) or  (VTT) or  (YTT) or  (WTVT) (24)

will fail because each T junction forces two labels. Other impossible subpictures

 (WT) or  (WWT) or  (WWT) or  (WYT) (25)

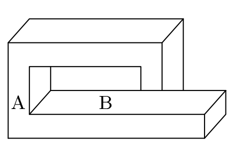
involve only one T; and exercise 76 has a small T-less example. The Swedish artist Oscar Reutersvärd has devised many amusing unlabelable pictures such as



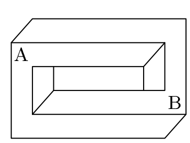
(26)

that fool our eyes when plausible side patterns are contradictory in the middle.

On the other hand, some HC pictures can be labeled perfectly, yet they don't correspond to any actual 3VP. Consider, for example, the pictures



or



, (27)

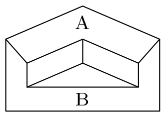
which look locally right although they're globally wrong. They "fail to compute" because each of them has two plane regions ('A' and 'B') that intersect in two *different* lines, contradicting a well-known principle of geometry.

A somewhat subtle distinction arises here, noted by Huffman in his original paper of 1971: There are locally consistent pictures that are globally inconsistent

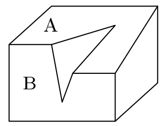
sphinx
Szilassi polyhedron
free boundary
Reutersvärd
intersection of planes+
Huffman

strongly realizable
weakly realizable
impossible
Huffman
truncated tetrahedron
rounding error
Penrose triangle
Penrose
Penrose

by virtue of the two-planes-determine-one-line principle, such as

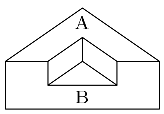


and

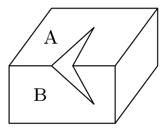


; (28)

yet certain globally *consistent* pictures have exactly the same HC networks:



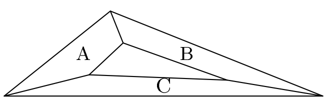
and



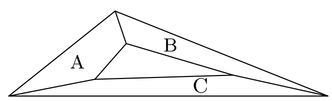
. (29)

Let's say that an HC picture H is *strongly realizable* if H is the projection of at least one 3VP X in general position. It is *weakly realizable* if there's an HC picture H' with the same HC network as H for which H' is strongly realizable. It is *impossible* if it's not weakly realizable. Thus, the pictures in (29) are strongly realizable; the pictures in (28) are weakly realizable; the picture in (26) is impossible. (The picture in (26) is not only impossible, it can't even be labeled.)

Huffman observed that a truncated tetrahedron gives another instructive example: Consider

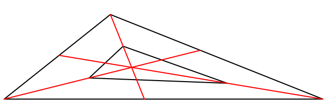


versus

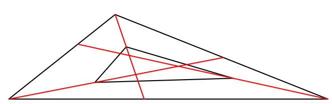


. (30)

The left picture is strongly realizable, but the right picture is not! In this case three planes are involved ('A', 'B', 'C'); three of the lines show the intersections of planes AB, BC, and CA. Those three planes always intersect in a single point, ABC, because no two of them are parallel. The relevant lines at the left of (30) do indeed share an invisible common point; but the lines at the right do not:



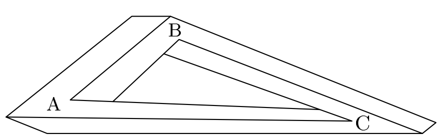
versus



. (30)

Thus we see that the notion of strong realizability is quite delicate — not at all robust: A tiny rounding error in one of the (x, y) coordinates can change a strongly realizable picture into one that can be realized only weakly.

The most famous impossible HC picture is probably the "Penrose triangle"

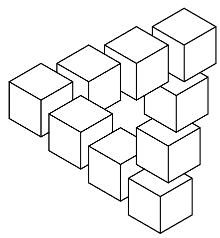


(31)

introduced by L. S. Penrose and R. Penrose in the *British Journal of Psychology* **49** (1958), 31–33. (Their version was slightly different: It was equilateral, and it included a few spurious "crack" lines.) Huffman's argument about nonconcurrent lines AB, BC, CA proves that (31) isn't even weakly realizable; and exercise 77 gives another proof of impossibility.

Reutersvärd
fun
HUFFMAN
Graph labeling
graceful labeling
complement

Oscar Reutersvärd, who is now known as the "father" of impossible pictures, discovered a paradoxical pattern akin to the Penrose triangle already in 1934:



(32)

This HC picture appears to be made of nine separate boxes that overlap in an impossible fashion. Surprisingly, however, it actually turns out to be strongly realizable! (See exercise 79.)

In fact, the theory of realizable objects is still far from complete, even when restricted to the 3VP world, and many fascinating problems remain to be solved.

I plead guilty to the charge that I deal with pictures of impossible objects because it is fun. It is, and that is reason enough. However, in addition to this I believe that much can be learned in the study of any language by asking 'Is that a nonsense sentence?' and 'Why is that a nonsense sentence?'.

— D. A. HUFFMAN (1971)

Graph labeling. Let's turn now to a completely different but equally fascinating way to attach labels to the vertices and edges of a graph. Our new goal is to give an identifying number to each vertex while simultaneously identifying each edge.

Consider, for example, Fig. 105(a), which is a graph of the 13 colonies that combined to form the original United States of America in 1776. Two vertices are adjacent if the corresponding colonies have a common boundary. Figure 105(b) shows that each colony can be represented by a cleverly chosen number, so that every edge is identified uniquely by the difference between the numbers of its endpoints:

$$\begin{array}{rrrrrr} 14-13=1 & 10-6=4 & 12-5=7 & 13-3=10 & 18-5=13 & 17-1=16 \\ 17-15=2 & 10-5=5 & 18-10=8 & 14-3=11 & 15-1=14 & 17-0=17 \\ 6-3=3 & 18-12=6 & 17-8=9 & 12-0=12 & 15-0=15 & 18-0=18 \end{array}. \qquad (33)$$

Numberings with this property are called "graceful." Formally speaking, if G is a graph with m edges, a *graceful labeling* of G is a function that assigns an integer $l(v)$ to each vertex v, in the range $0 \le l(v) \le m$, with the property that no two vertices have the same value of $l(v)$, and no two edges have the same value of $|l(v)-l(w)|$. We say that $l(v)$ is the label of vertex v, and $|l(v)-l(w)|$ is the label of edge $v \mathrel{-\!\!-} w$. Notice that $|l(v)-l(w)|$ is always positive, and it's at most $|m-0| = m$; therefore there's exactly one edge labeled d, for each d in $\{1,\ldots,m\}$.

Every graceful labeling has a "complement," obtained by setting $l(v) \leftarrow m - l(v)$ for all v. (See Fig. 105(c).) Complementation doesn't change the label of any edge. A labeling and its complement are considered to be essentially identical.

symmetry
automorphism
permutation
13-colonies graph
essentially different
puzzle
sudoku
data structure
isolated vertices

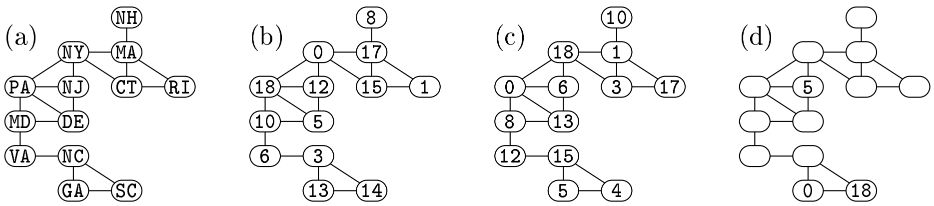


Fig. 105. (a) A famous graph G, which has 13 vertices and 18 edges. (b) One of the many graceful labelings of G. (c) The same labeling as (b), but complemented. (d) A puzzle: Complete this labeling to make it graceful. (The solution is unique.)

Every symmetry of a graph also preserves gracefulness. In other words, if α is an automorphism (a permutation of the vertices for which $v \mathrel{—} w$ implies $v\alpha \mathrel{—} w\alpha$), and if l is a graceful labeling, then the labeling $l'(v) = l(v\alpha)$ is also graceful. For example, Fig. 105(a) is symmetrical if we swap `GA` $\leftrightarrow$ `SC`; hence we could also swap the labels `13` $\leftrightarrow$ `14` in Fig. 105(b) and/or the labels `5` $\leftrightarrow$ `4` in Fig. 105(c). In this way every graceful labeling of the 13-colonies graph yields a set of four labelings that are mutually equivalent. (See exercise 91.)

That graph actually has hundreds of thousands of graceful labelings: 641952 altogether! Dividing by 4 gives us 160488 that are essentially different. They can be found quickly, using for example the XCC model of exercise 93. Each of the 18 edges can be the "longest," namely the edge that's labeled 18. That edge connects `NY` to `PA`, as it does in Fig. 105(b, c), in 22782 of those 160488 solutions; and it connects `NY` to `MA` in even more of them (24896). On the other hand only 24 of the 160488 have the longest edge between `GA` and `SC`, as in Fig. 105(d). (The latter labeling has been left as a puzzle, which we'll be discussing later. It's roughly as difficult as a "hard" sudoku, and we'll see that it can be solved by hand.)

A nice data structure can be used to represent a gracefully labeled graph inside a computer, using a few arrays of size $m + 1$. First, by including isolated vertices if necessary, we can assume that the vertices are named $0, 1, \ldots, m$, and that $l(v) = v$ for $0 \le v \le m$. (In other words, a vertex's label is also its name.) Then, if edge d connects vertices v and $v + d$, we set $\mathtt{LO}[d] \leftarrow v$. Consequently *two arbitrary vertices v and w with $v < w$ are adjacent if and only if* $\mathtt{LO}[w - v] = v$. With three further arrays, `FIRST`, `NEXTL`, and `NEXTH`, we can also visit all neighbors w of any given vertex v using a simple loop:

$$\text{Set } w \leftarrow \mathtt{FIRST}[v]\text{. While } w \ge 0, \text{ set } w \leftarrow \begin{cases} \mathtt{NEXTL}[v - w], & \text{if } w < v; \\ \mathtt{NEXTH}[w - v], & \text{if } w > v. \end{cases} \tag{34}$$

For example, the arrays might look like this in the case of Fig. 105(b):

$l =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
`LO`$[l] =$	—	13	15	3	6	5	12	5	10	8	3	3	0	5	1	0	1	0	0
`FIRST`$[l] =$	12	15	−1	6	−1	10	3	−1	17	−1	6	−1	18	14	13	17	−1	15	12
`NEXTL`$[l] =$	—	3	1	13	−1	12	5	18	−1	−1	14	−1	15	−1	17	17	−1	18	−1
`NEXTH`$[l] =$	—	3	8	10	5	18	10	0	5	1	−1	−1	−1	0	0	−1	0	−1	−1
`NAME`$[l] =$	NY	RI	—	NC	—	DE	VA	—	NH	—	MD	—	NJ	GA	SC	CT	—	MA	PA

(35)

(The `NAME` array shown here gives an optional *external* name for printouts. Entries marked '—' in these example arrays are unused.)

C_5
contiguous USA graph
`USA` graph
miracle
pi
Rokicki

When a graph has at least one graceful labeling it's called "graceful"; in that sense, the 13-colonies graph can be considered quite graceful indeed. Not all graphs are graceful, of course. For example, a disconnected graph with more than $m+1$ vertices can't possibly be graceful; there aren't enough labels to go around. And one can easily check that C_5, the 5-cycle, has no graceful labeling.

The 13-colonies graph is merely an induced subgraph of a much larger graph that we've been exploring in lots of examples in previous sections. The contiguous USA graph, introduced in 7–(17) and last seen in exercises 7.2.2.2–35 and 37, has 49 vertices and 107 edges. Could it possibly be graceful?

The answer is yes; and later in this section we shall discuss randomized techniques that are able to label it without a great deal of work. In fact, those techniques have revealed what can only be described as a *graceful miracle*: A solution can actually be achieved when the 15 states on the western and northern borders, from California to Maine, are labeled with the first 30 digits of pi!! This has to be seen to be believed (see Fig. 106).

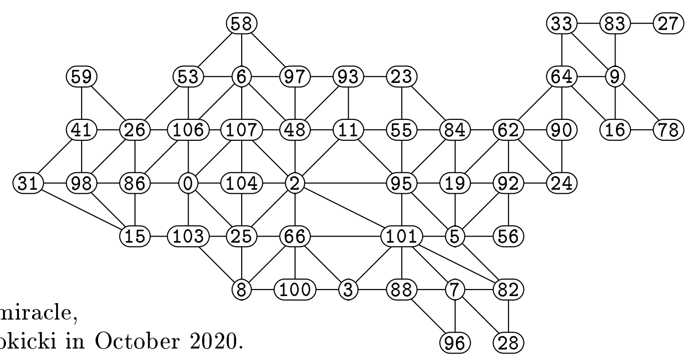


Fig. 106. A graceful miracle,
found by Tomas G. Rokicki in October 2020.

	$98-86=12$	$33-9=24$	$92-56=36$	$64-16=48$	$86-26=60$	$98-26=72$	$95-11=84$	$101-5=96$
$107-106=1$	$101-88=13$	$25-0=25$	$48-11=37$	$97-48=49$	$84-23=61$	$92-19=73$	$88-3=85$	$100-3=97$
$64-62=2$	$19-5=14$	$90-64=26$	$62-24=38$	$83-33=50$	$78-16=62$	$83-9=74$	$86-0=86$	$101-3=98$
$107-104=3$	$41-26=15$	$53-26=27$	$97-58=39$	$56-5=51$	$66-3=63$	$82-7=75$	$92-5=87$	$101-2=99$
$97-93=4$	$31-15=16$	$90-62=28$	$95-55=40$	$58-6=52$	$66-2=64$	$95-19=76$	$103-15=88$	$106-6=100$
$58-53=5$	$25-8=17$	$84-55=29$	$66-25=41$	$106-53=53$	$84-19=65$	$82-5=77$	$96-7=89$	$107-6=101$
$101-95=6$	$59-41=18$	$92-62=30$	$48-6=42$	$82-28=54$	$90-24=66$	$103-25=78$	$95-5=90$	$104-2=102$
$16-9=7$	$101-82=19$	$64-33=31$	$62-19=43$	$64-9=55$	$98-31=67$	$104-25=79$	$97-6=91$	$103-0=103$
$96-88=8$	$106-86=20$	$55-23=32$	$55-11=44$	$83-27=56$	$92-24=68$	$106-26=80$	$100-8=92$	$104-0=104$
$11-2=9$	$28-7=21$	$59-26=33$	$93-48=45$	$98-41=57$	$78-9=69$	$88-7=81$	$95-2=93$	$107-2=105$
$41-31=10$	$84-62=22$	$100-66=34$	$48-2=46$	$66-8=58$	$93-23=70$	$93-11=82$	$101-7=94$	$106-0=106$
$95-84=11$	$25-2=23$	$101-66=35$	$53-6=47$	$107-48=59$	$86-15=71$	$98-15=83$	$103-8=95$	$107-0=107$

The problem of labeling a given graph G of size m gracefully can be formalized as a CSP in many ways. For example, we can render the definition directly, by saying that the variables of the CSP are the vertices and edges of G; the domain of each vertex is $\{0,\ldots,m\}$ and the domain of each edge is $\{1,\ldots,m\}$; the constraints are that $l(e)=|l(v)-l(w)|$ when e is the edge v — w; furthermore the vertex labels should all be different and the edge labels should all be different.

domain
reverse model
complementation
unique graceful labeling
star graph
unique graceful labeling
path graph
graceful permutation

That direct model lets us solve small problems, of course. But experience shows that it doesn't scale up well. A much better method can be based on the `LO` and `NAME` arrays of the data structure in (35), where we take the attitude that vertex and edge labels are already given; our job is to attach them to the graph! More precisely, there's a variable for each vertex label in $\{0,\dots,m\}$; and they all have the domain $V \cup \{?\}$, meaning that each label l should be assigned a $\mathtt{NAME}[l]$, which is either a vertex of G or undefined. The defined labels should all be different. Furthermore, there's a variable for each edge label in $\{1,\dots,m\}$; and its value $\mathtt{LO}[l]$ has the domain $\{0,\dots,m-l\}$. The constraint is that

$$\mathtt{NAME}[\mathtt{LO}[l]] \mathbin{—\!—} \mathtt{NAME}[\mathtt{LO}[l]+l] \quad \text{is an edge of } G, \qquad \text{for } 1 \le l \le m. \tag{36}$$

Let's call this the "reverse model."

The reverse model has a big advantage, because $\mathtt{LO}[l]$ has a very small domain when l is large. Indeed, $\mathtt{LO}[m]$ must be 0; and $\mathtt{LO}[m-1]$ must be either 0 or 1. We can in fact assume that $\mathtt{LO}[m-1]=0$, because complementation changes $\mathtt{LO}[m-1]$ to $1-\mathtt{LO}[m-1]$. (See exercise 94.)

For example, the reverse model makes it easy to discover all of the graceful labelings when G is the complete graph K_n. In this case there are $m=\binom{n}{2}$ edges; and the constraint (36) is satisfied if and only if $\mathtt{NAME}[\mathtt{LO}[l]]$ and $\mathtt{NAME}[\mathtt{LO}[l]+l]$ are both defined, meaning that $\mathtt{LO}[l]$ and $\mathtt{LO}[l]+l$ are both among the n "real" vertices that belong to K_n.

If $n=1$, we're done: K_1 is graceful, with vertex 0.

Otherwise $m>0$ and $\mathtt{LO}[m]=0$. Hence 0 and m are real vertices, and we're done if $n=2$.

Otherwise $m>1$, and we may assume that $\mathtt{LO}[m-1]=0$ as stated above. That means $m-1$ is also real. So if $n=3$, we know that the three real vertices are $\{0,2,3\}$; hence $\mathtt{LO}[2]=0$ and $\mathtt{LO}[1]=2$. That settles K_3.

If $m>2$, edge $m-2$ is always either $0 \mathbin{—\!—} (m-2)$ or $1 \mathbin{—\!—} (m-1)$ or $2 \mathbin{—\!—} m$, and each case gives us a new real vertex. Consequently the four vertices when $n=4$ are either $\{0,4,5,6\}$, $\{0,1,5,6\}$, or $\{0,2,5,6\}$. Only the third alternative allows us to define $\mathtt{LO}[3]$ without introducing a fifth real vertex. That settles K_4.

Finally, if $n>4$, we get stuck (see exercise 95). So we've discovered that *K_n has a unique graceful labeling when $n \le 4$, but K_n is ungraceful when $n \ge 5$.*

The star graph $K_{1,n}$ is another instructive example. It consists of a central vertex that's joined to each of n other vertices; so it has lots of symmetry, like K_n, but it has only $m=n$ edges.

We might as well assume that $n>1$, because $K_{1,1}=K_2$. So we know that $\mathtt{LO}[n]=0$, and also $\mathtt{LO}[n-1]=0$. But that means 0 must be the central vertex, because no other vertex has more than one neighbor. Consequently $\mathtt{LO}[n-2]=0$, $\mathtt{LO}[n-3]=0$, and so on; *$K_{1,n}$ has a unique graceful labeling.*

That was easy. But what happens if G is the path graph P_n? A graceful labeling of P_n is called a *graceful permutation*, because P_n has $m=n-1$ edges, and the sequence $p_0p_1\dots p_{n-1}$ of labels on the path is a permutation of $\{0,1,\dots,n-1\}$. The permutation $p_0p_1\dots p_{n-1}$ is graceful if and only if

$$|p_0-p_1|\,|p_1-p_2|\,\dots\,|p_{n-2}-p_{n-1}| \quad \text{is a permutation of } \{1,\dots,n-1\}. \tag{37}$$

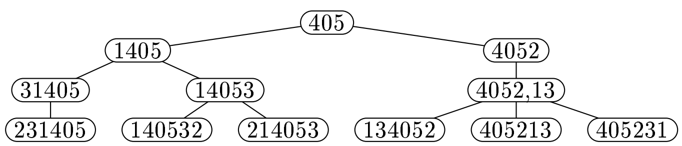
unique graceful labeling
reversal
complementation
data structure
ZDD
KP graphs
cliques

We know that P_3 has a unique graceful labeling, because $P_3 = K_{1,2}$. That fact can be confusing, because the six permutations $p_0p_1p_2$ of $\{0,1,2\}$ are

$$012,\ 021,\ 102,\ 120,\ 201,\ 210$$

and *four* of them satisfy (37)! Everything becomes clear, however, once we realize that the permutations $p_0p_1\ldots p_{n-1}$, $p_{n-1}\ldots p_1p_0$, $(n-1-p_0)\ldots(n-1-p_{n-1})$, and $(n-1-p_{n-1})\ldots(n-1-p_0)$ are considered to be essentially the *same*, because each of them is obtainable from the others by reversal and complementation. Similarly, the graceful labelings of P_4 and P_5 reduce to 1203 and either 21304 or 30421, which each represent four permutations. There are four times as many graceful permutations as there are ways to label P_n gracefully, when $n > 2$.

Let's take a look at P_6. We know that edges 5 and 4 will be $0 — 5$ and $0 — 4$, which we can abbreviate to 05 and 04, respectively. Thus $p_0p_1\ldots p_5$ will contain the substring 405 or 504, and we can assume that it's 405. Edge 3 must be 03 or 14 or 25; but 03 is impossible because 0 already has two neighbors. Two cases remain, 1405 and 4052. The tree of possibilities is, in fact,



(38)

as we choose edge 3, edge 2, then edge 1, leading to six solutions altogether.

Notice that this procedure chooses the values of `LO[5]`, `LO[4]`, `LO[3]`, ... sequentially. But it does *not* choose *any* values for the `NAME` array until the very last step. For instance, at one point in (38) we know that 4052 and 13, or their reflections, should be substrings of the final permutation; but we don't commit ourselves prematurely to exactly where those substrings will appear. Exercise 96 discusses a convenient data structure for dealing with such partial permutations.

The number of graceful permutations grows exponentially with n. For example, P_{41} can be labeled gracefully in 258,002,411,935,989,500 ways! Exercise 97 explains how a ZDD with fewer than 25 million nodes can represent them all.

Some dazzling patterns arise when we consider "KP graphs" of the form $K_n \mathbin{\square} P_r$, which consist of $r > 1$ cliques in a row, each of size $n > 2$. For example, here are two of the many graceful labelings of $K_4 \mathbin{\square} P_{10}$ and $K_5 \mathbin{\square} P_7$:

$$\begin{pmatrix} 0 & 96 & 4 & 93 & 5 & 90 & 11 & 88 & 22 & 84 \\ 1 & 3 & 13 & 65 & 89 & 14 & 62 & 25 & 81 & 58 \\ 91 & 9 & 87 & 7 & 77 & 50 & 18 & 72 & 51 & 69 \\ 95 & 28 & 73 & 12 & 55 & 17 & 82 & 33 & 68 & 27 \end{pmatrix};\quad \begin{pmatrix} 10 & 56 & 99 & 0 & 100 & 13 & 93 \\ 33 & 66 & 7 & 77 & 12 & 87 & 59 \\ 81 & 95 & 1 & 41 & 3 & 94 & 8 \\ 86 & 2 & 97 & 15 & 70 & 26 & 71 \\ 89 & 6 & 79 & 52 & 69 & 45 & 24 \end{pmatrix}. \qquad (39)$$

Each of the 10 columns on the left has six differences; in the first column they are $\{|0-1|, |0-91|, |0-95|, |1-91|, |1-95|, |91-95|\} = \{1, 91, 95, 90, 94, 4\}$. And each row also has nine differences between adjacent columns; in the first row they are $\{|0-96|, |96-4|, \ldots, |22-84|\} = \{96, 92, 89, 88, 85, 79, 77, 66, 62\}$. Those $60+36$ differences are all distinct! And so are the $70+30$ differences on the right!!

symmetries
automorphisms
complementation
break this symmetry
canonical form

In general $K_n \mathbin{\square} P_r$ has rn vertices and $m = r\binom{n}{2} + (r-1)n$ edges; that's exactly n^2 edges when $r = 2$. It has $2n!$ symmetries (aka automorphisms), because we can permute the rows of the matrix and/or reflect it left$\leftrightarrow$right.

Every graceful labeling of a KP graph can be represented as an $n \times r$ matrix (x_{ij}), for $1 \le i \le n$ and $1 \le j \le r$, as in (39). When complementation is taken into acount, every suitable matrix is therefore equivalent to a set of $4n!$ such solutions. The usual way to break this symmetry, in order to generate only inequivalent solutions, is to add additional constraints so that the matrix is in a "canonical form." For example, we can insist as above that 0 is adjacent to $m-1$, or at least it's in the same column, and that

$$x_{11} < x_{21} < \cdots < x_{n1}; \quad x_{11} < x_{1r}. \tag{40}$$

(See exercise 100.) The matrices in (39) are canonical in this sense. Constraints like (40), which significantly prune the search tree, are supposedly helpful.

But in this case a far more efficient approach is possible, based on the label-oriented philosophy suggested by the reverse model and exemplified by the way we've already handled K_n and P_r. Figure 107 illustrates the smallest KP graph:

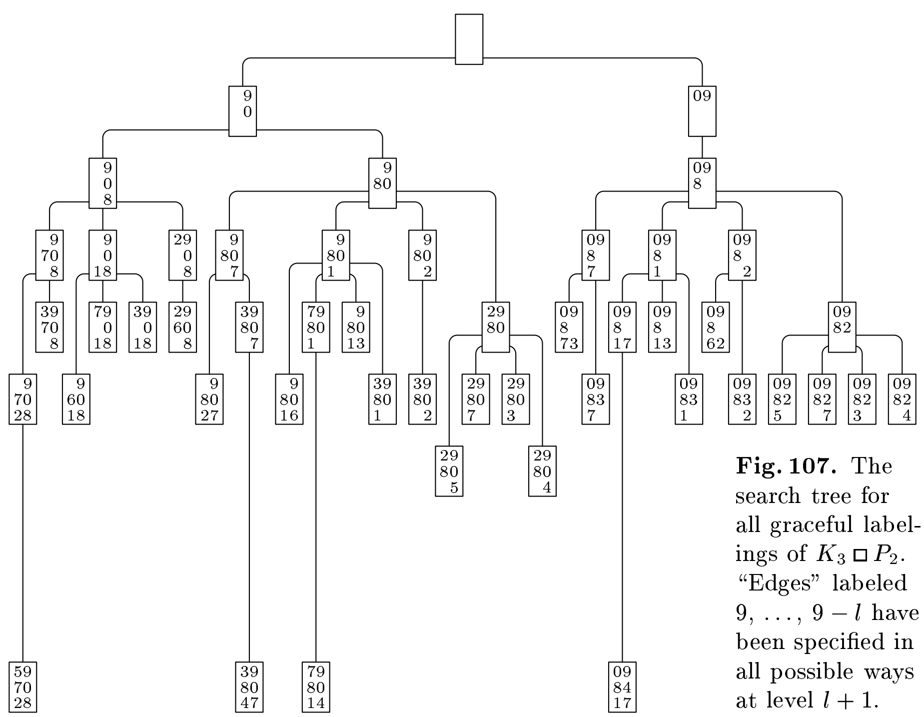


Fig. 107. The search tree for all graceful labelings of $K_3 \mathbin{\square} P_2$. "Edges" labeled 9, ..., $9-l$ have been specified in all possible ways at level $l+1$.

This problem has four solutions, which appear at the bottom of the tree (level 9). The key idea here is that we construct a "home-grown" canonical representation on the fly, by filling the 3×2 matrix with the labels of vertices that we've chosen to be the endpoints of edges m, $m-1$, $m-2$, ... Sometimes the placement of a single new vertex will create more than one necessary edge (see exercise 101).

unique graceful labeling
KC graphs
wraparound edges
parity
Bosák
cycle graphs
C_5
C_6

Search trees analogous to Fig. 107 can be constructed for all $n > 2$, and it turns out that the trees for $n = 3, 4, 5, \ldots$ have respectively 49, 446, 2094, 5545, 8103, 8825, 8907, 8910, 8910, 8910, 8910, ... nodes. Also, the number of solutions for those n turns out to be respectively 4, 15, 1, 0, 0, 0, 0, 0, 0, 0, 0, ...

Hmmm — guess what? The algorithm runs through precisely the *same* calculations for all $n \geq 10$, except that the number m of edges keeps getting larger and larger. It never is able to get past row 10 of its partially filled matrix. This amounts to a computer-generated proof that *the graphs $K_n \,\square\, P_2$ are ungraceful for all $n > 5$.* (See exercise 103.) Furthermore, the maximum running time over all n, which is also the time needed to generate that proof, is only 1.6 megamems.

Of course the graphs $K_n \,\square\, P_3$ can be analyzed too, by filling $n \times 3$ matrices in a similar way. The calculations are harder, yet the running time is still quite reasonable: Only (700 Kμ, 80 Mμ, 3.6 Gμ, 60 Gμ, 360 Gμ) are needed for $n =$ (3, 4, 5, 6, 7) to show that they have respectively (284, 704, 101, 1, 0) graceful labelings. Furthermore, 1.9 Tμ suffice to prove that *$K_n \,\square\, P_3$ is ungraceful for all $n > 6$*, by constructing a tree of 5,463,149,994 nodes.

$$\begin{pmatrix} 0 & 24 \\ 6 & 22 \\ 7 & 19 \\ 21 & 11 \\ 25 & 2 \end{pmatrix} \quad \begin{pmatrix} 0 & 56 & 1 \\ 5 & 36 & 9 \\ 12 & 6 & 52 \\ 33 & 55 & 26 \\ 44 & 2 & 49 \\ 57 & 20 & 11 \end{pmatrix} \quad \begin{pmatrix} 0 & 78 & 4 & 76 \\ 16 & 37 & 67 & 25 \\ 40 & 69 & 17 & 53 \\ 62 & 3 & 72 & 70 \\ 73 & 2 & 60 & 6 \\ 77 & 51 & 7 & 45 \end{pmatrix} \quad \begin{pmatrix} 0 & 62 & 6 & 64 & 75 \\ 3 & 18 & 69 & 10 & 33 \\ 41 & 70 & 23 & 59 & 20 \\ 73 & 9 & 43 & 24 & 51 \\ 74 & 2 & 71 & 14 & 8 \end{pmatrix}$$

Fig. 108. Some graceful gems: The unique labelings of $K_5 \,\square\, P_2$ and $K_6 \,\square\, P_3$. Also a (less rare) $K_6 \,\square\, P_4$ and $K_5 \,\square\, C_5$.

There's another intriguing family of graphs, the "KC graphs" $K_n \,\square\, C_r$ for $n > 2$ and $r > 2$, which add wraparound edges to the KP graphs. These graphs have even more symmetry: Every vertex has degree $n+1$, so there are rn vertices and $m = r(n+1)n/2$ edges. An example appears at the right of Fig. 108, where one can check that the 50 column differences $|x_{ij} - x_{kj}|$ together with the 25 row differences $|x_{ij} - x_{i((j-1) \bmod r)}|$ are precisely $\{1, 2, \ldots, 75\}$.

A new phenomenon now appears. Experiments show that $K_3 \,\square\, C_r$ is *ungraceful* whenever r is odd; yet the number of graceful labelings for the even values $r = 4, 6, \ldots$ grows very rapidly: 3809, 41928684, ... There's a very simple mathematical reason for failure in the odd-r case:

Lemma O. *In any graceful labeling of a graph with $4k+1$ or $4k+2$ edges, the number of vertices with an odd degree and an odd label is always odd.*

Proof. We have $\sum_{u - v} \bigl|l(u) - l(v)\bigr| = 1 + 2 + \cdots + m = \binom{m+1}{2}$ when there are m edges; and a given vertex v appears exactly $\deg(v)$ times in this sum. Working modulo 2, we also have $\bigl|l(u) - l(v)\bigr| \equiv l(u) + l(v)$. Therefore $\sum_v \deg(v) l(v) \equiv \binom{m+1}{2}$. But $\binom{m+1}{2} \equiv 1$ when $m = 4k+1$ or $m = 4k+2$. ▮

Corollary E (J. Bosák)**.** *If all vertices of a graceful graph have even degree, the graph has $4k$ or $4k+3$ edges for some integer k.* ▮

In particular, $K_3 \,\square\, C_r$ is ungraceful when r is odd, because it has $6r$ edges. Furthermore, the simple cycle graphs C_5, C_6, C_9, C_{10}, C_{13}, ... can't be graceful.

Sheppard
isolated vertex
symmetry
unique graceful labeling
fork
paw
free tree
superexponentially
Otter

The reverse model tells us another basic fact about gracefulness in general:

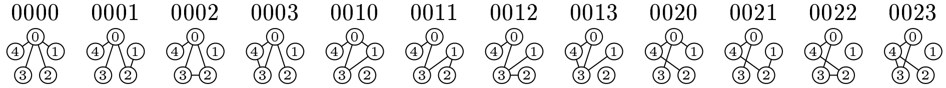
Theorem S. *There are exactly $m!$ graceful labelings with m edges.*

Proof. There are exactly $k+1$ ways to make $0 \le \mathtt{LO}[m-k] \le k$, for $0 \le k < m$. ▮

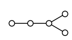
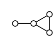
More precisely, if we insist that $\mathtt{LO}[m-1] = 0$ in order to rule out complementary solutions, *there are exactly $m!/2$ essentially distinct graceful labelings with m edges, for all $m \ge 2$.* [D. A. Sheppard, *Discr. Math.* **15** (1976), 379–388.]

Here, for example, are the $4!/2 = 12$ labelings when $m = 4$:

0000 0001 0002 0003 0010 0011 0012 0013 0020 0021 0022 0023



Each instance is accompanied by its four-digit LO string, LO[4]LO[3]LO[2]LO[1]. There are $m+1$ vertices in general, namely $\{0, 1, \ldots, m\}$; but some of them may be isolated — not participating in any edge. We can think of each isolated vertex in two ways: It's either present in the graph, representing its label; or it's absent, representing an unused label.

One of the nice things about this listing of $m!/2$ labelings is that symmetry is automatically handled as it should be. A highly symmetrical graph will appear only as often as it has truly distinct labelings, because labelings that differ only because of an automorphism are seen just once. For example, we observed earlier that $K_{1,4}$ has a unique graceful labeling, while P_5 has two; sure enough, we obtain $K_{1,4}$ only in case 0000, but P_5 in cases 0011 and 0021. Notice that C_4 also has a unique labeling (case 0022). The tree , which is often called the "fork," has three distinct labelings (cases 0001, 0012, 0020). The "paw" , otherwise known as $K_1 — (K_1 \oplus K_2)$, has the most (cases 0002, 0003, 0010, 0013, 0023).

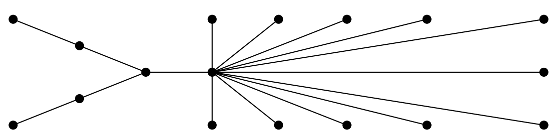
We can see gracefulness in action by looking at all $m!/2$ cases, when m isn't too large, and we're immediately faced with a host of interesting unsolved questions: How many of those cases yield graphs that are connected? planar? bipartite? triangle-free? When we omit the isolated vertices, how many of the resulting graphs are connected? cubic? And so on. (See exercises 116–123.)

In particular, how many of those graceful labelings yield a *free tree* on the vertices $\{0, 1, \ldots, m\}$? Equivalently, how many of those $m!/2$ sets of m edges have no cycles? In such cases no vertex is isolated. (See Theorem 2.3.4.1A.) The free trees shown above when $m = 4$ are 0000, 0001, 0011, 0012, 0020, and 0021.

Experimentation now reveals a striking phenomenon: *The number of graceful labelings of free trees grows superexponentially, as m increases, while the number of free trees grows only exponentially.* (There are nice ways to compute both numbers, without explicitly generating labelings or trees; see exercise 130 and 2.3.4.4–(9). On the other hand, according to R. Otter in *Annals of Mathematics* (2) **49** (1948), 583–599, the number of free trees with n vertices is proportional to $\alpha^n/n^{5/2}$, where $\alpha \approx 2.955765$.) For example, when $m = 30$, there are 902,745,276,529,593,126,158,482,120 essentially different labelings, but only 40,339,829,030 free trees with 31 vertices. That's an average of more than 2×10^{16} labelings per tree!

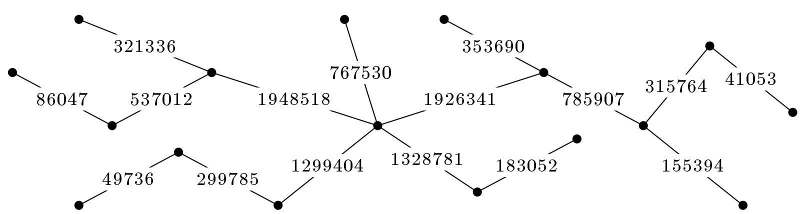
Anton Kotzig conjectured in 1965 that every tree is graceful, and his conjecture soon became famous, even infamous — because nobody could figure out how to prove it, yet all other questions about trees have generally been fairly easy to resolve. Indeed, there are hundreds of people for whom the initials GTC now mean only one thing: Not Green Templeton College, not Girls' Training Corps, not GPU Technology Conference, but Graceful Tree Conjecture.

The GTC is almost certainly true. For example, Alexander Rosa, who invented the concept of graceful graphs while completing his dissertation under Kotzig's direction, proved it already in 1965 for all trees of at most 16 vertices, and for many infinite families of trees. A careful study of the case $m = 16$ by David Anick [*Discrete Applied Mathematics* **198** (2016), 65–81] showed that only a handful of the 48629 free trees with 17 vertices have fewer than 50 labelings; and those few turned out to be obviously graceful, because they all are "caterpillars" (see exercise 145) except for this one of diameter 4:



(41)

At the other extreme, the champion tree has 10,399,350 different labelings. Here it is, with each edge showing the number of times it can be the edge of length 16:



(42)

(Long edges seem to prefer vertices of high degree.) Anick's analysis suggests strongly that all trees of larger sizes will also be easy to label.

Kotzig
GTC
Rosa
Anick
caterpillars
diameter 4
Graceful digraphs
digraphs, graceful
set inclusion
Boolean lattice

***Graceful digraphs.** There's also a nice way to define the concept of a graceful *directed* graph. Suppose D is a simple, loopfree digraph with m arcs. As before we want to assign distinct integers $l(v)$ to its vertices, with $0 \le l(v) \le m$. But now we say that each directed arc $v \longrightarrow w$ implicitly receives the label $\bigl(l(w)-l(v)\bigr) \bmod (m+1)$, respecting the orientation of the arc; and D is *graceful* if those arc labels are distinct. It follows that gracefulness gives us exactly one arc labeled k, for each k between 1 and m.

For example, Fig. 109 shows a digraph that represents set inclusion in a 3-element universe, together with several of its graceful labelings. We can check labeling (b) for gracefulness, just as we did in (33) for the undirected graph in Fig. 105, but this time using the operator $y \ominus x = (y-x) \bmod 13$:

$$\begin{array}{llllll} 1\ominus 0=1 & 9\ominus 6=3 & 8\ominus 3=5 & 7\ominus 0=7 & 3\ominus 7=9 & 4\ominus 6=11 \\ 3\ominus 1=2 & 8\ominus 4=4 & 6\ominus 0=6 & 9\ominus 1=8 & 4\ominus 7=10 & 8\ominus 9=12 \end{array}. \qquad (43)$$

Symmetries
antiautomorphisms
antisymmetries
self-converse

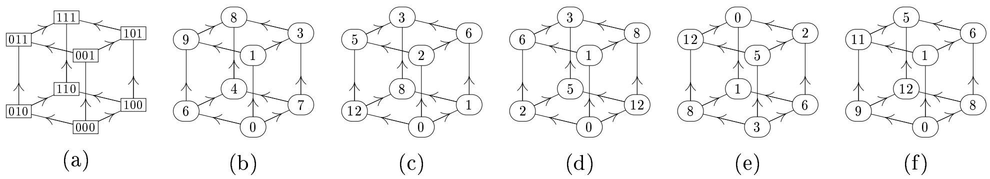


Fig. 109. This directed graph (a) can be gracefully labeled in many ways, some of which are readily derivable from each other. For example, (c) arises from (b) when every vertex label is doubled, modulo 13. (We work mod 13 in this digraph because it has 12 arcs.) Can you see how (d), (e), and (f) were obtained from the others?

Let $q = m + 1$. Cyclic labels mod q are much more versatile mathematically than the absolute-difference labels that we considered before, because (for example) we can add a constant to every vertex label without changing the implied label of any arc. This means we can arbitrarily choose any vertex v and look only for labelings with $l(v) = 0$, when we're trying to decide whether or not a given digraph is graceful. Any graceful labeling with $l(v) = b$ yields one with $l(v) = 0$ after b is subtracted from each label.

Furthermore, when q is a prime number as it is in Fig. 109, we can arbitrarily choose any *two* vertices v and w, and look only for labelings with $l(v) = 0$ and $l(w) = 1$: Given any labeling with $l(v) = 0$, we can multiply all the vertex labels by the number a for which $a \cdot l(w) \equiv 1$ (modulo q). This operation preserves gracefulness, because it implicitly multiplies every arc label by a (modulo q). For example, multiplying Fig. 109(b) by 2 changes the label of vertex 100 from 7 to 1.

Symmetries of the digraph give us yet another way to derive one labeling from another, just as the symmetry GA $\leftrightarrow$ SC did in Fig. 105. For example, labeling (d) arises from (c) when the label currently assigned to vertex $x_1x_2x_3$ is moved to vertex $x_2x_3x_1$, for each binary vector $x_1x_2x_3$.

Digraphs also bring a new notion into the picture, because they can have *antiautomorphisms* (antisymmetries), which are permutations α of the vertices for which $v \longrightarrow w$ implies $v\alpha \longleftarrow w\alpha$. In general, every digraph D has a *converse* D^T whose arcs all go the other way. A digraph is *self-converse* if and only if it has an antiautomorphism. For example, the mapping $x_1x_2x_3\alpha = \bar{x}_1\bar{x}_2\bar{x}_3$ is an antiautomorphism of the digraph in Fig. 109; hence the labeling in (e), obtained from (d) when each $l(v)$ is replaced by $l(v\alpha)$, gracefully negates each arc label.

Two labelings of a digraph are regarded as essentially the same if we can get one from the other by (i) subtracting b mod q, or (ii) multiplying by a mod q when a is relatively prime to q, or (iii) using an automorphism or antiautomorphism to permute the vertex labels, or (iv) using any combination of transformations (i), (ii), (iii). In this sense, 156 different labelings are essentially equivalent to Fig. 109(b) — including Fig. 109(f). (See exercises 156 and 157.)

Exercise 160 explains how to find all graceful labelings of a given digraph D, by finding representatives of each of its equivalence classes. The first step is to solve an appropriate CSP, using methods adapted from those that work for undirected graphs. Some instructive case studies appear in exercises 161 and 168.

data structures
digraph representation
compact representation

We saw above in (35) that any graceful graph can be represented conveniently within a computer by a set of five compact arrays. Directed graphs turn out to be even *more* attractive in this respect, because only four arrays suffice; a single array `NEXT` replaces the former `NEXTL` and `NEXTH`. For example, here's a compact representation of Fig. 109(a) that corresponds to Fig. 109(b):

$l =$	0	1	2	3	4	5	6	7	8	9	10	11	12
`LO`$[l] =$	—	0	1	6	4	3	0	0	1	7	7	6	9
`FIRST`$[l] =$	7	9	−1	8	8	−1	4	4	−1	8	−1	−1	−1
`NEXT`$[l] =$	—	−1	−1	−1	−1	−1	1	6	3	−1	3	9	−1
`NAME`$[l] =$	000	001	—	101	110	—	010	100	111	011	—	—	—

(44)

As before, the general idea is to include isolated vertices if necessary so that the vertices of the graceful digraph D are $\{0, 1, \ldots, m\}$, the same as their labels. The `NAME` array connects these internal numbers with D's external representation, if those vertex names are needed for communication with users.

The `LO` array is crucial. For $1 \le l \le m$, we have `LO`$[l] = v$ if and only if the arc labeled l goes from v to $(v + l) \bmod q$, where $q = m + 1$. Consequently it's easy to test whether or not $v \longrightarrow w$ is an arc of D, given v and w, by inspecting a single element of the `LO` array: *That arc is present if and only if* `LO`$[(w - v) \bmod q] = v$.

The `FIRST` and `NEXT` arrays are set up so that we can easily visit every successor of a given vertex v, using the following efficient algorithm:

$$\begin{aligned}&\text{Set } w \leftarrow \texttt{FIRST}[v];\\ &\text{while } w \ge 0, \text{ visit } w, \text{ then set } w \leftarrow \texttt{NEXT}[(w - v) \bmod q].\end{aligned} \qquad (45)$$

Exercise 164 explains one way to derive `FIRST` and `NEXT` from `LO`.

Every array `LO` with $0 \le$ `LO`$[l] \le m$ for $1 \le l \le m$ defines a graceful digraph with m arcs on the vertices $\{0, \ldots, m\}$. Thus the total number of m-arc graceful labelings is exactly $(m + 1)^m$. That's much larger than the $m!$ graceful labelings with m *edges* (see Theorem S); exercise 172 shows, however, that we can decrease it by a factor of approximately $2m^2$ when equivalent labelings are lumped together. Thus the complete set of graceful digraphs can be explored without difficulty when m isn't too large.

Digraphs often do turn out to be graceful; for example, 844161 of the 1540944 nonisomorphic digraphs on six vertices can be labeled successfully. But of course there are many exceptions—including half of the "most basic" ones:

Theorem H. *The oriented path $P_n^{\rightarrow}$ and the oriented cycle $C_n^{\rightarrow}$ are both graceful when n is even, but they're both ungraceful when n is odd.*

Proof. The arcs are $v_0 \longrightarrow v_1 \longrightarrow \cdots \longrightarrow v_m$, where $m = n - 1$ for $P_n^{\rightarrow}$ and $m = n$ (and $v_m = v_0$) for $C_n^{\rightarrow}$. Suitable labels exist when n is even (see exercise 175).

But there's an unsurmountable problem when n is odd, because the sum (modulo q) of all arc labels, $\bigl(l(v_1) - l(v_0)\bigr) \bmod q + \cdots + \bigl(l(v_m) - l(v_{m-1})\bigr) \bmod q$, is congruent to $l(v_m) - l(v_0)$. This sum should *not* be congruent to zero in the case of the path, but it *should* be congruent to zero in the cycle.

In a graceful digraph the sum of all the arc labels must be $1+2+\cdots+m$, which is $q(q-1)/2$. Hence it's congruent to 0 when q is odd, and it's an odd multiple of $q/2$ when q is even. Contradiction. ▮

An undirected graph is called *digraceful* if there's at least one way to convert it to a graceful digraph by orienting each of its edges. There are 2^m possible orientations of m edges, so this gives us lots of flexibility.

A graceful graph is obviously digraceful as well, because we can orient each edge towards its endpoint whose label is largest. Furthermore, the ungraceful graphs C_{4n+2} are digraceful, because $\vec{C}_{4n+2}$ is graceful by Theorem H. On the other hand, exercise 182 proves that the graphs C_{4n+1} are *not* digraceful.

Is the complete graph K_n digraceful? This is probably the most interesting unsolved question about digracefulness, because every orientation of K_n is called a *tournament*. Graceful tournaments have been studied in other disguises, and they are known to exist for $n = 1, 2, 3, 4, 5$, and 9. (See exercise 185.)

There is, however, a much nicer and more natural way to regard an undirected graph G as a digraph, namely to treat it as the symmetric digraph $G^{\leftrightarrow}$, in which every edge $u \mathrel{—} v$ has been replaced by two arcs $u \longrightarrow v$ and $v \longrightarrow u$. Indeed, as discussed just before 7–(26), G and $G^{\leftrightarrow}$ have essentially the same properties, so we represent them both in the same way inside a computer.

If G has m edges, $G^{\leftrightarrow}$ has $2m$ arcs. Thus the vertex labels of $G^{\leftrightarrow}$ should be chosen modulo $q = 2m+1$. The labels of $u \longrightarrow v$ and $v \longrightarrow u$ are then negatives of each other, modulo q; and there are just m possibilities, namely $\{\pm1, \pm2, \ldots, \pm m\}$. Consequently we define the label of edge $u \mathrel{—} v$ in $G^{\leftrightarrow}$ to be

$$\begin{aligned} d_L\bigl(l(u), l(v)\bigr) \;&=\; \min\bigl(\bigl(l(u)-l(v)\bigr) \bmod q, \bigl(l(v)-l(u)\bigr) \bmod q\bigr) \\ &=\; \min\bigl(\bigl|l(u)-l(v)\bigr|, q-\bigl|l(u)-l(v)\bigr|\bigr). \end{aligned} \qquad (46)$$

(This is the *Lee distance* between the points $l(u)$ and $l(v)$ on a q-cycle; see exercise 7.2.1.1–18.) And now a pleasant thing happens: When we draw K_{2m+1} with its vertices in a circle, it has exactly $2m+1$ edges of Lee distance 1, exactly $2m+1$ edges of Lee distance 2, …, and exactly $2m+1$ edges of Lee distance m. Therefore if $G^{\leftrightarrow}$ is a graceful digraph with m edges, we can pack $2m+1$ copies of G perfectly into K_{2m+1}. (Figure 110 illustrates the case $m = 5$.)

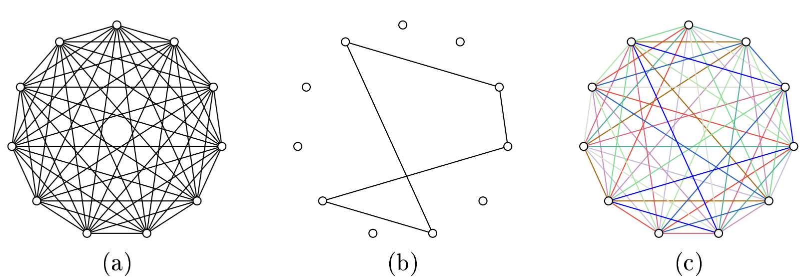


Fig. 110. K_{11} has eleven edges of distance 1, …, and eleven of distance 5. A 5-cycle can be drawn with one edge of each distance. Hence eleven 5-cycles exactly cover K_{11}. "Eleven people can form eleven rings of five, where everybody meets everybody else."

digraceful
orientations
tournament
symmetric digraph
representation of graphs and digraphs
Lee distance

Let's say that a graph G with m edges is *rainbow graceful* if the corresponding digraph $G^{\leftrightarrow}$ is graceful. This means that we can assign a label $l(v)$ to each vertex v, with $-m \le l(v) \le m$, in such a way that the edge labels $d_L\bigl(l(u), l(v)\bigr)$ defined in (46) are distinct for all m edges $u \,—\, v$.

A graceful graph is automatically rainbow graceful, because $d_L\bigl(l(u), l(v)\bigr) = \bigl|l(u) - l(v)\bigr|$ when $l(u)$ and $l(v)$ are nonnegative. Furthermore Fig. 110(b) shows that C_5 is rainbow graceful, although it neither graceful nor digraceful. In fact — see exercise 190 — there's an astonishingly simple way to prove that *every cycle C_n is rainbow graceful*, for $n \ge 3$, because of the elegant labeling

$$l(k) = (-1)^{k+[2k<n]} k, \qquad \text{for } 1 \le k \le n. \tag{47}$$

A great many graphs are in fact known to be rainbow graceful, and more are being discovered every day. For example, according to the systematic study in exercise 193, every graph on at most 6 vertices is rainbow graceful, except for $K_6 \setminus K_2$ (the 14-edge graph obtained by deleting one of the edges of K_6).

We've seen that graphs with lots of edges are often impossible to label gracefully, because so many labels have to avoid interfering with each other. Yet rainbow labeling is different, because the complete graphs K_5 and K_6 — which have the *maximum* number of edges — do turn out to be labelable! In fact, exercise 197 shows that *K_{n+1} is rainbow graceful whenever n is prime or a power of a prime*. It's remarkable, but true, that K_8, K_9, K_{10}, and K_{12} are rainbow graceful. (On the other hand, K_7, K_{11}, and K_{13} are not.)

The first major steps towards proving the Graceful Tree Conjecture were taken by R. Montgomery, A. Pokrovskiy, and B. Sudakov, who developed new methods in order to prove an asymptotic form of a weaker conjecture:

Theorem M. *All sufficiently large trees are rainbow graceful.*

Proof. See *Geometric and Functional Analysis* **31** (2021), 663–720. ▮

Numerous unresolved questions about gracefulness remain under active investigation, because the number of interesting graphs and digraphs is essentially boundless. Joseph A. Gallian has been actively maintaining a dynamic survey of what is currently known. His annual reports [*Electronic Journal of Combinatorics*, #DS6] began in 1998 with a 46-page review containing 306 references; its 23rd edition (2020) had 553 pages (with an 18-page index) and 2922 references.

Graph embedding. Graph G is said* to be *embedded* in graph H if it is isomorphic to a subgraph of H. Informally, this means that H contains a "copy" of G. Formally, it means that there's a function f from the vertices of G to the vertices of H such that two conditions are satisfied:

i) if $v \ne w$ then $f(v) \ne f(w)$;
ii) if $v \,—\, w$ in G then $f(v) \,—\, f(w)$ in H.

When that happens, we say that "H contains G," and the set of all vertices $\{f(v) \mid v \text{ is a vertex of } G\}$ is called the *image* of G in H.

rainbow graceful
Graceful Tree Conjecture
Montgomery
Pokrovskiy
Sudakov
Gallian
embedding
subgraph
image

* People also talk about a graph "embedded in a surface"; that's an entirely different topic.

strict embedding
isometric embedding
shortest distance
`WORDS`(1000)
Stanford GraphBase
five-letter words
clique
induced
Cartesian product of graphs

Embeddings actually come in three flavors. An ordinary vanilla-flavored embedding simply satisfies (i) and (ii); but a stronger version, called a *strict* embedding, also satisfies a third condition:

iii) if $v \not\!\!\relbar\!\!\relbar\, w$ in G then $f(v) \not\!\!\relbar\!\!\relbar\, f(w)$ in H.

Stronger yet is an *isometric embedding*, which satisfies even more:

iv) $d_G(v,w) = d_H(f(v), f(w))$, where d denotes the shortest distance.

Notice that condition (iv) by itself implies (i), (ii), and (iii).

For example, suppose G is the five-cycle C_5, and suppose H is `WORDS`(1000), the Stanford GraphBase graph that represents the thousand most common five-letter words of English. One of the zillions of five-cycles in H is

`share` — `spare` — `stare` — `store` — `shore` — `share`. (48)

Formally we could say that the vertices of G are $\{0,1,2,3,4\}$, and that G's edges are $v \relbar\!\!\relbar ((v+1) \bmod 5)$ for $0 \le v < 5$; then $f(0) =$ `share`, ..., $f(4) =$ `shore`. But such formalities are needlessly complicated when we're talking about graphs as simple as C_5; the embedding is immediately clear just from (48).

Example (48) is not a *strict* embedding of C_5, because we have `share` — `stare` in H but $0 \not\!\!\relbar\!\!\relbar\, 2$ in G. We could in fact have come up with a five-cycle such as

`share` — `shape` — `shade` — `shake` — `shame` — `share`, (49)

in which all five words are mutually adjacent in H; but that seems like cheating, because *any* graph is trivially isomorphic to a subgraph of a complete graph. (This graph `WORDS`(1000) actually contains the 8-clique {`right`, `might`, `night`, `light`, `sight`, `fight`, `tight`, `eight`}; hence it contains a copy of *every* G with up to eight vertices!) The essence of a five-cycle is present in (48), at least partly, but it has been drowned out in (49). A strict embedding retains the full structure, because (ii) and (iii) say that G appears as an *induced* subgraph of H.

There's no way to embed C_5 *strictly* into `WORDS`(1000), because `WORDS`(1000) is a subgraph of $K_{26} \mathbin{\square} K_{26} \mathbin{\square} K_{26} \mathbin{\square} K_{26} \mathbin{\square} K_{26}$; and that graph has no induced C_5 (see exercise 207(f)). Thus a weak embedding like (48) is the best we can get.

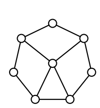
Surprisingly, however, there *is* a strict embedding of the next odd cycle, C_7:

`likes` — `lakes` — `cakes` — `caves`
— `waves` — `wives` — `lives` — `likes`. (50)

This one even turns out to be isometric, in the target graph `WORDS`(1000).

But — surprise, surprise — the induced cycle (50) is *not* isometric in the larger graph `WORDS`(5757) — because that graph contains the somewhat unusual word `laves`. The distance from `lakes` to `waves` in the larger graph is therefore 2, not 3; and the same is true for the distance from `caves` to `lives`.

Notice that if we add the word `laves` to (50), we get an isometric embedding of the graph



(51)

into $K_{26} \mathbin{\square} K_{26} \mathbin{\square} K_{26} \mathbin{\square} K_{26} \mathbin{\square} K_{26}$.

Evidently isometric embeddings are somewhat tricky. Some of their basic properties are explored in exercises 208–216, but we shall concentrate on embeddings of the other two kinds.

Given a pattern graph G and a target graph H, the problem of visiting all embeddings of the pattern in the target is called the *subgraph isomorphism problem* (SIP), and the problem of visiting all of the *strict* embeddings is called the *induced subgraph isomorphism problem* (ISIP). These should be distinguished from the *graph isomorphism problem* (GIP), which is to test whether or not G and H are essentially the same. The GIP is obviously equivalent to testing SIP or ISIP in both directions; but it's much simpler, and it can be attacked by many methods that don't work for the SIP or ISIP. We'll study the GIP in Section 7.2.3.

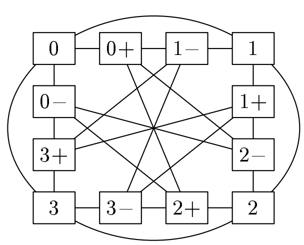
Let's write $G \subseteq H$ if the SIP for pattern G and target H is solvable, and $G \sqsubseteq H$ if the ISIP is solvable. (This is a slight abuse of notation; the relation $G \subseteq H$ really means that $G \cong H'$ for some $H' \subseteq H$, and $G \sqsubseteq H$ really means that $G \cong H \mid U$ for some vertices U of H. But we think of the embedded graph as actually present inside its host.)

The SIP is easily seen to be a CSP, with variables, domains, and constraints: The variables are the vertices of G, the domains are the vertices of H, and the constraints are conditions (i) and (ii). Indeed, we've already noted this characterization of embedding in (6) above. The SIP is, in essence, the CSP that's constrained to be a homomorphism of a given binary relation, together with the all-different constraint.

To fix the ideas, it will be helpful to consider an "organic" example. Figure 111 shows the principal interconnections of a typical human brain, together with two of the subgraphs obtained when only the strongest links are considered.*

Clearly `BRAIN83`(250) is embedded in `BRAIN83`; but a moment's thought shows that it would be pointless to use a subgraph-isomorphism test to verify that fact: The big graph is so rich and twisted, almost *any* not-too-big graph can probably be found within it, in zillions of ways. The interesting question is rather whether a smaller graph with nice structure can be found within `BRAIN83`(250).

Consider, for example, the attractive 4-regular graph called Chvátal's graph. We looked at it long ago in Figure 2(f), near the beginning of Chapter 7; here it is again, with convenient names given to the vertices:



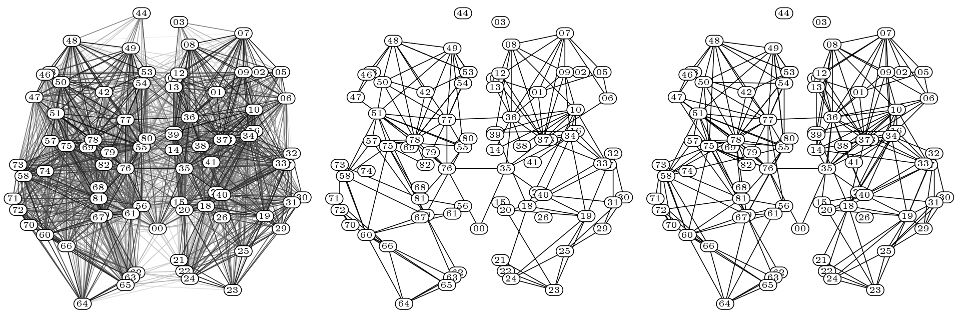
(52)

Can this graph be embedded in the somewhat sparse graph `BRAIN83`(250)?

* See `http://cs.stanford.edu/~knuth/brain83.html` for complete details about this graph, which was constructed from data compiled and simplified by Alain Goriely.

subgraph isomorphism problem
SIP
induced subgraph isomorphism problem
ISIP
graph isomorphism problem
GIP
CSP
homomorphism
all-different constraint
Goriely
internet
BRAIN83+
4-regular graph
Chvátal's graph

exact cover problem
Human Connectome Project



`BRAIN83`, 1654 edges (a) `BRAIN83`(250), 170 edges (b) `BRAIN83`(300), 242 edges (c)

Fig. 111. The graph `BRAIN83`, based on hundreds of high-resolution brain scans performed by the Human Connectome Project, shows the "wiring diagram" of a healthy human brain. The full graph (a) has 83 vertices (representing the major regions of interest) and 1654 edges (representing channels between them). Vertex 00 is the brain stem; vertices 01–41 form the right brain; and vertices 42–82 form the left brain, with $v+41$ on the left corresponding to v on the right.

Each edge is labeled with an integer $l \geq 0$, which is a logarithmic measure of its importance: The strength of an interconnection is proportional to $e^{-l/1000}$. (However, l is depicted linearly here, with a line that's shaded $l/1350$ of the way from black to white.) The subgraph `BRAIN83`(250) in (b), which retains only the edges with $l \leq 250$, illustrates some of the strongest interconnections. For example, vertices 77 and 36 are the left and right caudate nuclei, and they are connected by an edge with $l = 33$.

One way to decide this is to set it up as an exact cover problem, following the lead of exercise 7.2.2.1–77, which considered the special case where G and H have the same number of vertices. In general, let there be a primary item v for each vertex v of G, and a secondary item V for each vertex V of H. Let there also be secondary items $e \cdot E$ for every edge e of G and every *non*-edge E of H. The exact cover problem then has one option for each pair (v, V), representing the potential mapping $v \mapsto V$, namely

$$\text{`}v\;\; V\;\; \bigcup\{e \cdot E \mid e = (u \mathrel{-\!\!-} v) \text{ and } E = (U \mathrel{-\!\!\!\!/\!\!\!-} V) \text{ for some } u \text{ and } U\}\text{'}. \qquad (53)$$

The solutions to this exact cover problem are precisely the embeddings we want, because (i) every vertex v of G is paired with a distinct vertex V of H; and (ii) we cannot pair u with U and v with V in cases where $u \mathrel{-\!\!-} v$ and $U \mathrel{-\!\!\!\!/\!\!\!-} V$.

For example, when G is Chvátal's graph (52) and H is `BRAIN83`(250), G has 12 vertices and 24 edges; H has 68 non-isolated vertices, with $\binom{68}{2} - 170 = 2108$ nonedges between them. Our exact cover problem therefore has 12 primary items, $68 + 24 \cdot 2108 = 50660$ secondary items, and $12 \cdot 68 = 816$ options.

The options are long: Graph H has 65 nonedges involving vertex 00, so every option that pairs v with 00 contains $2+4\cdot 65 = 262$ items. The 816 options therefore have more than 200,000 entries altogether, and Algorithm 7.2.2.1X takes 6 gigamems just to input them before getting started! But then it needs only 2 gigamems to solve the problem — and the result is *no solutions* (no embeddings).

automorphisms
essentially different solutions
United States

We can discover the lack of solutions by being a little smarter when we set up the problem. In the first place, there's no point in considering 00 as a potential target vertex, because that vertex has degree 2; every target vertex will clearly have degree 4 in the image of G, so it must have degree ≥ 4 in H. We can therefore eliminate from `BRAIN83`(250) not only the 15 isolated vertices, but also the 28 vertices of degrees 1, 2, and 3.

Furthermore, after those vertices go away, other vertices no longer have degree ≥ 4. If we keep pruning vertices of low valency from the graph, we finally reach a graph $\widehat{H}$ that has only 22 vertices, all of degree ≥ 4. This reduced target graph has only $\binom{22}{2} - 63 = 168$ nonedges; and the reduced exact cover problem has fewer than 17000 entries in its 264 options. Algorithm 7.2.2.1X needs just 40 megamems to input them, and 90 megamems to prove them unsolvable.

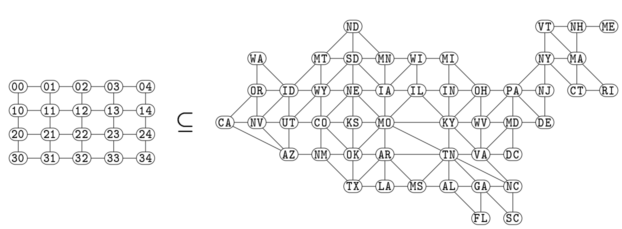
In fact, exercise 235 shows that there's a sneaky way to see that $G \not\subseteq \widehat{H}$ without even running the algorithm.

OK, `BRAIN83`(250) is too sparse to contain Chvátal's graph G. But what about `BRAIN83`(300)? That graph H has 70 nonisolated vertices, and we can prune it down to a min-degree-4 graph $\widehat{H}$ with only 58 vertices and 211 edges. Now we get an exact cover problem that Algorithm 7.2.2.1X can input in 3 Gμ and solve in 8 Gμ; there are 72 solutions. Therefore G is indeed embeddable into the graph of Fig. 111(c), in 72 ways. (That fact has little or no biological significance, of course; but somehow it's comforting to know that we all have Chvátal's graph rather firmly embedded in our brains.)

All 72 solutions turn out, in fact, to lie entirely within the left brain. But the right brain will contain (52) too, if we add a few more edges of the full graph.

It's significant that 72 is a multiple of 8, because Chvátal's graph has 8 automorphisms (see exercise 7–44). If G is any graph with exactly r automorphisms, the number of functions f that embed G into H is always a multiple of r, because we obtain r distinct embedding functions $f(v\alpha)$ when α ranges over all the automorphisms. Thus there really are only 9 essentially *different* ways to embed (52) into `BRAIN83`(300). One of them takes $0 \mapsto 48$, $0{+} \mapsto 49$, $1{-} \mapsto 51$, $1 \mapsto 47$, $1{+} \mapsto 77$, $2{-} \mapsto 53$, $2 \mapsto 78$, $2{+} \mapsto 55$, $3{-} \mapsto 58$, $3 \mapsto 75$, $3{+} \mapsto 50$, $0{-} \mapsto 54$; it's essentially the same as the embedding $1 \mapsto 48$, $1{+} \mapsto 49$, $2{-} \mapsto 51$, $2 \mapsto 47$, …, $1{-} \mapsto 54$, and to six others. (This solution does not belong to `BRAIN83`(298), because the edge 48 —— 54 has the label $l = 299$. There are $2 \cdot 8$ embeddings into `BRAIN83`(293), but none into `BRAIN83`(292).)

That was fun. Let's try another example, this time with a smaller target so that we can see more closely what is going on. Here's a question about the United States that has perhaps never been asked before:

Is  ? (54)

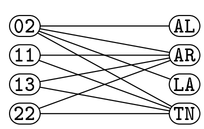
On the left is $P_4 \,\Box\, P_5$, a 4×5 grid. On the right is the 49-vertex, 107-edge graph of the continental USA that we saw most recently in Fig. 106. At first glance, smallish grids are visible within the right-hand graph, but a 4×5 seems unlikely.

There are, in fact, three different ways to solve the embedding problem of (54) — that is, $4 \cdot 3$ actual embedding functions, because the grid has four automorphisms. The reader is encouraged to find at least one of them now, by hand, before turning the page to peek at the answer.

Meanwhile let's look at how a computer might attack this problem intelligently. Call the graphs G and H. In the first place, the six interior vertices of G have degree 4; so their domains cannot include any of the 15 states {`CA`, `CT`, `DC`, `DE`, `FL`, `LA`, `ME`, `MI`, `ND`, `NH`, `NJ`, `RI`, `SC`, `VT`, `WA`} of smaller degree.

We can shrink the domains even further by looking at the degrees of neighbors. For example, the neighbors of `11` in G have degrees $\{3,3,4,4\}$, while the neighbors of `GA` in H have degrees $\{2,2,4,4,8\}$. Therefore no embedding of G into H can map `11` $\mapsto$ `GA`. (See exercise 242.) In a similar way we can remove `AL`, `GA`, `MA`, `NC`, `OR` from the domains of `11`, `12`, `13`, `21`, `22`, and `23`. Furthermore the neighbors of `NY` in H have degrees $\{3,3,3,5,6\}$; this doesn't rule out `11` $\mapsto$ `NY`, but it does show that we can't map `12` $\mapsto$ `NY` or `22` $\mapsto$ `NY`. That leaves just 28 possibilities in the initial domains of G's "middle" vertices `12` and `22`.

An even closer look shows that we can't take `12` $\mapsto$ `MS`. For if we did, there would be a matching of size 4 in the bipartite graph



. (55)

The left part here shows the neighbors of `12`; they must each match a vertex in their domain that also happens to be a neighbor of `MS`. There's no such matching. Similar analyses rule out the mappings `11` $\mapsto$ `OR`, `02` $\mapsto$ `MA`, and so on. This technique for domain reduction was introduced by C. Solnon [*Artificial Intelligence* **174** (2010), 850–864], who called it LAD filtering (for "Locally All Different").

We now begin to form a search tree, with 27 possibilities to try for the image of `12`. The first of these, alphabetically, is `AZ`, so let's tentatively map `12` $\mapsto$ `AZ`. This means we remove `AZ` from every other domain, and restrict the domains of `02`, `11`, `13`, and `22` to neighbors of `AZ`. Hmm; we soon reach an impasse, because `21` has no place to go: It must map to a neighbor of the domains of `11` and `22`, namely a neighbor of {`NM`, `NV`, `UT`}; but LAD filtering proves that impossible.

The next thing to try is `12` $\mapsto$ `AR`. This option is somewhat more plausible; LAD filtering whittles the domains down quite a bit, but not too far. They are

$$\begin{pmatrix} \mathtt{i} & \mathtt{e} & \mathtt{d} & \mathtt{e} & \mathtt{i} \\ \mathtt{h} & \mathtt{b} & \mathtt{a} & \mathtt{b} & \mathtt{h} \\ \mathtt{g} & \mathtt{c} & \mathtt{b} & \mathtt{c} & \mathtt{g} \\ \mathtt{j} & \mathtt{g} & \mathtt{f} & \mathtt{g} & \mathtt{j} \end{pmatrix}; \quad \begin{array}{ll} \mathtt{a} = \{\mathtt{AR}\}, & \mathtt{d} = \mathtt{b} \cup \{\mathtt{LA}, \mathtt{MS}, \mathtt{TX}\}, \\ \mathtt{b} = \{\mathtt{MO}, \mathtt{OK}, \mathtt{TN}\}, & \mathtt{e} = \mathtt{b} \cup \mathtt{c} \cup \{\mathtt{AL}, \mathtt{MS}, \mathtt{NM}, \mathtt{TX}\}, \\ \mathtt{c} = \{\mathtt{KS}, \mathtt{KY}, \mathtt{MO}\}, & \mathtt{f} = \mathtt{b} \cup \mathtt{c} \cup \{\mathtt{CO}, \mathtt{IA}, \mathtt{IL}, \mathtt{NE}, \mathtt{VA}\}, \\ \mathtt{g} = \mathtt{f} \cup \{\mathtt{IN}, \mathtt{WV}\}, & \mathtt{i} = \mathtt{e} \cup \mathtt{g} \cup \mathtt{h} \cup \{\mathtt{GA}\}, \\ \mathtt{h} = \mathtt{f} \cup \{\mathtt{NC}, \mathtt{NM}\}, & \mathtt{j} = \mathtt{g} \cup \mathtt{h} \cup \{\mathtt{MD}, \mathtt{OH}, \mathtt{SD}, \mathtt{WI}, \mathtt{WY}\}. \end{array} \qquad (56)$$

grid
continental USA
automorphisms
initial domains
maximum bipartite matching
matching
bipartite graph
Solnon
LAD filtering–

GAD filtering
all-different
Sudoku

For example, the domain of 11, 13, and 22 is {MO, OK, TN}; the domain of 02 is {LA, MO, MS, OK, TN, TX}; and the domain of 32 has 10 elements.

At this point we turn to a complementary technique, known as GAD filtering (for "Globally All Different"). The idea is again to solve a bipartite matching problem; but our goal this time is to match *every* pattern vertex with some element of its current domain. (Because if no such matching exists, the current domains are too small and we must backtrack.)

The domains in (56) readily yield such a matching. For example, here's one:

$$\begin{pmatrix} \texttt{VA} & \texttt{NM} & \texttt{TX} & \texttt{MS} & \texttt{AL} \\ \texttt{NC} & \texttt{TN} & \texttt{AR} & \texttt{OK} & \texttt{CO} \\ \texttt{IA} & \texttt{KY} & \texttt{MO} & \texttt{KS} & \texttt{WV} \\ \texttt{SD} & \texttt{NE} & \texttt{IL} & \texttt{IN} & \texttt{WY} \end{pmatrix}. \tag{57}$$

Of course this doesn't solve our subgraph isomorphism problem — Virginia is nowhere near New Mexico, and there are many other faults. But VA does belong to the current domain of 00, according to (56), and NM does belong to the domain of 01. The advantage of (57) is that the theory of bipartite matching gives us an efficient way to trim off all the "excess fat" from the domains of variables that are required to be all-different. Indeed, the algorithm of exercise 253 uses (57) to reduce (56) substantially, so that only the following domains are left:

$$\begin{pmatrix} \texttt{i} & \texttt{e} & \texttt{d} & \texttt{e} & \texttt{i} \\ \texttt{h} & \texttt{b} & \texttt{a} & \texttt{b} & \texttt{h} \\ \texttt{g} & \texttt{c} & \texttt{b} & \texttt{c} & \texttt{g} \\ \texttt{j} & \texttt{g} & \texttt{f} & \texttt{g} & \texttt{j} \end{pmatrix}; \quad \begin{array}{ll} \texttt{a} = \{\texttt{AR}\}, & \texttt{d} = \{\texttt{LA},\texttt{MS},\texttt{TX}\}, \\ \texttt{b} = \{\texttt{MO},\texttt{OK},\texttt{TN}\}, & \texttt{e} = \{\texttt{AL},\texttt{MS},\texttt{NM},\texttt{TX}\}, \\ \texttt{c} = \{\texttt{KS},\texttt{KY}\}, & \texttt{f} = \{\texttt{CO},\texttt{IA},\texttt{IL},\texttt{NE},\texttt{VA}\}, \\ \texttt{g} = \texttt{f} \cup \{\texttt{IN},\texttt{WV}\}, & \texttt{i} = \texttt{e} \cup \texttt{g} \cup \texttt{h} \cup \{\texttt{GA}\}, \\ \texttt{h} = \texttt{f} \cup \{\texttt{NC},\texttt{NM}\}, & \texttt{j} = \texttt{g} \cup \texttt{h} \cup \{\texttt{MD},\texttt{OH},\texttt{SD},\texttt{WI},\texttt{WY}\}. \end{array} \tag{58}$$

Notice, for example, that (56) had MO in 19 of the 20 domains; the only exception was 'a', the domain of the pattern vertex 12 that we've tentatively mapped to AR. But in (58), MO belongs only to 'b', which is the domain of pattern vertices 11, 13, and 22.

Sudoku experts will see why MO can be dropped from 16 of the 19 domains where it was formerly present: Any all-different assignment using (56) must map {11, 13, 22} into {MO, OK, TN}. Hence those three values can't be used elsewhere.

Similarly, we now know that 21 and 23 can't be mapped to MO; so they must map to {KS, KY}. We can therefore eliminate KS and KY from all domains but c.

GAD filtering, which reduces (56) to (58), is not specific to the subgraph isomorphism problem; it applies to *any* CSP with an all-different constraint. No further reduction from (58) is possible, from that global standpoint.

But the smaller domains in (58) now let us make further progress on our SIP, (54), by going back to LAD filtering, because the local bipartite graphs have gotten significantly smaller. Indeed, exercise 243 shows that a contradiction soon arises. Thus we learn that the tentative mapping 12 $\mapsto$ AR is impossible.

So we try 12 $\mapsto$ CO next. LAD filtering is now able to remove 300 elements from the other 19 domains; that's good, yet it's significantly fewer than the 379

GAD filtering
Supplemental
McCreesh
Prosser
Trimble
Chvátal

LAD deletions that we had in the previous case. So our new LAD-consistent domains are not as constrained as those in (56) above:

$$\begin{pmatrix} \mathtt{i} & \mathtt{e} & \mathtt{d} & \mathtt{e} & \mathtt{i} \\ \mathtt{h} & \mathtt{b} & \mathtt{a} & \mathtt{b} & \mathtt{h} \\ \mathtt{g} & \mathtt{c} & \mathtt{b} & \mathtt{c} & \mathtt{g} \\ \mathtt{j} & \mathtt{g} & \mathtt{f} & \mathtt{g} & \mathtt{j} \end{pmatrix}; \quad \begin{array}{ll} \mathtt{a} = \{\mathtt{CO}\}, & \mathtt{c} = \mathtt{x} \cup \{\mathtt{AZ},\mathtt{ID},\mathtt{MO},\mathtt{SD},\mathtt{TX}\}, \\ \mathtt{b} = \mathtt{d}, & \mathtt{d} = \mathtt{x} \cup \{\mathtt{KS},\mathtt{NM},\mathtt{UT}\}, \\ \mathtt{f} = \mathtt{h}, & \mathtt{e} = \mathtt{c} \cup \{\mathtt{KS}\}, \\ \mathtt{g} = \mathtt{i}, & \mathtt{h} = \mathtt{e} \cup \{\mathtt{AR},\mathtt{IA},\mathtt{MT},\mathtt{NV},\mathtt{UT}\}, \\ \mathtt{x} = \{\mathtt{NE},\mathtt{OK},\mathtt{WY}\}, & \mathtt{i} = \mathtt{h} \setminus \{\mathtt{AZ}\} \cup \mathtt{y} \cup \{\mathtt{IL},\mathtt{LA},\mathtt{MN},\mathtt{ND},\mathtt{TN}\}, \\ \mathtt{y} = \{\mathtt{CA},\mathtt{NM},\mathtt{OR}\}, & \mathtt{j} = \mathtt{i} \cup \{\mathtt{AZ},\mathtt{KY},\mathtt{MS},\mathtt{WA},\mathtt{WI}\}. \end{array} \tag{59}$$

In this situation GAD filtering makes no change. So we need to branch again; let's try $\mathtt{11} \mapsto \mathtt{OK}$. Hurray! LAD filtering now reduces most of the domains to singletons:

$$\begin{pmatrix} \{\mathtt{LA}\} & \{\mathtt{TX}\} & \{\mathtt{NM}\} & \{\mathtt{AZ}\} & \{\mathtt{CA},\mathtt{NV}\} \\ \{\mathtt{AR}\} & \{\mathtt{OK}\} & \{\mathtt{CO}\} & \{\mathtt{UT}\} & \{\mathtt{ID},\mathtt{NV}\} \\ \{\mathtt{TN}\} & \{\mathtt{MO}\} & \{\mathtt{NE}\} & \{\mathtt{WY}\} & \{\mathtt{ID},\mathtt{MT}\} \\ \{\mathtt{KY}\} & \{\mathtt{IL}\} & \{\mathtt{IA}\} & \{\mathtt{SD}\} & \{\mathtt{MT},\mathtt{ND}\} \end{pmatrix}. \tag{60}$$

So we're almost done. Branching on $\mathtt{04} \mapsto \mathtt{CA}$ gives us Fig. 112; and the other branch gives a second solution (see exercise 244).

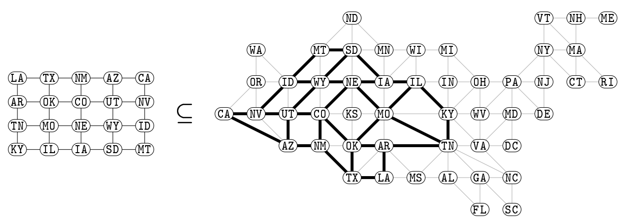


Fig. 112. One of the three ways to embed $P_4 \,\square\, P_5$ into the graph USA.

***Supplemental labels and graphs.** We've now seen how to solve problem (54), using a mixture of LAD and GAD filtering to keep the backtrack tree reasonably small. And there's another important technique that we could also have used, based on the fact that subgraph isomorphism is quite a strong property. [See C. McCreesh and P. Prosser, *LNCS* **9255** (2015), 295–312; C. McCreesh, P. Prosser, and J. Trimble, *LNCS* **12150** (2020), 316–324.] Notice, for example, that one subgraph isomorphism always implies another:

$$\text{If } G \subseteq H, \text{ then } G^{\leq 2} \subseteq H^{\leq 2}, \text{ with the same embedding.} \tag{61}$$

Here $G^{\leq 2}$ denotes the graph whose vertices are the same as those of G, but whose edges $u \mathbin{—} v$ exist if and only if there's a path of length ≤ 2 between u and v in G. If the function f embeds G into H, and if there's such a path in G, then there's clearly also a path of length ≤ 2 between $f(u)$ and $f(v)$ in H.

With (61) we can improve on what we did before. For example, suppose G is Chvátal's graph (52). Then $G^{\leq 2} = K_{12}$ and every vertex has degree 11, since the diameter is 2. But if H is BRAIN83(300), its vertices 30, 70, and 71 have degree only 9 in $H^{\leq 2}$. Therefore we can omit those three vertices from all domains, and it turns out that the SIP computation will take only 83% as long as before.

We didn't actually need the full strength of (61) in this particular case; all we used was the *degrees* of vertices in $G^{\leq 2}$ and $H^{\leq 2}$. In general, a *supplemental label* for a vertex is any function d_G for which the following property holds:

$$\text{If } G \subseteq H \text{ via embedding function } f\text{, then } d_G(v) \leq d_H(f(v)) \text{ for all } v \in G. \quad (62)$$

The degree of v in $G^{\leq 2}$ is just one example of a supplemental label.

Suppose S is an arbitrary graph, with a designated vertex s, and let $d_G^S(v)$ be the number of embeddings of S into G that map v to s. Then d_G^S is a supplemental label, because those embeddings of S into G will also be embeddings of S into the image $f(G)$ within H. We can think of S as a local "motif."

If we can somehow discover a motif S that occurs frequently in the pattern G but less often in the target H, the labels d_G^S and d_H^S will help reduce the size of initial domains when we try to embed G into H. (See also exercise 242.)

Supplemental labels can be combined in numerous ways. If d_G and d'_G are any two supplemental labels, so are $\min(d_G, d'_G)$, $\max(d_G, d'_G)$, and $\alpha d_G + \beta d'_G$ whenever $\alpha, \beta \geq 0$; indeed, so is *any* monotone combination of d_G and d'_G.

Furthermore, supplemental labels can be derived for edges as well as vertices. A supplemental edge label is a function ℓ_G for which we can prove the following:

$$\begin{aligned}&\text{If } G \subseteq H \text{ via embedding function } f,\\ &\quad\text{then } \ell_G(u,v) \leq \ell_H(f(u), f(v)) \text{ whenever } u \mathrel{—} v \text{ in } G.\end{aligned} \quad (63)$$

(It's possible to have $\ell_G^S(u,v) \neq \ell_G^S(v,u)$.) For example, let S be a motif graph in which two adjacent vertices, $s \mathrel{—} t$, have been designated; and let $\ell_G^S(u,v)$ be the number of ways we can embed S into G with $u \mapsto s$ and $v \mapsto t$. Then ℓ_G^S is a supplemental edge label, by the same reasoning we used for d_G^S above. And supplemental edge labels can be combined monotonically as before. Notice that, when S is the cycle C_k, ℓ_G^S is the number of k-cycles in G that contain a given edge.

A well-chosen supplemental edge label can significantly enhance LAD filtering. Let's go back to the USA problem of (54) and label each edge $u \mathrel{—} v$ by the number $\ell_G(u,v)$ of 4-cycles that it supports. Then ℓ_G equals 2 on every internal edge of $G = P_4 \square P_5$; and ℓ_H has interesting diversity on the edges of $H =$ `USA`. We can now, for example, prove that `11` $\mapsto$ `NY` is impossible: The neighbors of `11` are `01`, `10`, `12`, and `21`, all linked by edges with $\ell_G = 2$; the neighbors of `NY` are `CT`, `MA`, `NJ`, `PA`, `VT`, whose ℓ_H labels are respectively 2, 2, 1, 1, 2. LAD filtering rules this out, because the bipartite problem requires the four pattern vertices to match only three target vertices {`CT`, `MA`, `VT`}. Similar reasoning shows that `11` $\not\mapsto$ `AZ`, `NM`, `WI`, and 17 other targets that non-supplemental arguments had previously ruled out. The same pruning applies also, of course, to the domain of `12`.

More generally, a *supplemental pair label* ℓ_G satisfies a stronger condition:

$$\begin{aligned}&\text{If } G \subseteq H \text{ via embedding function } f,\\ &\quad\text{then } \ell_G(u,v) \leq \ell_H(f(u), f(v)) \text{ for all vertices } u \text{ and } v \text{ in } G.\end{aligned} \quad (64)$$

One way to get such a function is to designate two *non*-adjacent vertices s and t in a motif graph, and to define ℓ_G^S just as we did above. A supplemental pair label obtained in this way might turn out to be nonzero when $u \mathrel{—} v$.

supplemental label
motif
initial domains
monotone
supplemental edge label
cycle C_k
LAD filtering
supplemental pair label

Finally there's an even more powerful notion, a *supplemental graph*, which is a (possibly directed) graph on the same vertices but usually with a different adjacency relation. Suppose the following statement is true:

$$\text{If } G \subseteq H, \text{ then } G^{\Sigma} \subseteq H^{\Sigma}, \text{ with the same embedding.} \qquad (65)$$

Then we say that G^{Σ} and H^{Σ} are a pair of supplemental graphs. (We began this discussion with such a pair, in (61).)

For example, if ℓ_G is a supplemental pair label, we get a supplemental graph by letting $u \longrightarrow v$ if and only if $\ell_G(u,v) \geq k$, for any threshold k. (And we conventionally write $u \;\text{—}\; v$ if and only if we have both $u \longrightarrow v$ and $v \longrightarrow u$.) Let's say that $G^{S,k}$ is the supplemental graph we obtain in this way from the supplemental pair label ℓ_G^S. (Examples can be found in exercises 268 and 270.) The union and intersection of supplemental graphs is a supplemental graph.

And once we have a supplemental graph, we can use it to define *further* supplemental labels and graphs, based on *its* motifs!

We're clearly faced here with an embarrassment of riches. Innumerable supplemental labels and graphs can potentially be computed, perhaps turning a huge search tree into a mere shrub. On the other hand, supplemental data based on motifs that don't occur anywhere in the pattern is totally useless. A delicate balancing act is required when solving an SIP, and indeed when solving *any* CSP: It's great to reduce the number of search nodes by a factor of 10, but not when the computation time per node increases by a factor of 100, and not when there aren't extremely many nodes in the first place.

Thus a well-engineered SIP solver does its best to concentrate on supplemental data that justifies the time and space needed to compute it. We can judiciously relax our standards of LAD and GAD filtering, if our data structures allow us to do a pretty-good-but-incomplete job at high speed, as long as we don't change the set of solutions. Maximum bipartite matching problems are solved quickly by the Hopcroft–Karp algorithm (Algorithm 7.5.1H on page vi); but the existence of a suitably large matching can often be ruled out even more quickly by rudimentary tests. (See exercises 277–280.)

When C. Solnon surveyed the state of the art of SIP solving [*LNCS* **11510** (2019), 1–13], she observed that it's wise to feed your problem first to a comparatively simple solver that polishes off easy instances quickly. You can solve more problems in a given amount of time if you start in that way, but switch to heavier artillery if that solver doesn't finish in, say, 0.1 second.

Some SIP problems are extremely difficult indeed. So we can expect continued progress towards methods that ameliorate their solution — perhaps by understanding more about how to find fruitful motifs in a given pattern and target.

Special cases of subgraph isomorphism. The general SIP has many special cases that are well known by other names. For example, when the pattern graph is a path or a cycle having the same number of vertices as the target graph, the problem is to find a Hamiltonian path or Hamiltonian cycle. Special techniques apply to that problem, and we shall discuss them at length in Section 7.2.2.4. Similarly, when the pattern graph is a clique, the special methods discussed in

supplemental graph
LAD
GAD
Maximum bipartite matching
Hopcroft–Karp algorithm
Solnon
Hamiltonian
clique

Section 7.2.2.5 become available. And when the pattern graph is the same as the target graph, the solutions to the SIP are the automorphisms of that graph.

An n-vertex graph G is three-colorable if and only if $G \subseteq K_{n,n,n}$. It has bandwidth $\le k$ if and only if $G \subseteq P_n^k$, where P_n^k is the graph on $\{0, 1, \ldots, n-1\}$ with $u \mathrel{-\!\!-} v$ if and only if $|u - v| \le k$.

The special case when both pattern and target are free trees is perhaps the nicest of all, for in that case the SIP can be solved with a beautiful algorithm published by David W. Matula in 1978. His algorithm (see exercise 295) has a running time of $O(m^{1.5}n)$ in the worst case, when the pattern size is m and the target size is n; and its running time in practice is typically of order mn.

The fact that subtree isomorphism can be handled so efficiently might lead us to suspect that "subdag isomorphism" — when both pattern and target are directed acyclic graphs — might also be fairly easy. All such hopes are dashed, however, by the simple construction in exercise 228, which shows that *every* SIP can be regarded as a special case of subdag isomorphism.

The special case of trees cannot even be extended to forests: If the pattern graph G consists of disconnected trees, the problem of deciding whether or not $G \subseteq H$ turns out to be NP-hard, even when H is a free tree and G has an extremely simple form. (See exercise 220.)

On the other hand, if the pattern G is simply a collection of disjoint edges, $P_2 \oplus \cdots \oplus P_2$, an embedding of G is the same thing as a *matching*, and again we can test $G \subseteq H$ efficiently. The Hopcroft–Karp algorithm (page vi) does this well when H is bipartite, and other methods work for *arbitrary* H (see Section 7.5.5).

Solving a CSP. So far we've been looking at lots of different kinds of constraint satisfaction problems; and an endless variety of further applications beckons. But it's time now to think systematically about general approaches that we might take when we're faced with a new CSP.

In the first place, we can always basically start from scratch, and write a standalone program that's specifically tailored to whatever special problem we have in mind. In fact, Algorithm 7.2.2B, the basic backtrack algorithm, is still the method of choice for sufficiently simple tasks,* as well as for comparatively unstructured tasks like those in exercises 7.2.2–71 and 79. The CSP framework of variables, domains, and constraints has also suggested *refinements* of backtracking, such as backmarking (see exercise 310).

In the second place, we can formulate any CSP as an XCC problem — exact coloring with colors — and use the versatile methods of Section 7.2.2.1. Exercise 4 is a simple example of this general principle, and further examples can be found in exercises 61 (line labeling) and 93 (graceful labeling). Similarly, exercise 30 solves the car sequencing problem as an MCC, using Algorithm 7.2.2.1M for *non*exact covering. The notions of items and options often turn out to be more directly related to a problem than the notions of variables, domains, and constraints; for example, we saw in (53) that subgraph isomorphism is conveniently expressed

* The author still finds himself turning back to that algorithm about once a month, since customizations of 7.2.2B continue to be useful and fun, even after 60 years of experience!

automorphisms
three-colorable
bandwidth
free trees
trees
Matula
subtree isomorphism
subdag isomorphism
directed acyclic graphs
forests
NP-hard
matching
Hopcroft
Karp
bipartite
author
backtracking
backmarking
XCC
line labeling
graceful labeling
car sequencing problem
MCC
subgraph isomorphism

as an XC problem — exact covering *without* colors. Another instructive example is the "rainbow path problem" in the answer to exercise 291.

In the third place, we can formulate any CSP as a satisfiability problem, and use the extremely well-developed SAT solvers discussed in Section 7.2.2.2. This approach is often the way to go, especially if we want to find only one solution instead of the complete set, and we'll soon examine it in greater detail.

In the fourth place, we can choose from many well-designed computer programs that have been developed specifically for problems that conform explicitly to the CSP model. The task of designing a complete, general-purpose CSP solver is beyond the scope of this book; however, we shall study several of the important techniques that have been devised for such systems. A large community of researchers in constraint processing has developed new methods that enhance what we've already seen in Sections 7.2.2.1 and 7.2.2.2.

Translating CSP to SAT. The most obvious difference between the satisfiability problem that we considered in Section 7.2.2.2 and the more general CSP is the fact that satisfiability is based on *Boolean* variables, while the variables of a CSP usually have domains with *more* than two values. Large domains can, however, be represented with small domains, if we increase the number of variables.

Let's look first at the simplest non-binary case, where all CSP variables have the ternary domain $\{0,1,2\}$. (We could consider the "balanced" domain $\{-1,0,+1\}$ instead; and indeed, $\{-1,0,+1\}$ is the domain of choice in many applications. But all ternary domains are essentially equivalent to $\{0,1,2\}$; and we'll soon be studying domains $\{0,1,\ldots,d-1\}$ for $d>3$.)

One natural way to represent a ternary variable v SATwise is to encode it as three binary variables, $\{v_0,v_1,v_2\}$, where $v_j=[v=j]$. The three possible triplets $v_0v_1v_2$ are then $\{100,010,001\}$; and the other five triplets, {000, 011, 101, 110, 111} can be excluded by introducing four clauses into our SAT problem:

$$(v_0 \vee v_1 \vee v_2); \tag{66}$$

$$(\bar v_0 \vee \bar v_1) \;\wedge\; (\bar v_0 \vee \bar v_2) \;\wedge\; (\bar v_1 \vee \bar v_2). \tag{67}$$

Clause (66) says that v has *at least one* value, namely that $v_0+v_1+v_2\ge 1$; clauses (67) say that v has *at most one* value, namely that $v_0+v_1+v_2\le 1$. We've often seen this so-called *direct encoding* before, for instance in Eq. 7.2.2.2–(13).

A closer look shows that v_0 is really unnecessary here, because the three allowable pairs $v_1v_2=\{00,10,01\}$ are distinct. In fact, if we read those pairs in the opposite order, v_2v_1, we get 00, 01, and 10, which are the values 0, 1, and 2 in binary notation! When v_0 is dropped, we need only one constraint to ensure uniqueness of v's value,

$$(\bar v_1 \vee \bar v_2), \tag{68}$$

instead of the four in (66) and (67). This method is called the *log encoding*, because it generalizes to a representation of d values with only $\lceil \lg d\rceil$ binary variables. (At least $\lceil \lg d\rceil$ of them are needed, to distinguish between d cases.)

Many other encodings are also possible. Indeed, we've already made an extensive study of the mappings $x \mapsto x_l x_r$ by which a ternary variable x can

XC problem
rainbow path problem
satisfiability
SAT solvers
CSP solver
satisfiability problem
ternary domain
balanced
clauses
at least one
at most one
direct encoding
binary notation
log encoding

inequality relation: $x \neq y$
disequality, see inequality relation
not equality, see inequality rel
direct encoding

Table 2

ENCODING '$u \neq v$' WITH TERNARY DOMAINS

Name	Clauses for u	Clauses for v	Clauses for u and v
Direct	$(u_0 \vee u_1 \vee u_2)$ $(\bar u_0 \vee \bar u_1)$ $(\bar u_0 \vee \bar u_2)$ $(\bar u_1 \vee \bar u_2)$	$(v_0 \vee v_1 \vee v_2)$ $(\bar v_0 \vee \bar v_1)$ $(\bar v_0 \vee \bar v_2)$ $(\bar v_1 \vee \bar v_2)$	$(\bar u_0 \vee \bar v_0)$ $(\bar u_1 \vee \bar v_1)$ $(\bar u_2 \vee \bar v_2)$
Multivalued	$(u_0 \vee u_1 \vee u_2)$	$(v_0 \vee v_1 \vee v_2)$	$(\bar u_0 \vee \bar v_0)$ $(\bar u_1 \vee \bar v_1)$ $(\bar u_2 \vee \bar v_2)$
Log	$(\bar u_1 \vee \bar u_2)$	$(\bar v_1 \vee \bar v_2)$	$(u_2 \vee u_1 \vee v_2 \vee v_1)$ $(\bar u_1 \vee \bar v_1)$ $(\bar u_2 \vee \bar v_2)$
Binary			$(u_2 \vee u_1 \vee v_2 \vee v_1)$ $(u_2 \vee \bar u_1 \vee v_2 \vee \bar v_1)$ $(\bar u_2 \vee u_1 \vee \bar v_2 \vee v_1)$ $(u_2 \vee u_1 \vee \bar v_2 \vee \bar v_1)$ $(\bar u_2 \vee \bar u_1 \vee v_2 \vee v_1)$ $(\bar u_2 \vee \bar u_1 \vee \bar v_2 \vee \bar v_1)$
Support	$(u_0 \vee u_1 \vee u_2)$ $(\bar u_0 \vee \bar u_1)$ $(\bar u_0 \vee \bar u_2)$ $(\bar u_1 \vee \bar u_2)$	$(v_0 \vee v_1 \vee v_2)$ $(\bar v_0 \vee \bar v_1)$ $(\bar v_0 \vee \bar v_2)$ $(\bar v_1 \vee \bar v_2)$	$(\bar u_0 \vee v_1 \vee v_2)$ $(\bar u_1 \vee v_0 \vee v_2)$ $(\bar u_2 \vee v_0 \vee v_1)$ $(u_0 \vee u_1 \vee \bar v_2)$ $(u_0 \vee u_2 \vee \bar v_1)$ $(u_1 \vee u_2 \vee \bar v_0)$
Weakened	$(u_0 \vee u_1 \vee u_2)$	$(v_0 \vee v_1 \vee v_2)$	$(\bar u_0 \vee u_1 \vee u_2 \vee \bar v_0 \vee v_1 \vee v_2)$ $(\bar u_1 \vee u_2 \vee \bar v_1 \vee v_2)$ $(\bar u_2 \vee \bar v_2)$
Reduced			$(u_1 \vee u_2 \vee v_1 \vee v_2)$ $(\bar u_1 \vee \bar v_1)$ $(\bar u_2 \vee \bar v_2)$
Prefix			$(u_2 \vee u_1 \vee v_2 \vee v_1)$ $(u_2 \vee \bar u_1 \vee v_2 \vee \bar v_1)$ $(\bar u_2 \vee \bar v_2)$
Order	$(\bar u^2 \vee u^1)$	$(\bar v^2 \vee v^1)$	$(u^1 \vee v^1)$ $(\bar u^1 \vee u^2 \vee \bar v^1 \vee v^2)$ $(\bar u^2 \vee \bar v^2)$

be represented by a pair of binary variables, as part of our study of Boolean techniques: Equations 7.1.3–(110) through 7.1.3–(131) showed that the best such mapping depends heavily on the context in which the representation is used.

The context of a SAT encoding within a CSP is, of course, the set of constraints that involve the encoded variable. So let's consider how to express a given relation between two ternary variables u and v, when u and v have both been suitably encoded. We might as well begin with the simplest such relation that arises frequently in applications, namely inequality: '$u \neq v$'.

Table 2 shows nine ways to represent ternary inequality via SAT clauses. Some clauses are usually needed for u by itself and for v by itself; then there are clauses that involve both u and v. In the direct encoding, for example, Table 2 lists (66) and (67) for both variables, followed by three clauses $(\bar u_j \vee \bar v_j)$ to ensure that we don't simultaneously have $u = j$ and $v = j$.

The *multivalued encoding* is like the direct encoding, except that it omits the at-most-one clauses (67). If, say, there's a solution with $u_0 = u_1 = 1$, we can obtain two other solutions by changing either u_0 or u_1 to zero; in either case u will remain unequal to v, because $u_0 = u_1 = 1$ implies that $v_0 = v_1 = 0$.

The three clauses of the *log encoding* that forbid $u = v$ in Table 2 are the ones that don't allow the quadruple $u_2u_1v_2v_1$ to be 0000, $*1*1$, or $1*1*$.

The *binary encoding* is similar to the log encoding, but it allows *both* 11 and 00 as acceptable encodings of the domain value 0. Therefore we must forbid not only 0000, 0101, and 1010, but also 0011, 1100, and 1111.

The *support encoding* (see exercise 7.2.2.2–399) starts out like the direct encoding; but its clauses that make $u \neq v$ are quite different. For example, the clause '$(\bar{u}_0 \vee v_1 \vee v_2)$' says that $u = 0$ implies $v = 1$ or $v = 2$.

Exercise 300 explains the *weakened encoding* and the *prefix encoding*.

The *reduced encoding* is the most economical of all. Eight values of the quadruple $u_0u_1v_0v_1$ are permissible, each of which forces $u \neq v$ (see exercise 301).

Finally, Table 2 concludes with the *order encoding*, also called the *unary encoding*, which is another important idea that we've studied earlier. In this case $v^j = [v \geq j]$ (see Eq. 7.2.2.2–(163)). However, order encoding is not a really new alternative when $d = 3$, because the possible values $v^1v^2 = 00$, 10, 11 are equivalent to the log-encoded values $v_2v_1 = 10$, 00, 01, if $v^1 \leftrightarrow \bar{v}_2$ and $v^2 \leftrightarrow v_1$.

It's a nice theory. How well do these encodings work in practice? Notice that the CSP with domains $\{0, 1, \ldots, d-1\}$ and constraints $u \neq v$ between certain pairs of variables is precisely the problem of *coloring a graph with d colors*. So we can apply any of the nine encodings to the vertices and edges of any given graph G, and use a SAT solver to see whether or not G is 3-colorable. [In fact the first seven encodings of Table 2, generalized to d colors for arbitrary d, were used to test the colorability of dozens of graphs by S. Prestwich in *LNCS* **2919** (2004), 105–119, using Algorithm 7.2.2.2W (WalkSAT) as the solver.]

multivalued encoding
log encoding
binary encoding
support encoding
weakened encoding
prefix encoding
order encoding
unary encoding
coloring a graph
Prestwich
WalkSAT
Sierpiński gasket graph–
3-coloring

Fig. 113. *The Sierpiński gasket graph* $S_n^{(3)}$, shown here for $n = 4$, is created by pasting together the corners of 3^{n-1} triangles in an interesting way. Each triangle has a ternary label $\alpha = a_1 \ldots a_{n-1}$, and its corners are labeled $\alpha 0$ (top), $\alpha 1$ (lower left), $\alpha 2$ (lower right). Every vertex whose label has the form $\alpha = a_1 \ldots a_{k-1} a_k a_n \ldots a_n$, where $a_k \neq a_{k+1} = \cdots = a_n$, is pasted together with the vertex labeled $\alpha' = a_1 \ldots a_{k-1} a_n a_k \ldots a_k$. This rule gives two labels to all vertices, except for $\{0\ldots0, 1\ldots1, 2\ldots2\}$; so there are $(3^n+3)/2$ distinct vertices altogether.

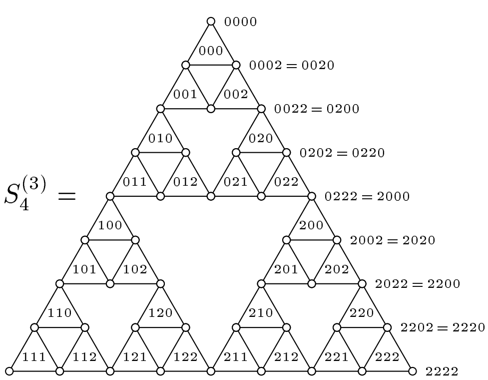
$S_4^{(3)} =$ 

Figure 113 illustrates a family of graphs for which 3-coloring is particularly instructive. The reader will have no trouble coloring the vertices of $S_4^{(3)}$ with three colors; but the interesting thing is that this coloring is essentially *unique*!

Fibonacci numbers
pinched Sierpiński gasket graph
triangles

Indeed, vertices u and v must have the same color whenever u and v lie on the same vertical line, or on any diagonal whose slope is $\pm 30°$ (see exercise 303).

Computer programmers have little difficulty verifying the uniqueness, in their heads; but it's a different story for computers themselves. Suppose, for example, that the machine has found a way to color the lower-right third of Fig. 113. Then there are two legal colors for vertices 0202 and 0212 (whose other names are 0220 and 0221). One of those colors is correct; but the other one leads to a dead end, which the machine might not discover for a long, long time. If a conventional backtrack search is used, the running time needed to color $S_{n+1}^{(3)}$ will actually be about $3+\sqrt{5} \approx 5.24$ times as long as the time that's needed for $S_n^{(3)}$. (In fact, exercise 311 shows that Fibonacci numbers have a surprising connection to this problem.)

The corners of a Sierpiński gasket graph have different colors in any 3-coloring. Let's therefore define the *pinched Sierpiński gasket graph* $\widehat{S}_n^{(3)}$ to be the same as $S_n^{(3)}$ but with the corner vertices $0\ldots 0$ and $1\ldots 1$ pasted together. This graph *cannot* be 3-colored. (Notice that $\widehat{S}_n^{(3)}$ has $\lceil 3^n/2 \rceil$ vertices, each of which has degree 4 except for the remaining corner vertex $2\ldots 2$; see page x.)

One way to compare the encodings of Table 2 is to see how long it takes for a SAT solver to prove the unsatisfiability of the clauses produced from $\widehat{S}_n^{(3)}$, with each encoding. We might save a factor of six if we introduce clauses to force the colors of the top three vertices $0\ldots 00$, $0\ldots 01$ and $0\ldots 02$ (see exercise 307).

Detailed statistics are reported in exercise 309, and the bottom line is that

$$\text{Log} \approx \text{Reduced} < \text{Prefix} \approx \text{Direct} \approx \text{Multi} \approx \text{Support} < \text{Weakened} \ll \text{Binary},$$

at least with respect to this 3-coloring problem. For example, the running times in gigamems, when Algorithm 7.2.2.2C was applied to the clauses for $\widehat{S}_9^{(3)}$, were Log (8.1), Reduced (8.6), Prefix (11.2), Direct (12.0), Multivalued (13.1), Support (13.3), Weakened (27.0), Binary (338.0), showing the median of nine runs in each case. (The binary encoding is *terrible*; we won't discuss it further.)

We can actually do better, however, because the graph $\widehat{S}_n^{(3)}$ contains lots of triangles (3-cliques); and that means we can give *clique hints* to the SAT solver. For example, whenever $u \longrightarrow v \longrightarrow w \longrightarrow u$ is a 3-clique in a graph that we want to 3-color, we can include the clauses

$$(u_0 \vee v_0 \vee w_0) \ \wedge\ (u_1 \vee v_1 \vee w_1) \ \wedge\ (u_2 \vee v_2 \vee w_2) \qquad (69)$$

when we're using the direct encoding, multivalued encoding, or support encoding, because each color must appear on one of those vertices. The other encodings also have appropriate clique hints (see exercise 315). So the running times for $\widehat{S}_9^{(3)}$ go down: Prefix (4.8), Log (5.8), Reduced (6.5), Multivalued (7.5), Direct (7.9), Support (9.6), Weakened (39.2). The prefix encoding has jumped into the lead!

Let's take a look under the hood, in order to understand a bit of what's going on. The SAT solver used in these experiments, Algorithm 7.2.2.2C, gets much of its prowess from its ability to learn new clauses, as it tries random possibilities and notices the reasons for contradictions. For example, in one attempt when given the small example $\widehat{S}_4^{(3)}$ of Fig. 113 (but pinched), the first thing that it learned after inputting the prefix-encoded clauses was

$$(\overline{0202}_2 \vee 0122_2). \tag{70}$$

It means, "if vertex 0202 has color 2, so does vertex 0122." Can you guess why? The machine tried to assume the truth of 0202_2; and that implies both $\overline{0212}_2$ and $\overline{0201}_2$; but the clique hint $(0212_2 \vee 0122_2 \vee 0201_2)$ then implies 0122_2.

Exercise 316 discusses the machine's next discovery, which was the clause '$(0202_2 \vee 0222_2)$'. Its eighth major deduction was '$(0112)_2$'; and after learning 21 clauses it was ready to deduce the empty clause, namely unsatisfiability.

Thus the magic of Boolean algebra allows a SAT solver to pursue lines of reasoning that go well beyond anything that a conventional backtracking approach would ever contemplate. But when we look at the running times by which the prefix encoding verifies uncolorability, our hopes are actually dashed:

$\widehat{S}_3^{(3)}$	$\widehat{S}_4^{(3)}$	$\widehat{S}_5^{(3)}$	$\widehat{S}_6^{(3)}$	$\widehat{S}_7^{(3)}$	$\widehat{S}_8^{(3)}$	$\widehat{S}_9^{(3)}$	$\widehat{S}_{10}^{(3)}$	$\widehat{S}_{11}^{(3)}$
1.36 Kμ	27.6 Kμ	345 Kμ	2.98 Mμ	23.0 Mμ	299 Mμ	4.77 Gμ	72.9 Gμ	1460 Gμ

This is the best of our SAT-oriented methods for $\widehat{S}_n^{(3)}$; yet when n increases by 1, its running time eventually grows by a factor exceeding 15. That's *much* worse than the factor of $3+\sqrt{5} \approx 5.236$, which we know from exercise 311 is achievable by simple backtracking! Indeed, Algorithm 7.2.2.1X is able to handle the case $n = 11$ in only 2.34 Gμ (see exercise 312), more than 600 times faster.

All is not lost, however. Algorithm 7.2.2.2C has ten tunable parameters, and the running times above were all obtained with the default settings shown in 7.2.2.2–(194). But a quite different set of parameters, 7.2.2.2–(196), is known to work much better with problems of the form *waerden*$(3, k; n)$. Filip Stappers has discovered that a similar phenomenon occurs for the pinched gasket benchmarks: He used ParamILS on small cases to obtain the somewhat eccentric settings

$$\begin{aligned}\alpha = 0.6,\ \ \rho = 0.6,\ \ \varrho = 0.99,\ \ \Delta_{\rm p} = 10000,\ \ \delta_{\rm p} = 5000,&\\ \tau = 20,\ \ w = 1,\ \ p = 0.02,\ \ P = 0,\ \ \psi = 0.15.&\end{aligned} \tag{71}$$

Those parameters make the algorithm run dramatically faster as n grows:

$\widehat{S}_3^{(3)}$	$\widehat{S}_4^{(3)}$	$\widehat{S}_5^{(3)}$	$\widehat{S}_6^{(3)}$	$\widehat{S}_7^{(3)}$	$\widehat{S}_8^{(3)}$	$\widehat{S}_9^{(3)}$	$\widehat{S}_{10}^{(3)}$	$\widehat{S}_{11}^{(3)}$
1.84 Kμ	49.0 Kμ	583 Kμ	2.84 Mμ	18.0 Mμ	90.8 Mμ	521 Mμ	2.27 Gμ	13.2 Gμ

And indeed the ratio for $\widehat{S}_{n+1}^{(3)}/\widehat{S}_n^{(3)}$ is now close to $3+\sqrt{5}$, as when backtracking.

The fact that $\widehat{S}_{11}^{(3)}$ can be proved 3-uncolorable in only 13 Gμ is quite impressive, considering that it's a problem with $3^{11} + 1 = 177148$ Boolean variables and $4 \cdot 3^{11} + 6 = 708594$ clauses! As the author was conducting these experiments in 2022, he considered also Armin Biere's "Kissat," one of the world's best contemporary solvers. Kissat, which is the fruit of a decade's further research since Section 7.2.2.2 was written, is more than twice as fast as the best solvers of 2012, on a majority of difficult problems. Kissat tunes its

empty clause
Boolean algebra
parameters, tuning
waerden
Stappers
ParamILS
author
Biere
Kissat

own internal parameters; and its running time when applied to $\widehat{S}_n^{(3)}$ turns out to have the same order of growth, $(3+\sqrt{5})^n$. (See exercise 338). It appears that this kind of machine learning cannot break through that asymptotic barrier.

Recall that we did see, way back in Fig. 92 when Algorithm 7.2.2.2C was originally defined, that SAT technology does dramatically speed up similar proofs with respect to *another* family of graphs. In that problem, which deals with the "flower snark line graphs" $L(J_q)$, the graphs in question have only $6q$ vertices and $12q$ edges, so they lead to far fewer Boolean variables. Those graphs aren't 3-colorable when q is odd; so they give us lots more cases on which we can compare the effectiveness of different SAT encodings. Let's therefore pursue the exploration of flower snarks by extending the results reported in Fig. 92.

Exercise 7.2.2.2–176(c) defines clauses called *fsnark*(q), which represent the multivalued encoding for the problem of 3-coloring the graph $L(J_q)$. We know now, however, that we can improve those clauses by also including clique hints. (Indeed, the $12q$ edges of $L(J_q)$ arise from $4q$ 3-cliques, because J_q is a cubic graph.) Furthermore we can of course consider the same problem with respect to the other encodings in Table 2. Exercise 320 shows that when $q = 99$ the respective running times, in megamems, are Log (240), Reduced (305), Prefix (339), Weakened (402), Direct (448), Multivalued (520), Support (1091).

Surprise: Those *aren't* the rankings that our experience with pinched gaskets has led us to expect, although both coloring problems seem to be quite similar.

A second surprise awaits us when we study the running times for larger and larger q. According to Fig. 92, those times grow linearly for $q \le 99$; thus if we change q to $2q+1$ we should expect the proof of unsatisfiability to take about twice as long. That's not what happens, however. Considering only the log encoding, which appears to be best for these graphs, we find

$L(J_{99})$	$L(J_{199})$	$L(J_{399})$	$L(J_{799})$	$L(J_{1599})$	$L(J_{3199})$	$L(J_{6399})$
249 Mμ	1.10 Gμ	4.66 Gμ	21.2 Gμ	48.2 Gμ	171.7 Gμ	630 Gμ

which is roughly quadratic behavior. The reasons are by no means clear, nor is much known about the effect of adapting Algorithm 7.2.2.2C's parameters $(\alpha, \rho, \varrho, \ldots, \psi)$ to the various encodings. SAT solvers are full of surprises!

So far we've been looking only at ternary domains. Domains of size 4 or more lead of course to many further questions, with seemingly endless possibilities to explore. The encodings for $d = 3$ in Table 2 can be extended to arbitrary d in interesting ways (see exercise 332). And the graphs $S_n^{(3)}$ can also be extended to *Sierpiński simplex graphs* $S_n^{(d)}$ for arbitrary d; the case $d = 4$ and $n = 3$ is illustrated in Fig. 114.

When $d = 4$, $S_n^{(d)}$ is called the Sierpiński *tetrahedron* graph of order n. Notice that $S_n^{(d)}$ is essentially a $(d-1)$-dimensional object; that makes it a bit of a challenge (but fun) to imagine when $d > 4$.

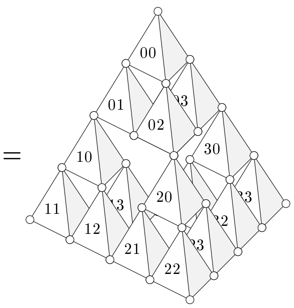
We obtain a *pinched* version $\widehat{S}_n^{(d)}$ by pasting vertices $0\ldots0$ and $1\ldots1$ together as before. Exercise 330 points out that the graph $\widehat{S}_n^{(d)}$ cannot be d-colored, when d is odd; but the situation is quite different when d is even. For

flower snark line graphs
line graphs
fsnark(q)
cubic graph
log encoding
parameters
Sierpiński simplex graphs
simplex graphs
cliques
pure vertices
pasting graphs together
Sierpiński *tetrahedron* graph
Sierpiński [sub] triangle graph see gasket
pinched

augmented Sierpinski simplex gr
prefix encoding
order encoding
Hints for d-cliques

Fig. 114. The graph $S_n^{(d)}$ is obtained by pasting together d^{n-1} cliques of size d, using the same rules that were specified for $d = 3$ in Fig. 113: Each d-clique has a d-ary label $\alpha = a_1 \ldots a_{n-1}$, and its "corner vertices" are labeled αj for $0 \le j < d$. Each vertex has two d-ary labels α and α' as before, except for the d "pure" vertices labeled $j \ldots j$; exercise 323 gives examples. Therefore there are $(d^n + d)/2$ vertices altogether.

$S_3^{(4)} =$



example, it's easy to 4-color the graph in Fig. 114 by putting the same color at each of the four corners. Yet we can't 4-color it with all-*different* corner colors.

Let's therefore define the *augmented* Sierpiński simplex graph to be

$$\overline{S}_n^{(d)} \;=\; S_n^{(d)} \text{ plus } d-1 \text{ edges } 0\ldots0 \text{ — } j\ldots j \text{ for } 0<j<d. \qquad (72)$$

This graph cannot be d-colored when $n > 1$ and d is even.

As we saw when $d = 3$, instructive results are obtained when we experiment with various SAT encodings to verify the d-uncolorability of $\widehat{S}_n^{(d)}$ for odd domain sizes d, and of $\overline{S}_n^{(d)}$ for even domain sizes. The principal contenders when $d = 4$ are the direct, multivalued, log, support, weakened, reduced, and order encodings. (See Table 2 and exercise 332; prefix encoding is the same as log encoding when $d = 4$, and order encoding becomes distinct from the others when $d > 3$.) Hints for d-cliques, discussed in exercise 333, prove to be enormously beneficial.

Detailed statistics for $d = 4$ and $n \le 7$ show that for these problems we have

$$\text{Direct} \approx \text{Multi} \approx \text{Ordered} < \text{Reduced} < \text{Support} < \text{Log} \ll \text{Weakened},$$

roughly speaking, as reported in exercise 336. The best results overall, obtained with the direct encoding, make those relative rankings quantitative:

$\overline{S}_3^{(4)}$	$\overline{S}_4^{(4)}$	$\overline{S}_5^{(4)}$	$\overline{S}_6^{(4)}$	$\overline{S}_7^{(4)}$
57.8 Kμ	1.99 Mμ	23.8 Mμ	1.16 Gμ	135 Gμ

(possibly indicating superexponential growth in the running time as n increases).

That's great news: Those running times are a huge win for SAT-based methods — because the $\overline{S}_n^{(4)}$ problem has a much, much larger search space than the $\widehat{S}_n^{(3)}$ problem does. For example, its backtrack tree appears to have about 10^{13} nodes already when $n = 4$, and more than 10^{50} when $n = 5$. The methods that we used to beat SAT in the two-dimensional case are now hopelessly inadequate.

Moving on to domains of size $d = 5$, again there are surprises (see exercise 337). The log encoding now becomes totally outclassed, and the new champion is the *reduced* encoding! Typical running times for the latter are

$\widehat{S}_3^{(5)}$	$\widehat{S}_4^{(5)}$	$\widehat{S}_5^{(5)}$	$\widehat{S}_6^{(5)}$
156 Mμ	1.78 Gμ	17.9 Gμ	172 Gμ

although the backtrack tree for $\widehat{S}_4^{(5)}$ has $\approx 10^{17}$ nodes. These are tough problems.

order encoding
2SAT
3SAT
log encoding
auxiliary variables
table constraint
Direct encoding
at-least-one
at-most-one
nogood
preclusion clause
conflict clause, see preclusion

SAT encodings of general relations. We've now seen a variety of Boolean representations of d-ary domains; but we've looked at only one constraint, '$\neq$'.

The next most important way to constrain two variables u and v is probably the relation '$u \le v$' — or perhaps '$u < v$', which is the same as '$u \le v - 1$'. The order encoding is particularly good for constraints such as this. Indeed, in the d-ary domain $\{0, 1, \ldots, d-1\}$, with the Boolean variable u^j standing for $[u \ge j]$, the relation '$u \le v - t$' for any fixed t is equivalent to the clauses

$$\bigwedge_{0 \le j \le d-t} (\bar{u}^j \vee v^{j+t}), \quad \text{if } t > 0; \qquad \bigwedge_{-t < j < d} (\bar{u}^j \vee v^{j+t}), \quad \text{if } t \le 0. \tag{73}$$

(We omit $\bar{u}^0$ or v^d if they are present.) In the case $t = 1$, for example, we get

$$\text{`}u < v\text{'} \quad \Longleftrightarrow \quad (v^1) \wedge (\bar{u}^1 \vee v^2) \wedge (\bar{u}^2 \vee v^3) \wedge \cdots \wedge (\bar{u}^{d-2} \vee v^{d-1}) \wedge (\bar{u}^{d-1}). \tag{74}$$

And we can even go much further: Exercises 7.2.2.2–405 and 406 give encodings for '$au + bv \le c$' as well as '$uv \le a$' and '$uv \ge a$', for any constants a, b, c, using only clauses that belong to 2SAT. Exercise 7.2.2.2–407 gives a 3SAT equivalent of the *ternary* relation '$u + v \le w$', when all three variables are order-encoded.

There's also a good way to translate the relation '$u \le v$' into SAT clauses when u and v have the log encoding, thanks to Eq. 7.2.2.2–(169). For example, suppose $d = 16$, $u = (u_8u_4u_2u_1)_2$, and $v = (v_8v_4v_2v_1)_2$, using four bits to represent each variable. Then we have $u \le v$ if and only if

$$(\bar{u}_8 \vee v_8) \wedge (\bar{u}_8 \vee a_1) \wedge (v_8 \vee a_1) \wedge (\bar{a}_1 \vee \bar{u}_4 \vee v_4) \wedge (\bar{a}_1 \vee \bar{u}_4 \vee a_2) \wedge (\bar{a}_1 \vee v_4 \vee a_2) \wedge \\ (\bar{a}_2 \vee \bar{u}_2 \vee v_2) \wedge (\bar{a}_2 \vee \bar{u}_2 \vee a_3) \wedge (\bar{a}_2 \vee v_2 \vee a_3) \wedge (\bar{a}_3 \vee \bar{u}_1 \vee v_1). \tag{75}$$

Notice that these clauses introduce *auxiliary variables* a_k; such variables must not be used in the encoding of any other constraint. (For instance, if we also require $v \le w$, we'd need to introduce auxiliaries called a_4, a_5, and a_6, say.) Exercise 341 shows that a similar scheme can encode '$u \le v - t$' for any t.

In general, however, a CSP can involve arbitrary constraints that don't have nice properties like the relation '$u \le v - t$'. A so-called "table constraint" is specified by tabulating the pairs (u, v) that satisfy it. (Or by listing the pairs that *don't* satisfy it, if the bad pairs are easier to specify than the good ones.) If we can deal with any table constraint, we can handle any constraint whatsoever.

Table constraints are usually translated into SAT by letting the Boolean variable v_a represent $[v = a]$, for each value a in the domain of each variable v, as we've done in most of the examples above. Here are the most popular schemes:

- *Direct encoding.* Start with the at-least-one and at-most-one clauses for each variable, as in (66) and (67). Then, for each pair of values (a, b) such that the assignments $u = a$ and $v = b$ do *not* satisfy the given relation — a so-called *nogood* — add the "preclusion clause" $(\bar{u}_a \vee \bar{v}_b)$, also called a "conflict clause."

(Thus Table 2, which encodes '$u \neq v$' in ternary domains, has three nogoods.)

Notice that the direct encoding works naturally for k-ary constraints as well as for binary constraints: If the values $(a_1, \ldots, a_k)$ don't satisfy a given relation on the variables $(v_1, \ldots, v_k)$, the preclusion clause is $(\bar{v}_{1a_1} \vee \cdots \vee \bar{v}_{ka_k})$.

Support encoding
k-ary to binary
hidden variable
hypergraph
bipartite graph
direct encoding
support encoding

- *Support encoding.* Given a binary relation $R(u,v)$, start with the at-least-one and at-most-one clauses as above. Then add the "support clauses"

$$\bigwedge_{a\in D_u}\Bigl(\bar u_a \vee \bigvee\{v_b \mid ab \in R(u,v)\}\Bigr) \wedge \bigwedge_{b\in D_v}\Bigl(\bar v_b \vee \bigvee\{u_a \mid ab \in R(u,v)\}\Bigr). \qquad (76)$$

(The domains are D_u and D_v. In Table 2, $D_u = D_v = [0\,..\,3)$, $R(u,v) = [u \neq v]$.)

The support encoding can also be defined for k-ary relations $R(v_1,\ldots,v_k)$. But in this case we use a trick by which any k-ary relation can be regarded as a set of k *binary* relations $R_j(v_j,R)$; here R_j relates the original variable v_j to a new "hidden variable" R, whose domain D_R is the set $\{a_1\ldots a_k \mid R(a_1,\ldots,a_k)\}$ of all tuples that satisfy R. If $a \in D_{v_j}$ and $a_1\ldots a_k \in D_R$, then we have

$$R_j(a, a_1\ldots a_k) \iff a = a_j, \quad \text{for } 1 \le j \le k. \qquad (77)$$

(The idea concealed in this daunting notation is basically an elaboration of the way in which we represented a hypergraph as a bipartite graph in 7–(57).)

Let's study a simple example, by considering the case where $R = R(u,v,w)$ is the following more-or-less random ternary relation on ternary variables $\{u,v,w\}$:

$$R(u,v,w) \iff uvw \in \{000, 001, 010, 012, 020, 121, 211\}. \qquad (78)$$

The direct encoding for R has $3^3 - 7 = 20$ nogoods, because R has seven tuples; so it consists of the at-least-one and at-most-one clauses together with

$$\begin{aligned}
&(\bar u_0\vee\bar v_0\vee\bar w_2) \wedge (\bar u_0\vee\bar v_1\vee\bar w_1) \wedge (\bar u_0\vee\bar v_2\vee\bar w_1) \wedge (\bar u_0\vee\bar v_2\vee\bar w_2) \wedge (\bar u_1\vee\bar v_0\vee\bar w_0) \wedge\\
&(\bar u_1\vee\bar v_0\vee\bar w_1) \wedge (\bar u_1\vee\bar v_0\vee\bar w_2) \wedge (\bar u_1\vee\bar v_1\vee\bar w_0) \wedge (\bar u_1\vee\bar v_1\vee\bar w_1) \wedge (\bar u_1\vee\bar v_1\vee\bar w_2) \wedge\\
&(\bar u_1\vee\bar v_2\vee\bar w_0) \wedge (\bar u_1\vee\bar v_2\vee\bar w_2) \wedge (\bar u_2\vee\bar v_0\vee\bar w_0) \wedge (\bar u_2\vee\bar v_0\vee\bar w_1) \wedge (\bar u_2\vee\bar v_0\vee\bar w_2) \wedge\\
&(\bar u_2\vee\bar v_1\vee\bar w_0) \wedge (\bar u_2\vee\bar v_1\vee\bar w_2) \wedge (\bar u_2\vee\bar v_2\vee\bar w_0) \wedge (\bar u_2\vee\bar v_2\vee\bar w_1) \wedge (\bar u_2\vee\bar v_2\vee\bar w_2). \qquad (79)
\end{aligned}$$

The support encoding for R is obtained by combining the support encodings for the three binary relations $R_u(u,R)$, $R_v(v,R)$, and $R_w(w,R)$, namely

$$(R_{000}\vee R_{001}\vee R_{010}\vee R_{012}\vee R_{020}\vee R_{121}\vee R_{211}); \qquad (80)$$

$$\begin{array}{ll}
 & (\bar u_0\vee R_{000}\vee R_{001}\vee R_{010}\vee R_{012}\vee R_{020}),\\
(\overline R_{000}\vee u_0) \wedge (\overline R_{000}\vee v_0) \wedge (\overline R_{000}\vee w_0), & (\bar u_1\vee R_{121}),\\
(\overline R_{001}\vee u_0) \wedge (\overline R_{001}\vee v_0) \wedge (\overline R_{001}\vee w_1), & (\bar u_2\vee R_{211});\\
(\overline R_{010}\vee u_0) \wedge (\overline R_{010}\vee v_1) \wedge (\overline R_{010}\vee w_0), & (\bar v_0\vee R_{000}\vee R_{001}),\\
(\overline R_{012}\vee u_0) \wedge (\overline R_{012}\vee v_1) \wedge (\overline R_{012}\vee w_2), & (\bar v_1\vee R_{010}\vee R_{012}\vee R_{211}),\\
(\overline R_{020}\vee u_0) \wedge (\overline R_{020}\vee v_2) \wedge (\overline R_{020}\vee w_0), & (\bar v_2\vee R_{020}\vee R_{121});\\
(\overline R_{121}\vee u_1) \wedge (\overline R_{121}\vee v_2) \wedge (\overline R_{121}\vee w_1), & (\bar w_0\vee R_{000}\vee R_{010}\vee R_{020}),\\
(\overline R_{211}\vee u_2) \wedge (\overline R_{211}\vee v_1) \wedge (\overline R_{211}\vee w_1); & (\bar w_1\vee R_{001}\vee R_{121}\vee R_{211}),\\
 & (\bar w_2\vee R_{012}); \qquad (81)
\end{array}$$

$$\begin{aligned}
&(u_0\vee u_1\vee u_2) \wedge (\bar u_0\vee\bar u_1) \wedge (\bar u_0\vee\bar u_2) \wedge (\bar u_1\vee\bar u_2);\\
&(v_0\vee v_1\vee v_2) \wedge (\bar v_0\vee\bar v_1) \wedge (\bar v_0\vee\bar v_2) \wedge (\bar v_1\vee\bar v_2);\\
&(w_0\vee w_1\vee w_2) \wedge (\bar w_0\vee\bar w_1) \wedge (\bar w_0\vee\bar w_2) \wedge (\bar w_1\vee\bar w_2). \qquad (82)
\end{aligned}$$

At-most-one clauses for R, such as $(\overline R_{000}\vee\overline R_{001})$, aren't needed (see exercise 347).

projections
lossless join dependency
join dependency
n queens
queens
forcing
unit propagations

- *Encoded projections.* A k-ary relation can be "projected" onto any subset of its variables, obtaining a weaker relation that must also be true. The conjunction of these weaker relations is an approximation to the overall one.

For example, the ternary relation (78) has three projections onto binary relations:

$$uv \in \{00, 01, 02, 12, 21\}; \tag{83}$$
$$uw \in \{00, 01, 02, 11, 21\}; \tag{84}$$
$$vw \in \{00, 01, 10, 11, 12, 20, 21\}. \tag{85}$$

We need $3+3+1$ preclusion clauses to rule out their inadmissible pairs. That leaves the seven tuples of R, and also 011; so one more preclusion clause,

$$(\bar{u}_0 \vee \bar{v}_1 \vee \bar{w}_1), \tag{86}$$

will give us the equivalent of (79) in the direct encoding.

(In database theory, a relation that's equal to the intersection of some of its projections is said to have a *lossless join dependency* on those projections.)

Exercise 353 shows that about 1.2% of all ternary relations on ternary domains can be decomposed losslessly into their binary projections. The remaining 98.8% are inherently ternary; but nearly half of them are *almost* decomposable, needing only five or fewer additional preclusions such as (86). (See exercise 354.)

Notice that the direct encoding is smallest when there are comparatively few nogood tuples, as we saw in the relation '$u \neq v$'; contrariwise, the support encoding is smallest when there are comparatively few *good* ones. The tradeoff is often tricky. When trying to place n queens, for example, exercise 7.2.2.2–400 concludes that the direct encoding is preferable when trying to find just one solution to that problem, but the support encoding is better for finding all solutions.

We needn't choose a single encoding scheme; the best solution for some applications might be to use two different encodings simultaneously.

Recall from Eq. 7.2.2.2–(180) that some encodings are *forcing*, in the sense that every implied consequence with respect to the individual (nonauxiliary) literals can be found efficiently by a SAT solver using only unit propagations. Furthermore, exercise 7.2.2.2–433 showed that the log encoding in (75) is forcing for the relation '$u \leq v$'. Thus, for example, if $u_4 = u_1 = 1$ and $v_2 = v_1 = 0$, then unit propagation in (75) will force $v_8 = 1$ and $u_8 = 0$.

Forcing clauses are obviously desirable, if they don't take up too much space. The direct encoding usually doesn't have the forcing property; for instance, if we assert $u_0 = 0$ in (79), unit propagation does nothing. By contrast, however, asserting $u_0 = 0$ in (81) immediately implies $\overline{R}_{000}$, $\overline{R}_{001}$, $\overline{R}_{010}$, $\overline{R}_{012}$, $\overline{R}_{020}$, $\bar{v}_0$, $\bar{w}_0$, $\bar{w}_2$; hence w_1, by (82). The good news is that *the support encoding is always forcing.* (See exercise 358; we can regard variables R_{000}, R_{001}, ... as auxiliary.)

* * *

Consistency.

* * *

Who knows what I might eventually say next? (There will soon be an Algorithm D, to establish domain consistency.)

Algorithm D
Algorithm C
dancing cells
dancing links

* * *

Another thing I'm planning is Algorithm C, which will be an XCC *solver based on "dancing cells" as an alternative to dancing links. A prototype is online at* `http://cs.stanford.edu/~knuth/programs/ssxcc.w`.

A brief history. The notion of "constraint satisfaction problems" was introduced and named by Richard E. Fikes in *Artificial Intelligence* **1** (1970), 27–120, 299. He implemented an elaborate system that generated a sequence of CSPs from a given nondeterministic program in a fairly general language; the goal was to solve one or more of the resulting CSPs. His system included more than a dozen constraint manipulation methods by which it was possible to eliminate variables and/or to reduce their domains and/or to discover contradictions.

Before the 1970s, a search for combinatorial patterns was generally specified by prescribing one or more *global* constraints that the variables of a problem were supposed to satisfy. A more nuanced understanding, by which such objectives could often best be regarded as networks of *local* constraints, was then formulated by Ugo Montanari in *Information Sciences* **7** (1974), 95–132.

Montanari limited his discussion to the special case in which all constraints are binary. In other words, he considered n-tuples $(x_1, \ldots, x_n)$ such that $x_j \in D_j$ for $1 \le j \le n$, and such that $(x_i, x_j) \in R_{ij}$ for certain ordered pairs (i, j), where each D_j was a given finite set and each $R_{ij} \subseteq D_i \times D_j$ was a given binary relation. He'd been working with digitized pictures, containing $n \approx 1000$ pixel values x_j, where each domain D_j had roughly 20 values. In such problems he expected almost all of the constraints to involve geometrically adjacent pixels x_i and x_j, so that only $O(n)$ or $O(n \log n)$ relations would need to be specified. His goal was to reduce the search space by doing some sort of preprocessing to simplify them.

He required each relation R_{ii} between a variable and itself to be a subset of the identity relation $x = y$; but (curiously and unnecessarily) he allowed R_{ij} and R_{ji} to be independent of each other. His main contribution was the following algorithm to refine the given network of relations:

$$\text{For } 1 \le k \le n, \text{ set } R_{ij} \leftarrow R_{ij} \cap R_{ik} R_{kk} R_{kj} \text{ for } 1 \le i, j \le n. \qquad (200)$$

Here each R_{ij} is regarded as a $|D_i| \times |D_j|$ matrix of 0s and 1s, and the matrix multiplication is Boolean (namely ORs of ANDs, not sums of products). If any R_{ij} is changed by this process, the entire computation (200) is supposed to be repeated, until no further changes occur. Finally a form of path consistency will have been achieved (see exercise 502).

Algorithm (200) was inspired by an algorithm for all shortest paths due to R. W. Floyd [*CACM* **5** (1962), 345], which in turn was related to the solution of simultaneous linear equations by Gaussian elimination. It's *not* very efficient; notice, for example, that it may well constrain variables that were initially unconstrained, because R_{ij} might change from $|D_i| \times |D_j|$ to something smaller. But it was a start, and it encouraged other researchers to find improvements.

Meanwhile, as we have seen, D. A. Huffman and M. B. Clowes had independently come up with an interesting system of constraints, both binary and ternary, between adjacent lines in digitized images. Their ideas about line labeling were considerably extended by D. L. Waltz, who showed how to deal not only with the edges of polyhedra but also with the complex *shadows* that are cast by such objects. [See his Ph.D. thesis (MIT report TR-271, November 1972),

historical remarks–
Fikes
Montanari
pixel
identity relation
0s and 1s
Boolean matrix multiplication
path consistency
all shortest paths
shortest paths
Floyd
Gaussian elimination
Huffman
Clowes
line labeling
Waltz
shadows

349 pages; partially summarized in *The Psychology of Computer Vision*, edited by P. Winston (McGraw–Hill, 1975), 19–91.] He found that the propagation of such local constraints led to enormous speedups in the recognition of scenes, and his approach became known as the "Waltz filter."

But let's backtrack. Several years before computer scientists had been attaching interesting symbolic labels to lines in scenes, combinatorial mathematicians had been attaching interesting numbers to the vertices of graphs. Alexander Rosa published an influential paper [in *Theory of Graphs* (Paris: Dunod, 1967), 349–355], based on his dissertation written in 1965, that introduced four kinds of labelings called α-valuations, β-valuations, σ-valuations, and ρ-valuations. Every α-valuation was a β-valuation; every β-valuation was a σ-valuation; every σ-valuation was a ρ-valuation; and every ρ-valuation was enough to show that the m edges of the underlying graph could cover all edges of the complete graph K_{2m+1} in rainbow fashion, when rotated cyclically as in Fig. 110(c).

S. W. Golomb began to think about graph labels independently, because he wanted a convenient way to identify the terminals of communication networks and the interconnections between them. He decided to call a graph "graceful" if it had an ideal labeling by his criterion; and of course he told his good friend Martin Gardner about these ideas. Martin wrote about "The graceful graphs of Solomon Golomb, or how to number a graph parsimoniously" in *Scientific American* **226**, 3 (March 1972), 108–112; Golomb's own publication appeared at about the same time in *Graph Theory and Computing* (Academic Press, 1972), 23–37. People soon discovered that Rosa's β-valuations were exactly the same as Golomb's graceful labelings, and interest in the subject began to take off.

Rosa's ρ-valuations eventually became known as "rainbow graceful" — a nice coincidence, because "ρ" stands for both "rainbow" and "Rosa."

The first significant algorithm for subgraph isomorphism was developed by E. H. Sussenguth, Jr., motivated by queries to databases of chemical compounds [*J. Chemical Documentation* **5** (1965), 36–43]. He considered induced subgraphs of labeled structures, and based his method on supplemental labels that he called "properties," such as the length of a shortest cycle (if any) from a vertex to itself. His implementation used bitwise operations to represent the sets of pattern and target vertices that have various combinations of label values. Several years later, J. R. Ullmann independently described bitwise techniques for finding *non*-induced copies of a given pattern in a given target [*JACM* **23** (1976), 31–42].

* * *

(more history to come, when more subsections are written)

* * *

Many other historical notes appear with the answers to particular exercises. They can be located by consulting "Historical notes" in the index.

Winston
propagation
Waltz filter
Rosa
Golomb
networks
Gardner
rainbow graceful
subgraph isomorphism
Sussenguth
chemical compounds
induced subgraphs
supplemental labels
bitwise operations
Ullmann

EXERCISES

1. [*01*] Find all solutions to the CSP in (1) and (2).

2. [*21*] Every 3SAT problem with m clauses on n Boolean variables can be regarded as a CSP with n variables, binary domains, and m ternary constraints. (See (3).)

a) Instead, represent it with m variables, *ternary* domains, and *binary* constraints.

b) What CSP does your method construct from the 3SAT problem R' in 7.2.2.2–(7)?

c) Reduce the number of binary constraints to $3m$, by adding n binary variables.

d) What CSP do you get from 7.2.2.2–(7) now?

3. [*18*] Express the CSP of (1) and (2) as a SAT problem.

4. [*15*] Express the CSP of (1) and (2) as an XCC problem.

5. [*M05*] The Cartesian product D^0 of 0 copies of a set D consists of a single element, the 0-tuple, denoted by ϵ. Describe all of the possible nullary relations.

▸ **6.** [*M16*] When f is a function from a set A to a set B, textbooks of mathematics traditionally say that A is the "domain" and B is the "range." But when h is the function in a CSP that takes i to x_i, the literature of constraint processing traditionally says that x_i lies in the domain — *not* the range! Discuss.

8. [*15*] True or false: If there's a homomorphism from the cyclic graph C_9 to a given graph G, that graph must contain either a 3-cycle or a 9-cycle.

▸ **9.** [*M25*] Is it hard to decide if there's a homomorphism from a given graph to C_5?

▸ **10.** [*25*] Explain why the following problems are special cases of the GCP.

a) Does graph $G = (V, E)$ have an independent set of size k? (Can we choose k distinct vertices in G without selecting any neighbors?)

b) Does graph $G = (V, E)$ have a vertex cover of size k? (Are there k vertices that "hit" every edge of G at least once?)

c) Are graphs $G = (V, E)$ and $G' = (V', E')$ isomorphic? (Is there a one-to-one correspondence between their vertices so that $u \mathrel{—} v$ in $G \iff h(u) \mathrel{—} h(v)$ in G'?)

d) Does graph $G = (V, E)$ have bandwidth k? (Can its vertices be given distinct integer labels so that $u \mathrel{—} v$ implies $|h(u) - h(v)| \le k$?)

e) Does the directed graph $G = (V, A)$ have an Eulerian trail? (Can we "walk" through it, traversing every arc exactly once?)

11. [*20*] (P. Jeavons.) The k-tuple $x_1 \ldots x_k$ is said to be *unlike* the k-tuple $x'_1 \ldots x'_k$ if $x_j \ne x'_j$ for $1 \le j \le k$. It's convenient to write '$x_1 \ldots x_k \parallel x'_1 \ldots x'_k$' when this is true.

Let R be a k-ary relation on a set V. What's a "natural" way to understand the significance of a homomorphism from $(V, \ne)$ to $(R, \parallel)$?

▸ **15.** [*M12*] Why is the general combinatorial problem (GCP) a special case of the CSP?

18. [*HM34*] Let $G(z) = G_N(z) = \sum z^{E(\Sigma)}$ be the generating function for energy, summed over all 2^N one-dimensional Ising configurations Σ, as defined in (9).

a) Find a "closed-form" expression for $G(z)$, when B is (i) 0; (ii) arbitrary.

b) What is the *average* energy per particle, $zG'(z)/(NG(z))$, when $z = e^{-\beta}$?

c) Express those quantities asymptotically as $N \to \infty$.

d) Also evaluate $G_k(z) = \sum \sigma_k z^{E(\Sigma)}$, and the "average magnetization" $\frac{1}{N}\sum_{k=1}^{N} \frac{G_k(z)}{G(z)}$.

▸ **20.** [*20*] Is the all-different constraint *really* necessary, when the crystal maze puzzle (11) already has seventeen constraints like (12)? How about when there are just seven constraints like (15)?

3SAT
CSP represented as SAT
SAT representation of CSP
CSP represented as XCC
XCC representation of CSP
Cartesian product
0-tuple
nullary relations
domain
range
homomorphism
cyclic graph
independent set
vertex cover
isomorphic
bandwidth
Eulerian trail
Jeavons
unlike
Notations: $\parallel$
general combinatorial problem
GCP
generating function
Ising configurations
partition function
asymptotically
magnetization
all-different constraint
crystal maze puzzle

symmetry
car sequencing problem
MCC problem
(p/q)-string
generating functions
extreme
plane partitions
Szilassi
polyhedron

21. [*21*] Since the graph in (11) is symmetric, every essentially different solution to the CSP models in the text will be found four times. Explain how to exploit symmetry.

▶ **22.** [*20*] Express (11) as an exact cover problem with primary items $\{1,\ldots,8,\mathtt{A},\ldots,\mathtt{H}\}$.

23. [*22*] Express (11) as a CSP with only 7 variables. *Hint:* Use edges, not vertices.

26. [*20*] Solve the car sequencing problem of Fig. 100 and (16).

27. [*15*] Why can the solution to exercise 26 assume $f < 5$, in the text's formulation?

▶ **28.** [*M25*] The redundant constraints in (18) are asymmetrical: They all apply at the *left* of the sequence, because they involve f_{0k}. We could generalize them, and require

$$f_{(l'q_k)k} + f_{(l'q_k+1)k} + \cdots + f_{(t-l''q_k-1)k} \geq r_k - (l' + l'')p_k$$

in the "middle" of the sequence, where $l' + l'' < \lceil r_k/p_k \rceil$. Would that be a good idea?

▶ **30.** [*21*] Express the car sequencing problem as an MCC problem without using colors.

31. [*21*] Improve the previous answer by incorporating the redundant constraints (18).

▶ **32.** [*20*] Extend (16) to two new types of car: Model G has premium audio and heated seats only; Model H is "loaded" with every feature *except* heated seats. Then the 30 cars $\{7\cdot\mathrm{A}, 2\cdot\mathrm{B}, 5\cdot\mathrm{C}, 4\cdot\mathrm{D}, 4\cdot\mathrm{E}, 2\cdot\mathrm{F}, 4\cdot\mathrm{G}, 2\cdot\mathrm{H}\}$ have overall requirements $(r_0, \ldots, r_4) = (15, 20, 10, 12, 6)$, which are the maximum that could conceivably be installed in 30 cars.

Does that "tight" car sequencing problem have a solution? Answer this question by applying Algorithm 7.2.2.1M to the MCC encoding of (a) exercise 30; (b) exercise 31.

33. [*21*] If we double all the requirements of exercise 32, we get a 60-car problem. Unfortunately that problem has no solution. Is there, however, a solution to the 61-car problem in which we manufacture one extra "Model 0" car (with *no* optional features)?

35. [*M25*] Inspired by the car sequencing problem, let's say that a "(p/q)-string" is a binary string in which no q consecutive bits contain more than p 1s.

a) How many strings of length 10 are $(1/2)$-strings? $(1/3)$-strings? $(2/3)$-strings?
b) What is the maximum number of 1s in a (p/q)-string of length n?
c) Find the generating functions $G_{pq}(z) = \sum_{n\geq 0} C_{pqn} z^n$ for $0 < p < q \leq 5$, where C_{pqn} is the number of (p/q)-strings of length n.

▶ **36.** [*M35*] A (p/q)-string with the maximum number of 1s is called *extreme*.

a) Let $e_{pq}(m)$ be the number of (p/q)-strings of length qm that contain exactly pm 1s. Prove that $e_{pq}(m)$ is the number of plane partitions that fit in a $p \times (q-p) \times m$ box (see answer 7.2.2.1–262). *Hint:* Find a one-to-one correspondence.
b) Let c_{pqn} be the number of extreme (p/q)-sequences of length n. Express c_{pqn} in terms of the numbers in part (a).

39. [*M21*] (L. Szilassi, 1986.) Regard each of the following 14 triples ijk of digits

$$023, 134, 245, 356, 460, 501, 612,\ \ 054, 165, 206, 310, 421, 532, 643$$

as a cycle that contains the pairs ij, jk, and ki. Then every pair of distinct digits $i \neq j$ with $0 \leq i, j < 7$ occurs exactly once. Show that those triples can be assigned to points (x, y, z) in such a way that every triple containing digit j belongs to plane j, where plane 0 is '$z = 0$'; plane 1 is '$4y + z = 200$'; plane 2 is '$2x + z = -280$'; plane 3 is '$5x - 5y + 7z = -700$'; plane 4 is '$-5x + 5y + 7z = -700$'; plane 5 is '$-2x + z = -280$'; plane 6 is '$-4y + z = 200$'. Furthermore, the six triples containing j form the boundary of a polygon that defines the face of a polyhedron, for $0 \leq j < 7$.

▸ **40.** [*M28*] Three-dimensional space can be discretized into little "cubies," where cubie (i,j,k) consists of all points (x,y,z) with $i \le x \le i+1$, $j \le y \le j+1$, and $k \le z \le k+1$. (Each cubie therefore shares a common face with 6 adjacent cubies, a common edge with 12 diagonally adjacent cubies, and a common vertex with 8 corner-adjacent cubies.)

Given an $m \times n$ matrix (a_{ij}) for $0 \le i < m$ and $0 \le j < n$, its *histoscape* is the set of cubies (i,j,k) for $0 \le k < a_{ij}$. (For example, Fig. 101(d) is the histoscape for $\binom{4\,3}{1\,2}$.)

How many 2×2 matrices with $0 \le a_{ij} < 10$ have a histoscape that's a 3VP?

▸ **41.** [*M27*] Continuing exercise 40, how many of the 10^{64} 8×8 matrices whose entries satisfy $0 \le a_{ij} < 10$ for $0 \le i,j < 8$ have a histoscape that's a 3VP? *Hint:* Formulate this question as a constraint satisfaction problem.

42. [*24*] Extend the algorithm of the previous exercise so that it will find the kth $m \times n$ histoscape whose entries satisfy $0 \le a_{ij} < t$, given k, m, n, and t, when those histoscapes are listed in some convenient order. Then, by choosing k at random, use your method to find a uniformly random solution to the 8×8 problem.

▸ **43.** [*M26*] Given an $m \times n$ matrix whose histoscape is a 3VP, what are its vertices, and what polygons define its faces? (Design an algorithm that answers these questions.)

▸ **44.** [*M21*] (*Whirlpool permutations.*) An $m \times n$ matrix has $(m-1)(n-1)$ submatrices of size 2×2. An $m \times n$ "whirlpool permutation" is an $m \times n$ matrix containing mn distinct numbers, in which the relative order of the elements in each of those submatrices is a "vortex" — that is, it travels a cyclic path from smallest to largest, either clockwise or counterclockwise.

Thus there are eight 2×2 whirlpool permutations of $\{1,2,3,4\}$:

$$\begin{pmatrix}1&2\\4&3\end{pmatrix}\quad\begin{pmatrix}1&4\\2&3\end{pmatrix}\quad\begin{pmatrix}2&1\\3&4\end{pmatrix}\quad\begin{pmatrix}2&3\\1&4\end{pmatrix}\quad\begin{pmatrix}3&2\\4&1\end{pmatrix}\quad\begin{pmatrix}3&4\\2&1\end{pmatrix}\quad\begin{pmatrix}4&1\\3&2\end{pmatrix}\quad\begin{pmatrix}4&3\\1&2\end{pmatrix}.$$

a) The 4×4 matrix at the right is not quite a whirlpool permutation. Fix the problem by interchanging two rookwise adjacent elements.

$$\begin{pmatrix}16&3&2&13\\9&7&8&10\\5&6&12&11\\4&14&15&1\end{pmatrix}$$

b) Prove that if any two rookwise adjacent elements of a whirlpool permutation are interchanged, the result is *not* a whirlpool.

c) What is the lexicographically smallest $m \times n$ whirlpool permutation of $\{1,\ldots,mn\}$?

d) True or false: The histoscape of an $m \times n$ matrix with distinct elements is a 3VP if and only if that matrix is a whirlpool permutation. (See Fig. 101(d).)

e) If M exceeds the difference between the largest and smallest elements of a whirlpool permutation, and if x is any number, prove that the matrix obtained after replacing each element a_{ij} by $(a_{ij}+x) \bmod M$ is also a whirlpool permutation.

▸ **45.** [*M30*] How many 5×5 matrices are whirlpool permutations of $\{0,1,\ldots,24\}$? *Hint:* An algorithm similar to that of exercise 41 can be used to count them.

▸ **46.** [*HM35*] An *up-up-or-down-down permutation* of $2n-1$ elements is a permutation $a_1a_2\ldots a_{2n-1}$ for which $a_{2k-1} < a_{2k}$ if and only if $a_{2k} < a_{2k+1}$, for $0 < k < n$. Let U_n be the number of such permutations; for example, $(U_1,\ldots,U_5) = (1,2,14,204,5104)$.

a) Prove that $U_{n+1} = \sum_k \binom{2n}{2k} Q_k Q_{n-k}$, where $Q_k = (k=0?\ 1 :\ kU_k)$.

b) Find the exponential generating function $U(z) = U_1z/1! + U_2z^3/3! + U_3z^5/5! + \cdots$.

c) What is the asymptotic behavior of U_n, correct to relative error $(1+O(1/4^n))$?

d) The number of $2 \times n$ whirlpool permutations is $2nU_n$. Prove this by establishing a one-to-one correspondence between up-up-or-down-down permutations and $2 \times n$ whirlpool permutations of $\{0,\ldots,2n-1\}$ with first element 0.

cubies
histoscape
constraint satisfaction problem
uniformly random solution
Whirlpool permutations
vortex
Dürer
lexicographically smallest
up-up-or-down-down permutation
exponential generating function
generating function
whirlpool permutations

47. [*21*] Which of the following partially filled 5×5 matrices can be completed to a whirlpool permutation of $\{1, 2, \ldots, 25\}$ in exactly one way?

$$\text{(i)}\ \begin{array}{|c|c|c|c|c|}\hline 1&3&5&7&9\\\hline 17&&&&\\\hline &&25&&\\\hline &&&&\\\hline 2&4&6&8&10\\\hline\end{array}\,;\quad \text{(ii)}\ \begin{array}{|c|c|c|c|c|}\hline 3&14&15&9&2\\\hline &6&&5&\\\hline &&&&\\\hline &&&&\\\hline &&&&\\\hline\end{array}\,;\quad \text{(iii)}\ \begin{array}{|c|c|c|c|c|}\hline 3&14&15&&\\\hline &9&2&6&\\\hline 5&&&&\\\hline &1&25&22&\\\hline &&11&21&19\\\hline\end{array}\,;\quad \text{(iv)}\ \begin{array}{|c|c|c|c|c|}\hline 3&&14&&15\\\hline 9&&2&&6\\\hline &&5&&\\\hline 1&&21&&25\\\hline 4&&18&&22\\\hline\end{array}\,.$$

▸ **50.** [*M25*] The *skeleton* of a polyhedron is the graph formed by its vertices and edges. Hence the skeleton of a 3VP is a cubic graph. Make sketches of four 3VPs, each of which has the same skeleton as the 3-cube, but they differ in the number of concave edges.

51. [*M20*] The *signed skeleton* of a polyhedron is like its skeleton, but each edge is also identified as being either concave or convex. In illustrations we can indicate a convex edge by a solid line and a concave edge by a dashed line; for example, the signed skeletons of the objects in answer 50 are

What is the signed skeleton of the Szilassi polyhedron?

52. [*HM46*] Is there an algorithm to decide whether or not a given signed cubic graph can be realized as the signed skeleton of some 3VP?

54. [*HM20*] Let v_0 be a vertex of X, where X is a 3VP. Let the three neighbors of v_0 in the skeleton of X be $\{v_1, v_2, v_3\}$, and let each v_i have Cartesian coordinates (x_i, y_i, z_i).

a) Show that we can always choose the subscripts in such a way that

$$D(v_0, v_1, v_2, v_3) > 0, \qquad \text{where } D(v_0, v_1, v_2, v_3) = \det\begin{pmatrix} x_0 & y_0 & z_0 & 1\\ x_1 & y_1 & z_1 & 1\\ x_2 & y_2 & z_2 & 1\\ x_3 & y_3 & z_3 & 1\end{pmatrix}.$$

b) Let p_{12} be the plane that contains v_0, v_1, and v_2. What equation defines the set of all vectors $v = (x, y, z)$ that lie in p_{12}?

c) What inequality characterizes all $v = (x, y, z)$ that lie on the same side of p_{12} as v_3?

d) Define p_{23} and p_{31} by analogy with p_{12}. Then the three planes p_{12}, p_{23}, p_{31} divide three-dimensional space into eight "octants": Every point v lies on one side or the other of each plane, unless it belongs to that plane. Devise a computer-friendly way to number the octants *0* to *7* in octal notation.

e) Using your numbering scheme, what octant contains the "three-famous-constants" point (π, ϕ, γ) when $v_0 = (0, 0, 0)$, $v_1 = (1, 0, 0)$, $v_2 = (0, 1, 0)$, $v_3 = (0, 0, 1)$?

f) Same as (e), but $v_0 = (0, 0, 0)$, $v_1 = (1, 1, 0)$, $v_2 = (0, 1, 1)$, $v_3 = (1, 0, 1)$.

▸ **55.** [*HM25*] Continuing exercise 54, let $\epsilon > 0$ be smaller than the distance from v_0 to any other vertex of X, and let X_ϵ be the interior of the closed set $X \cap S_\epsilon(v_0)$, where

$$S_\epsilon(v_0) = \{v \mid \|v - v_0\| \le \epsilon\} = \bigl\{(x, y, z) \mid (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \le \epsilon^2\bigr\}.$$

a) Explain how to decide precisely which of the edges from v_0 to v_1, v_2, and v_3 are concave and which are convex, if told which of the octants are intersected by X_ϵ.

b) Explain how to compute the angles between the pairs of planes that meet at v_0.

pi as random
skeleton
3-cube
concave edges
signed skeleton
convex edge
Szilassi polyhedron
realized
coordinates
octants
octal notation
pi as source
phi as source
gamma as source

57. [*HM25*] Using Cartesian coordinates (x, y, z), state quantitative conditions for the notion of "general position," under which we can be sure that a given 3VP X has a well-defined HC picture after projection to the (x, y)-plane.

58. [*M29*] Derive Table 1 by considering the $2^8 = 256$ different ways that up to eight cubies can be placed into a $2 \times 2 \times 2$ box.

a) Show that exactly 64 of those placements make a 3VP in which the center of the box is a vertex.
b) Furthermore, if that 3VP is in general position, we'll be able to *see* its central vertex in exactly 32 cases.
c) Draw those 32 pictures, and verify that the different possibilities for V, W, and Y junctions are precisely those shown in Table 1.
d) Also explain why Table 1 is correct for T junctions.

59. [*10*] If an HC network has respectively (t, v, w, y) junctions of types T, V, W, and Y, how many variables does the corresponding CSP have? How many constraints?

▸ **60.** [*18*] The line labeling problem has also been modeled as a CSP in quite a different way from (21) and (22): Instead of having one variable for each line, let there be one variable for each junction. The domain of variable j is then either $\{1, 2, 3, 4\}$ or $\{1, 2, 3, 4, 5, 6\}$ or $\{1, 2, 3\}$ or $\{1, 2, 3, 4, 5\}$, depending on whether j has type T, V, W, or Y; and j's value represents the index of the legal labeling in Table 1. There's one constraint for each line between junctions.

a) What is the constraint for line ab of (20) in this scheme?
b) How about the lines np and op?
c) What's the answer to exercise 59, with respect to *this* model?
d) Which model do you think is better?

61. [*15*] Translate the line labeling problem (22) into an XCC problem.

62. [*15*] What standard labeling of Szilassi's polyhedron differs from Fig. 104(b)?

64. [*M20*] If H is the HC network that corresponds to an HC picture, explain how to construct the HC network H^R that corresponds to the mirror image of that picture, when H and H^R both have the same junctions and the same oriented lines. Find a simple relation between the line labeling problems for H and H^R.

▸ **65.** [*M25*] An HC network is called *realizable* if it corresponds to at least one actual HC picture. Furthermore, that HC picture must not have a T junction whose collinear lines both lie on the outer boundary. (Such a T cannot be the image of a 3VP in general position. Notice that the line labeling problem for H is well defined regardless of whether or not H can be physically realized.)

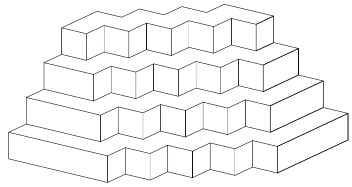
a) What is the smallest unrealizable HC network? *Hint:* It has three junctions.
b) Characterize all realizable HC networks whose junctions all have type V.
c) Find an HC network, consisting entirely of type W junctions, that is unrealizable because it doesn't define a planar graph.
d) Prove that every realizable HC network contains at least three junctions of type V or W. *Hint:* Consider the boundary cycle of any connected component.
e) True or false: If the junction $T(\mathrm{a}, \mathrm{b}, \mathrm{c})$ in a realizable network is changed to either $W(\mathrm{c}, \mathrm{b}, \mathrm{a})$ or $Y(\mathrm{a}, \mathrm{b}, \mathrm{c})$, the resulting network is still realizable.

66. [*M46*] Is there an algorithm to decide whether a given HC network is realizable?

general position
HC picture
projection
HC network
XCC problem
Szilassi
mirror image
reflection of an HC network
realizable
planar graph
boundary cycle
connected component
decision problem

standard labeling
smile of order n
standard
bow tie
biconnected
standard
Lucas number
Fibonacci numbers
twindragon fractal
fractal

67. [*22*] Cover up the boundary of the HC picture



and watch the disconnected interior images as they jump in and out, before your eyes.

a) Show that this picture has only one standard labeling.

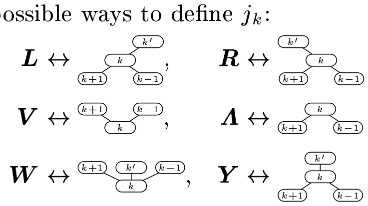
b) In how many ways can the boundary junctions be labeled consistently, without regard to any of the interior junctions?

c) How many labelings are possible altogether, standard or not?

▸ **68.** [*M30*] Let $(j_0 j_1 \ldots j_{q-1})$ be the boundary cycle of a realizable HC network.

a) For $0 \le k < q$, show that there are only six possible ways to define j_k:

- $j_k = T(j_{k+1}, j_{k-1}, j'_k)$, called case $\boldsymbol{L}$;
- $j_k = T(j'_k, j_{k+1}, j_{k-1})$, called case $\boldsymbol{R}$;
- $j_k = V(j_{k+1}, j_{k-1})$, called case $\boldsymbol{V}$;
- $j_k = V(j_{k-1}, j_{k+1})$, called case $\boldsymbol{\Lambda}$;
- $j_k = W(j_{k+1}, j'_k, j_{k-1})$, called case $\boldsymbol{W}$;
- $j_k = Y(j_{k+1}, j'_k, j_{k-1})$, called case $\boldsymbol{Y}$.

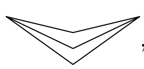
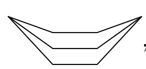
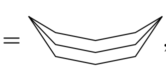


(The subscripts in '$j_{k\pm1}$' are to be understood mod q. The line j_k — j'_k in cases $\boldsymbol{L}$, $\boldsymbol{R}$, $\boldsymbol{W}$, and $\boldsymbol{Y}$ is called an "inner line," although j'_k might lie on the boundary.)

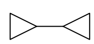
b) What combinations of line labels for $j_{k-1}j_k$, $j_k j_{k+1}$, $j_k j'_k$ can occur in each case?

c) Design an efficient way to test whether any inner line label can be assigned more than one value, when only the q constraints of the boundary cycle are imposed.

69. [*M23*] The "smile of order n" is a realizable HC network S_n with $3n+2$ junctions:

$S_1 =$ , $S_2 =$ , $S_3 =$ , ...

How many line labelings does S_n have? How many of them are standard?

70. [*16*] In how many ways can the "bow tie"  be labeled?

71. [*M22*] Does a biconnected realizable HC network have a unique boundary cycle?

72. [*22*] Construct a realizable HC network that has a unique line labeling, although it doesn't have a *standard* labeling.

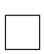
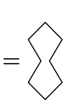
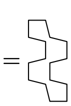
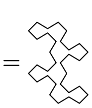
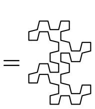
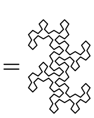
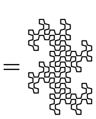
73. [*HM39*] Suppose each junction j_k of a boundary cycle $(j_0 j_1 \ldots j_{q-1})$ is $\boldsymbol{V}$ or $\boldsymbol{\Lambda}$.

a) Let $M_k = A$ if $j_k = \boldsymbol{V}$ and $M_k = B$ if $j_k = \boldsymbol{\Lambda}$, where $A = \binom{1\;1}{1\;0}$ and $B = \binom{\;1\;\;-1}{-1\;\;0}$ are 2×2 matrices. Prove that the number of ways to label the boundary cycle $(j_0 j_1 \ldots j_{q-1})$ is $\operatorname{trace}(M_0 M_1 \ldots M_{q-1}) + L_q$, where L_q is a Lucas number.

b) Show that $2F_q \le \operatorname{trace}(A^p B^{q-p}) + L_q \le 2L_q$ for $0 \le p \le q$. What p gives equality?

c) In fact, the number of labelings is between $2F_q$ and $2L_q$ in all cases.

74. [*HM21*] The *twindragon fractal* (see Fig. 1 in Chapter 4) can be approximated by a sequence of polygonal paths T_n for $n \ge 2$, where T_n has 2^n junctions:

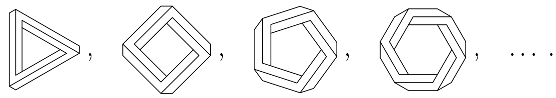
$T_2 =$ , $T_3 =$ , $T_4 =$ , $T_5 =$ , $T_6 =$ , $T_7 =$ , $T_8 =$ , ...

The clockwise path T_n turns left or right at step k and at step $k+2^{n-1}$ according as the Jacobi symbol $\left(\frac{-1}{k}\right)$ is $=-1$ or $+1$, for $1 \le k \le 2^{n-1}$. (See exercise 4.5.4–23.)

In how many ways can T_n be labeled? *Hint:* Use exercise 73.

76. [*20*] Combine a V junction, a W junction, and a Y junction in such a way that the resulting subpicture cannot be labeled. (See (24) and (25).)

▸ **77.** [*M25*] The Penrose triangle, Penrose square, Penrose pentagon, Penrose hexagon, . . . , are

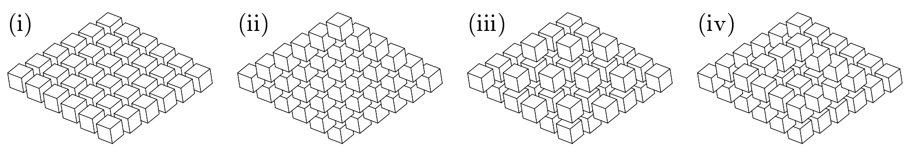


a) What is the HC network for the Penrose n-gon?
b) In how many ways can the Penrose n-gon be labeled consistently?
c) Is the Penrose n-gon weakly realizable for any $n \ge 3$?

78. [*20*] Explain how to obtain (32) as the projection of nine "squashed" cubes.

79. [*M22*] In how many ways can Reutersvärd's (32) be labeled (standard or not)?

80. [*24*] We can extend the idea in (32) to larger arrays of partially overlapping boxes:



(This is essentially a hexagonal grid, because each box can potentially overlap with six neighbors.) How many standard labelings are possible for (i), (ii), (iii), and (iv)?

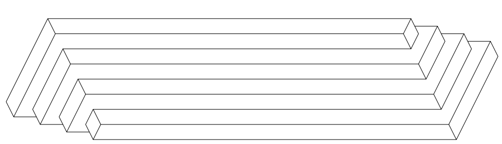
81. [*23*] The 36 boxes in the 6×6 hexagonal arrays of exercise 80 involve 85 pairs (A, B) of adjacent boxes: $30 = 6\cdot 5$ pairs in direction $\nearrow$; $30 = 5\cdot 6$ pairs in direction $\nwarrow$; and $25 = 5\cdot 5$ pairs in direction $\uparrow$. In every case we're allowed to specify either $A < B$ or $A > B$, meaning that A lies behind or in front of B in the image. Example (iv) illustrates the fact that this relation need not be transitive.

Thus those 36 boxes might be depicted in 2^{85} different ways. However, it turns out that the boxes are too close together to allow all possibilities: When boxes A, B, and C are mutually adjacent, we cannot simultaneously specify $A < B$, $B < C$, and $C < A$.

a) In how many ways can those 85 relations be specified, without any such non-transitive triplets? *Hint:* This is a CSP.
b) Generalize to $m \times n$ hexagonal arrays of boxes, for $1 \le m \le n \le 10$.

▸ **83.** [*M30*] Let H be a labeled HC picture, whose junctions have known (x, y) coordinates. Explain how to construct a system of linear equations and linear inequalities that have a solution whenever H is the projection of some 3VP X in general position.

85. [*22*] Is the following HC picture impossible? (It uses the right half of (26), twice.)



86. [*M25*] (L. Kirousis and C. Papadimitriou, 1988.) Prove that it's NP-complete to decide whether or not a realizable HC picture can be labeled.

87. [*HM46*] Is it decidable whether or not a given HC network is weakly realizable?

Jacobi symbol
Penrose n-gon
squashed
Reutersvärd
hexagonal grid
linear equations and linear inequalities
impossible
Kirousis
Papadimitriou
NP-complete
decision problem

90. [*15*] If we change `6` to `7` in Fig. 105(b), we get *another* graceful labeling, since the edge labels $10-6=4$ and $6-3=3$ become $10-7=3$ and $7-3=4$. Show that further graceful labelings can be obtained by changing only the labels of vertices `13` and `14`.

91. [*M21*] True or false: If graph G has k automorphisms, every graceful labeling of G is equivalent to $2k-1$ others, under symmetry and complementation.

▶ **93.** [*21*] To model the graceful labeling problem of Fig. 105 as an XCC problem, we can introduce 18 primary items $\{1,\ldots,18\}$ for the edge labels, 18 primary items $\{\texttt{NH-MA},\ldots,\texttt{GA-SC}\}$ for the edges, 13 secondary items $\{\texttt{NH},\ldots,\texttt{SC}\}$ for the colonies, and 19 secondary items $\{h_0,\ldots,h_{18}\}$ for the holders of vertex labels. These items are to be governed by $18\cdot 19\cdot 18 = 6156$ options, such as

'6 `PA-DE` `PA`:3 `DE`:9 h_3:`PA` h_9:`DE`',

namely one for each edge label d, each edge, and each way to assign labels j and k with $0\le j<k=j+d\le 18$ to the endpoints of that edge. (The example shown covers edge label 6 and edge `PA-DE` when `PA` is labeled 3 and `DE` is labeled 9.) Given those options, Algorithm 7.2.2.1C needs about 90 gigamems to find the 641952 solutions.

a) Modify the model so that only the 160488 essentially different solutions are found.
b) Modify the model so that it solves the puzzle of Fig. 105(d).

94. [*M21*] The arrays `LO`, `FIRST`, `NEXTL`, `NEXTH`, `NAME` in (35) correspond to the labeling in Fig. 105(b). What arrays `LO`$'$, …, `NAME`$'$ correspond to its complement, Fig. 105(c)?

95. [*M20*] (S. Golomb, 1972.) Complete the proof that K_n is ungraceful when $n\ge 5$.

▶ **96.** [*25*] Design a backtrack algorithm to find all the graceful labelings of P_n as in (38).

97. [*26*] The search tree for graceful labelings of P_{10}, analogous to (38), contains 206 nodes, two of which are labeled 1738092 and 1809372. Those two nodes have *identical* subtrees, because they both represent a partial path between 1 and 2 that lacks the elements $\{4,5,6\}$. Modify the algorithm of exercise 96 so that it avoids such redundant computations, by identifying nodes that are obviously equivalent. (Think of ZDDs.)

98. [*M25*] (M. Adamaszek, 2013.) Consider n points that all lie on a straight line L.

a) What's the length of the longest path within L that doesn't hit any point twice?
b) Prove that if $p_1\ldots p_{2m}$ is a graceful permutation of $\{1,\ldots,2m\}$ with $p_{2m}=p_1+m$, then $p_{2k}>m$ for $1\le k\le m$.
c) Conversely, if $p_1\ldots p_{2m}$ is graceful and $p_{2k}>m$ for $1\le k\le m$, then $p_{2m}=p_1+m$.

▶ **99.** [*M30*] Determine all of the essentially different graceful labelings of $K_{1,1,n}$.

100. [*M16*] Prove that exactly one of the $4n!$ equivalent matrices (x_{ij}) that gracefully label a KP graph $K_n \mathbin{\square} P_r$ has $0 \mathbin{—} (m-1)$ and satisfies (40).

101. [*16*] Study Fig. 107. Why doesn't $\boxed{\substack{29\\80}}$ appear in level 3 of that tree?

▶ **102.** [*21*] If $n>5$, one of the branches in the search tree analogous to Fig. 107 will set $x_{12}=m$ and $x_{22}=0$ at level 1, $x_{32}=m-1$ at level 2, $x_{42}=2$ at level 4 (and level 3), and $x_{52}=m-4$ at level 5. What are the immediate descendants of that level-5 node, if (a) $r=2$? (b) $r=3$?

103. [*M25*] Explain why the exhaustive search for graceful labelings of $K_n \mathbin{\square} P_2$, illustrated for $n=3$ in Fig. 107, performs essentially identical calculations for all sufficiently large values of n, never finding a solution.

▶ **104.** [*20*] Draw levels 0, 1, and 2 of the search tree for $K_3 \mathbin{\square} P_3$, analogous to Fig. 107.

graceful labeling
automorphisms
symmetry
complementation
XCC problem
symmetry breaking
puzzle
complement
Golomb
K_n
ZDDs
Adamaszek
longest path
graceful permutation
KP

105. [*46*] Determine the number of graceful labelings of $K_n \mathbin{\square} P_4$ for all n.

▸ **106.** [*20*] Is it possible to prove that $K_3 \mathbin{\square} P_{17}$ is graceful by constructing a 3×17 matrix whose first row contains the first 34 digits of π?

107. [*M24*] Prove that $K_3 \mathbin{\square} P_r$ is graceful for all $r \geq 1$, by constructing an appropriate $3 \times r$ matrix whose top row is $(0, m-2, 4, m-6, 8, \ldots)$.

108. [*46*] Is $K_4 \mathbin{\square} P_r$ graceful for all $r \geq 1$?

109. [*M11*] How many symmetries does a KC graph have?

110. [*M18*] For what $n > 2$ and $r > 2$ does Lemma O prove that $K_n \mathbin{\square} C_r$ isn't graceful?

▸ **111.** [*20*] Does Lemma O tell us anything useful about KP graphs?

112. [*20*] A graceful square: Show that $K_4 \mathbin{\square} K_4$ is graceful(!).

113. [*12*] Is every graph with four edges graceful?

▸ **115.** [*M24*] A "random graceful graph" G_m^π can be based on π using the factorial series

$$\pi = 3 + \sum_{k=1}^{\infty} \frac{a_k}{(k+1)!}, \quad \text{where } 0 \leq a_k \leq k.$$

The vertices are $\{0, \ldots, m\}$; the edges are $0 — m$ and $a_k — a_k + m - k$, for $1 \leq k < m$.
a) Show that these integers a_k are unique, and compute them for $k \leq 20$.
b) How many isolated vertices does G_m^π have, for $m \leq 20$? How many components?
c) Determine the chromatic numbers $\chi(G_1^\pi)$, …, $\chi(G_{20}^\pi)$.

116. [*22*] Among the 16! graceful labelings with 16 edges, how many of them define an n-vertex graph, for each n, after removing isolated vertices? How many are connected?

117. [*22*] Repeat exercise 116, but restrict the counts to *bipartite* graphs.

▸ **118.** [*22*] Explain how to compute all possible graceful labelings of r-regular graphs with m edges, given m and r. What are the smallest such labelings when $2 \leq r \leq 8$?

119. [*22*] Continuing exercise 118, make a complete survey of all graceful labelings of 2-regular graphs with ≤ 16 edges. How many such graphs are graceful?

120. [*32*] Continuing exercise 118, make a complete survey of all graceful labelings of 3-regular graphs (cubic graphs) with ≤ 14 *vertices*. How many such graphs are graceful?

121. [*46*] Is every connected cubic graph graceful?

▸ **123.** [*40*] Fun fact: Exactly 12345 different graphs have at most 8 nonisolated vertices. Study their gracefulness: How many of them are graceful? Which of them are *uniquely* graceful? Which of them are *maximally* graceful — graceful in the most different ways?

▸ **125.** [*28*] A graceful labeling is called *rooted* if every edge has a vertex in common with a longer edge, except edge m itself. For example, the first three graceful permutations in (38) are rooted; but the other three are not, because edge $1 — 3$ doesn't touch any of the longer edges $2 — 5$, $0 — 4$, $0 — 5$.
a) Is the 13-colonies labeling in Fig. 105(b) rooted?
b) How many of the 160488 graceful labelings of that graph are rooted?
c) How many of the 16! labelings in exercise 116 are rooted?
d) Compute the number of rooted graceful labelings of P_n, for $n \leq 16$.

126. [*30*] Find a connected graceful graph that has no *rooted* graceful labeling.

128. [*24*] Find all of the essentially distinct graceful labelings of (41).

π
KC graph
random graceful graph
G_m^π
π
factorial series
isolated vertices
components
chromatic numbers
isolated vertices
bipartite
r-regular graphs
2-regular graphs
3-regular graphs
cubic graphs
trivalent graphs
Fun fact
graphs, small
uniquely graceful
rooted
graceful permutations
13-colonies
P_n

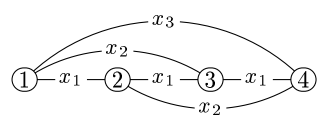
Anick
free trees
superexponential
components
Elkies
complete graph
spanning trees
complement
determinant
binomial tree
n-cube
Gray code
binary recurrence

129. [*M30*] (D. Anick, 2016.) To backtrack through all graceful labelings of free trees on the vertices $\{0,\ldots,m\}$, we can successively choose `LO[`k`]` for $k=m,\ m-1,\ \ldots,\ 1$, in such a way that the edge `LO[`k`]` — `LO[`k`]` $+\,k$ doesn't produce a cycle in the graph-so-far. We shall prove that the number of choices is superexponential, by showing that there always are at least t_k choices for `LO[`k`]`, where t_k is suitably large.

At the moment we choose `LO[`k`]`, the current graph has exactly $k+1$ connected components (possibly singletons). Let's write $x \asymp y$ if vertices x and y belong to the same component; also $x \equiv y$ if $x \bmod k = y \bmod k$. Call r a "residue" if $0 \le r < k$, and call it "bad" if $x \equiv y \equiv r$ implies $x \asymp y$. Say also that x is bad if $x \bmod k$ is bad; a component is bad if all its vertices are bad. Furthermore, "good" means "not bad."

a) Show that there's always at least one good residue.
b) If there are g good residues, then $t_k \ge g$.
c) If there are G good components, then $t_k \ge G-g$.
d) If $k < m/2$, a bad component contains at least two different bad residues.
e) Hence we may let $t_k = \lfloor (k+4)/3 \rfloor$ when $k < m/2$.
f) When $k \ge m/2$ we may let $t_k = 2 + \lfloor (m-k)/2 \rfloor$. *Hint:* Prove that if $x \asymp x+k$, there are vertices $y < x$ and $z > x+k$ such that $y \asymp x$ and $z \asymp x+k$.

▶ **130.** [*HM25*] (N. Elkies, 2002.) In the complete graph on vertices $\{1,\ldots,n\}$, assign the weight x_d to edge k — $(k{+}d)$, for $1 \le k \le n-d$ and $1 \le d < n$, as illustrated here for $n=4$. This graph has n^{n-2} spanning trees in general, by exercise 2.3.4.4–22; and we can form the sum $S(x_1,\ldots,x_{n-1})$ of the products of all edge weights, over each of those trees. For example, when $n=4$ we have



$$S(x_1,x_2,x_3) \;=\; x_1^3 + 4x_1^2x_2 + 3x_1x_2^2 + 3x_1^2x_3 + 4x_1x_2x_3 + x_2^2x_3,$$

because there's one spanning tree that uses all three x_1's, and four that use two x_1's and an x_2, etc. Notice that $[x_1x_2x_3]\,S(x_1,x_2,x_3)=4$ is twice the total number of graceful labelings of 4-vertex trees, since a labeling and its complement are both counted.

a) Express $S(x_1,\ldots,x_{n-1})$ as a determinant. *Hint:* See exercise 2.3.4.2–20.
b) Explain how to compute $\tau(n-1) = [x_1\ldots x_{n-1}]\,S(x_1,\ldots,x_{n-1})$ in $O(2^n n^3)$ steps.

131. [*HM46*] Determine the asymptotic value of the function $\tau(n)$ in exercise 130.

▶ **132.** [*21*] The *binomial tree* T_n has 2^n nodes $\{0,1,\ldots,2^n-1\}$, rooted at 0, where the parent of node $x \ne 0$ is node $x \,\&\, (x-1)$. (See 7.2.1.3–(21).) If $x=(x_{n-1}\ldots x_1x_0)_2$, let $l(x)=(l_{n-1}\ldots l_1l_0)_2$, where $l_k = x_0 \oplus \cdots \oplus x_k$. Show that these labels make T_n graceful.

133. [*24*] Continuing exercise 132, determine the exact number of essentially different graceful labelings of T_3 and T_4. Also estimate that number for T_5 and T_6.

136. [*M23*] Prove that the n-cube is graceful by means of the following labeling based on Gray code and an auxiliary sequence $0=a_0<a_1<a_2<\cdots$: Let $g(2k)$ and $g(2k+1)$ be labeled a_k and $m-k-a_k$, respectively, where $m=n2^{n-1}$. For example,

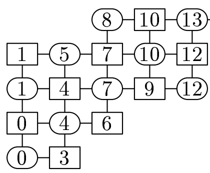
$$\begin{array}{rcllllllll} v &=& 000 & 001 & 011 & 010 & 110 & 111 & 101 & 100\\ l(v) &=& a_0 & 12{-}a_0 & a_1 & 11{-}a_1 & a_2 & 10{-}a_2 & a_3 & 9{-}a_3 \end{array}$$

when $n=3$. (See 7.2.1.1–(4).) Assume that $a_{2^n+r} = a_{2^n} + a_r$ for $0 \le r < 2^n$.

a) Let V_j be the vertices of the form $j\alpha$, and let L_j be the labels of the edges in $G\,|\,V_j$, for $0 \le j \le 1$. (For example, when $n=3$ we have $V_0=\{000,001,010,011\}$ and $L_0=\{12-2a_0,\,12-a_0-a_1,\,11-2a_1,\,11-a_1-a_0\}$.) Express L_1 in terms of L_0.
b) What values of $a_1,\ a_2,\ a_4,\ a_8,\ \ldots$ make the labeling graceful?

▸ **137.** [*M25*] A *parallomino graph* (see exercise 7.2.2.1–303) has vertices (x, y) for integers $0 \le x \le r$ and $s_x \le y \le t_x$, where $0 = s_0 \le s_1 \le \cdots \le s_r$, $t_0 \le t_1 \le \cdots \le t_r$, and $s_{k+1} \le t_k$ for $0 \le k < r$; edges go from (x, y) to $(x+1, y)$ and $(x, y+1)$ when possible.

For example, the parallomino graph with $r = 6$, $(s_0, t_0) = (s_1, t_1) = (0, 3)$, $(s_2, t_2) = (1, 4)$, $(s_3, t_3) = (s_4, t_4) = (2, 4)$, and $(s_5, t_5) = (s_6, t_6) = (4, 4)$ can be decorated with labels in two closely related ways:

(i) 

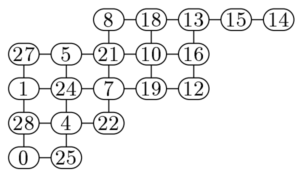
(ii) 

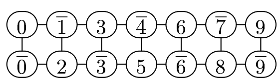
Illustration (ii) is in fact a remarkable *graceful* labeling, where the edges whose labels are 1, 2, …, 28 appear in strict order, from right to left and top to bottom!

a) How many vertices and edges does a parallomino graph have, in general?
b) Decipher the rule that connects illustration (i) with illustration (ii).
c) Reverse-engineer the rule by which illustration (i) was labeled.
d) Can *every* parallomino be gracefully labeled, using these rules?

▸ **138.** [*M25*] Let $\bar{l}$ denote $m - l$. A graph is *α-graceful* if its edges can be written

$$u_0 \,\text{---}\, \overline{v_0}, u_1 \,\text{---}\, \overline{v_1}, \ldots, u_{m-1} \,\text{---}\, \overline{v_{m-1}},$$
$$\text{where } u_k + v_k = k,\ 0 \le u_k < l, \text{ and } 0 \le v_k < m+1-l, \text{ for some } l.$$

Here u_k and $\overline{v_k}$ are labels of vertices in the graph. For example, the labels



show that $K_2 \,\square\, P_7$ is α-graceful; and a similar construction works for $K_2 \,\square\, P_r$ in general.

a) Prove that an α-graceful graph is graceful and bipartite.
b) For which n is the cycle C_n α-graceful?
c) Prove that every α-graceful labeling has an "edge complement" in which edge k becomes edge $m + 1 - k$, for $1 \le k \le m$.
d) Find a tree with seven nodes that's not α-graceful.

139. [*23*] A bipartite graph with parts U and V has an *ordered* graceful labeling if it has a graceful labeling such that $l(u) < l(v)$ for every edge $u \,\text{---}\, v$ with $u \in U$, $v \in V$.

a) Show that every α-graceful graph has an ordered graceful labeling.
b) Show that the non-α-graceful tree of exercise 138(d) *also* has such a labeling.
c) Let G have m edges and an ordered graceful labeling. Prove that m copies of G can be perfectly packed into the complete bipartite graph $K_{m,m}$.
d) A bipartite graph G with m edges $u \,\text{---}\, v$ between parts U and V leads naturally to a bipartite graph $G^{(t)}$ with tm edges $u \,\text{---}\, v_i$ between parts U and $V_1 \cup \cdots \cup V_t$. If G has an ordered graceful labeling, show that $G^{(t)}$ does too.

140. [*M21*] Continuing exercises 138 and 139, how many (a) α-graceful labelings (b) ordered graceful labelings have m edges? (Compare with Theorem S.)

142. [*M20*] The direct product of bipartite graphs always has at least two components.

a) Prove this, by determining the components of $K_{a,b} \otimes K_{c,d}$.
b) When G and H are bipartite with parts (U, V) and (X, Y), let $(G \otimes H)'$ and $(G \otimes H)''$ be $G \otimes H$ restricted respectively to parts $(U \times X, V \times Y)$ and $(U \times Y, V \times X)$.

parallomino graph
grid
skeleton
α-graceful
bipartite
cycle C_n
complement
ordered graceful labeling
near α-labeling, see ordered graceful labeling
direct product
tensor product, see direct product

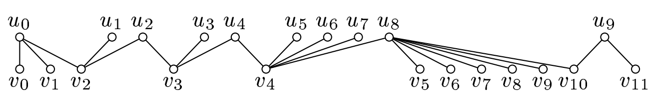
Describe (i) $(P_{2m} \otimes P_{2n})'$ and $(P_{2m} \otimes P_{2n})''$; (ii) $(P_{2m} \otimes P_{2n+1})'$ and $(P_{2m} \otimes P_{2n+1})''$; (iii) $(P_{2m+1} \otimes P_{2n+1})'$ and $(P_{2m+1} \otimes P_{2n+1})''$; (iv) $(C_{2m} \otimes C_{2n})'$ and $(C_{2m} \otimes C_{2n})''$; (v) $(Q_m \otimes Q_n)'$ and $(Q_m \otimes Q_n)''$, where Q_n is the n-cube.

c) If G and H each have an ordered graceful labeling, prove that $(G \otimes H)'$ and $(G \otimes H)''$ do too.

▸ **145.** [*M28*] (*Caterpillar nets.*) A "caterpillar" is a graph with at least two vertices that becomes a path (or empty) when you remove all of its vertices of degree 1. More precisely, an (s,t)-*caterpillar* is a bipartite graph with vertices $\{u_0, \ldots, u_s; v_0, \ldots, v_t\}$ and edges defined by a binary vector $e = e_1 \ldots e_{s+t}$ that has s 0s and t 1s:

$$u_{s_i} \mathbin{\frac{\quad}{}} v_{t_i} \;\text{ for } 0 \le i \le s+t, \;\text{ where } s_i = \bar e_1 + \cdots + \bar e_i \text{ and } t_i = e_1 + \cdots + e_i.$$

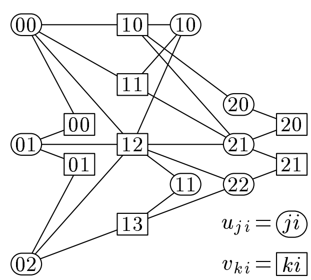
For example, here's the $(9,11)$-caterpillar whose edge vector is 11001001000011111101:



a) Draw the eight (s,t)-caterpillars for which $s+t=3$.

b) Prove that every (s,t)-caterpillar is α-graceful.

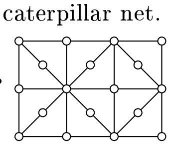
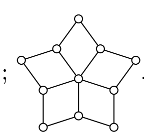
c) Given an (s,t)-caterpillar, a "caterpillar net" is a graph obtained when we replace the vertices u_j and v_k by disjoint sets of vertices $U_j = \{u_{j0}, \ldots, u_{jp_j}\}$ and $V_k = \{v_{k0}, \ldots, v_{kq_k}\}$, for $0 \le j \le s$ and $0 \le k \le t$. The edges are (p_{s_i}, q_{t_i})-caterpillars between U_{s_i} and V_{t_i}, for $0 \le i \le s+t$. For example, a caterpillar net with $e = 1001$, $p_0 = p_2 = 2$, $p_1 = q_0 = q_2 = 1$, and $q_1 = 3$ is illustrated here. How many edges does a caterpillar net have?



d) Prove that every caterpillar net is α-graceful.

e) Prove that the complete bipartite graph $K_{n,r}$ is a caterpillar net.

f) Prove that the grid $P_n \,\square\, P_r$ is a caterpillar net.

g) Are either of the following graphs caterpillar nets?  ;  .

146. [*23*] The grid graph $P_2 \square P_6$ is the "skeleton" of a pentomino (showing the outlines of its five cells). Prove that the skeletons of all twelve pentominoes are α-graceful.

148. [*HM36*] Exercise 145(e) proved that $K_{n,r}$ is α-graceful. Let $A(n,r)$ be the exact number of different α-labelings that $K_{n,r}$ has, times 2 if $n = r > 1$. (We know that $K_{2,2} = C_4$ has a unique graceful labeling; but $A(2,2) = 2$ because the edges can be written either as $0 \mathbin{\frac{\quad}{}} \bar 0$, $1 \mathbin{\frac{\quad}{}} \bar 0$, $0 \mathbin{\frac{\quad}{}} \bar 2$, $1 \mathbin{\frac{\quad}{}} \bar 2$ or $0 \mathbin{\frac{\quad}{}} \bar 0$, $0 \mathbin{\frac{\quad}{}} \bar 1$, $2 \mathbin{\frac{\quad}{}} \bar 0$, $2 \mathbin{\frac{\quad}{}} \bar 1$ in the notation of exercise 138.)

a) Prove that $A(n,r)$ is the number of ways to write the polynomial $F_m(x) = 1 + x + \cdots + x^{m-1}$ as a product $G(x)H(x)$, where $m = nr$, $G(1) = n$, $H(1) = r$, and all coefficients of G and H are either 0 or 1. (For example, $A(2,2) = 2$ because $F_4(x) = (1+x)(1+x^2) = (1+x^2)(1+x)$; $A(6,2) = 4$ because $F_{12}(x) = (1+x+x^2+x^3+x^4+x^5)(1+x^6) = (1+x+x^2+x^6+x^7+x^8)(1+x^3) = (1+x+x^4+x^5+x^8+x^9)(1+x^2) = (1+x^2+x^4+x^6+x^8+x^{10})(1+x)$.)

b) Prove that if $F_m(x) = G(x)H(x)$ and the coefficients of G and H are real, both G and H are *palindromials* (palindromic polynomials): Their coefficients are the same when read in either direction. (That is, $G(x) = x^{\deg(G)}G(1/x)$.)

n-cube
grid
torus
Path P_n
cycle C_n
ordered graceful labeling
Caterpillar nets
(s,t)-caterpillar
bipartite graph
pi, as random example
complete bipartite graph
$K_{n,r}$
grid
grid graph
skeleton
pentominoes
$K_{n,r}$
complete bigraph
polynomial
real
palindromials

c) Furthermore if all coefficients of G and H are between 0 and 1, they're all 0 or 1.
d) Furthermore, if $n > 1$ and $r > 1$, either $G(x)$ or $H(x)$ has the special form $F_k(x)T(x)$, where $1 < k < m$ and all coefficients of T are 0 or 1.
e) Furthermore, $G(x) = F_k(x)T(x)$ implies that H and T are polynomials in x^k.
f) Conclude that $A(p, q) = 2$ whenever p and q are prime. What is $A(p^e, q^f)$?
g) What is $A(p_1p_2, q_1q_2)$, when p_1, p_2, q_1, q_2 are prime and $p_1 \neq p_2$, $q_1 \neq q_2$?
h) Use trace theory (Theorem 7.2.2.2F) to prove that $A(p_1^{e_1} \ldots p_s^{e_s}, q_1^{f_1} \ldots q_t^{f_t}) = [p_1^{e_1} \ldots p_s^{e_s} q_1^{f_1} \ldots q_t^{f_t}]\, 1/\bigl((1-p_1)\ldots(1-p_s) + (1-q_1)\ldots(1-q_t) - 1\bigr)$.
i) In particular, $A(p^e, q_1^{f_1} \ldots q_t^{f_t}) = \binom{e+f_1}{e} \ldots \binom{e+f_t}{e}$.

149. [*M22*] Show that $K_{n,r}$ sometimes has graceful labelings that are *not* α-graceful:
a) If $r = 2$ and $2n+1 = pq$ with $p, q > 1$, use labels $\{2n, 2n-p\}$ in the second part, with labels $\bigcup_{k=0}^{\lfloor q/2\rfloor -1} [2kp \mathinner{.\,.} 2kp+p)$ and $\lfloor p/2\rfloor$ others in the first part.
b) If $n = 3k+1$, use labels $[0 \mathinner{.\,.} 2k] \cup [nr-k \mathinner{.\,.} nr-1]$ in the first part.

150. [*M46*] Does $K_{n,r}$ have graceful labelings besides those of exercises 148 and 149?

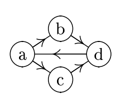
155. [*20*] Given a simple digraph D without loops, construct an XCC problem whose solutions are the graceful labelings of D. *Hint:* Modify the construction in exercise 93.

156. [*13*] For what a and b does $x \mapsto (ax+b) \bmod 13$ take Fig. 109(e) into Fig. 109(f)?

157. [*22*] Find all of the essentially different graceful labelings of Fig. 109(a).

▸ **160.** [*M28*] Two graceful labelings l and l' of a digraph D with $q-1$ arcs are called *affinely equivalent* if $l'(v) = \bigl(a(l(v)-b)\bigr) \bmod q$ for all vertices v, where a and b are integers with $a \perp q$. (This notion matches transformations (i) and (ii) discussed in the text.)
a) Let v and w be distinct vertices of D. Show that every graceful labeling l is affinely equivalent to a graceful labeling l' for which $l'(v) = 0$ and $l'(w) = d$ for some $d\backslash q$.
b) Exactly how many such labelings l' exist, given d and q?
c) Now explain how to take the labelings found in (a) and find all of the "essentially different" ones, by taking account of D's symmetries and antisymmetries.

161. [*19*] What are the essentially different ways to label these digraphs gracefully?

(a)  ; (b) $a \to b \leftarrow c \leftarrow d \leftarrow e \to f$.

164. [*16*] Design an algorithm to create the `FIRST` and `NEXT` arrays of a graceful digraph, given its `LO` array.

▸ **165.** [*M25*] Let l be a graceful labeling of D, and let `LO`, `FIRST`, `NEXT`, and `NAME` be the corresponding representation as in (44). A labeling l' equivalent to l will then correspond to certain arrays `LO`$'$, `FIRST`$'$, `NEXT`$'$, and `NAME`$'$. (Compare with exercise 94.)
a) Compute them when $l'(v) = \bigl(a(l(v)-b)\bigr) \bmod q$, given a and b with $a \perp q$.
b) Compute them when $l'(v) = l(v\alpha)$, given an automorphism α of D.
c) Compute them when $l'(v) = l(v\alpha)$, given an antiautomorphism α of D.

▸ **168.** [*M24*] The digraph D in Fig. 109 doesn't fully represent set inclusion in a 3-element universe because it isn't transitive. Let D^* be the digraph obtained when the arcs $000 \longrightarrow 011$, $000 \longrightarrow 101$, $000 \longrightarrow 110$, $000 \longrightarrow 111$, $001 \longrightarrow 111$, $010 \longrightarrow 111$, $100 \longrightarrow 111$ are added to D. What are its classes of equivalent graceful labelings?

169. [*22*] When the digraph in Fig. 109 is extended to a 4-element universe, it has 16 vertices and 32 arcs. Is it still graceful?

trace theory
digraph
XCC problem
essentially different graceful labelings
affinely equivalent
essentially different
symmetries
antisymmetries
`FIRST`
`NEXT`
`LO`
digraph representation
automorphism
antiautomorphism
set inclusion
transitive
Boolean lattice
Boolean lattice
set inclusion

▶ **172.** [*HM35*] Let $\mathcal{D}_m$ be the set of m-tuples $x = x_1 \dots x_m$ with $0 \le x_l \le m$ for $1 \le l \le m$. If $x \in \mathcal{D}_m$, the digraph $aD(x)+b$ has $m+1$ vertices $\{0, \dots, m\}$ and m arcs, $(ax_l+b) \bmod q \longrightarrow (ax_l+al+b) \bmod q$, where $q = m+1$. Furthermore, say that $x \equiv x'$ in $\mathcal{D}_m$ if $aD(x)+b$ equals $D(x')$ or its converse $D(x')^T$, for some a and b with $a \perp q$.

a) What are the equivalence classes of $\mathcal{D}_2$ and $\mathcal{D}_3$?

b) What's a good way to visit each equivalence class of $\mathcal{D}_m$, when m isn't too large?

c) What's a good way to count the number of equivalence classes, when m is larger?

175. [*25*] Let $l_k = l(v_k)$ be the kth vertex label in a path or cycle $v_0 \longrightarrow \cdots \longrightarrow v_m$.

a) Show that $l_{2k} = r-1-k$ and $l_{2k+1} = r+k$ gracefully label the oriented path $P_{2r}^{\rightarrow}$.

b) Find a somewhat similar pattern of graceful labels for $C_{2r}^{\rightarrow}$. *Hint:* Use vertex labels $< r$ and arc labels $\equiv r-1$ (modulo 2) in the first half of the cycle.

176. [*20*] (G. S. Bloom and D. F. Hsu.) If D is a graceful digraph with m arcs and $m+1$ vertices, prove that $D \longrightarrow \overline{K_n}$ is also graceful. (It has $mn+m+n$ arcs, $m+n+1$ vertices.)

177. [*22*] Find an ungraceful digraph D with 2 arcs and 3 vertices such that $D \longrightarrow \overline{K_n}$ is graceful for all $n > 0$.

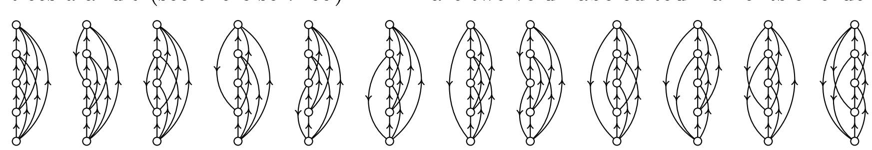
178. [*16*] Is the oriented complete bipartite graph $K_{m,n}^{\rightarrow} = \overline{K_m} \longrightarrow \overline{K_n}$ graceful?

180. [*41*] Investigate all of the graceful digraphs that have at most 6 nonisolated vertices. (Compare with exercise 123; the number rises from 12345 to 1540943.)

182. [*M20*] (C. Delorme.) Let D be a digraph with m arcs for which the total degree $d^+(v)+d^-(v)$ is *even* at every vertex v. Prove that D cannot be graceful if $m \bmod 4 = 1$.

183. [*20*] (G. S. Bloom and D. F. Hsu.) Show that the m edges of a digraceful graph can always be oriented in at least $2^{\lfloor m/2 \rfloor}$ graceful ways.

▶ **185.** [*M30*] A *tournament* is a digraph in which either $u \longrightarrow v$ or $v \longrightarrow u$ for every pair of vertices u and v (see exercise 7–59). There are twelve unlabeled tournaments of order 5:



A B C D E F G H I J K L

a) What are the *converses* of A, B, ..., L? (For example, $\mathrm{A}^T = \mathrm{A}$.)

b) How many essentially distinct graceful labelings does each of them have?

c) What are the graceful tournaments of orders 3 and 4?

d) A *cyclic (v,k,λ)-difference set* is a set $\{a_1, \dots, a_k\} \subseteq \{0, 1, \dots, v-1\}$ such that the $k(k-1)$ differences $(a_j - a_k) \bmod v$ for $j \ne k$ contain each nonzero residue exactly λ times. For example, $\{0,1,3\}$ is a cyclic $(4,3,2)$-difference set because

$$0 \ominus 1 = 3,\ 0 \ominus 3 = 1,\ 1 \ominus 0 = 1,\ 1 \ominus 3 = 2,\ 3 \ominus 0 = 3,\ 3 \ominus 1 = 2,$$

writing '$x \ominus y$' for $(x-y) \bmod v$. Prove that there exists a graceful n-vertex tournament if and only if there exists a cyclic $\bigl(\binom{n}{2}+1, n, 2\bigr)$-difference set.

e) Show that $\{1, 7, 7^2 \bmod 37, \dots, 7^8 \bmod 37\}$ is a cyclic $(37, 9, 2)$-difference set.

187. [*46*] An undirected graph with m edges can be converted to a directed graph in 3^m ways, because each edge $u \,\text{—}\, v$ can become $u \longrightarrow v$ or $u \longleftarrow v$ or both. Is *every* graph "weakly digraceful," in the sense that at least one of those 3^m possibilities is graceful?

190. [*20*] (M. Buratti and A. Del Fra.) Show that (47) gracefully labels $C_n^{\leftrightarrow}$.

converse
affine equivalence
oriented path
Bloom
Hsu
oriented complete bipartite graph
graphs, small
digraphs, small
Delorme
parity
$d^+(v)$ (out-degree)
Bloom
Hsu
digraceful graph
tournament
converses
essentially distinct
cyclic (v,k,λ)-difference set
weakly digraceful
Buratti
Del Fra

▸ **191.** [*23*] (R. Montgomery, A. Pokrovskiy, and B. Sudakov.) Prove that every tree T with m edges and a vertex v adjacent to at least $2m/3$ leaves is rainbow graceful.

192. [*24*] Find rainbow graceful labelings of (i) $\overline{K_1 \oplus 3K_2}$; (ii) $\overline{4K_1 \oplus C_3}$; (iii) $\overline{2C_4}$.

193. [*30*] Which of the 12345 graphs of exercise 123 are *rainbow* graceful?

194. [*23*] Is every digraceful graph also rainbow graceful?

196. [*HM20*] A *projective plane of order n* has n^2+n+1 points and n^2+n+1 lines, where every line contains exactly $n+1$ points and every point belongs to exactly $n+1$ lines. Furthermore, every two points belong to exactly one line, and every two lines intersect in exactly one point. The following construction defines such a plane whenever F is a finite field of n elements (see exercise 4.6.2–16): Each point is a nonzero triple (a_1, a_2, a_3), and each line is a nonzero triple $[b_1, b_2, b_3]$, where the a's and b's belong to F. Two triples are considered equal if one is a multiple of the other; for example, $(a_1, a_2, a_3) = (2a_1, 2a_2, 2a_3)$ in the field of three elements. Point (a_1, a_2, a_3) lies on line $[b_1, b_2, b_3]$ if and only if $a_1b_1 + a_2b_2 + a_3b_3 = 0$ in F.

a) Explain why this construction gives n^2+n+1 points and n^2+n+1 lines.
b) Which points belong to the line $[1, 0, 2]$ when $n = 3$?
c) Why do two lines intersect in a unique point?

▸ **197.** [*HM27*] (J. Singer, 1938.) Suppose K_{n+1} has graceful rainbow labels $\{l_0, \ldots, l_n\}$.

a) Show that they're a cyclic $(n^2+n+1, n+1, 1)$-difference set (see exercise 185(d)).
b) If $n = p$ is prime, let $f(x) = x^3 - c_1x^2 - c_2x - c_3$ be a primitive polynomial modulo p for the field F of p^3 elements (see 3.2.2–(9)). Consequently the nonzero elements of F are $\{1, \pi, \pi^2, \ldots, \pi^{p^3-2}\}$, where π is a root of f in F. What are the other two roots of f? *Hint:* $(x+y)^p \equiv x^p + y^p$ (modulo p).
c) Continuing (b), find a transformation $(a_1, a_2, a_3)\alpha = (a'_1, a'_2, a'_3)$ of triples with the property that $\pi^k = a_1\pi^2 + a_2\pi + a_3$ implies $\pi^{k+1} = a'_1\pi^2 + a'_2\pi + a'_3$.
d) Find a transformation $[b_1, b_2, b_3]\alpha = [b'_1, b'_2, b'_3]$ of triples, to go with the transformation in (c), with the property that $a_1b_1 + a_2b_2 + a_3b_3 = a'_1b'_1 + a'_2b'_2 + a'_3b'_3$.
e) As a consequence of (c), there are triples (a_{k1}, a_{k2}, a_{k3}) of integers mod p for which we have $\pi^k = a_{k1}\pi^2 + a_{k2}\pi + a_{k3}$, for $0 \le k < p^3 - 1$. List those triples in the special case when $p = 5$ and $f(x) = x^3 - 4x^2 - 3$. (You can stop at $k = 31$.)
f) Construct a projective plane of order p as in exercise 196, and show that we may take the points to be (a_{k1}, a_{k2}, a_{k3}) for $0 \le k < p^2 + p + 1$. Furthermore, $L = \{k \mid a_{k1} = 0 \text{ and } 0 \le k < p^2 + p + 1\}$ is a set of graceful rainbow labels for K_{p+1}.
g) Extend the ideas of (b)–(f) to the case when $n = p^e$ is an arbitrary power of the prime p, and work out the details when $n = 8$.

199. [*HM33*] Let $\mathcal{R}_m$ be the set of m-tuples $x = x_1 \ldots x_m$ with $0 \le x_l \le 2m$ for $1 \le l \le m$. If $x \in \mathcal{R}_m$, the graph $aG(x) + b$ has $2m+1$ vertices $\{0, \ldots, 2m\}$ and m edges, $(ax_l + b) \bmod q \mathrel{—\!—} (ax_l + al + b) \bmod q$, where $q = 2m+1$. Furthermore, say that $x \equiv x'$ in $\mathcal{R}_m$ if $aG(x) + b$ equals $G(x')$, for some a and b with $a \perp q$.

a) What are the equivalence classes of $\mathcal{R}_2$ and $\mathcal{R}_3$? (Compare with exercise 172.)
b) What's a good way to visit each equivalence class of $\mathcal{R}_m$, when m isn't too large?
c) What's a good way to count the number of equivalence classes, when m is larger?

200. [*46*] Is every *forest* rainbow graceful?

203. [*15*] True or false: A *subgraph* of H is any graph that we obtain from H by removing zero or more edges, then removing zero or more isolated vertices. An *induced subgraph* of H is any graph that we obtain from H by removing zero or more vertices, then removing every edge that touched at least one of those vertices.

Montgomery
Pokrovskiy
Sudakov
rainbow graceful labelings
projective plane
uniform hypergraph
finite field
Singer
primitive polynomial modulo p
projective plane
affine equivalence
subgraph
induced subgraph

204. [*16*] Find, by hand, an induced C_7 of common English words, including `chord`.

205. [*17*] Is `cords` — `colds` — `colts` — `costs` — `casts` — `carts` — `cards` — `cords` an isometric embedding of C_7 into `WORDS`(5757)?

▶ **207.** [*M21*] A *Hamming graph* is a graph of the form $K_{n_1} \mathbin{\square} K_{n_2} \mathbin{\square} \cdots \mathbin{\square} K_{n_r}$. Thus it has $n_1 \ldots n_r$ vertices $x_1x_2 \ldots x_r$, where $0 \le x_k < n_k$ for $1 \le k \le r$; and we have $x_1x_2 \ldots x_r \mathrel{—} y_1y_2 \ldots y_r$ if and only if $x_k \ne y_k$ for exactly one index k.

a) How many edges does $K_{n_1} \mathbin{\square} K_{n_2} \mathbin{\square} \cdots \mathbin{\square} K_{n_r}$ have?

b) How many automorphisms does $K_{n_1} \mathbin{\square} K_{n_2} \mathbin{\square} \cdots \mathbin{\square} K_{n_r}$ have?

c) Compute the distance between 141421 and 271828 in a Hamming graph.

d) If a clique G is embedded in $K_{n_1} \mathbin{\square} K_{n_2} \mathbin{\square} \cdots \mathbin{\square} K_{n_r}$, prove that its image is constant in all but one of the constituents K_{n_k}.

e) What 4-vertex graph G can't be strictly embedded in a Hamming graph?

f) Prove that the five-cycle C_5 can't be strictly embedded into a Hamming graph.

208. [*27*] Exactly how many induced seven-cycles are present in `WORDS`(5757)? How many of them are isometrically embedded?

209. [*22*] A strict embedding into a Hamming graph is called a *Hamming embedding.* More precisely, if G is a graph with vertices $\{v_0, v_1, \ldots, v_{n-1}\}$, a Hamming embedding of G is a function $f(v_i) = x_{i1} \ldots x_{ir}$ with the property that, for $0 \le i < j < n$, we have $x_{i1} \ldots x_{ir} \mathrel{—} x_{j1} \ldots x_{jr}$ in a Hamming graph if and only if $v_i \mathrel{—} v_j$ in G.

a) Assume that G is connected, and that each vertex v_i for $i > 0$ has a "parent vertex" $v_{i'}$ with $i' < i$ and $v_{i'} \mathrel{—} v_i$. Show that every Hamming embedding of G can be "normalized" so that (i) $x_{0k}x_{1k} \ldots x_{(n-1)k}$ is a restricted growth string, as defined in 7.2.1.5–(4), for $1 \le k \le r$; and (ii) $x_{i(k+1)} > 0$ for $i > 0$ implies that $x_{jk} > 0$ for some $j < i$. (Condition (ii) means that we don't "invade" coordinate $k+1$ until coordinate k has been used. In particular, a normalized embedding always has $x_{01}x_{02} \ldots x_{0r} = 00 \ldots 0$ and $x_{11}x_{12} \ldots x_{1r} = 10 \ldots 0$.)

b) Design an algorithm that visits every normalized Hamming embedding of G.

210. [*18*] A graph G is called *minimal non-Hamming* (MNH) when its induced subgraphs G' are Hamming embeddable if and only if $G' \ne G$.

a) Is G Hamming embeddable if and only if it has no induced MNH subgraph?

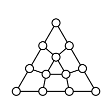
b) Prove that an MNH subgraph is connected.

c) True or false: If G is connected and not Hamming embeddable and not MNH, one of its subgraphs $G \setminus v$ is connected and not Hamming embeddable.

▶ **211.** [*24*] Find all MNH graphs that have at most nine vertices.

▶ **212.** [*25*] (P. M. Winkler, 1984.) If graph G satisfies the conditions of exercise 209(a), prove that it has at most one normalized *isometric* embedding into a Hamming graph. Also design a polynomial-time algorithm that discovers the embedding, if it exists.

213. [*M25*] (P. M. Winkler, 1984.) Let $(u \mathrel{—} v) \bowtie (u' \mathrel{—} v')$ be the relation $d(u,u') - d(u,v') \ne d(v,u') - d(v,v')$, when $u \mathrel{—} v$ and $u' \mathrel{—} v'$ are edges of a graph and $d(u,v)$ denotes shortest distance in that graph.



a) Determine the $\bowtie$ relation between the 18 edges of the graph shown.

b) True or false: In a complete graph, $(u \mathrel{—} v) \bowtie (u' \mathrel{—} v') \iff \{u,v\} \cap \{u',v'\} \ne \emptyset$.

c) A *ternary Hamming graph* is a graph of the form $K_3 \mathbin{\square} \cdots \mathbin{\square} K_3$, "a Cartesian product of triangles." If G can be isometrically embedded in a ternary Hamming graph, prove that the $\bowtie$ relation in G is *transitive* (so it's an equivalence relation).

d) Conversely, if $\bowtie$ is transitive in G, there's an isometric ternary embedding of G.

isometric embedding
`WORDS`(5757)
Hamming graph
Cartesian product
automorphisms
isometrically embedded
strict embedding
Hamming embedding
parent vertex
restricted growth string
minimal non-Hamming
induced subgraphs
Winkler
isometric
Winkler
relation
ternary Hamming graph
Cartesian product
transitive
equivalence relation

214. [*24*] Find the smallest graph that (i) can be embedded as an induced subgraph, but not isometrically; (ii) can be embedded isometrically, but has an induced subgraph that cannot. How many graphs of n vertices, for $1 \le n \le 9$, can be isometrically embedded in a Hamming graph? (See exercise 211.)

▸ **216.** [*M37*] (*Subcube labels.*) A string of 0s, 1s, and $*$s conventionally represents a subcube of a cube, where each $*$ is a "wild card" that stands for either 0 or 1. For example, $0{*}1{*}$ represents $\{0010, 0011, 0110, 0111\}$, which is a subcube of $****$.

It's easy to work with subcubes inside a computer, using the asterisks-and-bits representation of exercise 7.1.1–30. For example, $0{*}1{*}$ is represented by the two bitstrings $a = 0101$ and $b = 0010$, showing respectively the $*$s and the 1s.

The vertices of a connected graph can always be labeled with subcubes in such a way that *the distance between any two vertices is exactly equal to the distance between their labels*(!). One such labeling of the five-cycle $0 — 1 — 2 — 3 — 4 — 0$ is

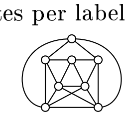
$$l(0) = 0000,\ \ l(1) = 1000,\ \ l(2) = 11{*}0,\ \ l(3) = {*}{*}11,\ \ l(4) = 0{*}01;$$

for example, the distance $d(1,4)$ from 1 to 4 is 2; so is the distance from 1000 to $0{*}01$.

a) Give a formula for the distance between subcubes represented by (a,b) and (a',b').

b) Find all of the subcube representations of C_5 that have 4 coordinates per label.

c) Show that the eight-vertex graph illustrated here has a subcube representation, with 4 coordinates per label, in which the vertices of the induced five-cycle have the same labels as shown above.



d) Let T be a tree with n vertices, rooted at r. Assign labels with $n-1$ coordinates to each vertex v of T, with one coordinate v_w for each $w \ne r$, defined by the rule

$$v_w \ = \ [w \text{ is an inclusive ancestor of } v] \ = \ [d(v,w) + d(w,r) = d(v,r)].$$

Exactly $d(v,r)$ coordinates of $l(v)$ are 1. Show that these are valid subcube labels.

e) Given any graph G on n vertices, let T be a spanning tree rooted at r, with every vertex v at level $d(r,v)$ of that tree. Construct labels as in (d), with $v_w = 1$ if w is an inclusive ancestor of v; but otherwise $v_w = (0, ?, *)$ if $d(v,w) - d(v,w') = (1, 0, -1)$, respectively. Here '?' is a special value that contributes $\frac{1}{2}$ to the distance when matched with 1, but 0 when matched with 0 or $*$ or ?. For example, if $G = C_5$ and $r = 0$, and if T has all edges but $2 — 3$, we get

$$l(0) = 0000;\ \ l(1) = 10?0;\ \ l(2) = 11{*}?;\ \ l(3) = ?{*}11;\ \ l(4) = 0?01.$$

Now the "distance" between, say, $l(2)$ and $l(4)$ is $1 + \frac{1}{2} + 0 + \frac{1}{2} = 2$. Prove that, in general, the "distance" between $l(u)$ and $l(v)$ is $d(u,v)$, for any graph G. Also exhibit the labels when G is the Petersen graph, $\mathit{subsets}(2, 1, -4, 0, 0, 0, {}^{\#}1, 0)$.

f) In order to obtain a subcube labeling, we need to find a rule that changes each '?' to either '0' or '$*$', like flipping a coin but smarter. Show that there is such a rule. *Hint:* Thinking of T as an ordered tree, v_w can depend on whether v precedes or follows w in preorder, as well as on the parity of the distances of v and w from r.

217. [*M15*] Which of the following potential "transitive laws" are true in general?

i) $G \subseteq G' \subseteq G''$ implies $G \subseteq G''$.	v) $G \sqsubseteq G' \subseteq G''$ implies $G \subseteq G''$.
ii) $G \subseteq G' \subseteq G''$ implies $G \sqsubseteq G''$.	vi) $G \sqsubseteq G' \subseteq G''$ implies $G \sqsubseteq G''$.
iii) $G \subseteq G' \sqsubseteq G''$ implies $G \subseteq G''$.	vii) $G \sqsubseteq G' \sqsubseteq G''$ implies $G \subseteq G''$.
iv) $G \subseteq G' \sqsubseteq G''$ implies $G \sqsubseteq G''$.	viii) $G \sqsubseteq G' \sqsubseteq G''$ implies $G \sqsubseteq G''$.

218. [*M16*] Suppose G_1, G_2, H_1, and H_2 are connected graphs, with $G_1 \oplus G_2 \subseteq H_1 \oplus H_2$. True or false: Either $G_1 \subseteq H_1$ and $G_2 \subseteq H_2$ or $G_1 \subseteq H_2$ and $G_2 \subseteq H_1$.

Subcube labels
cube
a-code, see Asterisk codes for subcubes
Asterisk codes for subcubes
tree
ancestor
inclusive ancestor
subsets graphs (SGB)
Petersen graph
preorder
transitive laws
connected graphs

219. [*M17*] True or false: If $G \subseteq H$, G is connected, and H is a forest, then $G \sqsubseteq H$.

▸ **220.** [*20*] Let G be the pattern graph $K_{1,m} \oplus P_{a_1} \oplus \cdots \oplus P_{a_t}$, where $A = \{a_1, \ldots, a_t\}$ is a multiset of positive integers. Let T be the tree with root r and $mn+m$ additional vertices x_{jk} for $1 \le j \le m$, $0 \le k \le n$, whose edges are $r \text{ --- } x_{j0}$ and $x_{jk} \text{ --- } x_{j(k+1)}$. Prove that $G \subseteq T$ if and only if A can be partitioned into m multisets whose sums are each $\le n$. (And special cases of this partitioning problem are known to be NP-complete.)

▸ **221.** [*M23*] If G is a graph on vertices V, let $q(G)$ be the graph whose vertices are pairs (v,k) with $v \in V$ and $0 \le k < 5$, and whose edges $(v,k) \text{ --- } (v',k')$ are of three kinds: (i) $v = v'$ and $\{k,k'\} \in \{\{0,1\},\{1,2\},\{2,3\},\{3,4\},\{4,0\},\{2,4\}\}$; (ii) $v \text{ --- } v'$ and $\{k,k'\} = \{0,1\}$; (iii) $v \ne v'$, $v \not\text{---}\, v'$, and $\{k,k'\} = \{0,3\}$.

a) If G has n vertices, how many vertices does $q(G)$ have? How many edges?

b) Prove that G can be strictly embedded in H if and only if $q(G)$ can be embedded in $q(H)$. (Thus unlabeled ISIP is a special case of unlabeled SIP.)

222. [*M25*] Continuing exercise 221, reduce the unlabeled SIP to the unlabeled ISIP.

224. [*M20*] (*Labeled graph embedding.*) An SIP often has side constraints in practice. For example, when graphs represent molecules, each vertex might represent a particular kind of atom (carbon, hydrogen, etc.), and each edge might be labeled strong or weak.

In general, a *labeled subgraph isomophism problem* is defined by a pattern graph G and a target graph H, where every vertex has zero or more labels l_i and every edge has zero or more labels L_j. Relations of compatibility are also defined between the pattern and target labels. The problem is to find every function f from the vertices of G to the vertices of H that satisfies four conditions: (i) If $v \ne w$ then $f(v) \ne f(w)$. (ii) If $v \text{ --- } w$ in G then $f(v) \text{ --- } f(w)$ in H. (iii) $l_i(v)$ is compatible with $l_i(f(v))$, for all i. (iv) If $v \text{ --- } w$ in G then $L_j(v,w)$ is compatible with $L_j(f(v), f(w))$, for all j.

a) Prove that every ISIP, possibly labeled, is a labeled SIP.

b) Given a labeled SIP, a vertex u of G, and a vertex $\hat{u}$ of H, show that the problem of finding all solutions with $f(u) = \hat{u}$ is a labeled SIP on the graphs $G \setminus u$ and $H \setminus \hat{u}$.

226. [*M30*] Show that the problem of testing $G \sqsubseteq H$ is NP-complete, even when G is a (free) tree and all vertices of G and H have degree ≤ 3. *Hint:* Reduce from 3SAT.

228. [*20*] If G is a graph with n vertices and m edges, let $\widehat{G}$ be the directed acyclic graph with $m+n$ vertices and $2m$ arcs obtained by replacing each edge $u \text{ --- } v$ by $u \longrightarrow uv \longleftarrow v$. Prove or disprove: (a) $G \subseteq H \iff \widehat{G} \subseteq \widehat{H}$; (b) $G \sqsubseteq H \iff \widehat{G} \sqsubseteq \widehat{H}$.

229. [*21*] Given an integer $M \ge 3$ and a graph H, is it hard to test if $C_M \sqsubseteq H$?

231. [*20*] A suitably small SIP problem can be solved as an exact cover problem using the options (53). Can an ISIP problem be solved in a similar way?

232. [*20*] Encode SIP and ISIP problems for *directed* graphs as exact cover problems.

233. [*HM30*] (S. Chatterjee and P. Diaconis, 2021.) Let $\mathcal{G}_N$ be a random graph on N vertices; each of the $\binom{N}{2}$ potential edges is independently present with probability 1/2.

a) Prove that $\mathcal{G}_{\lfloor 2\lg n+2+\delta \rfloor} \not\sqsubseteq \mathcal{G}_n$ a.s., for fixed $\delta > 0$ as $n \to \infty$.

b) Prove that $\mathcal{G}_{\lceil 2\lg n-\delta \rceil} \sqsubseteq \mathcal{G}_n$ a.s., for fixed $\delta > 0$ as $n \to \infty$.

▸ **235.** [*21*] The reduced target graph $\widehat{H}$ obtained from `BRAIN83`(250) has 11 vertices in the left brain and 11 vertices in the right brain, with only two edges between them. Why does that make $G \subseteq \widehat{H}$ impossible, when G is Chvátal's graph (52)?

236. [*23*] Embed Chvátal's graph (52) into `BRAIN83` with 6 vertices in each half-brain.

forest
partitioned
NP-complete
strictly embedded
Labeled graph embedding
chemistry
molecules
atom
compatibility
bounded degree
NP-complete
tree
3SAT
directed acyclic graph
exact cover
ISIP problem
directed graphs
Chatterjee
Diaconis
random graph
a.s.: Asymptotically almost surely
`BRAIN83`
Chvátal's graph

237. [*24*] When $k \geq 12$ is a multiple of 6, Chvátal's graph of order k has k vertices $\{0, 0+, 1-, 1, 1+, \ldots, ((k/3)-1)+, 0-\}$ and $2k$ edges $j \mathrel{-\!\!-} (j+1)$, $j \mathrel{-\!\!-} j+$, $j \mathrel{-\!\!-} j-$, $j+ \mathrel{-\!\!-} (j+1)-$, $j+ \mathrel{-\!\!-} (j+k/6)+$, $j+ \mathrel{-\!\!-} (j+k/6)-$ (modulo k) for $0 \leq j < k/3$. (Thus (52) is the case of order 12.) Can his 18-vertex graph be embedded in `BRAIN83`?

238. [*23*] Is the flower snark graph J_5 (exercise 7.2.2.2–176) embeddable into `BRAIN83`?

▸ **239.** [*20*] Constrain the embeddings of (52) so that only the essentially different solutions are found (thus only 1/8 of the total number).

242. [*M22*] If A and B are multisets of integers, say that A *surpasses* B if A's kth largest element is greater than or equal to B's kth largest element, for $1 \leq k \leq |B| \leq |A|$.

a) Given a vertex v of a graph G, let $s(v) = \{\deg(u) \mid u \mathrel{-\!\!-} v\}$ be the multiset of its neighbors' degrees. Prove that, whenever $G \subseteq H$ with an embedding function f, the multiset $s(f(v))$ surpasses $s(v)$, for all vertices v of G.

b) The obvious way to test whether or not $s(w)$ surpasses $s(v)$ is to sort the neighbors of w and v by their degrees, then to do a pairwise comparison of the sorted elements. But sorting might introduce a logarithmic factor into the running time. Explain how to perform that test in only $O(p + \deg(w))$ steps, where p is the maximum degree of any pattern vertex.

243. [*21*] Explain why LAD filtering from (58) forces `02` $\mapsto$ `LA`, after which further assignments to `01` and `03` and their neighbors get into trouble.

244. [*23*] What two solutions to the embedding problem (54) differ from Fig. 112?

245. [*24*] What's the largest n for which (a) $P_2 \mathbin{\square} P_n \subseteq$ `USA`? (b) $P_3 \mathbin{\square} P_n \subseteq$ `USA`?

246. [*15*] Do exercise 245 with $\sqsubseteq$ in place of $\subseteq$.

247. [*20*] If possible, embed half of a dodecahedron (namely, a pentagon surrounded by five other pentagons) into the `USA` graph.

250. [*21*] Explore the embedding of *simplex* graphs (triangular grids) into `USA`.

▸ **253.** [*M25*] (*Globally All Different filtering.*) When variables $x_1, \ldots, x_m$ are subject to an all-different constraint, the domains $D_1, \ldots, D_m \subseteq \{1, \ldots, n\}$ are said to be *feasible* if there's a matching of size m in the bipartite graph on vertices $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$ whose edges are $x_i \mathrel{-\!\!-} y_j$ when $j \in D_i$. A value $j \in D_i$ is said to be *removable* if $x_i \mathrel{-\!\!-} y_j$ isn't in any feasible matching.

Let $x_1 \mathrel{-\!\!-} y_{j_1}, \ldots, x_m \mathrel{-\!\!-} y_{j_m}$ be a matching, and construct the following tripartite digraph T on $\{x_1, \ldots, x_m\}$, $\{y_1, \ldots, y_n\}$, and $\{\perp\}$: $x_i \longrightarrow y_{j_i}$ and $y_{j_i} \longrightarrow \perp$, for $1 \leq i \leq m$; $x_i \longleftarrow y_j$, if $j \in D_i$ and $j \neq j_i$, for $1 \leq i \leq m$; $y_j \longleftarrow \perp$, if $j \notin \{j_1, \ldots, j_m\}$. Prove that $j \neq j_i$ is removable from D_i if and only if x_i, y_j, and $\perp$ belong to different strong components of T.

▸ **254.** [*M26*] Continuing exercise 253, further theory elucidates the situation.

a) If $I \subseteq \{1, \ldots, m\}$, let $D(I) = \bigcup\{D_i \mid i \in I\}$. Prove that the domains are feasible if and only if $|D(I)| \geq |I|$ for all subsets I. *Hint:* Use Algorithm 7.5.1H (see page vi).

b) A subset I for which $|D(I)| = |I|$ is called a "Hall set." Prove that if a feasible family of domains has no nonempty Hall sets, it has no removable values.

c) In particular, nothing is removable if $|D_i| \geq m$ for $1 \leq i \leq m$.

d) If I is a Hall set, explain why we can remove $D(I)$ from all domains D_j for $j \notin I$.

e) Prove that Hall sets of feasible domains are closed under union and intersection.

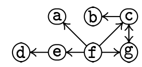
f) Prove that a feasible family of domains has no removable elements if and only if there's a partition of $\{1, \ldots, m\}$ into disjoint sets $I_0, I_1, \ldots, I_r$ with disjoint

Chvátal's graph
flower snark
essentially different solutions
multisets
comparison of multisets
sorting
LAD filtering
dodecahedron
simplex graphs
triangular grids
Globally All Different filtering
GAD
all-different
matching
bipartite graph
removable
tripartite digraph
strong components
Hopcroft–Karp algorithm
Hall set
critical block, see Hall set

domains $D(I_0)$, $D(I_1)$, …, $D(I_r)$ such that the Hall sets are precisely the 2^r sets obtainable by unions of $\{I_1,\ldots,I_r\}$. (GAD filtering always yields such a family.)

g) Relate the partition of (f) to the tripartite digraph T of exercise 253.

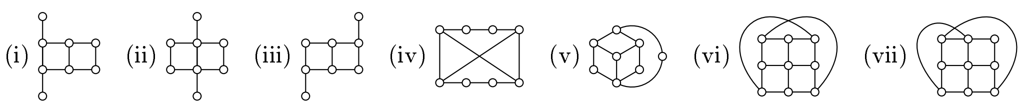
▸ **255.** [*21*] When $m=n$ in exercise 253, every solution $x_1\ldots x_n$ will be a *permutation* of $\{1,\ldots,n\}$. Improve the GAD filtering algorithm in that case.

256. [*29*] Find all (a) embeddings (b) strict embeddings of the digraph 

into Agatha Christie's "Orient Express digraph" (Fig. 3 near the beginning of Chapter 7). As in the text's solution of (54), determine the initial domains; then repeatedly branch on a variable with smallest domain, using LAD and GAD filtering.

259. [*22*] Is the Petersen graph, minus two edges, embeddable in Chvátal's graph?

260. [*25*] Which of the following graphs are *strictly* embeddable in Chvátal's graph?

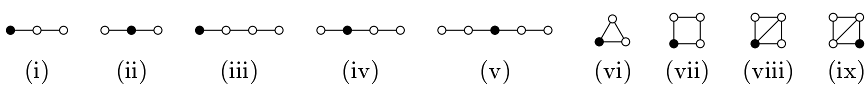


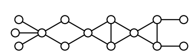
263. [*15*] True or false: $G\sqsubseteq H$ implies $G^{\le 2}\sqsubseteq H^{\le 2}$.

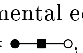
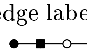
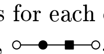
264. [*20*] Compute the vertex degrees of $G^{\le 2}$ and $H^{\le 2}$ when $G=P_4\,\square\,P_5$ and $H=\texttt{USA}$. What do those statistics imply about the domain of G's "middle vertex" `12`?

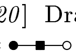
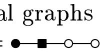
265. [*M18*] Explain informally the meaning of the supplemental label d_G^S when S is the path P_{k+1} of length k, placing the designated vertex s at one end. Show that the degree of vertex v in $G^{\le 2}$ can be expressed in terms of $d_G^{P_2}(v)$ and $d_G^{P_3}(v)$.

266. [*23*] Consider the following motif graphs S, with designated vertex $s=$ '•':



Compute the supplemental vertex labels $d_G^S(v)$, for each $v\in G=$ .

267. [*21*] Compute supplemental edge labels for each edge u — v of that same graph, using each of the motifs $S=$ , , . (Here • $=s$, ▪ $=t$.)

268. [*20*] Draw the supplemental graphs $G^{S,k}$, for the graph G of exercise 266, when (i) $S=$  and $k=1$; (ii) $S=$  and $k=2$.

269. [*20*] Consider supplemental pair labels based on the motif $S=C_4$, with s and t at distance 2. Show that, in problem (54) of embedding $P_4\,\square\,P_5$ into `USA`, such labels tell us that we can't map both `00` $\mapsto$ `MN` and `11` $\mapsto$ `MO`.

▸ **270.** [*24*] Using the supplemental graph $G^{S,2}$, where $S=P_2\,\square\,P_3$ and its vertices of degree 3 are s and t, show that the initial domains for all six interior vertices of $P_4\,\square\,P_5$ in the `USA` problem can be reduced to size 13 — less than half of what we had without it!

▸ **273.** [*20*] Restate the rules for LAD filtering in the presence of supplemental edge labels, pair labels, and graphs: Precisely what bipartite graph is required to have a matching of size deg(u) when we're trying to ascertain whether $u\mapsto v$ is locally feasible?

▸ **274.** [*20*] Extend the concept of supplemental labels and graphs to strict embeddings: Show that it's possible to construct functions $d_G(v)$, $\ell_G(v,w)$, and digraphs G^Σ such that $G\sqsubseteq H$ implies $d_G(v)\le d_H(f(v))$, $\ell_G(v,w)\le \ell_H(f(v),f(w))$, and $G^\Sigma\sqsubseteq H^\Sigma$, by analogy with (62), (64), and (65).

permutation
GAD filtering
strict embeddings
Christie
initial domains
LAD
GAD
Petersen graph
Chvátal's graph
supplemental label
supplemental vertex labels
supplemental edge labels
supplemental graphs
supplemental pair labels
`USA`
LAD filtering
bipartite graph
strict embeddings

▶ **277.** [*22*] Many graph embedding problems are simple enough to be solved efficiently without maintaining a separate domain for each pattern variable. Instead, it suffices to keep track of the vertices adjacent to the ones already assigned. Say that an unassigned vertex is *near* if it has at least one assigned neighbor; otherwise it's *far*.

a) Show that the pattern vertices can be prearranged into a fixed (static) order $p_1p_2\ldots p_m$ so that, at level l of the search, vertices $\{p_1,\ldots,p_l\}$ have been assigned and $\{p_{l+1},\ldots,p_{r_l}\}$ are near, for some $r_l \ge l$. Furthermore $r_l > l$ for $0 < l < m$ if and only if the pattern is connected.

b) Explain how to maintain a permutation $t_1t_2\ldots t_n$ of the target vertices (dynamically) so that, at level l of the search, the current assignments are $t_j = f(p_j)$ for $1 \le j \le l$, and the vertices $\{t_{l+1},\ldots,t_{s_l}\}$ are near, for some $s_l \ge r_l$.

c) If p_{l+1} has q near neighbors, must $f(p_{l+1})$ have at least q near neighbors?

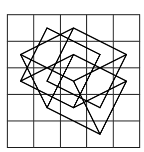
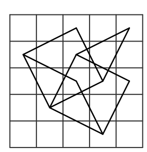
d) If p_{l+1} has q far neighbors, must $f(p_{l+1})$ have at least q far neighbors?

278. [*20*] Let D_1, $\ldots$, D_m be domains $\subseteq \{1,\ldots,n\}$, with $|D_1| \le \cdots \le |D_m|$. In practice, much of the benefit of GAD filtering (exercise 253) can be achieved more cheaply: "Set $H \leftarrow U \leftarrow \emptyset$, and do the following for $1 \le j \le m$: Set $D_j \leftarrow D_j \setminus H$ and $U \leftarrow U \cup D_j$; then if $D_j = \emptyset$ or $|U| < j$, the domains aren't feasible; otherwise if $|U| = j$, set $H \leftarrow H \cup U$." Show that all values removed from D_j were indeed removable.

▶ **279.** [*25*] One of the main subtasks of a SIP solver is to assign a target value v' to a pattern vertex v, and to update all domains appropriately. Suggest appropriate data structures for making such assignments, when GAD filtering is relaxed as in exercise 278. Consider also the use of supplemental graphs. How can your structures efficiently propagate the constraints until all remaining domains have size 2 or more?

280. [*22*] Write an `MMIX` program for the algorithm of exercise 278, assuming that $n \le 64$ and that each domain is represented bitwise. Process the domains in order of increasing size, *without* assuming that $|D_1| \le \cdots \le |D_m|$, and show that the running time for the entire computation is only $O(m)$. *Hint:* Sort into $m+1$ buckets.

283. [*22*] (*Knight's grids.*) The graphs $P_2 \,\square\, P_7$ and $P_3 \,\square\, P_3$ can be seen as knight moves

 and 

within a 5×5 board; in other words, $P_2 \,\square\, P_7 \subseteq N_5$ and $P_3 \,\square\, P_3 \subseteq N_5$, where N_n is the $n\times n$ knight graph. (This scenario generalizes the classic notion of a "knight's tour.")

a) Find the largest n with $P_m \,\square\, P_n \subseteq N_8$ when $m = 2$, 3, 4, 5, 6.

b) Find the largest n with $P_m \,\square\, P_n \sqsubseteq N_8$ when $m = 2$, 3, 4, 5, 6.

c) Find the largest n with $P_2 \,\square\, C_n \subseteq N_8$.

d) Find the largest n with $P_2 \,\square\, C_n \sqsubseteq N_8$.

e) Find the largest n with $P_3 \,\square\, C_n \subseteq N_8$.

f) Find the largest n with $P_3 \,\square\, P_3 \,\square\, P_n \subseteq N_8$.

284. [*40*] Continuing exercise 283, let $f_m(t)$ be the largest n such that $P_m \,\square\, P_n \subseteq N_t$, and let $\overline{f}_m(t)$ be the largest n such that $P_m \,\square\, P_n \sqsubseteq N_t$. Compute $f_m(t)$ and $\overline{f}_m(t)$ for as many values of $t \ge 3$ as you can, when $m = 2$, 3, and 4. [These problems make interesting benchmark tests for SIP and ISIP solvers — and the results are attractive too.]

▶ **285.** [*30*] (*Knights and queens.*) Hundreds of benchmarks for use in comparing and improving SIP and ISIP solvers have been proposed by J. Larrosa and G. Valiente

domain
near vertices
far vertices
connected
GAD filtering
approximate GAD filtering
supplemental graphs
`MMIX`
bitwise
Sort
buckets
Knight's grids
knight moves
chessboard
knight graph
benchmark tests
Knights and queens
queens
benchmarks
Larrosa
Valiente

[*Math. Structures in Comp. Sci.* **12** (2002), 403–422], who selected a wide variety of graphs from the Stanford GraphBase and proceeded to test all pairs. The smallest SIP instance that couldn't be solved within a reasonable time limit, according to C. Solnon's survey in 2018, turned out to be, "Is $N_8 \subseteq Q_8$?" In other words, are the knight moves on a chessboard isomorphic to a subset of the queen moves? Investigate this problem.

286. [*40*] Continuing exercise 285, study other values of $n \geq 3$ for which $N_n \subseteq Q_n$.

287. [*M25*] Is the $n \times n$ knight graph embeddable into the $n \times n$ rook graph for any n?

288. [*30*] Continuing exercise 285, the smallest ISIP instance that resisted solution in 2018 was quite weird: "Is *book*("`jean`", 0, 5, 0, 178, 1, 0, 0) $\sqsubseteq$ *games*(0, 0, 0, 0, 0, 0, 0, 0)?" (The pattern graph has 75 vertices; the target graph has 120.) Investigate this problem.

▶ **290.** [*30*] (*Universal graphs.*) A five-vertex graph called the "bull" ( ) is *3-universal*, in the sense that it contains every 3-vertex graph at least once as an induced subgraph.

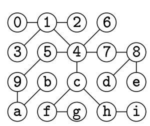
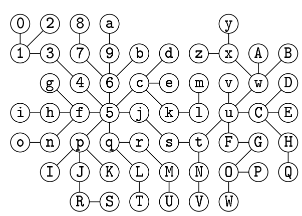
a) Find a 4-universal eight-vertex graph in which every vertex has degree 3 or 6.

b) Find a 5-universal ten-vertex graph that contains an induced 4-universal graph with eight vertices.

291. [*27*] Find a "revolving-door Gray code for 4-vertex graphs" by finding 4-vertex subsets $V_1, V_2, \ldots, V_{11}$ of the graph H in exercise 290(a) such that the induced subgraphs $H \mid V_1, H \mid V_2, \ldots, H \mid V_{11}$ are the eleven possible graphs on four vertices. Each V_{j+1} should share three vertices with V_j.

▶ **293.** [*34*] (*Subtree isomorphism.*) Let S and T be free trees, having m nodes and n nodes, respectively. A remarkably efficient algorithm, due to D. W. Matula, is able to decide whether or not $S \subseteq T$ (and $S \sqsubseteq T$) in only $O(mn\sqrt{s})$ steps, where s is the maximum inner degree of any node in S (the number of nonleaf neighbors).

a) Get ready to understand Matula's algorithm by solving the problem by hand when

S =  ($m = 19$, $s = 4$); T =  ($n = 59$).

b) In general, let the nodes of S be $\{0, 1, \ldots, m-1\}$, where $\deg(0) = 1$. We think of 0 as S's *root*; every other node r has a *parent*, $p(r)$, which is the first node on the path from r to 0. Similarly, the nodes of T are $\{0, 1, \ldots, n-1\}$; but instead of regarding T as rooted, we consider it to have $2(n-1)$ directed arcs $u \longrightarrow v$, one for each edge $u \,\text{---}\, v$ of T. This arc e is denoted for convenience by $e = {}^{u}_{v}$. Let S_r be the subtree of S consisting of all nodes whose path to 0 passes through r. Similarly, when $e = {}^{u}_{v}$, let T_e be the subtree of T consisting of all nodes whose path to u passes through v. Is $S_r \subseteq T_e$ in (a), when $r = 7$, $u = \texttt{u}$, and $v = \texttt{w}$?

c) Let $\{r_1, \ldots, r_k\}$ be the children of r in S, let $e = {}^{u}_{v}$, and let $\{w_1, \ldots, w_l\}$ be the children of v in T. Under what conditions is it possible to embed S_r into T_e, with $r \mapsto v$, based on the embeddability of smaller subtrees?

d) Let $\texttt{sol}[r][e] = [S_r \subseteq T_e \text{ with } r \mapsto \mathrm{root}(T_e)]$, for $0 < r < m$ and $0 \leq e < 2n-2$. Explain how to compute all elements of this $(m-1) \times (2n-2)$ matrix by solving $O(mn)$ maximum bipartite matching problems.

e) Furthermore, if v has $l+1$ neighbors in T, the $l+1$ matching problems with $\mathrm{root}(T_e) = v$ are almost the same and they can be solved simultaneously.

Stanford GraphBase
Solnon
rook graph
book graphs
games graphs
benchmarks
Universal graphs
bull
4-vertex graphs
5-vertex graphs
revolving-door Gray code for 4-vertex graphs
Gray code for 4-vertex graphs
Subtree isomorphism.
free trees
Matula
inner degree
root
parent
subtree
maximum bipartite matching
bipartite matching
matching

f) Sketch the details of a complete implementation, using Algorithm 7.5.1H (the Hopcroft–Karp algorithm) for matching. What's the `sol` matrix for problem (a)?

294. [*29*] Evaluate Matula's algorithm (exercise 293) empirically by applying it to several classes of free trees:

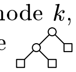
a) Let S run through all 551 free trees with $m = 12$, and let T run through all 19320 free trees with $n = 16$.
b) Let S and T be uniformly random free trees with $m = 25$ and $n = 1000$.
c) Let T be a random free tree with $n = 1000$; obtain S by repeatedly removing a random leaf, 100 times.

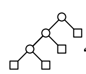
▸ **295.** [*20*] The *feedback vertex set* problem asks whether a given digraph D has a set of k vertices that cover every directed cycle. Show that it's a special case of ISIP.

▸ **297.** [*23*] Exercise 4 illustrates how any finite CSP can be encoded as an XCC problem by listing its positive table constraints — the tuples that satisfy the given relations. Show that any finite *binary* CSP can be encoded as an XC problem by listing its *negative* table constraints — the ordered pairs that do *not* satisfy the given relations.

Illustrate your method by explaining how to find all *radio colorings* of a given graph, using the colors $\{0, 1, \ldots, d-1\}$. (See exercise 7.2.2.2–36.)

298. [*21*] Apply exercise 297 to enumerate all optimum radio colorings of (a) $P_3 \,\square\, P_3$; (b) Petersen's graph; (c) Chvátal's graph; (d) Mycielski's graph M_4.

300. [*20*] Any extended binary tree with d leaves and height h defines an h-bit *prefix code* for a d-element domain: The representation of k is the path to external node k, using 0 for a left branch and 1 for a right branch. For example, the binary tree  defines the 2-bit codewords $(00, 01, 1*)$ for $k = (0, 1, 2)$.

a) Is this the same as Table 2's "prefix encoding"?
b) What's the prefix code for the extended binary tree  ?
c) Relate that code to the "weakened encoding" of Table 2.

301. [*20*] Reverse-engineer Table 2's "reduced encoding." What makes it tick?

302. [*20*] How many variables, clauses, and total literals are generated by each of the encodings in Table 2, when the given graph has V vertices and E edges?

303. [*17*] Why is the Sierpiński gasket graph $S_n^{(3)}$ uniquely 3-colorable?

304. [*20*] True or false: The graph $S_n^{(3)}$ minus any edge is *not* uniquely 3-colorable.

▸ **306.** [*M25*] Since $S_n^{(3)}$ is a subgraph of the triangular grid, we can also name its edges and vertices by using the barycentric even/odd coordinate system of answer 7.2.2.1–124. Give formulas for the barycentric coordinates of triangle $a_1 \ldots a_{n-1}$ and its vertices, assuming that vertex $12\ldots2 = 21\ldots1 \mapsto (0,0,0)$. What are the coordinates of $0\ldots0$, $1\ldots1$, and $2\ldots2$? *Hint:* Show that every odd number between -2^n and $+2^n$ has a unique *binary representation* $(b_1 \ldots b_n)_2$ in which every digit b_j is ±1.

307. [*18*] What clauses can be used with Table 2 to ensure that vertices u, v, and w will have the respective colors 0, 1, and 2?

309. [*29*] Apply the encodings of Table 2 to the problem of 3-coloring $\widehat{S}_n^{(3)}$ for small n. How well do they work with Algorithms 7.2.2.2L and 7.2.2.2C?

▸ **311.** [*M30*] Find a simple formula for the size of the backtrack tree that arises when proving that $\widehat{S}_n^{(3)}$ cannot be 3-colored. Each node should branch on a vertex with fewest available colors, breaking ties by choosing the lexicographically smallest.

Hopcroft
Karp
Matula
feedback vertex set
cover
directed cycle
positive table constraints
table constraints
CSP represented as XCC
XCC representation of CSP
negative table constraints
radio colorings
L(2,1) labeling, see radio coloring
Petersen's graph
Chvátal's graph
Mycielski's graph M_4
extended binary tree
binary tree
prefix code
reduced encoding
Sierpiński gasket graph
triangular grid
barycentric even/odd coordinate system
even/odd coordinate system
binary representation
search tree size
analysis of algorithms
MRV heuristic

313. [*40*] The pinched Sierpiński gasket $\widehat{S}_4^{(3)}$ remains uncolorable with three colors even if we remove the edges 0000 — 0001, 0101 — 0111, 0222 — 2002, 2202 — 2222, 2212 — 2222. What's the largest number of edges that can be removed from $\widehat{S}_n^{(3)}$ before it becomes 3-colorable?

▸ **315.** [*21*] What clique hints, analogous to (69), are most appropriate for the (a) log, (b) weakened, (c) reduced, and (d) prefix encodings?

316. [*24*] How could a SAT solver learn '$(0202_2 \vee 0222_2)$' from the prefix-encoded clauses for 3-coloring $\widehat{S}_4^{(3)}$? (See (70); assume that the clique hints have been given.)

▸ **318.** [*25*] Exercise 7.2.2.1–117 shows that graph coloring is an XC problem. Empirically, how long does it take Algorithm 7.2.2.1X to show that $\widehat{S}_n^{(3)}$ cannot be 3-colored?

319. [*M46*] Can an exponential lower bound be proved on the refutation length of the clauses for 3-uncolorability of $\widehat{S}_n^{(3)}$? (See Theorem 7.2.2.2B.)

320. [*24*] Repeat exercise 309, but test flower snark line graphs $L(J_q)$ instead of $\widehat{S}_n^{(3)}$.

321. [*40*] The flower snark line graph $L(J_q)$ for odd q actually remains 3-uncolorable even if we remove any one of its $12q$ edges. What's the largest number of edges that can be removed before it becomes 3-colorable?

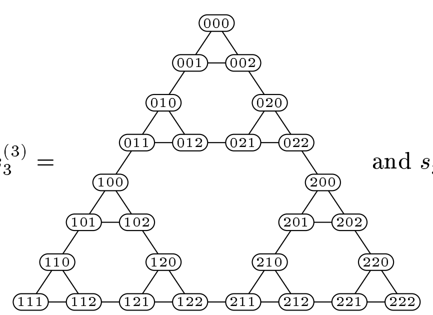
323. [*16*] The graph $S_3^{(4)}$ in Fig. 114 has $(4^3+4)/2 = 34$ vertices, but only 27 of them are visible. What are the names of the seven hidden vertices? (Give both names.)

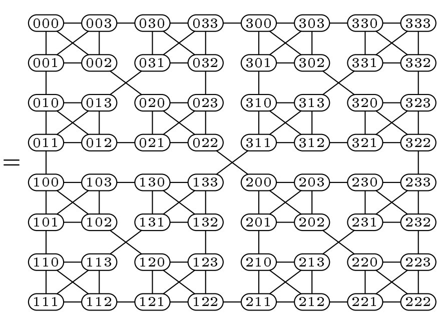
324. [*10*] What's a simpler name for the Sierpiński simplex graph $S_n^{(d)}$ when $d = 2$?

325. [*M15*] True or false: $S_n^{(d)}$ is an induced subgraph of $S_n^{(d')}$ when $d \le d'$.

326. [*16*] Almost every vertex of $S_n^{(d)}$, $\widehat{S}_n^{(d)}$, and $\overline{S}_n^{(d)}$ has degree $2d-2$. What vertices are the exceptions?

▸ **328.** [*M17*] The "proper" *Sierpiński graphs* $s_n^{(d)}$, exemplified by

$s_3^{(3)} =$ 

and $s_3^{(4)} =$ 

,

are different from but strongly related to the Sierpiński simplex graphs $S_n^{(d)}$. In general, $s_n^{(d)}$ has d^n vertices $a_1 \ldots a_n$, for $0 \le a_j < d$, and two kinds of edges:

i) clique edges $a_1 \ldots a_{n-1}j$ — $a_1 \ldots a_{n-1}k$, for $0 \le j < k < d$;
ii) nonclique edges $a_1 \ldots a_i jk \ldots k$ — $a_1 \ldots a_i kj \ldots j$, for all $0 \le i < n-1$ and $0 \le j < k < d$.

Notice that almost every vertex has degree d; this property is akin to exercise 326.

a) Give a formula for the total number of edges in $s_n^{(d)}$.
b) What's an intuitive way to obtain $S_n^{(d)}$ from $s_{n+1}^{(d)}$?
c) What's an intuitive way to obtain $S_n^{(d)}$ from $s_{n-1}^{(d)}$?

clique hints
log
weakened
reduced
prefix encoding
XC problem
exact covering problem
lower bounds for resolution refutation length
flower snark line graphs
Sierpiński simplex graph
induced subgraph
subgraph
Sierpiński graphs

330. [*M25*] Show that, in every d-coloring of $S_n^{(d)}$, for $n > 1$, the number of pure vertices having a given color is congruent to d (modulo 2).

▶ **332.** [*22*] Generalize the encodings of '$u \neq v$' in Table 2 from ternary to d-ary.

▶ **333.** [*23*] Generalize the clique hints of exercise 315 to d-ary. Illustrate the case $d = 5$.

335. [*M26*] When we try to prove that $\overline{S}_n^{(4)}$ isn't 4-colorable, we can assume without loss of generality that vertices $0\ldots00$, $0\ldots01$, $0\ldots02$, $0\ldots03$ have the respective colors 0, 1, 2, 3. Show that the remaining problem still has 6-fold symmetry. How could that symmetry be exploited?

336. [*24*] Repeat exercise 309, but test $\overline{S}_n^{(4)}$ instead of $\widehat{S}_n^{(3)}$. (Use clique hints.)

337. [*24*] Repeat exercise 309, but test $\widehat{S}_n^{(5)}$ instead of $\widehat{S}_n^{(3)}$. (Use clique hints.)

338. [*34*] Apply a state-of-the-art SAT solver to the clauses for $\widehat{S}_n^{(3)}$, $\overline{S}_n^{(4)}$, $\widehat{S}_n^{(5)}$, and $L(J_q)$ for various encodings, and compare the results to those obtained with Algorithm 7.2.2.2C in exercises 309, 320, 336, and 337.

341. [*25*] Explain how to generate SAT clauses that efficiently encode the relation '$u \leq v - t$', when variables u and v are represented with the log encoding and t is constant. Illustrate your construction in the cases '$u \leq v + 1$' and '$u \leq v - 2$', assuming that $u = (u_8u_4u_2u_1)_2$ and $v = (v_8v_4v_2v_1)_2$.

342. [*20*] Shorten the direct encoding of (78) by simplifying (79). (For example, $(\bar{u}_0 \vee \bar{v}_1 \vee \bar{w}_1) \wedge (\bar{u}_0 \vee \bar{v}_2 \vee \bar{w}_1)$ can be replaced by $(\bar{u}_0 \vee v_0 \vee \bar{w}_1)$.)

343. [*17*] What are the direct and support encodings of '$uv \in \{00, 01, 12, 20\}$'?

▶ **346.** [*20*] If the binary relation of exercise 343 is treated as a k-ary relation with $k = 2$ and "binarized" by the general strategy of (77), what support clauses do we get?

347. [*11*] Derive $\overline{R}_{001}, \overline{R}_{010}, \ldots, \overline{R}_{211}$ from (80)–(82) and (R_{000}) by unit propagation.

350. [*M16*] Let $R(v_1, \ldots, v_k)$ be a k-ary relation, where variable v_j has domain $[0\,.\,.\,d_j)$ for $1 \leq j \leq k$. If R contains exactly G tuples, how many total literals are in the (a) preclusion (b) support clauses, when R is encoded for SAT?

351. [*M20*] Prove that the direct encoding doesn't need the at-most-one clauses.

▶ **352.** [*M22*] Use resolution to derive the clauses for $b \in D_v$ in (76) from the clauses for $a \in D_u$. (Thus half of the support clauses for R are redundant.)

353. [*22*] How many of the 2^{27} ternary relations on variables whose domain size is 3 can be expressed as the conjunction of *binary* relations on those variables?

354. [*23*] Two of the 2^{27} ternary relations on ternary domains are equivalent to each other if they differ only with respect to permuting the elements of the domains or permuting the order of the variables (or both). Thus, an equivalence class might contain as many as $3!^4 = 1296$ different relations. How many equivalence classes are there? How many of them satisfy the special condition of exercise 353? How many "come close"?

356. [*20*] When variables u, v, and w all have the domain $[0\,.\,.\,d)$, let $R(u, v, w)$ be the median-fixing relation '$\langle uvw \rangle = c$'. Is R the conjunction of its three binary projections?

▶ **357.** [*20*] Let $R(a, b, c, d, e)$ be the quinary relation whose tuples are `WORDS(1000)`, the most common 1000 five-letter words of English: `which`, `there`, ..., `ditch`. What tuples are not in R, but are in all of its projections $R_a(b, c, d, e)$, $R_b(a, c, d, e)$, ..., $R_e(a, b, c, d)$?

▶ **358.** [*21*] One way to perform unit propagation is to (i) delete any clause that contains a true literal; (ii) remove all false literals from all clauses; (iii) regard a unit clause as

pure vertices
symmetry
augmented Sierpinski tetrahedron
clique hints
pinched Sierpinski simplex
clique hints
encode
log encoding
direct encoding
direct
support encoding
unit propagation
at-most-one clauses
resolution
ternary relations
binary relations
equivalent
median
`WORDS(1000)`
unit propagation

a true literal; (iv) regard an empty clause as a contradiction. If this process has been applied to the support encoding S for some nonempty relation $R(v_1, \ldots, v_k)$, prove:

a) There will be no contradiction.

b) If no clauses remain, R is satisfied by the true literals $v_{1a_1}, \ldots, v_{ka_k}$.

c) Otherwise the remaining clauses are the support encoding for some relation R'.

d) If literal v_a remains, there's a solution with v_a true and another with v_a false.

e) If literal v_a remains, statements (a), (b), and (c) hold also for the clauses $S \wedge (v_a)$.

f) If literal v_a remains, statements (a), (b), and (c) hold also for the clauses $S \wedge (\bar{v}_a)$.

▸ **360.** [*24*] (*The haystack problem.*) Consider n^2 variables x_{ij} for $0 \le i, j < n$, each with domain $\{0, 1, \ldots, n-1\}$, subject to the following constraints: (i) $x_{ij} \ne x_{ij'}$ when $j \ne j'$. (ii) $x_{i0} + x_{ij} > 1$ when $0 < i, j < n$. (iii) $x_{i0} = x_{0i}$ when $0 < i < n$.

a) Prove that this CSP is unsatisfiable.

b) Formulate it as an exact cover problem, and try it with algorithms of §7.2.2.1.

c) Formulate it as a satisfiability problem, and try it with algorithms of §7.2.2.2.

▸ **365.** [*M30*] Consider the $d \times d'$ matrix (r_{ij}), where $r_{ij} = [ij \in R]$ characterizes a binary relation R. When doing domain reduction, we want to know the support vectors $s_i = [\text{row } i \text{ of } r \text{ is nonzero}]$ and $s'_j = [\text{column } j \text{ of } r \text{ is nonzero}]$, for $0 \le i < d$ and $0 \le j < d'$. It's easy to compute s_i and s'_j by simply scanning row i or column j until we see a 1. But let's suppose that it's *expensive* to access the array r (that is, to decide whether or not $ij \in R$); so we want to avoid checking r_{ij} whenever possible.

The following two-pass procedure has been suggested, using an auxiliary $d \times d'$ Boolean matrix m to remember where we've already looked in r. Initially m, s, and s' are zero. "Pass 1. For $0 \le i < d$ do this: For $0 \le j < d'$, set $m_{ij} \leftarrow 1$; if $r_{ij} = 1$, set $s_i \leftarrow 1$, $s'_j \leftarrow 1$, and break out of the loop on j. Pass 2. For $0 \le j < d'$ with $s'_j = 0$ do this: For $0 \le i < d$ with $m_{ij} = 0$, if $r_{ij} = 1$, set $s'_j \leftarrow 1$, and break out of the loop on i."

a) Analyze that algorithm, assuming that each entry of the matrix is independently random, with $\Pr(r_{ij} = 1) = p$ for all i and j. Given i and j, what is the probability that r_{ij} will be examined in Pass 1? In Pass 2?

b) Improve Pass 1. *Hint:* We can often avoid looking at r_{ij} if we know that $s'_j = 1$.

c) Experiment with the improved algorithm when, say, $d = d' = 100$.

366. [*M46*] Does the algorithm of exercise 365(b) have minimum expected cost, over all support-finding algorithms for random $d \times d'$ matrices of density p?

▸ **370.** [*M25*] The *chain problem* is a CSP with n variables $x_1, \ldots, x_n$, of which x_1 through x_m are "sources" and x_n is a "sink." All variables have domain $\{0, 1, 2\}$. There are m binary constraints, '$x_i \ne x_n$' for $1 \le i \le m$; also $n - m$ ternary constraints,

$$\text{'either } x_i = x_{j(i)} \text{ or } x_i = x_{k(i)}\text{'} \qquad \text{for } m < i \le n,$$

where two indices with $0 < j(i) < k(i) < i$ are prescribed for every such i. (Notice the similarity with addition chains, Boolean chains, resolution chains, etc.)

a) Explain why every chain CSP is unsatisfiable.

b) Express any given chain CSP as an XCC problem with $\le 15n$ options.

c) Exactly how many chain CSPs are possible, given m and n with $1 \le m \le n$?

d) Experiment with XCC solvers on uniformly random chain CSPs that have been formulated as in (b), when $m = 24$ and n varies.

e) Exhibit supports that establish domain consistency for every chain CSP. But show that Algorithm D will find a contradiction just after x_n is assigned a value.

support encoding
haystack problem
domain reduction
support vectors
Analyze
random
support-finding algorithms
chain problem
sources
supports
domain consistency

371. [*M28*] Analyze the problems of exercise 370: Let $P_{m,n}$ be a random chain problem, where every possible choice of the pairs $(j(i), k(i))$ for $i > m$ is equally likely.

a) Let $S_{m,n}$ be the expected total number of sinks in $P_{m,n}$. (A sink is a variable x_i that isn't in $\{j(i+1), k(i+1), \ldots, j(n), k(n)\}$.) Find a simple formula for $S_{m,n}$.

b) A sink x_i for which $i < n$ is not connected to x_n. Neither is a variable that's constrained only by unconnected variables. Find a recurrence by which we can compute $C_{m,n}$, the expected number of variables of $P_{m,n}$ that are connected to x_n. (For example, $C_{3,5} = 66/18$.) What is $C_{24,64}$?

c) Find a recurrence by which we can compute $c_{m,n}$, the probability that all variables of $P_{m,n}$ are connected to x_n. (For example, $c_{3,5} = 3/18$.) What is $c_{24,64}$?

372. [*HM46*] What's the asymptotic behavior of $C_{m,n}$ and $c_{m,n}$, for large m and/or n?

380. [*M20*] Exactly how many permutations of $\{1, 2, \ldots, n\}$ have $p_{j+1} < p_j + d$, for $1 \le j < n$, given a number d with $1 \le d \le n$?

381. [*M21*] For every subset $S \subseteq \{1, \ldots, n-1\}$, prove that exactly one slow growth permutation of $\{1, 2, \ldots, n\}$ has the property "$p_{j+1} > p_j$ if and only if $j \in S$."

382. [*M20*] True or false: The inverse of a slow growth permutation has slow growth.

▶ **383.** [*23*] Construct an exact cover problem whose solutions are the 2^{n-1} slow growth permutations of $\{1, 2, \ldots, n\}$. There should be n^2 options, each containing $O(\log n)$ items. *Hint:* Use the pairwise ordering trick of exercise 7.2.2.1–20.

384. [*21*] Use exercise 382 to solve exercise 383 with more restrictive options.

▶ **390.** [*M33*] (*Backmarking.*) Suppose we are solving a CSP by assigning values to variables x_1, x_2, …, in that order. Step t of the search process begins at level $l = l_t$, at which time we've made certain provisional assignments $x_1 \leftarrow a_1$, …, $x_l \leftarrow a_l$ and we want to select a consistent value a_{l+1} for x_{l+1}. If we succeed, this step is a "forward step," and we'll have $l_{t+1} = l + 1$; otherwise it's a "backward step," and $l_{t+1} = l - 1$. (Initially $l_0 = 0$. A backward step from level 0 terminates the search.)

After the first backward step from level l, subsequent steps at that level tend to repeat much of the previous computations. Indeed, there's a value $s = s_t \le l$ for which the previous backward step at level l dealt with exactly the same assignments $x_j \leftarrow a_j$ for $1 \le j < s$. Thus we already "know" the results of all tests on s-ary relations between a_1, …, a_{s-1} and a_{t+1}, and we could have saved that information in an auxiliary array.

a) Forward and backward steps can be represented by a sequence of nested parentheses as in 7.2.1.6–(1). What values of l_t and s_t for $0 \le t < 30$ correspond to the sequence '(())((())((()(()))())(()(()))))'? (Use $s = 0$ before backward steps.)

b) Devise a way to calculate s_0, s_1, …, from a given level sequence l_0, l_1, … . *Hint:* Maintain a sequence of intervals $[p_0 \,..\, q_0]$, $[p_1 \,..\, q_1]$, …, $[p_r \,..\, q_r]$, where $0 = p_0 < p_1 < \cdots < p_r$, such that $s_t = p_k$ when k is maximum with $p_k \le l_t \le q_k$.

c) Show that the s values can indeed be rather complicated, by constructing a level sequence l_0, l_1, … for which the intervals in the preceding hint are

$$[0 \,..\, \infty],\ [2 \,..\, 8],\ [4 \,..\, 6],\ [5 \,..\, 5],\ [10 \,..\, 15],\ [11 \,..\, 12],\ [14 \,..\, 14].$$

d) Find levels $0 = l_0$, l_1, …, $l_{30} = 0$ for which $s_0 + s_1 + \cdots + s_{29} \ge 107$.

e) What's the average of $s_0 + \cdots + s_{29}$ over *all* level sequences $0 = l_0$, …, $l_{30} = 0$?

f) The amount of nonrepeated computation at step t can be measured by $l_t - s_t$. Generate random sequences of nested parentheses, 1000000 of each, and estimate the average value of $l_t - s_t$ for $0 \le t < 2000000$. *Hint:* See Algorithm 7.2.1.6W.

chain problem
sink
recurrence
asymptotic behavior
slow growth permutation
inverse
exact cover problem
pairwise ordering trick
Backmarking
forward step
backward step
nested parentheses
average

g) Let $D_j = \{1, \ldots, d_j\}$ be x_j's domain. Explain how to use s_t to avoid recomputation at step t, by maintaining a "mark" M_{ja} for each variable x_j and each $a \in D_j$.

▸ **400.** [*31*] A *constraint satisfaction automaton* (CSA) is a nondeterministic automaton based on a given CSP. Like all automata, it has a set Q of states, which contains a set $I \subseteq Q$ of input states and a set $\Omega \subseteq Q$ of output states, together with a transition rule that takes us from state to state. In this case the transitions have the general form

$$q \;\mapsto\; v_1 \setminus a_1, \ldots, v_t \setminus a_t,\; (v \leftarrow a?\; q'\colon q''), \quad \text{for some } t \geq 0,$$

where the v's are variables, the a's are domain elements, and the q's are states. The meaning is, "Begin deterministically: For $1 \leq j \leq t$, if v_j is unassigned, remove a_j from its domain if a_j was present. Then branch nondeterministically: Either (i) assign a as the value of variable v, and go to state q', or (ii) remove a from the domain of v and go to state q''." Variable v must not previously have been assigned a value. Case (i) is permitted only when a is in v's current domain. It means that the domain of v is reduced to the single value $\{a\}$; furthermore, the domain of every other unassigned variable w is also reduced, if necessary, so that every constraint for which all variables but w are assigned is fully satisfied by every value in w's remaining domain.

A CSA computation begins in an initial state, with all variables unassigned, and with all domains equal to the initial domains but restricted by the unary constraints. It ends successfully in an output state when all variables have been assigned; or it can end unsuccessfully, either in a state q for which some domain is empty, or for which all variables are assigned but $q \notin \Omega$, or for which no transition rule was specified. The *solutions* of a CSA are the tuples of assigned values that a successful computation can produce. (In particular, those solutions will also solve the given CSP.)

Either v or a in the '$v \leftarrow a$' part of a transition rule, or both, can be replaced by an *asterisk* ($*$), meaning that the automaton itself is supposed to choose the variable and/or the value to be assigned, deterministically, using an arbitrary heuristic. Of course such a "wildcard" transition is inapplicable when no valid assignment is possible.

For example, the CSA with $Q = I = \Omega = \{q\}$ and the wildcard transition rule '$q \;\mapsto\; (* \leftarrow *?\; q\colon q)$' simply has the same solutions as the given CSP. The CSA with $Q = \{q_0, q_1, q_2\}$, $I = \{q_0\}$, $\Omega = \{q_2\}$, and transitions

$$q_0 \;\mapsto\; (v \leftarrow a?\; q_1\colon q_2); \quad q_1 \;\mapsto\; w \setminus b, (* \leftarrow *?\; q_2\colon q_2); \quad q_2 \;\mapsto\; (* \leftarrow *?\; q_2\colon q_2)$$

has all solutions except those for which $v = a$ and $w = b$.

The domain element in a transition rule can also be a *named wildcard* of the form 'a^*', where a is a local identifier. It means that the value a chosen by the automaton can be used in the specification of the states q' and q''. For example, the transition rule

$$q \;\mapsto\; (v \leftarrow a^*?\; q_a\colon q_{-a})$$

will cause the automaton to choose an arbitrary value a in v's domain. Then if, say, $a = 3$, it will branch nondeterministically, either assigning $v \leftarrow 3$ and going into state q_3 or making no assignment and going into state q_{-3}.

Notice that a CSA essentially adds a global constraint to the given CSP. "Find all solutions that correspond to a sequence of states in the CSA from I to Ω." It can be simulated by any procedure that makes further domain reductions, for example to maintain consistency, as long as those reductions don't eliminate any solutions.

The following examples exhibit some of the versatility provided by this CSA formalism. Let the variables of a given CSP be $\{v_1, \ldots, v_n\}$, each with domain $[0\,..\,d) = \{0, 1, \ldots, d-1\}$, and subject to any number of further constraints.

constraint satisfaction automaton
CSA
nondeterministic automaton
automata
states
input states
output states
transition rule
variables
domain elements
solutions
asterisk
wildcard
global constraint
consistency

a) Define a CSA whose solutions $v_1 \ldots v_n$ are those with $(v_1+\cdots+v_n) \bmod 5 \in \{1,3\}$.
b) Define a CSA for the solutions where each value occurs at most twice.
c) Define a CSA for the solutions where each value occurs either twice or not at all.
d) Similarly, design a CSA for all solutions $v_1 \ldots v_n$ that are *restricted growth strings*. (See Section 7.2.1.5; in particular, $v_1 = 0$ and v_2 is 0 or 1.)
e) Let $d = 2$, and restrict the solutions to binary strings $v_1 \ldots v_n$ that correspond to *nested parentheses* when $0 \leftrightarrow$ `(` and $1 \leftrightarrow$ `)`. (In particular, $v_1 = 0$ and $v_n = 1$.)

401. [*20*] Suppose the reflection $v_n \ldots v_2 v_1$ solves a certain CSP whenever $v_1 v_2 \ldots v_n$ does. All domains are $[0\,..\,d)$. Design a CSA that yields only one of those solutions.

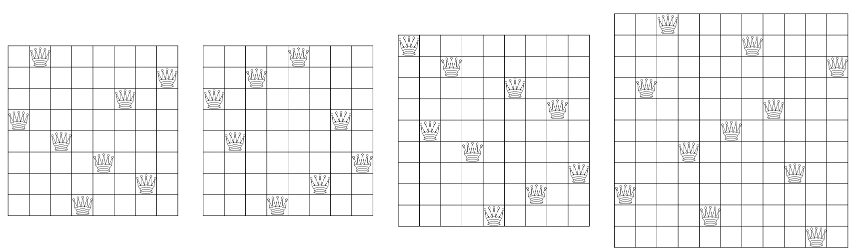
402. [*23*] Suppose the cyclic permutation $v_2 \ldots v_n v_1$ solves a certain CSP whenever $v_1 v_2 \ldots v_n$ does. Design a CSA that yields just one solution in each equivalence class under cyclic shifts. All domains are $[0\,..\,d)$. *Hint:* Consider *prime strings* (Section 7.2.1.1).

▸ **403.** [*28*] Solutions to the n queens problem belong to the same equivalence class if they differ only by a reflection and/or rotation of the board. The purpose of this exercise is to define *canonical solutions*, of which there's exactly one in each class.

Denote the cells by (i,j) for $0 \le i,j < n$. Let R_i be the column containing a queen in row i, and let C_j be the row containing a queen in column j; thus $R_i = j$ if and only if $C_j = i$. Let $\bar{x} = n-1-x$; notice that rotation by 90° changes R_i to $C_{\bar{\imath}}$ and C_j to $\bar{R}_j$.
a) Let $(a_i, b_i, c_i, d_i) = (R_i, C_{\bar{\imath}}, \bar{R}_{\bar{\imath}}, \bar{C}_i)$. Can we have $\{a_i, b_i, c_i, d_i\} \cap \{\bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{d}_i\} \ne \emptyset$?
b) How does reflection of the board change the numbers (a_i, b_i, c_i, d_i) of a solution?
c) Let $n' = \lfloor n/2 \rfloor$. Write out the eight values of the $4\lceil n/2 \rceil$-tuple

$$(a_{n'}, b_{n'}, c_{n'}, d_{n'}; a_{n'+1}, b_{n'+1}, c_{n'+1}, d_{n'+1}; \ldots; a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1})$$

that occur when the following solutions are rotated and/or reflected:



d) Explain why the lexicographically least of eight such tuples is a canonical solution.
e) True or false: If $n = 2n'$, the canonical tuple begins with $a_{n'} \le n' - 2$.
f) Design a CSA for canonical solutions to the n queens problem.

404. [*25*] What's the lexicographically *largest* canonical solution that uses 32 queens?

405. [*24*] A *superqueen* (also called an "amazon") combines the moves of a queen and a knight. Use the methods of exercise 403 to determine the number of inequivalent solutions to the n superqueens problem for small n.

restricted growth strings
nested parentheses
reflection
symmetry breaking (removal)
breaking symmetries
cyclic shifts
Lyndon words, see prime strings
prime strings
n queens problem
queens problem
equivalence class
reflection
rotation
canonical solutions
lexicographically least
superqueen
amazon
knight
n superqueens problem

▶ **450.** [*20*] (*Fillomino.*) A "fillomino pattern" is a labeling of grid cells with positive integers in such a way that every cell labeled d is rookwise connected to exactly d cells that have the same label. (Equivalently, it's a way to pack a shape with polyominoes, where no two d-ominoes have an edge in common.) For example, a more-or-less random fillomino pattern is shown at the right.

```
33877722
31888777
44488179
33418899
13225999
88855926
81859926
88856666
```

A "fillomino puzzle" is a labeling with positive integers and blanks, for which exactly one fillomino pattern can be obtained by filling in the blanks.

If, for instance, we want to solve puzzle (i) below, it's clear that the upper left corner cell must be labeled 2, and that there must be a 3 at the lower left.

```
(i)      (ii)     (iii)
␣14␣     214␣     214␣
2␣␣␣  ;  2␣␣␣  ;  24␣␣  .
1␣␣2     1␣␣2     1␣␣2
␣3␣␣     33␣␣     33␣␣
```

So (ii) is forced; and with a bit of thought we see that the blank below the upper 1 can't be 3 or more than 4. Hence we reach (iii), and ultimately a unique solution.

Show that one of the six clues in puzzle (i) is actually redundant. But none of the other five can be removed, without spoiling the puzzle by allowing additional patterns.

451. [*M24*] Compute the exact number of $2 \times n$ fillomino patterns for $n = 1, 2, 3, \ldots,$ until reaching an n for which that number exceeds 10^{100}.

452. [*21*] The "fillomino problem" is to find every fillomino pattern that's consistent with a given partial labeling. Formulate it as an exact cover problem.

453. [*22*] Try your luck with the following selected fillomino puzzles:

```
(a)
2␣1␣2␣
1␣␣␣12
␣1␣3␣␣
␣␣␣␣␣␣ ;
␣␣3␣1␣
21␣␣␣1
␣2␣1␣2

(b)
33␣␣␣␣␣␣␣
31415926␣
␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣
535897932 ;
␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣
␣38462643
␣␣␣␣␣␣␣44

(c)
␣␣␣␣␣2␣␣␣␣
␣24␣␣8␣24␣
␣68␣␣6␣68␣
␣␣␣␣␣4␣␣␣␣
2684␣␣␣␣␣␣
␣␣␣␣␣␣6482 ;
␣␣␣␣4␣␣␣␣␣
␣24␣8␣␣24␣
␣68␣6␣␣68␣
␣␣␣␣2␣␣␣␣␣

(d)
1␣341412␣␣
␣␣␣␣␣␣␣␣␣4
2␣␣3␣2␣␣3␣
␣2␣3␣33␣1␣
␣1␣␣␣␣␣␣␣2
2␣␣␣␣␣␣␣4␣ ;
␣4␣44␣3␣2␣
␣3␣␣1␣3␣␣1
1␣␣␣␣␣␣␣␣␣
␣␣342242␣3

(e)
␣24␣␣␣␣␣␣␣
␣15␣␣␣␣␣13
␣␣␣36␣␣␣45
␣␣␣46␣63␣␣
␣␣␣␣␣␣31␣␣
␣␣65␣␣␣␣␣␣ .
␣␣52␣42␣␣␣
34␣␣␣13␣␣␣
23␣␣␣␣␣41␣
␣␣␣␣␣␣␣24␣
```

454. [*24*] There are 59,951 4×4 fillomino patterns $\varPhi$ whose labels don't exceed 5. Exhaustively study them all, finding every valid puzzle without redundant clues for which $\varPhi$ is the solution. What interesting statistics and extremal examples lurk among them?

▶ **455.** [*M26*] Prove that the solution to a fillomino puzzle whose maximum clue is s cannot include a d-omino with $d > 4s+2$. Can you construct such puzzles with $d = 4s+2$?

▶ **456.** [*M30*] Characterize all valid $m \times n$ fillomino puzzles whose clues are all 1s.

457. [*HM40*] Let $\#_d(\varPhi)$ be the number of cells labeled d in the fillomino pattern $\varPhi$, and let $\delta_d = \limsup_{n\to\infty} \#_d(\varPhi_n)/n^2$ be the maximum density of d's in any infinite sequence of $n \times n$ patterns $\varPhi_n$. Determine δ_d for as many d as you can, and show that $\delta_d = 1 - \Theta(1/\sqrt{d}\,)$ as $d \to \infty$.

500. [*15*] True or false: If Montanari's procedure (200) ever sets $R_{ij} \leftarrow O$ (the all-0 matrix) for at least one pair (i, j), it will eventually set $R_{i'j'} \leftarrow O$ for *all* pairs (i', j').

501. [*23*] Summarize what (200) will do when presented with each of the following inputs, assuming that every unspecified relation R_{ij} is the identity matrix when $i = j$, or the all-1s matrix when $i \neq j$. (Domain sizes can be deduced from the given matrices.)

a) $n = 5$, $R_{12} = R_{23} = R_{34} = R_{45} = R_{51} = \binom{0\,1}{1\,0}$.

fillomino
rookwise connected
polyominoes
pi, "random" example
e, "random" example
googol=10^{100}
exact cover problem
pi, random
density
Montanari
O, the all-0 matrix

b) $n = 5$, $R_{12} = R_{23} = R_{34} = R_{45} = \left(\begin{smallmatrix}0&1&0\\0&0&1\\1&0&0\end{smallmatrix}\right)$ and $R_{51} = \left(\begin{smallmatrix}0&1&1\\0&0&1\\1&0&0\end{smallmatrix}\right)$. Montanari

c) $n = 5$, $R_{12} = R_{23} = R_{34} = R_{45} = R_{51} = \left(\begin{smallmatrix}0&1&1\\1&0&1\\1&1&0\end{smallmatrix}\right)$.

d) $n = 3$, $R_{12} = \left(\begin{smallmatrix}1&1\\1&0\end{smallmatrix}\right)$, $R_{13} = \left(\begin{smallmatrix}1&1&0\\0&1&1\end{smallmatrix}\right)$, and $R_{23} = \left(\begin{smallmatrix}1&0&1\\1&0&0\end{smallmatrix}\right)$.

502. [*M25*] (U. Montanari, 1974.) If (200) makes no change to any relation, prove that the following property holds for every $(s, t) \in R_{ij}$ and every sequence $k_0k_1 \ldots k_r$ of indices with $k_0 = i$, $1 \le k_l \le n$ for $0 < l < r$, and $k_r = j$: There's a sequence of values $x_0x_1 \ldots x_r$ such that $x_0 = s$, $(x_l, x_{l+1}) \in R_{k_l,k_{l+1}}$ for $0 \le l < r$, and $x_r = t$.

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

[This blank page has temporarily been inserted so that the answers will begin on an even-numbered page.]

After [this] way of Solving Questions, a man may steale a Nappe,
and fall to worke again afresh where he left off.
— JOHN AUBREY, *An Idea of Education of Young Gentlemen* (c. 1684)

SECTION 7.2.2.3

1. Only BCAON, BCUOD, BLUED, and SCION satisfy R_1 and R_3; the first two fail R_2.

2. (a) The literals of each clause define the domain of the corresponding variable. If one clause contains x and the other contains $\bar{x}$, forbid the pair $x\bar{x}$. [See H. Bennaceur, *ECAI* **12** (1996), 155–159. Satisfiability/unsatisfiability is preserved, but the number of solutions may change; when $m=1$ the 3SAT problem has 7 solutions, the CSP has 3.]

(b) Seven variables $c_1 \in \{1,2,\bar{3}\}$, …, $c_7 \in \{\bar{3},\bar{4},\bar{1}\}$; $\binom{7}{2}=21$ constraints. Three constraints are satisfied in 6 ways (for example, $c_1c_5 \in \{1\bar{2},13,2\bar{1},23,\bar{3}\bar{1},\bar{3}\bar{2}\}$); the other 18 in 8 ways ($c_1c_7 \in D_1 \times D_7 \setminus 1\bar{1}$). The SAT problem has 2 solutions, the CSP has 48.

(c, d) Adding Boolean variables $\{x_1,x_2,x_3,x_4\}$, we need only 5-out-of-6 constraints such as $c_1x_1 \in \{11,20,21,\bar{3}0,\bar{3}1\}$. [See M. Järvisalo and I. Niemelä, *Workshop on Modelling and Reformulating Constraint Satisfaction Problems* **3** (2004), 111–124.]

3. Let $x_{1\mathtt{B}} = [x_1 = \mathtt{B}]$, etc. Then the clauses $(x_{1\mathtt{B}} \lor x_{1\mathtt{S}})$, $(\overline{x_{1\mathtt{B}}} \lor \overline{x_{1\mathtt{S}}})$, $(x_{2\mathtt{C}} \lor x_{2\mathtt{L}})$, $(\overline{x_{2\mathtt{C}}} \lor \overline{x_{2\mathtt{L}}})$, $(x_{3\mathtt{A}} \lor x_{3\mathtt{I}} \lor x_{3\mathtt{U}})$, $(\overline{x_{3\mathtt{A}}} \lor \overline{x_{3\mathtt{I}}})$, $(\overline{x_{3\mathtt{A}}} \lor \overline{x_{3\mathtt{U}}})$, $(\overline{x_{3\mathtt{I}}} \lor \overline{x_{3\mathtt{U}}})$, $(x_{4\mathtt{E}} \lor x_{4\mathtt{O}})$, $(\overline{x_{4\mathtt{E}}} \lor \overline{x_{4\mathtt{O}}})$, $(x_{5\mathtt{D}} \lor x_{5\mathtt{N}})$, $(\overline{x_{5\mathtt{D}}} \lor \overline{x_{5\mathtt{N}}})$ establish the domains. And the clauses $(R_{11} \lor R_{12} \lor R_{13})$, $(\overline{R}_{11} \lor x_{1\mathtt{B}})$, $(\overline{R}_{11} \lor x_{3\mathtt{A}})$, $(\overline{R}_{11} \lor x_{5\mathtt{N}})$, $(\overline{R}_{12} \lor x_{1\mathtt{B}})$, $(\overline{R}_{12} \lor x_{3\mathtt{U}})$, $(\overline{R}_{12} \lor x_{5\mathtt{D}})$, $(\overline{R}_{13} \lor x_{1\mathtt{S}})$, $(\overline{R}_{13} \lor x_{3\mathtt{I}})$, $(\overline{R}_{13} \lor x_{5\mathtt{N}})$, …, $(\overline{R}_{33} \lor x_{2\mathtt{L}})$, $(\overline{R}_{33} \lor x_{4\mathtt{E}})$, $(\overline{R}_{33} \lor x_{5\mathtt{D}})$ establish the relations.

(Many other encodings are possible; this one is systematic and avoids trickery.)

4. Primary R_1, R_2, R_3; secondary x_1, …, x_5. Options 'R_1 x_1:B x_3:A x_5:N', 'R_1 x_1:B x_3:U x_5:D', 'R_1 x_1:S x_3:I x_5:N', …, 'R_3 x_2:L x_4:E x_5:D'. (See exercise 7.2.2.1–100.)

5. There are just two subsets of $\{\epsilon\}$, namely $\emptyset$ and $\{\epsilon\}$. The first of those relations is always false, so it's a constraint that wipes out all solutions. The second is a tautology, always true; it doesn't really constrain anything. (In general, there are $2^{d_1 \ldots d_k}$ k-ary relations on $(D_1,\ldots,D_k)$, when each D_i has d_i elements; hence there are 2^{d^k} k-ary relations over any d-element set. One of them is always false; another is always true.)

6. Given any binary relation on $A \times B$, consisting of ordered pairs (a,b), math texts say furthermore that the "domain" is the set of left coordinates and the "range" is the set of right coordinates. Yet constraint satisfiers have happily spoken of the domains of variables ever since Mackworth's paper of 1977 introduced the terminology.

Mackworth was influenced by earlier work in computer vision, where the value of a variable was often a rectangle (say) where some object might be found in a digital image; that would be an extramathematical sense of the word "domain," like a "dominion." Moreover, his main focus was on constraints, not variables; the domains of the constraints are the values of the variables.

[Fikes had actually used the term "range," *not* domain, in his original paper of 1970. See also the concept of "range consistency" in exercise ??.]

8. False. For example, (012343434) is a homomorphism from C_9 to C_5. (The most that can be concluded, from the existence of a homomorphism from $C_{\rm odd}$ to G, is that G isn't bipartite, because it contains an odd cycle.)

9. (Solution by P. Jeavons.) Construct a new graph G' by replacing every edge $u \mathbin{—} v$ of G by a path $u \mathbin{—} uv \mathbin{—} vu \mathbin{—} v$, where uv and vu are new vertices. Then there's a homomorphism from G' to C_5 if and only if there's a homomorphism from G to K_5. Hence the problem is NP-complete. (In general the "H-coloring problem," to decide whether

AUBREY
Bennaceur
Järvisalo
Niemelä
tautology
binary relation
Mackworth
computer vision
historical notes
Fikes
Jeavons
NP-complete
H-coloring problem

or not a homomorphism from G to H exists, is trivial when H is bipartite; otherwise it's NP-complete [P. Hell and J. Nešetřil, *J. Comb. Theory* **B48** (1990), 92–110].)

10. (a) Let $\overline{E} = \{\{u,v\} \mid u \neq v \text{ and } \{u,v\} \notin E\}$ be the edges of the complement graph $\overline{G}$. (See Eqs. 7–(15) and 7–(35).) An independent set in G is a clique in $\overline{G}$. "Is there a homomorphism from K_k to $(V,\overline{E})$?"

(b) The vertices *not* in a cover are independent. Use (a) with $k \leftarrow |V| - k$.

(c) They're isomorphic if and only if each is embeddable in the other. It's a *single* GCP if $|V| = |V'|$: "Is there a homomorphism from $(V,E,\overline{E})$ to $(V',E',\overline{E'})$?"

(d) Let G' be the graph on $\{1,\ldots,|V|\}$ for which $i \mathrel{-\!\!-} j$ if and only if $|i-j| \le k$. "Is G embeddable in G'?"

(e) Let A' be the relation $\{(uv, u'v') \mid v = u'\}$ on ordered pairs of vertices in V, and let $(\{0,\ldots,m-1\}, O)$ be the oriented cycle $C_{\vec{m}}$, where $m = |A|$ and $O = \{ij \mid j = (i+1) \bmod m\}$. "Is there a homomorphism from $(\{0,\ldots,m-1\}, O, \neq)$ to $(A, A', \neq)$?"

11. "$u \neq v$ implies $h(u) \parallel h(v)$" is the same as saying that $|V|$ mutually unlike k-tuples satisfy relation R. And that's precisely the kDM problem (k-dimensional matching).

15. Given similar relational structures $S = (U, R_1, \ldots, R_t)$ and $S' = (U', R'_1, \ldots, R'_t)$, the corresponding CSP has variables U, each with domain U'. Suppose $U = \{1,\ldots,n\}$. The values $x_{i_1} \ldots x_{i_k}$ of every k-tuple $i_1 \ldots i_k \in R_j$, where $k = k_j$, are constrained to satisfy the relation R'_j, for $1 \le j \le t$.

18. (a) Let T be the matrix $\binom{wz^- \;\; w^-z}{wz \;\; w^-z^-}$, where z^- denotes $1/z$. By induction we have $G_N(z) = {}^{(w\;w^-)}T^{N-1}\binom{1}{1}$. For example, $G_1(z) = w^- {+} w$ and $G_2(z) = w^{-2}z^- {+} 2z {+} w^2 z^-$.

Now let $u = (\frac{w+w^-}{2})/z$, $v = ((\frac{w-w^-}{2})/z)^2 + z^2$, $\lambda = u + \sqrt{v}$, $\mu = u - \sqrt{v}$. Then we have $T = S\binom{\lambda\;0}{0\;\mu}S^-$, where $S = \Bigl(\begin{smallmatrix}\frac{\lambda - w^-z^-}{wz} & \frac{\mu - w^-z^-}{wz}\end{smallmatrix}\Bigr)$. Hence $G_N(z) = a\lambda^{N-1} + b\mu^{N-1}$, with coefficients $a = \lambda z + (z - z^3)/\sqrt{v}$, $b = \mu z - (z - z^3)/\sqrt{v}$. (Notice that when $B = 0$, everything simplifies enormously because $w = 1$. For example, $\lambda = z^- + z$.)

(b) Differentiate and plug in. (The exact formulas are hairy, until we get to (c).)

(c) When N is large we can ignore μ. Thus $G'(z)/G(z)$ in (b) is $\frac{d}{dz}\ln G(z) \sim \frac{d}{dz}N\ln\lambda$, where $\lambda = e^\beta\cosh\beta B + \sqrt{e^{2\beta}\sinh^2\beta B + e^{-2\beta}}$.

(d) Now we have $G_k(z) = {}^{(w\;w^-)}T^{k-1}XT^{N-k}\binom{1}{1}$, where $X = \binom{-1\;0}{0\;1}$. To put this in closed form, let $Y = S^-XS$, so that $T^{k-1}XT^{N-k} = S\binom{\lambda\;0}{0\;\mu}^{k-1}Y\binom{\lambda\;0}{0\;\mu}^{N-k}S^-$. Hence $G_k(z) = \hat{a}\lambda^{N-1} + \hat{b}\mu^{N-1} + c\lambda^{N-k}\mu^{k-1} + c\lambda^{k-1}\mu^{N-k}$, where $\hat{a} = (\frac{w-w^-}{2z})a/\sqrt{v}$, $\hat{b} = (\frac{w^- - w}{2z})b/\sqrt{v}$, $c = \frac{w-w^-}{2}(1-z^2)/v$. So the average comes to $(\hat{a} + c\lambda/(N\sqrt{v}\,))\lambda^{N-1} + (\hat{b} - c\mu/(N\sqrt{v}\,))\mu^{N-1}$, divided by $G(z)$; asymptotically, it's $\sinh\beta B/\sqrt{\sinh^2\beta B + e^{-4\beta}}$.

[This answer is based on §2.5.1 of the book by Mézard and Montanari.]

20. It turns out that 17 constraints like (12) are sufficient to force $x_i \neq x_j$ whenever $i \neq j$. (The problem without (14) is in fact equivalent to "radio coloring" as in exercise 7.2.2.2–36; the graph in (11) can't be 7-colored radiowise.) But the second model, with only 7 constraints like (15), has 20,358 solutions without the all-different constraint! We can, for instance, set $\mathtt{A} \leftarrow \mathtt{C} \leftarrow \mathtt{E} \leftarrow \mathtt{G} \leftarrow 1$ and $\mathtt{B} \leftarrow \mathtt{D} \leftarrow \mathtt{F} \leftarrow \mathtt{H} \leftarrow 8$.

[The inventor of this puzzle is unknown. After Martin Gardner publicized it in *Scientific American* **206**, 2 (February 1962), 150, Fred Gruenberger told him that he'd learned of the problem in 1961 from a friend at Walt Disney Studios, "where it had already consumed a fair amount of Mr. Disney's staff time." Gruenberger had used it that year in a TV documentary, "How a Digital Computer Works," featuring three high-school students who solved it from scratch in about five minutes, working at a blackboard, while a computer would supposedly have to run through 8! = 40320

Hell
Nešetřil
complement graph
clique
embeddable
oriented cycle
$C_{\vec{m}}$
kDM problem
k-dimensional matching
3DM
Mézard
Montanari
radio coloring
Gardner
Gruenberger
historical notes
Disney

permutations in order to find the answer! Ten years later, D. K. Cohoon called (11) the "no-touch puzzle" in *Math. Mag.* **45** (1972), 261–265, without mentioning his source.]

Notice that the CSP model using (15) is essentially based on the *complement* of the graph in (11), which has only 11 edges and is easy to draw. According to that model, the problem is to make $(\mathtt{A}, \mathtt{B}, \ldots, \mathtt{H})$ label a *Hamiltonian path* in the complement graph — an observation made independently by T. H. O'Beirne and H. Koplowitz in letters to Gardner, and later by Cohoon. There are four such paths, easy to find.

21. We can save a factor of 2 by assuming that A occurs in the left half of the graph: Remove A from the domains of $\{x_2, x_5, x_6, x_8\}$ in the first model; remove $\{2, 5, 6, 8\}$ from the domain of A in the second.

To save another factor of 2, we can add the constraint $x_2 < x_8$ (say) in the first model. That can't be done in the second, without probing deeper into the solution.

22. Let there be $17 \cdot 7$ secondary items juv, one for every combination of a letter j with $\mathtt{A} \le j < \mathtt{H}$ and an edge $u — v$, where $u < v$. There are 64 options (v, k), where $1 \le v \le 8$ and $\mathtt{A} \le k \le \mathtt{H}$; option (v, k) contains the primary items v and k, meaning that vertex v is labeled with letter k. To prevent adjacent letters in edge $u — v$, add secondary item juv to options (u, j), (v, j), $(u, j{+}1)$, and $(v, j{+}1)$. For example, option $(2, \mathtt{E})$ is '2 E D12 D24 D25 D26 E12 E24 E25 E26'. (This construction nicely incorporates both of the text's CSP models; notice that the all-different constraint "comes for free.")

That XC problem has 4 solutions, found in 300 kilomems with 485 nodes in the search tree. To break the symmetry as in exercise 21, first remove options $(2, \mathtt{A})$, $(5, \mathtt{A})$, $(6, \mathtt{A})$, $(8, \mathtt{A})$; then also remove options $(2, \mathtt{H})$ and $(8, \mathtt{B})$, and use the pairwise ordering trick of exercise 7.2.2.1–20 with $m = 6$, $\alpha_i = (2, \mathtt{B} + i)$, $\beta_i = (8, \mathtt{C} + i)$ to ensure that the label of 2 is less than the label of 8. (This introduces secondary items $y_1, \ldots, y_5$; it also puts y_2 and y_3 into option $(2, \mathtt{E})$.) The resulting XC problem has 1 solution, costs 108 kilomems, and examines 146 nodes. [If we cleverly change 5 to #5 and use the sharp preference heuristic of exercise 7.2.2.1–10, thereby forcing the first branch to be on vertex 5, the search tree decreases to just 43 nodes and the running time to just 35 Kμ.]

23. Let variables (AB, BC, CD, DE, EF, FG, GH) each have the 11-element domain of all edges not in the graph. Constrain each of (AB, BC), …, (FG, GH) to be one of the 48 ordered pairs of edges that have one vertex in common. Also constrain each of the nonoverlapping pairs of variables, namely (AB, CD), (AB, DE), …, (EF, GH), to be one of the other 62 ordered pairs of edges. (The all-different constraint would be redundant.)

26. FABABACDCE (and its mirror image ECDCABABAF).

27. The mirror image of a solution with $f \ge 5$ has $f < 5$. (Alternatively, we could have assumed that $d < 5$, or $e < 5$, or even that $a_1 < 5$; but F is probably harder to place. When t is even, the symmetry can be broken by choosing *any* model of odd multiplicity, and requiring more than half of its occurrences to be $< t/2$.)

28. (Solution by B. C. Dull.) No. If that new constraint is violated, so is (18) when $l = l' + l''$, because we have $f_{0k} + f_{1k} + \cdots + f_{(l'q_k - 1)k} \le l' p_k$ by (17).

But that "solution" is *wrong*! The new constraints *are* useful, for example, when $l'' = 0$ and we have a partial solution for which f_{ik} is known only when $i > t/2$.

30. Introduce a primary item, representing slot i, for $0 \le i < t$. Also a primary item for the name of each model type, with its given multiplicity. (In Fig. 100, for example, item A has multiplicity 3.) There will be one option for each slot and each type.

To implement the constraints (17), introduce primary items u_{jk} for $0 \le j \le t - q_k$ and $0 \le k < m$, having multiplicity $[0\,.\,.\,p_k]$. (If $p_k = 1$, this item could be secondary.)

Cohoon
no-touch puzzle
Hamiltonian path
O'Beirne
Koplowitz
symmetry
pairwise ordering trick
XC: exact cover
sharp preference heuristic
Kμ: kilomems
symmetry
Dull
slot

Include u_{jk} in the option for every model that uses feature k in slot i, for $j \le i < j+q_k$. (Thus, one option for Fig. 100 is '2 B u_{10} u_{20} u_{03} u_{13} u_{23}'.)

31. Notice that $f_{0k}+\cdots+f_{(t-lq_k-1)k} \ge r_k - lp_k$ if and only if $\bar f_{0k}+\cdots+\bar f_{(t-lq_k-1)k} \le s_{lk} = t - lq_k - r_k + lp_k$. Therefore introduce primary items v_{lk} for $0 < l < \lceil r_k/p_k \rceil$ and $0 \le k < m$, having multiplicity $[0\,..\,s_{lk}]$. Include v_{lk} in the option for every model that does *not* use feature k, for every slot i in the range $0 \le i < t - lq_k$. (If $s_{lk} = 0$, any options that would include v_{lk} should be omitted, like the options for 0 B and 0 D in Fig. 100. The option in answer 30 becomes '2 B u_{10} u_{20} v_{41} v_{71} v_{72} u_{03} u_{13} u_{23}'. Other redundant constraints such as those of exercise 28 can be implemented in a similar way.)

32. Yes: The only solutions are FEBAGAHDCAGECDACDCEGACDHAGABEF and its mirror image (change 'AC' to 'CA' in the middle). The running time is (a) 28 gigamems, with 22 meganodes in the search tree; (b) 4 megamems, with 1670 nodes.

33. No; Algorithm 7.2.2.1M verifies this in 202 Gμ, with 158 meganodes.

There's actually an easy way to prove the impossibility by hand, because Model F can only appear at the beginning, or at the end, or next to Model 0; furthermore F0F is impossible. Hence the shortest possible way to produce four Model Fs is to put one at each end and to have two occurrences of F0 or 0F inside the sequence.

One way to solve the 62-car problem is to place '00' between two solutions of the 30-car problem. That 62-car problem actually has 19050 solutions, of which 18 are unchanged under left-right reflection and the others form 9516 mirror pairs. Only 69 Gμ of computation are needed to find the symmetric ones. Every solution begins with FEBA and ends with ABEF. Six of the palindromic solutions, such as FEBAF0HDCAGECDC-AGEBAGAHDCAGECDAADCEGACDH ... GACDH0FABEF, have two F's near each end.

35. (a) We've seen equivalent problems before (for example, in Sections 5.4.2, 7.2.1.1, and 7.2.1.7); but let's start from scratch. Consider the digraph whose vertices are the q-bit patterns α with $\nu\alpha \le p$, having arcs $\alpha \longrightarrow \beta$ when the last $q-1$ bits of α match the first $q-1$ bits of β. (It's a subgraph of the digraph in exercise 2.3.4.2–23.) The answer is the number of walks of length 10 that start from vertex 0^q in this digraph: 144 when $(p,q)=(1,2)$; 60 when $(p,q)=(1,3)$; 504 when $(p,q)=(2,3)$.

(b) $p\lfloor n/q \rfloor + \min(p, n \bmod q)$.

(c) In general, the generating function $G(z)$ for walks of length n from vertex σ in a given digraph is $\sum_\alpha G^\alpha(z)$, where $G^\alpha(z) = [\alpha=\sigma] + z\sum_{\beta\to\alpha} G^\beta(z)$ for each vertex α. For example, when $(p,q)=(1,2)$ and $\sigma = 00$ we have $G(z) = G^{00}(z)+G^{01}(z)+G^{10}(z)$; $G^{00}(z) = 1+z(G^{00}(z)+G^{10}(z))$; $G^{01}(z) = z(G^{00}(z)+G^{10}(z))$; $G^{10}(z) = zG^{01}(z)$; hence $G_{12}(z) = G(z) = (1+z)/(1-z-z^2)$. (They're Fibonacci numbers: $C_{12n} = F_{n+2}$.)

Similarly $G_{13}(z) = (1+z+z^2)/(1-z-z^3)$ (Narayana numbers); $G_{23}(z) = (1+z+z^2)/(1-z-z^2-z^3)$ (Tribonacci numbers). In general, $G_{1q}(z) = (1+z+\cdots+z^{q-1})/(1-z-z^q)$ and $G_{(q-1)q}(z) = (1+z+\cdots+z^{q-1})/(1-z-z^2-\cdots-z^q)$. But the other cases don't fit any evident pattern: $G_{24}(z) = (1+z+z^2+z^3-z^4-z^5)/(1-z-z^2-z^4+z^6)$; $G_{25}(z) = (1+z+2z^2+2z^3+2z^4-z^5-z^6-2z^7-z^8-z^9)/(1-z-z^3-2z^5+z^8+z^{10})$; $G_{35}(z) = (1+z+z^2+2z^3+2z^4-z^5-z^6-z^8-z^9)/(1-z-z^2-z^4-2z^5+z^7+z^{10})$.

36. (a) Given a plane partition whose elements P_{ij} satisfy $0 \le P_{ij} \le m$, $P_{ij} \ge P_{i(j+1)}$, $P_{ij} \ge P_{(i+1)j}$, and $P_{ij} = 0$ for $i > p$ or $j > q-p$, construct an extreme (p/q)-string as follows: For $k = 1, 2, \ldots, m$, form the tableau shape whose boxes are the elements with $P_{ij} \ge k$, and write down its *rim representation*, as in 7.2.1.4–(13) and (14). (This will be a binary string of length q that contains exactly p 1s.)

For example, suppose $p=2$, $q=5$, $m=6$, and consider the plane partition $\substack{441\\210}$. The rim representations for $k = 1, 2, 3, 4, 5, 6$ are respectively 10100, 01010, 01001,

palindromic
walks
digraph
Fibonacci numbers
Narayana numbers
Tribonacci numbers
tableau shape
rim representation

01001, 00011, 00011; and the concatenation of those strings is extreme. (This beautiful construction, devised by Ira Gessel in March 2020, is clearly reversible.)

(b) Let $r = n \bmod q$. Then c_{pqn} is $e_{(p-r)(q-r)}(\lfloor n/q\rfloor)$, if $r < p$; 1, if $r = p$; $e_{pr}(\lceil n/q\rceil)$, if $r > p$.

39. Each point (x, y, z) satisfies three equations in three unknowns, so the respective vertices are $((-140, 0, 0), (75, 75, -100), (0, 252, -280), (40, -100, -200), (90, -50, 0), (140, 50, 0), (-240, 0, 200), (140, 0, 0), (240, 0, 200), (-140, -50, 0), (-90, 50, 0), (-40, 100, -200), (0, -252, -280), (-75, -75, -100))$. Then the seven hexagons 023 — 310 — 501 — 054 — 460 — 206, 134 — 421 — 612 — 165 — 501 — 310, ..., 612 — 206 — 460 — 643 — 356 — 165 do the job, because we can construct a model (with stiff paper or computer graphics). [*Structural Topology* #13 (1986), 69–80.]

40. The simplest example whose histoscape is *not* a 3VP is the identity matrix $\binom{1\,0}{0\,1}$, because more than three edges (in fact, five of them) touch vertex $(1, 1, 1)$. Moreover, the edge from $(1, 1, 1)$ to $(1, 1, 0)$ is adjacent to *four* faces! [*Beware:* The standard row-and-column convention for coordinates ij of a matrix are sometimes confusingly at odds with the standard Cartesian coordinates (x, y, z) of three-dimensional geometry.]

In general, consider the histoscape for $\binom{a\,b}{c\,d}$ when $a = \max\{a, b, c, d\}$ and $b \ge c$. It fails to be 3VP when $d > b$, because the cubies $(0, 0, b)$ and $(1, 1, b)$ have a boundary edge in common. A milder violation occurs when $a > b$ and $c > d$, because four faces meet at vertex $(1, 1, b)$. Four faces meet at that vertex also when $a > b = d > c$.

But the other cases are fine: *Case 1*, $a = b = c \ge d$. *Case 2*, $a = b > \max\{c, d\}$. *Case 3*, $a > b = c = d$. *Case 4*, $a > b > d \ge c$. When we take symmetry into account, these cases contribute respectively $(\binom{10}{1}+4\binom{10}{2}, 4\binom{10}{2}+8\binom{10}{3}, 4\binom{10}{2}, 8\binom{10}{3}+8\binom{10}{4})$ valid 3VPs, a total of 4150.

(And the B^4 histoscapes of 2×2 matrices with $a_{ij} < B$ yield $B^4/3+O(B^3)$ 3VPs.)

41. An $m \times n$ histoscape is a 3VP if and only if $r(a_{(i-1)(j-1)}, a_{(i-1)j}, a_{i(j-1)}, a_{ij})$ holds for $1 \le i < m$ and $1 \le j < n$, where r is the relation in the previous answer, because the vertices (x, y, z) for which $x = i$ and $y = j$ depend only on those four matrix entries.

The best way to enumerate the solutions to a CSP whose relations are enforced in such a structured manner is to use the techniques of "dynamic programming," which is the topic of Section 7.7. This problem offers us a nice preview of those coming attractions, because the following remarkable algorithm finds the total number of $m\times n$ matrices whose 2×2 submatrices all satisfy an *arbitrary* quaternary relation r. We assume that each variable has the domain $0 \le a_{ij} < t$; and we use an $(n+1)$-dimensional array of t^{n+1} potentially large integers $c(x_0, \ldots, x_n)$, all initially 1.

Q1. [Iterate on rows.] Do step Q2 for $i = 1, \ldots, m-1$; then go to Q3.

Q2. [Iterate on columns.] Do subroutine (i, j) below for $j = 1, \ldots, n-[i = m-1]$.

Q3. [Sum.] The answer is $\sum\{c(x_0, \ldots, x_n) \mid 0 \le x_0, \ldots, x_n < t\}$. ▌

Subroutine (i, j) is the following: Set $q \leftarrow (j-i) \bmod (n+1)$. For all t^n choices of $(x_0, \ldots, x_n)$ such that $x_q = 0$, compute t sums for $0 \le d < t$, namely

$$s_d \leftarrow \sum_{0\le k<t} \bigl[r_{ij}(k, x_{(q+1) \bmod (n+1)}, x_{(q-1) \bmod (n+1)}, d)\bigr]c(x_0, \ldots, x_{q-1}, k, x_{q+1}, \ldots, x_n);$$

then set $c(x_0, \ldots, x_{q-1}, d, x_{q+1}, \ldots, x_n) \leftarrow s_d$ for $0 \le d < t$. (Notice that this computation is rather similar to the discrete Fourier transform in Eq. 4.6.4–(40).)

Gessel
coordinates
matrix coordinates
Cartesian coordinates
structured
dynamic programming
quaternary relation
discrete Fourier transform
Fourier transform

The relation r_{ij} in the formula for s_d is r when $j < n$; but r_{ij} is the universal relation (always true) when $j = n$. (One could in fact let r_{ij} be a different quaternary relation for each (i, j), where r_{in} constrains the joint values of $(a_{(i-1)(n-1)}, a_{i0}, a_{i(n-1)}, a_{(i+1)0})$. Imagine the $2 \times (m-1)n$ matrix $\left(\begin{smallmatrix} a_{00}a_{01}\ldots a_{0(n-1)}a_{10}\ldots a_{1(n-1)}a_{20}\ldots \\ a_{10}a_{11}\ldots a_{1(n-1)}a_{20}\ldots a_{2(n-1)}a_{30}\ldots \end{smallmatrix}\right)$!)

The method works because, when subroutine (i, j) begins, $c(x_0, \ldots, x_n)$ is the number of ways to set the initial matrix entries $a_{i'j'}$, for (i', j') lexicographically less than (i, j), so that all constraints on those variables are satisfied and

$$(a_{(i-1)(j-1)}, \ldots, a_{(i-1)(n-1)}, a_{i0}, \ldots, a_{i(j-1)}) = (x_q, x_{q+1}, \ldots, x_n, x_0, \ldots, x_{q-1}).$$

About 1.8 teramems of computation suffice to show that the desired number of 8×8 matrices is 1,927,084,607,409,168,698,157,388,476,170,741,096,757,035,906,066. (Those "mems" were however longer than usual, because 24 gigabytes of memory were needed.)

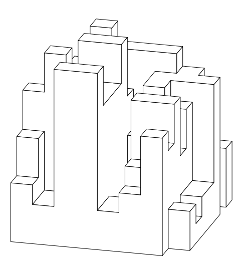
42. We essentially want to run that algorithm in reverse. To reverse step Q3, let the counts $c(x_0, \ldots, x_n)$ be renamed c_j for $0 \le j < t^{n+1}$, in any convenient way. Then for $j = 0, 1, \ldots,$ set $k \leftarrow k - c_j$ if $k \ge c_j$; but stop when $k < c_j$. That gives us suitable values of $(x_0, x_1, \ldots, x_n)$, which will be $(a_{(m-2)(n-1)}, a_{(m-1)0}, \ldots, a_{(m-1)(n-1)})$. And we'll want the kth solution for which those $n + 1$ values are prespecified.

Similarly, we can run subroutine (i, j) in reverse, if we're given the t^{n+1} counts that it ends with, because each of those counts was obtained as the sum of at most n counts c_j whose sum exceeds k. That will give us enough information to determine $a_{(i-1)(j-1)}$, as well as a new value of k. The remaining problem is then to find the kth solution when the final $(m - i + 1)n - j - 1$ elements are given.

We must rerun the algorithm for each $(i, j) = (m - 1, n - 1), (m - 1, n - 2), \ldots, (1, 1)$, because the previous counts have been discarded. However, we can save time by cleverly omitting the computation of counts that won't contribute to solutions having the prespecified final elements. (See the author's program HISTOSCAPE-UNRANK.)

The "random" 8×8 solution shown here was found by setting $k \leftarrow N/\phi$, where N is the total number of solutions. (It can be fabricated from sugar cubes.)

$$\begin{pmatrix} 5&4&2&3&2&2&6&3 \\ 6&7&9&8&8&8&7&0 \\ 5&1&1&6&4&7&7&7 \\ 5&3&9&9&1&7&6&2 \\ 5&4&5&7&1&1&4&2 \\ 7&9&6&7&1&7&7&1 \\ 5&2&5&7&2&4&5&2 \\ 3&2&9&9&2&3&6&0 \end{pmatrix}$$



Incidentally, this histoscape has 184 vertices and 94 faces. Only 89 of the vertices are visible in this particular view, and only 48 of the faces are at least partly visible. There are 35 T junctions, 24 V junctions, 42 W junctions, and 23 Y junctions. When half edges are forced at the boundary, the line labeling problem has six solutions, because of two independent ambiguities in the "central canyon"; all but four labels are forced.

43. It's convenient to use the even/odd coordinate system of exercise 7.2.2.1–145, with cubie (i, j, k) represented by $(2i + 1, 2j + 1, 2k + 1)$. In the following description we shall use the notation $\bar{k}$ to stand for $k \bmod 2$. Assume that $a_{ij} < t$ for all i and j, and set up a $(2m + 1) \times (2n + 1) \times (2t + 1)$ array b, initially zero.

universal relation
lexicographically less
mems
author
downloadable programs
golden ratio
junctions
half edges
boundary
line labeling
even/odd coordinate system

square torus
torus
vortex
dynamic programming
all different
inversions
pi, as random example

First, mark all the cubies, by setting $b_{(2i+1)(2j+1)(2k+1)} \leftarrow 1$ for $0 \le k < a_{ij}$.

Second, mark all the "visible" faces of cubies, by doing the following for all (i,j,k) with $\bar\imath\bar\jmath\bar k = 111$ and $b_{ijk} = 1$: If $b_{(i\pm 2)jk} = 0$, set $b_{(i\pm 1)jk} \leftarrow 1$; if $b_{i(j\pm 2)k} = 0$, set $b_{i(j\pm 1)k} \leftarrow 1$; if $b_{ij(k\pm 2)} = 0$, set $b_{ij(k\pm 1)} \leftarrow 1$. (We assume that $b_{ijk} = 0$ whenever $i < 0$ or $j < 0$ or $k < 0$ or $i > 2m$ or $j > 2n$ or $k > 2t$.)

Third, to mark all the "visible" edges, do the following for all (i,j,k) with $\bar\imath\bar\jmath\bar k = 011$ and $b_{ijk} = 1$: If $b_{i(j\pm 2)k} = 0$, set $b_{i(j\pm 1)k} \leftarrow 1$; if $b_{ij(k\pm 2)} = 0$, set $b_{ij(k\pm 1)} \leftarrow 1$. Also do this, for all (i,j,k) with $\bar\imath\bar\jmath\bar k = 101$ and $b_{ijk} = 1$: If $b_{(i\pm 2)jk} = 0$, set $b_{(i\pm 1)jk} \leftarrow 1$.

Fourth, mark all the vertices, by doing the following for all (i,j,k) with $\bar\imath\bar\jmath\bar k = 001$ and $b_{ijk} = 1$: If $b_{ij(k\pm 2)} = 0$, set $b_{ij(k\pm 1)} \leftarrow 1$.

Finally, now that we know the vertices, we're ready to output the face polygons (some of which might be "holes" enclosed in a larger polygon). Every vertex will be part of three polygons, one with constant i, another with constant j, another with constant k. All three cases are similar; the polygon with constant i can be found as follows, starting at ijk where $\bar\imath\bar\jmath\bar k = 000$: "While $b_{ijk} = 1$, do a j-step and a k-step." A j-step means, "Output vertex $(i/2)(j/2)(k/2)$; set $b_{ijk} \leftarrow 2$; set $\delta \leftarrow 2$ if $b_{i(j+1)k} > 0$, otherwise $\delta \leftarrow -2$; repeat $j \leftarrow j + \delta$ until $b_{ijk} > 0$." A k-step is similar. (The polygon will have an even number of vertices, because we alternate j-steps with k-steps.) After all faces with constant i have been output, all vertices will have $b_{ijk} = 2$.

For example, consider the histoscape for $\left(\begin{smallmatrix}1&1&1\\1&0&1\\1&1&1\end{smallmatrix}\right)$. It has 16 vertices: 000, 001, 030, 031, 110, 111, 120, 121, 210, 211, 220, 221, 300, 301, 330, 331. Its i-face polygons are 000 — 030 — 031 — 001 — 000, 110 — 120 — 121 — 111 — 110, 210 — 220 — 221 — 211 — 210, 300 — 330 — 331 — 301 — 300; its j-face polygons are 000 — 001 — 301 — 300 — 000, 030 — 031 — 331 — 330 — 030, 110 — 111 — 211 — 210 — 110, 120 — 121 — 221 — 220 — 120; its k-face polygons are 000 — 300 — 330 — 030 — 000, 001 — 301 — 331 — 031 — 001, 110 — 210 — 220 — 120 — 110, 111 — 211 — 221 — 121 — 111. It looks like a square torus.

44. (a) Swap 14 with 15.

(b) Swapping adjacent elements of a vortex changes it to a non-vortex. (Moreover, the 2×2 matrix $\left(\begin{smallmatrix}a&b\\c&d\end{smallmatrix}\right)$ is a vortex if and only if $[a<b]+[b<d]+[d<c]+[c<a]$ is odd.)

(c) First row $(1,\ldots,n)$, second row $(2n,\ldots,n+1)$, and so on.

(d) True, by answer 40 (case 4).

(e) It suffices to verify this for 2×2 matrices, when it's clearly true.

45. Let $r_{ij}(w,x,y,z)$ be any 4-ary relation that depends only on the relative order of four distinct elements $\{w,x,y,z\}$. (There are 2^{24} such relations.) We can enumerate all $m \times n$ matrices whose elements are a permutation of $\{0,1,\ldots,mn-1\}$ and whose 2×2 submatrices satisfy $r_{ij}(a_{(i-1)(j-1)}, a_{(i-1)j}, a_{i(j-1)}, a_{ij})$, with a dynamic programming algorithm structured as the method of answer 41. But this time we need counts $c(x_{n-t},\ldots,x_t)$ for each of the $t^{\underline{n+1}}$ choices of *distinct* elements with $0 \le x_{n-t},\ldots,x_t < t$, where $t = in + j$ when starting subroutine (i,j) and $t = in + j + 1$ when finishing. (For example, when $m = n = 5$, the number of counts is only $13^{\underline{6}} = 1235520$ when $(i,j) = (2,2)$, but it rises to $25^{\underline{6}} = 127512000$ during the last round when $(i,j) = (4,4)$.)

Two ideas make it possible to represent these numerous counts efficiently in memory. Count $c(x_{n-t},\ldots,x_t)$ is the number of partial solutions $x_0 \ldots x_t$ whose final $n+1$ elements are $x_{n-t} \ldots x_t$. Those counts can be represented by $y_{n-t} \ldots y_t$, where y_j is x_j minus the number of elements "inverted" by x_j (namely the smaller elements to its right, as in Section 5.1.1). For example, if $n = 3$ and $t = 8$, the final four elements of a permutation $x_0 \ldots x_8$ might be $x_5x_6x_7x_8 = 3142$; we represent them by $y_5y_6y_7y_8 = 1132$. Or, going the other way, if $y_5y_6y_7y_8 = 3141$, then $x_5x_6x_7x_8$ must have been 6251. This representation has the nice property that $0 \le y_j \le j$ for $n-t \le j \le t$, so there clearly are $t^{\underline{n+1}}$ possibilities.

Every permutation $x_0 \ldots x_t$ of $\{0,\ldots,t\}$ yields $t+2$ permutations $x'_0 \ldots x'_{t+1}$ of $\{0,\ldots,t+1\}$, if we choose x'_{t+1} arbitrarily and then set $x'_j \leftarrow x_j + [x_j \ge x'_{t+1}]$. For example, if $t=8$ and $x_5x_6x_7x_8 = 3142$, the ten permutations obtained from $x_0 \ldots x_8$ will have $x'_5x'_6x'_7x'_8x'_9 = 42530$, 42531, 41532, 41523, 31524, 31425, 31426, 31427, 31428, or 31429. And the representations $y'_5y'_6y'_7y'_8y'_9$ of those last five elements will simply be respectively 31420, 31421, ..., 31429! In general, we'll have $y'_j = y_j$ for $0 \le j \le t$, and $y'_{t+1} = x'_{t+1}$ will be arbitrary; this inversion-oriented representation works beautifully.

Furthermore, there's a beautiful way to arrange the counts in memory, so that subroutine (i,j) doesn't clobber any of the existing counts when it updates t to $t+1$. These details are all worked out in the author's program WHIRLPOOL-COUNT (online).

The answer to the stated problem is 2,179,875,344,187,129,600 (found in 10 Gμ).

46. (a) If $n>0$, $2Q_n = 2nU_n$ is the number of permutations $a_0 \ldots a_{2n-1}$ for which $a_{2k-1} < a_{2k} \iff a_{2k} < a_{2k+1}$. Hence Q_n counts those which also have $a_0 < a_1$. The permutations enumerated by U_{n+1} have the form $a_1 \ldots a_{2k}(2n+1)a_{2k+1}\ldots a_{2n}$, for some k, where $a_1 \ldots a_{2k}$ and $a_{2k+1}\ldots a_{2n}$ are independently counted by Q_k and Q_{n-k}.

(b) Hence $U'(z) = Q(z)^2$, where $Q(z) = 1+U_1z^2/2!+2U_2z^4/4!+\cdots = 1+zU(z)/2$. The solution to this differential equation, with $U(0)=0$, turns out to be slightly scary: $U(z) = \sqrt{2}\tanh(z/\sqrt{2})/(1-(z/\sqrt{2})\tanh(z/\sqrt{2}))$.

[Let $p_n(k)$ be the number of up-up-or-down-down permutations of the $2n+1$ numbers $\{-n,\ldots,0,\ldots,n\}$ that begin with k. For example, the values $(p_n(-n),\ldots,p_n(n))$ for $1 \le n \le 3$ are $(1,0,1)$; $(4,2,2,2,4)$; $(42,28,22,20,22,28,42)$. Ira Gessel has discovered a surprisingly simple formula for the bivariate exponential generating function

$$\sum_{m,n} p_{(m+n)/2}\left(\tfrac{m-n}{2}\right)\frac{w^m}{m!}\frac{z^n}{n!} = \frac{\cosh((w-z)/\sqrt{2})}{\cosh((w+z)/\sqrt{2}) - ((w+z)/\sqrt{2})\sinh((w+z)/\sqrt{2})}.$$

[To appear (2020); he used exercise 7.2.2.2–333.] One can also show that these curious numbers satisfy the unusual recurrence relation $p_{n+1}(k) = \sum_{j=-n}^{n} |j-k|\,p_n(j)$.]

(c) Let $V(z) = 1/(1-z\tanh z) = 1+V_1z^2/1!+V_2z^4/3!+\cdots$, where $V_n = 2^{n-1}U_n$, and let μ be the positive number that satisfies $\mu\tanh\mu = 1$. We have $z\tanh z = \sum_{k=0}^{\infty} c_k(z-\mu)^k$ when z is near μ, where $c_0 = \mu\tanh\mu = 1$, $c_1 = \mu+\tanh\mu-\mu\tanh^2\mu = \mu$, and $c_2 = 1-\mu\tanh\mu-\tanh^2\mu+\mu\tanh^3\mu = 0$. The only other root of $z\tanh z = 1$ for $|z| \le 2\mu$ is $z = -\mu$. Hence the function $V(z) - 2/(\mu^2(\mu^2-z^2))$ is analytic in $|z| \le 2\mu$; and we have $U_n/(2n-1)! = 2^{1-n}V_n/(2n-1)! = 2^{2-n}/\mu^{2n+2} + O(1/(2\mu)^{2n})$.

The constant μ is a well-studied number called the dual Laplace limit,

$$\mu = 1.19967\,86402\,57733\,83391\,63698\,48641\,14194\,42615-;$$

the even more famous Laplace limit constant $\sqrt{\mu^2-1}$ is

$$\lambda = 0.66274\,34193\,49181\,58097\,47420\,97109\,25290\,70562+.$$

[*Historical notes:* See P. S. Laplace, *Connaissance des Tems de 1828* (1825), 311–321, who thought the value was 0.66195. Cauchy published the correct value of λ to five decimals in an important memoir of 1831, which laid the foundations of complex variable theory; see his *Œuvres complètes* (2) **12** (1916), 101, where he also computed μ and μ^2.]

To get further accuracy, Philippe Jacquet observes that there are constants μ_k with $\mu_k\tan\mu_k = -1$ and $(k-.5)\pi < \mu_k < k\pi$, for all $k \ge 1$; for example, $\mu_1 \approx 2.79839$.

author
downloadable programs
differential equation
Gessel
bivariate exponential generating function
generating function
recurrence relation
dual Laplace limit
fundamental constants
Laplace limit constant
Historical notes
Laplace
Cauchy
complex variable theory
Jacquet

Thus $z = \pm i\mu_k$ is another root of $z \tanh z = 1$ and another pole of the meromorphic function $V(z)$. (Apparently these, together with $z = \pm\mu$, are the *only* poles.)

(d) See the author's note "Whirlpool permutations" (May 2020), available online.

47. To formulate an $m \times n$ whirlpool puzzle as a CSP, there's one variable x_{ij} for each empty cell, having as domain the numbers not yet present; those variables must be all different. Also introduce redundant variables r_{ij} for $0 \le i < m$ and $1 \le j < n$, with binary domains $\{<, >\}$, constrained to describe the result of comparing $x_{i(j-1)} : x_{ij}$. Similarly, c_{ij} describes $x_{(i-1)j} : x_{ij}$, for $1 \le i < m$ and $0 \le j < n$. Finally we constrain $(r_{ij}, c_{ij}, r_{(i-1)j}, c_{i(j-1)})$ to yield a vortex, for $1 \le i < m$ and $1 \le j < n$.

(This setup is easily expressed as an XCC problem. For example, puzzle (iv) has 72 primary items, 44 secondary items, and 1808 options; it is solved in 800 kilomems.)

Puzzles (i) and (iv) have unique solutions. But puzzle (ii) has none; indeed, two entries are required to be 4. Puzzle (iii) has two solutions (one can swap $7 \leftrightarrow 8$).

(i)

1	3	5	7	9
17	16	15	14	13
23	24	25	11	12
22	21	20	19	18
2	4	6	8	10

; (ii)

3	14	15	9	2
4?	6		5	4?

; (iii)

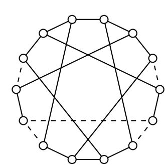
3	14	15	12	13
7	9	2	6	16
5	20	24	23	17
4	1	25	22	18
8	10	11	21	19

; (iv)

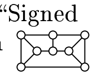
3	13	14	23	15
9	12	2	24	6
10	11	5	8	7
1	20	21	16	25
4	19	18	17	22

.

50. Start with a tetrahedron, and introduce a "crease" in one of its faces, either concave (◬) or convex (◬). That gives us an object with six vertices, nine edges, two triangular faces, and three quadrilateral faces. Now crease a quadrilateral face, between the two triangular faces; that gives us six quadrilateral faces and the desired skeleton:



51. (We've seen this graph before in 7–(57). It's called the Heawood graph, after its discovery by P. J. Heawood [*Quarterly Journal of Pure and Applied Mathematics* **24** (1890), 332–338 and fig. 16 following 386], and it has 336 automorphisms. At present this is its only known signed skeleton that is realizable as a 3VP, up to automorphism.)

52. Partial results on small graphs are discussed in the author's online note "Signed skeletons" (April 2020). For example, 13 signed realizations of the 8-vertex graph  are known(!), and there may be others. Does the 3-cube have more than four?

54. (a) The determinant is zero if and only if $\{v_0, v_1, v_2, v_3\}$ are coplanar; but they aren't. If it's negative, swap $v_2 \leftrightarrow v_3$. (Hence the cyclic order $(v_1 v_2 v_3)$ is unique.)

[See F. Joachimsthal, *Crelle* **40** (1850), 21–47, who observed that the volume of the tetrahedron formed by $\{v_0, v_1, v_2, v_3\}$ is $|D(v_0, v_1, v_2, v_3)|/6$. See also J. de la Grange, *Nouveaux Mém. Acad. Sciences et Belles-Lettres* **4** (Berlin: 1773), 85–120, §5.]

(b) $D(v_0, v_1, v_2, v) = 0$.

(c) $D(v_0, v_1, v_2, v) > 0$.

(d) For example use $(o_1 o_2 o_3)_8$, where $o_1 = [v$ is opposite v_1 with respect to $p_{23}]$, …, $o_3 = [v$ is opposite v_3 with respect to $p_{12}]$. (There's no standard convention for numbering octants; roman numerals are traditionally used in some arbitrary way.)

(e) With those v_i, that method gives octant *0* whenever x, y, z are all positive.

(f) It's now in octant *2*, because $\pi > \phi + \gamma$.

meromorphic function
author
internet
all different
XCC problem
crease
concave
convex
Heawood
automorphisms
author
internet
determinant
coplanar
Joachimsthal
volume
tetrahedron
de la Grange
Lagrange
historical notes
roman numerals

plane in three dimensions
ell

55. (a) A careful case analysis shows that edge $v_0 — v_1$ is concave if and only if X_ϵ intersects octant *3*. Similar conclusions hold for $v_0 — v_2$ with respect to *5*, and for $v_0 — v_3$ with respect to *6*.

(b) For example, if θ is the angle at edge $v_0 — v_1$, we have $(v_2 - v_0)\cdot(v_3 - v_0) = \|v_2 - v_0\|\|v_3 - v_0\|\cos\theta$. Choose $0 < \theta < 180^\circ$ if concave, otherwise $180^\circ < \theta < 360^\circ$.

57. First, if (x, y, z) is a vertex of X, there must be no edge containing a point (x, y, z') with $z \neq z'$. (In particular, there must be no vertex (x, y, z') with $z \neq z'$.)

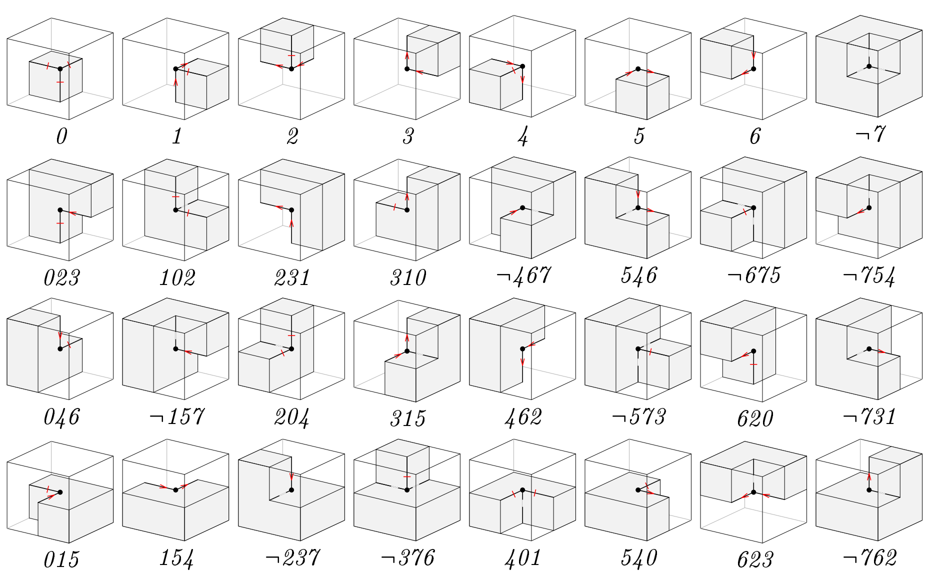
Second, X mustn't contain noncollinear edges whose projections are collinear. For example, if the line segment $\{(t, 0, 0) \mid 0 \le t \le 1\}$ is an edge of X, there shouldn't also be an edge of the form $(u, 0, u)$. Quantitatively, each edge has the form $\{(x_0 + \alpha t, y_0 + \beta t, z_0 + \gamma t) \mid t_0 \le t \le t_1\}$ for some $(\alpha, \beta, \gamma) \neq (0, 0, 0)$; by the first assumption, we have in fact $(\alpha, \beta) \neq (0, 0)$. Distinct edges must not have $\alpha\beta' = \alpha'\beta$.

[Consequently X has no faces perpendicular to the (x, y) plane. Indeed, every plane in three dimensions is characterized by an equation of the form $ax + by + cz = d$, where a, b, and c are not all zero. Since adjacent edges of a face aren't collinear, the equation for its plane must have $c \neq 0$. Hence we may assume that $c = 1$.]

58. (a) There obviously are 8 cases with one cubie. Three cubies that make the "ell" tricube can be placed in 24 ways. Five cubies whose complement is an "ell" can also be placed in 24 ways. Seven cubies can be placed in 8 ways. An even number of cubies can't make a 3VP with the center as vertex. Total, $8+24+24+8 = 64$. (Incidentally, a solution with (1, 3, 5, 7) cubies has respectively (0, 1, 2, 3) concave edges at the center.)

(b) Only the cubie in the corner closest to the camera obscures the center.

(c) This chart shows the octants that contain cubies, when octant *7* is closest:



(Notice that the rotation $x \mapsto y \mapsto z \mapsto x$ always gives an equivalent junction pattern.) By exercises 54 and 55, the possible labels of a V, W, or Y junction in an HC picture depend only on which octants adjacent to the corresponding vertex are occupied.

(d) By definition, the two "bars" of a T must be half edges that point left.

59. $(3t + 2v + 3w + 3y)/2$ variables and $t + v + w + y$ constraints.

60. (a) (a, b) $\in \{41, 51, 33, 62\}$, where '11' abbreviates $(1, 1)$, etc.

(b) (n, p) $\in \{12, 13, 22, 23, 32, 33, 42, 43\}$; (o, p) $\in \{13, 23, 36\}$.

(c) $t + v + w + y$ variables and $(3t + 2v + 3w + 3y)/2$ constraints (role reversal!).

(d) The text's model is most likely a winner, because it allows us to deduce some labels immediately (see (24) and (25)). Although we can deduce p = 3 from the two constraints in (b), the corresponding inference from (22) is just as easy. The total size of the new state space, $4^t 6^v 3^w 5^y$, does however tend to be quite a bit smaller than $4^{(3t+2v+3w+3y)/2}$; the ratio is $(1/2)^t (3/2)^v (3/8)^w (5/8)^y$, which is $\approx .00014$ in example (20). Computational experience is generally advisable when choosing between models, because different models typically suggest different heuristics for the order of branching. [See P. van Beek, *AAAI Conf.* **10** (1992), 447–452, Example 3.]

61. With 19 primary items $\{\mathrm{a, b}, \ldots, \mathrm{s}\}$ and 26 secondary items $\{\mathrm{ab, ac}, \ldots, \mathrm{rs}\}$ (see (21)), the options are 'a ab:< ac:+', 'a ab:< ac:>', ..., 's rs:+ ls:- qs:+', as in exercise 7.2.2.1–100. (In general, continuing exercise 59, there will be $t + 6v + 3w + 5y$ options.)

62. Change the lower Y labels to '---'. (That fills in the "hole".)

64. Whenever j is $T(l, m, r)$ or $V(l, r)$ or $W(l, m, r)$ or $Y(a, b, c)$ in H, j is respectively $T(r, m, l)$ or $V(r, l)$ or $W(r, m, l)$ or $Y(c, b, a)$ in H^R. (This rule defines H^R also in cases where H is unrealizable as an HC picture.)

Notice that H and H^R have the same variables and the same domains, but different relations. The values $x_1 \ldots x_n$ solve H if and only if $x_1^R \ldots x_n^R$ solves H^R, where $+^R = +$, $-^R = -$, $<^R = >$, $>^R = <$. (For example, in the reflection of (20) we have a $= V(\mathrm{c, b})$; the corresponding constraint is (ac, ab) $\in$ {<+, <>, +>, >-, ><, -<}, which is the same as (ab, ac) $\in$ {+<, ><, >+, ->, <>, <-}, which is the same as (ab, ac) $\in$ {>+, ><, +<, <-, <>, ->}.)

[People often say that mirror reflection interchanges left and right, but not top and bottom. Martin Gardner explains why in his book *The Ambidextrous Universe*.]

65. (a) For example, a $= V(\mathrm{b, c})$, b $= V(\mathrm{c, a})$, c $= V(\mathrm{b, a})$.

(b) H is realizable if and only if each of its connected components is realizable. If H is connected and its junctions $\{j_0, j_1, \ldots, j_{p-1}\}$ all have type V, we can assume that $j_k = V(j_{k+\sigma_k}, j_{k-\sigma_k})$ for $0 \le k < p$, with subscripts treated mod p, where each σ_k is ± 1. When $p = 3$, we must have $\sigma_0 = \sigma_1 = \sigma_2$. When $p = 4$, we must not have $\sigma_0 \ne \sigma_1 \ne \sigma_2 \ne \sigma_3$. When $p > 4$, we can assume (by switching to H^R if necessary) that $\sigma_0 = \sigma_k = \sigma_l = +1$ for some $0 < k < l < p$. Then H is realized by putting j_0, j_k, j_l at the vertices of a triangle, and placing the intervening junctions at roughly equidistant positions near the intervening edges of that triangle — perturbing them slightly so that each junction is convex or concave as desired, when seen from outside the triangle.

(c) The graph of a $=$ b $=$ c $= W(\mathrm{d, e, f})$, d $=$ e $=$ f $= W(\mathrm{a, b, c})$ is $K_{3,3}$.

(d) The interior angles of a polygon with m vertices sum to $(m-2)180°$; hence at most $m - 3$ of them are greater than $180°$.

(e) True. (Just jiggle the junction a little bit.)

66. If indeed this question is recursively decidable, what is its complexity?

67. (a) Each "level" has a sequence of junctions $j_1 = W(j_0, j_1', j_2)$, $j_2 = Y(j_1, j_2', j_3)$, $j_3 = W(j_2, j_3', j_4)$, ..., $j_9 = W(j_8, j_9', j_{10})$, whose connecting lines $j_0 j_1$, $j_1 j_2$, ..., $j_9 j_{10}$ must all be given the same label: either + or - or <. The standard boundary forces the labels <<<<<<<<<< at the bottom, but ---------- on the other levels. These, in turn, immediately force the labels in their vicinity, so the standard labeling is unique.

(b) Similarly, junctions of the form $j_0 = V(j_0', j_1)$, $j_1 = W(j_0, j_1', j_2)$, $j_2 = Y(j_1, j_2', j_3)$, ..., $j_6 = Y(j_5, j_6', j_7)$, $j_7 = W(j_6, j_7', j_8)$, $j_8 = V(j_7, j_8')$, which appear

van Beek
unrealizable
Gardner
$K_{3,3}$
interior angles
complexity

upwards at the right and downwards at the left, force the labels from j_0 to j_8 to be either `++++++++` or `--------` or `<<<<<<<<`. But `++++++++` is excluded, because it doesn't combine with `++++++++++` or `----------` or `<<<<<<<<<<` at the bottom.

Let the V junctions at the top be $t_1 = V(t_2, t_0)$, $t_2 = V(t_1, t_3)$, $t_3 = V(t_4, t_2)$, $\ldots$, $t_6 = V(t_5, t_7)$, $t_7 = V(t_8, t_6)$. We know that t_0t_1 must be `-` or `>`; the same holds for t_7t_8. Hence the legal labelings from t_0 to t_8 are `->+<->>-`, `->+<->>>`, `->+<<+>-`, `->+<<+>>`, `->+<<<->`, `->>-<+>-`, `->>-<+>>`, `->>-<<->`, `->>>+<->`, `->>>>>>-`, `->>>>>>>`, `>-<+>>>-`, `>-<+>>>>`, `>-<<->>-`, `>-<<->>>`, `>-<<<+>-`, `>-<<<+>>`, `>-<<<<->`, `>>+<->>-`, `>>+<->>>`, `>>+<<+>-`, `>>+<<+>>`, `>>+<<<->`, `>>>-<+>-`, `>>>-<+>>`, `>>>-<<->`, `>>>>+<->`, `>>>>>>>-`, `>>>>>>>>`: 4 from `-` to `-`, 7 from `-` to `>`, 7 from `>` to `-`, and 11 from `>` to `>`. The latter can be used with either `----------` or `<<<<<<<<<<` at the bottom. So the total number of boundary labelings is $4 + 7 + 7 + 11 + 11 = 40$.

(c) There are exactly 40, because each of those 40 boundary-only solutions imposes exactly the same constraints on the lines touching the boundary. [P. H. Winston, who presented this picture as Fig. 3-17 in the second edition of his book *Artificial Intelligence* (Addison–Wesley, 1984), noted that "The background border contributes considerable constraint to line-drawing analysis efforts." He may not have been aware, however, that *any* border constrains the interpretation of the interior in the same way!]

68. (a) These possibilities are essentially forced by the definition of boundary cycle.

(b) In each of the following 4×4 matrices, the rows and columns are indexed by (`>`, `<`, `-`, `+`), where rows represent the label of $j_{k-1}j_k$ and columns represent the label of j_kj_{k+1}. The entry is 0 when the row/column labels are illegal; otherwise it is 1 when there's no junction j'_k; otherwise it is (g, l, m, p) when $j_kj'_k$ must be labeled (`>`, `<`, `-`, `+`).

$$\boldsymbol{L} = \begin{bmatrix} l&0&0&0\\ l&0&0&0\\ l&0&0&0\\ l&0&0&0 \end{bmatrix},\ \boldsymbol{R} = \begin{bmatrix} g&g&g&g\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0 \end{bmatrix},\ \boldsymbol{V} = \begin{bmatrix} 1&0&1&0\\ 0&1&0&1\\ 1&0&0&0\\ 0&1&0&0 \end{bmatrix},\ \boldsymbol{\Lambda} = \begin{bmatrix} 1&0&1&0\\ 0&1&1&0\\ 0&1&0&0\\ 1&0&0&0 \end{bmatrix},\ \boldsymbol{W} = \begin{bmatrix} p&0&0&0\\ 0&0&0&0\\ 0&0&p&0\\ 0&0&0&m \end{bmatrix},\ \boldsymbol{Y} = \begin{bmatrix} m&0&0&0\\ 0&0&l&0\\ 0&g&m&0\\ 0&0&0&p \end{bmatrix}.$$

(c) Multiply the matrices of the boundary cycle together, treating $\{g, l, m, p\}$ as noncommuting variables. The diagonal entries of the resulting matrix then specify the permissible labelings of the internal lines. For example, the boundary cycle of (20) gives

$$\boldsymbol{VWYWVWVWYW} = \begin{bmatrix} 3pmpppmp & 0 & 2pmpppmp & 0\\ 0&0&0&0\\ 2pmpppmp & 0 & pmpppmp & 0\\ 0&0&0&0 \end{bmatrix};$$

hence the boundary (in isolation) can be labeled four ways, with $j_{-1}j_0$ labeled `>` in three cases and `-` in the other. (The sum of diagonal elements is called the "trace.") In all four cases the interior labels are respectively `+-+++-+`; hence (20) has a free boundary.

This free-boundary-testing algorithm needn't implement arithmetic on string polynomials in full generality. For each matrix entry, it needs to remember only whether that entry is (i) zero, (ii) a multiple of a certain string σ, or (iii) mixed. At the end, the boundary is free if and only if the sum of the four diagonal elements isn't mixed.

69. The trace of the boundary cycle matrix product $\boldsymbol{W\Lambda}^n\boldsymbol{WV}^n$ of exercise 68 is $F_{n+1}^2pp + 0 + F_{n-1}^2pp + F_{n-1}^2mm$. Therefore, to complete the labeling, we need to consider a sequence of n V junctions, preceded and followed by the same sign. That's equivalent to binary strings $1x_1 \ldots x_{n-1}1$ with no two consecutive 1s — of which there are F_{n-1} (see exercise 7.2.1.1–91). Altogether, then, there are $F_{n-1}(F_{n+1}^2 + 2F_{n-1}^2)$ labelings, of which F_{n-1} are standard. (*Nitpicking note:* S_1 has a free boundary, by definition, although it cannot be fully labeled.)

Winston
noncommuting variables
trace
free boundary
string polynomials
consecutive 1s
free boundary

bridge
boundary cycle
trace
organ pipe order, inverse
Kane
operator norm
eigenvalues

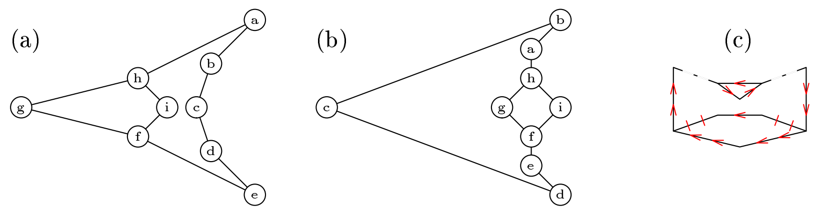


Fig. A–13. Unusual examples of HC pictures.

70. There are (1, 9, 1, 1) labelings with the bridge in the middle labeled (+, -, >, <), respectively. (This example shows that an HC picture need not have a boundary cycle consisting of distinct lines.)

71. No: The HC pictures in Figs. A–13(a) and A–13(b) have the same HC network but different cycles. (Consequently the algorithm of exercise 68(c) must be told the boundary cycle as well as the network.)

72. See Fig. A–13(c). (Answer 68(c) gives trace($\boldsymbol{VYYVWWVW}$) $= 4mmpp + glpp$; so the boundary cycle can be labeled in five ways. Only one of those ways, ><++, gives usable labels to the inner lines, because a V junction doesn't allow --.)

73. (a) Let P be the 4×4 matrix product $j_0 j_1 \ldots j_{q-1}$, and let $M = M_0 M_1 \ldots M_{q-1}$. By induction we can verify that $P_{ij} = P_{(i\oplus 1)(j\oplus 1)}$ for $0 \le i,j < 4$; $P_{00}+P_{01} = F_{q+1}$; $P_{02}+P_{03} = P_{20}+P_{21} = F_q$; $P_{22}+P_{23} = F_{q-1}$; and $M_{ij} = P_{(2i)(2j)} - P_{(2i)(2j+1)}$ for $0 \le i,j < 2$. Hence $\operatorname{trace} P = P_{00}+P_{11}+P_{22}+P_{33} = P_{00}+P_{00}+P_{22}+P_{22}$ and $\operatorname{trace} M = M_{00}+M_{11} = P_{00}-P_{01}+P_{22}-P_{23} = P_{00}-(F_{q+1}-P_{00})+P_{22}-(F_{q-1}-P_{22})$.

(b) The matrix products can be expressed in closed form using the identities

$$A^a = \begin{pmatrix} F_{a+1} & F_a \\ F_a & F_{a-1} \end{pmatrix}, \quad B^b = \begin{pmatrix} F_{b+1} & -F_b \\ -F_b & F_{b-1} \end{pmatrix}, \quad A^a B^b = \begin{pmatrix} \Delta_{a,b} & -\Delta_{a,b-1} \\ \Delta_{a-1,b} & -\Delta_{a-1,b-1} \end{pmatrix},$$

where $\Delta_{a,b} = F_{a+1}F_{b+1} - F_a F_b = \frac{1}{5}\bigl(L_{a+b+1} + 2(-1)^b L_{a-b}\bigr)$. Hence $\Delta_{a,b} - \Delta_{a-1,b-1} = \frac{1}{5}\bigl(L_{a+b} + 4(-1)^b L_{a-b}\bigr)$, and the values of $t_p = \operatorname{trace}(A^p B^{q-p})$ occur in a peculiar order:

$$t_1 < t_3 < \cdots < t_{\lfloor q/2\rfloor} = t_{\lceil q/2\rceil} < \cdots < t_2 < t_0, \quad \text{with } t_p = t_{q-p}.$$

The extremes are $t_p + L_q = 2F_q$ when $p \in \{1, q-1\}$; $t_p + L_q = 2L_q$ when $p \in \{0, q\}$.

(c) (Solution by D. M. Kane.) Note that $B = XAX$, where $X = \bigl(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\bigr)$. Thus M is a product of q As, but with m Xs inserted somehow, where m is the number of switches between $\boldsymbol{V}$ and $\boldsymbol{\Lambda}$ in the cycle. Our goal is to prove that $\operatorname{trace} M \ge 2F_q - L_q = -F_{q-3}$.

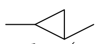
We can assume that $M = (AXA)A^{p_1}(AXA)A^{p_2}\ldots(AXA)A^{p_m}$, where $p_k \ge -1$ for $1 \le k \le m$. If all p_k are nonnegative, $\operatorname{trace} M \ge 0$, because $AXA = \bigl(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\bigr)$.

If $p_1 = p_m = -1$, we have $M = ABA^{p_2+2}B^{p_3+2}\ldots B^{p_{m-1}+2}$. And $ABA = BAB$ implies that $\operatorname{trace} M = \operatorname{trace}(BABA^{p_2+1}B^{p_3+2}\ldots B^{p_{m-1}+2}) = \operatorname{trace}(ABA^{p_2+1}B^{p_3+2}\ldots B^{p_{m-1}+3}) = \cdots = \operatorname{trace}(AB^{p_3+1}\ldots B^{p_{m-1}+p_2+4})$, thus reducing m by 2. Therefore we can assume that at least one p_k is -1, but no two -1s are consecutive.

Now let $\|M\|$ be the *operator norm* of M, namely $\sup|Mx|$ over all vectors x of length 1. Then we have $\|A\| = \|AXA\| = \phi$ and $\|AXAXA\| = 1$. Consequently $\|M\| \le \phi^{n-5}$ when $m \ge 4$. (We save a factor of ϕ^2 when $p_k = -1$, ϕ when $p_k \ge 0$.)

Finally, let M have eigenvalues λ and $\hat\lambda$, where $|\lambda| \ge |\hat\lambda|$. Then $\operatorname{trace} M = \lambda + \hat\lambda$, and $\lambda\hat\lambda = \det M = (-1)^q$. So $|\operatorname{trace} M| \le |\lambda| + 1/|\lambda| \le \phi^{n-5} + \phi^{5-n} \le F_{n-3}$, for $n > 6$.

74. Let $D_0 = I$ and $D_{n+1} = D_n A D_n^{RX}$, where ^{R} means left-right reflection and ^{X} means 'change A to B and B to A'. Thus $D_1 = A$, $D_2 = AAB$, $D_3 = AABAABB$, etc. We have $D_n^R = D_n^T$, because $A^T = A$ and $B^T = B$. Hence, using the matrices of answer 73, $D_{n+1} = D_n AXD_n^T X$; and the surprising formula $D_{n+3} = \left(\begin{smallmatrix} 1 & n-1 \\ -1-n & -n^2 \end{smallmatrix}\right)$ arises by induction for $n \geq 0$. Consequently T_n has $\mathrm{trace}(D_{n-1}AD_{n-1}A) + L_{2^n} = L_{2^n} - 1$ labelings, when $n \geq 4$(!). The same formula holds for $n = 3$; but T_2 has 14.

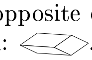
76. . [This is a subpicture of Figure 9(c) in D. A. Huffman's 1971 paper. Examples (24) and (25) come from his Figure 8.]

77. (a) The junctions are $t_k = T(t_{k-1}, t_{k+1}, u_k)$, $u_k = V(w_{k+1}, t_k)$, $v_k = V(w_k, w_{k-1})$, $w_k = W(v_k, u_{k+1}, v_k)$, with subscripts mod n, for $0 \leq k < n$.

(b) The Lucas number L_n. (But only one of these labelings is standard; these networks have a free boundary. Exercise 69 has similar considerations.)

(c) (Solution by K. Sugihara.) The network defines a graph that's uniquely embeddable as an HC picture H in the plane. Suppose H is the projection of some 3VP, X, and let F_k be the face of X that corresponds to the region of H bounded by the polygon $(w_k u_{k-1} t_{k-1} t_k u_k w_{k+1} v_{k+1})$. Let $P = (x, y)$ be a point in H's center region, and let L be the line through P perpendicular to the plane of the picture. Then L intersects F_k at some point (x, y, z_k). Since the edge $u_{k-1}w_k$ is convex, by part (a), we have $z_k > z_{k-1}$. But $z_{n-1} > z_{n-2} > \cdots > z_0 > z_{n-1}$ is impossible.

[See also the discussion by S. W. Draper in *Perception* **7** (1978), 283–296, as well as the comments by Bruno Ernst in Chapter 2 of his book *Adventures with Impossible Figures* (1986). Ernst shows the Penrose square and hexagon, together with a *different* pentagon(!). The Penrose pentagon of the present exercise is #85 in the comprehensive website *Impossible World* by Vlad Alexeev, `http://im-possible.info`, a gallery launched in 2001 that features more than 1000 mind-bending images.]

78. Take a cube and flatten it so that opposite corners are near each other. (Here's a view from the side, only 90% squashed: .) This gives a crumpled object very like a hexagonal tile; you can place such "chips" on a table with any desired overlaps.

Historical notes: A copy of Reutersvärd's original 'Opus 1' is held by Moderna Museet in Stockholm [NMH 42/1981]. It does not show the boxes in general position — the blank region in the middle is a symmetrical "star of David" — so HC picture (32) is slightly different. He told Bruno Ernst in 1986 that he discovered the pattern while doodling during a boring lecture about Latin! [See Figure 1 in Chapter 6 of Ernst's book *Optical Illusions* (1992). Figure 7 in Ernst's Chapter 1 is (26), 'perspective japonaise no. 231 aga', part of a series of more than 2500 artworks now prized by collectors.]

79. The central region has three V junctions, whose left lines can independently be labeled `-` or `<`. Hence there are 8 standard labelings — all realizable as in exercise 78.

There's a free boundary, since each of the corners can be labeled in three ways, and each of the other six in two ways; these $2^6 3^3 = 1728$ boundary labelings all force the same labels inside. So there are $8 \cdot 1728 = 13824$ labelings altogether.

80. Image (i) has a unique standard labeling. But (ii) has $33{,}554{,}432 = 2^{25}$, because each of 25 interior "box tops" has a V junction that can be labeled in two ways.

Image (iii) shows what happens when the 36 cells of the 6×6 hexagonal rhombus are partitioned into three independent sets of 12. One set of twelve boxes is placed in front, another in back. The front ones are labeled uniquely. The back ones are labeled uniquely at the edges, but in five ways when they appear only as a Y in the interior. The middle ones each have two labelings of a W near the edges (except at the very

transpose of matrix
Huffman
Lucas number
free boundary
Sugihara
Draper
Ernst
Alexeev
Historical notes
Reutersvärd
general position
star of David
Ernst
free boundary

bottom), but nine in the interior (when they show up as an unconstrained Y with three Ws). Altogether 11,809,800,000 $= 5^5 2^6 9^5$ standard labelings.

In image (iv) there's clockwise overlapping in the outer loop, enclosing a loop with counterclockwise overlapping; but it's realizable with "squashed boxes." As with the other three images, a large number of T junctions makes the labelings factor into small independent subnetworks, and we find 5,242,880 $= 2^{20} \cdot 5$ standard labelings altogether.

[An interesting mapping was used to draw these images: If x, y, and z are each ± 1, corner (x, y, z) of the box in row i and column j of the array is assigned to point $(6i + j - 2y - 2z, -i + 5j + 2x + 2z, -5i - 6j - 2z + 2y)$ in barycentric coordinates. (At most seven corners of each box are visible — all except corner $(1, 1, -1)$.) With this scheme, all points where the edges of two boxes intersect are distinct, and those points are also distinct from all corner points; thus the images appear in general position.]

squashed boxes
barycentric coordinates
general position
Dynamic programming
planar graph
background plane

81. (a, b) When $m = n = 6$ there are 85 Boolean variables, 50 ternary constraints; in general there are $m(n-1) + (m-1)n + (m-1)(n-1)$ Boolean variables and $2(m-1)(n-1)$ ternary constraints. Each constraint has the form $[A < B] + [B < C] + [C < A] \in \{1, 2\}$.

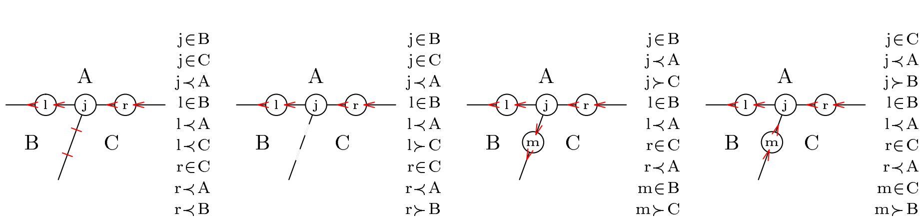
Dynamic programming works well, as in exercise 41, and this problem is considerably easier than that one: Let box (i, j) in row i, column j for $0 \le i < m$ and $0 \le j < n$ be adjacent to boxes $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$; and consider the number $c_n(x_1, \ldots, x_{m-1})$ of $m \times n$ solutions with $x_j = [(i-1, n-1) < (i, n-1)]$. After setting $c_1(x_1, \ldots, x_{m-1}) \leftarrow 1$, we can readily compute the 2^{m-1} counts $c_{n+1}(x_1, \ldots, x_{m-1})$ from the 2^{m-1} counts c_n. For example, when $m = 3$ we have $c_{n+1}(0,0) = 13c_n(0,0) + 11c_n(0,1) + 9c_n(1,0) + 6c_n(1,1)$; $c_{n+1}(0,1) = 11c_n(0,0) + 12c_n(0,1) + 10c_n(1,0) + 9c_n(1,1)$; $c_{n+1}(1,0) = c_{n+1}(0,1)$; $c_{n+1}(1,1) = c_{n+1}(0,0)$. (These 2^{2m-3} coefficients are themselves each precomputed in $O(m)$ steps by solving a small-and-simple CSP.)

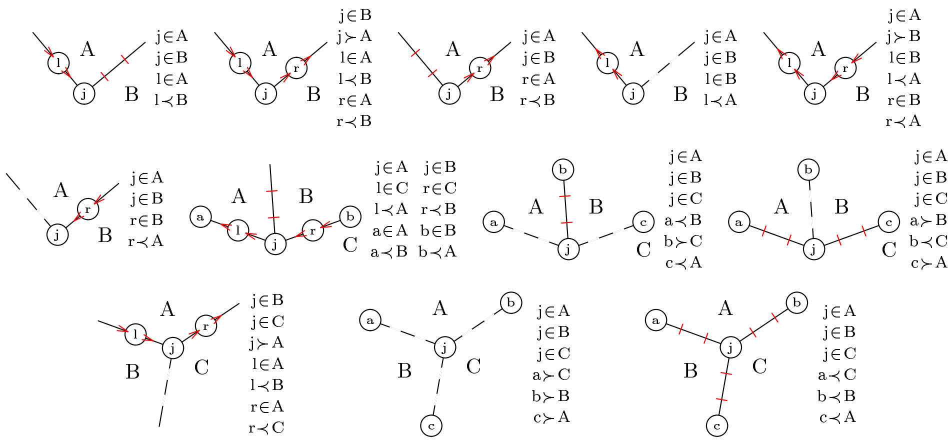
The total number of solutions is $t_{m,n} = \sum\{c_n(x_1, \ldots, x_{m-1}) \mid 0 \le x_k \le 1\}$. For example, $(t_{3,1}, t_{3,2}, t_{3,3}, \ldots, t_{3,n}, \ldots) = (4, 162, 6570, \ldots, \lceil cr^n \rceil, \ldots)$, where $r = (41 + \sqrt{1609})/2 \approx 40.556$ and $c = (1609 + 31\sqrt{1609})/28962 \approx .0985$. We also have $t_{6,6} = 22406540276117433798 \approx 2^{64.28}$; $t_{10,10} = 2333623171515704644702\ldots99558 \approx 2^{193.89}$.

83. Regarding H as an embedded planar graph, let F be the set of its faces. Each $f \in F$ will correspond to part (or all) of some face $\hat f$ of X — except that H's exterior face f_0 will corresponds to a "background plane" $\hat f_0$, which is sufficiently distant that it doesn't conceal any of X.

For each line of H that is labeled < or >, introduce a new "shadow junction" at the midpoint of that line; and let J be the set of all junctions (shadow or not). Each junction j of type V, W, or Y will correspond to a vertex $\hat\jmath$ of X. Every remaining junction j will correspond to an *artificial* vertex $\hat\jmath$, namely the point of X or of the background plane that lies just behind the point (x_j, y_j) of H.

Now use the following chart to establish relations between junctions and faces:



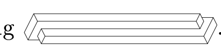


Here '($j \in A$, $j \prec A$, $j \succ A$)' means "$\hat{\jmath}$ lies (in, behind, in front of) the plane of $\hat{A}$."

To represent those relations linearly, we introduce a real variable z_j for each $j \in J$, meaning that $\hat{\jmath} = (x_j, y_j, z_j)$, where x_j and y_j are given constants. We also introduce three real variables (a_f, b_f, c_f) for each $f \in F$, meaning that plane $\hat{f}$ consists of all (x, y, z) for which $ax+by+z+c=0$. (See answer 57.) By convention, point (x, y, z) lies behind point (x, y, z') if and only if $z > z'$. Hence $j \in A \iff a_A x_j + b_A y_j + z_j + c_A = 0$; $j \prec A \iff a_A x_j + b_A y_j + z_j + c_A \geq 1$; $j \succ A \iff a_A x_j + b_A y_j + z_j + c_A \leq -1$.

(Actually '> 0' and '< 0' were expected here instead of '≥ 1' and '≤ -1'; but strict inequality is difficult to deal with, in general, while the theory of linear programming handles nonstrict inequality with ease. Fortunately the two notions are equivalent in this case: If there's a solution to the strict inequalities, the nonstrict ones will be satisfied after we multiply all variables $\{a_f, b_f, c_f, z_j\}$ by a suitably large positive constant.)

[This construction is based on Chapter 3 of K. Sugihara's book *Machine Interpretation of Line Drawings* (1986), where a considerably more general problem is treated. It is unknown whether or not this linear system is *sufficient* for a 3VP X to exist.]

85. No — it's strongly realizable as a 3VP. (Start by realizing .)

86. See *J. Computer and System Sciences* **37** (1988), 14–38. The construction is based on D. Lichtenstein's theorem [*SICOMP* **11** (1982), 329–343] that 3SAT is NP-complete even when the clauses are planar and severely restricted.

(The authors show, however, that labelability can be decided in *linear* time if the HC picture arises from an "orthohedral" 3VP, in which every plane face is perpendicular to the x-, y-, or z-axis. For example, a histoscape is orthohedral. In such a case all angles can be assumed to be multiples of 60°. Furthermore, the two entries of Table 1 for which a V junction has a + label can arise only for 60° angles; the other four possibilities for V can arise only for 120° angles.)

87. If indeed this question is recursively solvable, what is its complexity? [Partial results were given by Kirousis and Papadimitriou in the paper just cited. K. Sugihara presented polynomial time necessary and sufficient conditions for strong realizability, in his book cited in answer 83, based on a related but different mathematical model of the problem. Consequently the realizations constructed there aren't 3VP in general.]

90. Replace (13, 14) by (2, 13) or (4, 14) or (13, 2) or (14, 4) or (14, 13). (And to get two more solutions, change either (12, 6, 13) to (6, 13, 7) or (8, 6, 14) to (6, 7, 4).)

strict inequality
linear programming
Sugihara
strongly realizable
Lichtenstein
3SAT
planar
orthohedral
histoscape
complexity
Kirousis
Papadimitriou
Sugihara

91. Almost true (but false when $m = 1$). Given any graceful labeling l, we obtain $2k$ equivalent labelings $l(v\alpha)$ and $m - l(v\alpha)$ when α runs through G's automorphisms. If those labelings aren't distinct, there are automorphisms α and β for which $l(v\alpha) = m - l(v\beta)$ for all v. But then $\beta^-\alpha$ would be an automorphism satisfying $l(v\beta^-\alpha) = m - l(v)$; that is, complementation would be an automorphism.

That can't happen when $m > 1$: By adding isolated vertices if necessary, we can assume that the vertices are $\{0, \ldots, m\}$ and that $l(v) = v$ for $0 \le v \le m$. The edge labeled m must be $0 \mathrel{—} m$, and we can assume that the edge labeled $m-1$ is $1 \mathrel{—} (m-1)$. Then m is not adjacent to 1, so complementation isn't an automorphism.

93. (a) For example, eliminate all options with $l(\texttt{NY}) > l(\texttt{MA})$ or $l(\texttt{GA}) > l(\texttt{SC})$. (Then 5814 options remain, and the running time goes down to 33 gigamems.)

(b) Add a new primary item '`*`' and the new option '`* GA`:0 `SC`:18 `NJ`:5'. (The search tree now has 192 nodes. It can be reduced to 62 nodes via domain consistency.)

94. $\texttt{LO}'[l] = m - l - \texttt{LO}[l]$, $\texttt{NAME}'[l] = \texttt{NAME}[m-l]$, $\texttt{FIRST}'[l] = m - \texttt{FIRST}[m-l]$, for $0 \le l \le m$; $\texttt{NEXTL}'[l] = m - \texttt{NEXTH}[l]$, $\texttt{NEXTH}'[l] = m - \texttt{NEXTL}[l]$, for $1 \le l \le m$; but change $m+1$ to -1. (Other settings of $\texttt{FIRST}'$, $\texttt{NEXTL}'$, $\texttt{NEXTH}'$ are also possible.)

95. The first four real vertices can't be $\{0, m-2, m-1, m\}$ or $\{0, 1, m-1, m\}$, because only one edge can be labeled 1. Hence they are $\{0, 2, m-1, m\}$; and $\texttt{LO}[m-3] = 2$. That forces $\texttt{LO}[m-4] = 0$, leaving no choices for $\texttt{LO}[m-5]$.

96. The key idea is to have a good way to represent the partial path fragments formed by the already-chosen edges. If l is an unchosen vertex label, let $\texttt{MATE}[l] = l$; if l is chosen and the endpoint of a partial subpath, let $\texttt{MATE}[l]$ be the other endpoint; otherwise let $\texttt{MATE}[l]$ be the bitwise complement of the value it had when it was most recently an endpoint, during the backtracking. For example, the `MATE` table ($\texttt{MATE}[0], \ldots, \texttt{MATE}[5]$) at node '405' of (38) is $({\sim}5, 1, 2, 3, 5, 4)$; at node '4052,13' it's $({\sim}5, 3, 4, 1, 2, {\sim}4)$.

P1. [Initialize.] Set $\texttt{MATE}[l] \leftarrow l$ for $0 \le l < n$, then set $l \leftarrow 1$.

P2. [Enter level l.] If $l = n$, visit a solution and go to P5. Otherwise set $v \leftarrow 0$.

P3. [Try $\texttt{LO}[n-l] = v$.] Set $w \leftarrow v + n - l$, $v' \leftarrow \texttt{MATE}[v]$, $w' \leftarrow \texttt{MATE}[w]$. Go to P4 if $v' < 0$ or $w' < 0$ or $v' = w$. Otherwise set $\texttt{LO}[n-l] \leftarrow v$, $\texttt{MATE}[v] \leftarrow {\sim}v'$, $\texttt{MATE}[w] \leftarrow {\sim}w'$, $\texttt{MATE}[v'] \leftarrow w'$, $\texttt{MATE}[w'] \leftarrow v'$, $l \leftarrow l + 1$, and return to P2.

P4. [Try again.] Set $v \leftarrow v + 1$. If $v < l$ and $l > 2$, go to P3.

P5. [Backtrack.] Set $l \leftarrow l - 1$, and terminate if $l = 0$. Otherwise set $v \leftarrow \texttt{LO}[n-l]$, $w \leftarrow v + n - l$, $v' \leftarrow \texttt{MATE}[v]$, $w' \leftarrow \texttt{MATE}[w]$. If $v' \ge 0$ set $\texttt{MATE}[v] \leftarrow v$; otherwise set $\texttt{MATE}[v] \leftarrow {\sim}v'$ and $\texttt{MATE}[{\sim}v'] \leftarrow v$. If $w' \ge 0$ set $\texttt{MATE}[w] \leftarrow w$; otherwise set $\texttt{MATE}[w] \leftarrow {\sim}w'$ and $\texttt{MATE}[{\sim}w'] \leftarrow w$. Return to P4. ▌

97. A "blurred state" is obtained from `MATE` when all the negative entries are replaced by '$-$'. For example, 1738092 and 1809372 both have $(-, 2, 1, -, 4, 5, 6, -, -, -)$ as their blurred state. With a suitable hashing scheme we can maintain a dictionary of all the distinct blurred states that arise during the search.

We also maintain a list of branch specs (v_p, β_p, o_p) for $p = 1, 2, \ldots$; here v_p is a value of `LO`; β_p is the blurred state if v_p is chosen; and o_p is the branch when v_p isn't. If α represents a blurred state, $\texttt{FIRST}(\alpha)$ represents its first branch and $\texttt{LOC}(\alpha)$ represents the corresponding output node. Both `FIRST` and `LOC` are 0 unless changed.

In step P1, set $p \leftarrow 0$ and α_1 to the initial blurred state.

In step P2, "visit" a solution by setting $\texttt{LOC}(\alpha_l) \leftarrow 1$.

isolated vertices
domain consistency
bitwise complement
blurred state

At the end of step P3, do the following just before returning to P2: Set α_l to the current blurred state, and set $p \leftarrow p+1$, $v_p \leftarrow v$, $\beta_p \leftarrow \alpha_l$, $o_p \leftarrow \mathtt{FIRST}(\alpha_{l-1})$, and $\mathtt{FIRST}(\alpha_{l-1}) \leftarrow p$. If α_l has occurred before, jump to the second sentence of step P5.

Finally, after backtracking is complete, we can transform the branch specs into something like a ZDD with the following procedure: "Set $z \leftarrow 2$, $s \leftarrow 1$, $\beta_1 \leftarrow \alpha_1$, $o_1 \leftarrow 0$, $\mathtt{LOC}(\alpha_1) \leftarrow z$. While $s \neq 0$ do the following: "Set $p \leftarrow \mathtt{LOC}(\beta_s)$, $q \leftarrow \mathtt{FIRST}(\beta_s)$, $s \leftarrow o_s$. While $q \neq 0$ do the following: "Set $q' \leftarrow o_q$, $\alpha \leftarrow \beta_q$. If $\mathtt{LOC}(\alpha) = 0$ and $\mathtt{FIRST}(\alpha) \neq 0$, set $o_q \leftarrow s$, $s \leftarrow q$, $\mathtt{LOC}(\alpha) \leftarrow z$, $z \leftarrow z+1$. If $q' \neq 0$, output $I_p = (\bar{v}_q?\ z\colon \mathtt{LOC}(\alpha))$ and set $p \leftarrow z$, $z \leftarrow z+1$; otherwise output $I_p = (\bar{v}_q?\ 0\colon \mathtt{LOC}(\alpha))$. Set $q \leftarrow q'$." " "

The output isn't necessarily a true ZDD: Its "variables" have to be understood correctly, it isn't necessarily reduced, and its instructions can sometimes have the form $I_p = (\bar{v}?\ 0\colon 0)$. But many algorithms that manipulate ZDDs will handle it correctly. For example, the algorithm of exercise 7.1.4–208 will count the total number of solutions.

Equivalent nodes occur only on the same level, so it might seem that a breadth-first search is needed. But this method coexists nicely with (depth-first) backtracking.

This exercise is based on the ideas of M. Adamaszek [*J. Combin. Math. Combin. Computing* **87** (2013), 191–197], who was the first to enumerate graceful permutations for $20 < n \leq 40$. It gives a tremendous speedup over exercise 96; for example, when $n = 30$ the running time decreases from 25 teramems to 34 megamems!

[*Historical notes:* Graceful permutations were implicitly introduced by J. Abrham and A. Kotzig, *Cong. Numerantium* **72** (1990), 163–174, who proved that they have exponential growth. T. Kløve, *IEEE Trans.* **IT-41** (1995), 279–283, considered them independently and used them to design certain error-correcting codes. See J. McGill and M. A. Ollis, *Discrete Math.* **342** (2019), 793–799, for further developments.]

98. (a) Let l_1 and l_2 be the longest two distinct lengths. If we perturb each point by less than $|l_1 - l_2|/(2n)$, we change the path length by less than $|l_1 - l_2|$. So we may assume that the points $p_1 \ldots p_n = (x_1, 0) \ldots (x_n, 0)$ of a longest path have *distinct* x's.

The path can't be longest if $\max(x_{i-1}, x_i) < \min(x_j, x_{j+1})$ or if $\min(x_{i-1}, x_i) > \max(x_j, x_{j+1})$ for some $1 < i < j < n$: $p_1 \ldots p_{i-1} p_j \ldots p_i p_{j+1} \ldots p_2 p_n$ would be longer.

Let S be the $\lfloor n/2 \rfloor$ points with *smallest* x's, and let T be the other $\lceil n/2 \rceil$ points. No two points of S can be consecutive in the path; otherwise there would also be two consecutive points of T. Hence we can assume that $S = \{x_2, x_4, \ldots, x_{2\lfloor n/2 \rfloor}\}$.

The maximum path length is therefore $(x_1 - x_2) + (x_3 - x_2) + (x_3 - x_4) + \cdots = 2\sum T - 2\sum S - x_1 + x_n [n \text{ even}]$, where x_1 is the smallest x in T and x_n is the largest x in S.

Similarly, the longest *cycle* $(p_1 \ldots p_n)$ has length $2\sum T - 2\sum S - 2x_1 [n \text{ odd}]$.

(b) A graceful permutation with $p_n = p_1 + m$ yields a cycle $(p_1 \ldots p_n)$ of length $1 + \cdots + (2m-1) + (p_n - p_1) = 2m^2$, which is maximum because $\sum T - \sum S = m^2$.

(c) The path length is $2\sum T - p_n - 2\sum S + p_1 = 1 + \cdots + (2m-1) = 2m^2 - m$.

99. There are $m = 2n+1$ edges. Call S a (d,m)-set if $S \cup \{\, |k-d| \mid k \in S \,\} \cup \{d\} = \{1, \ldots, m\}$. A canonical graceful labeling of $K_{1,1,n}$ has vertices 0 and d in the first two parts, where $1 \leq d \leq m$, and the vertices S of the third part are a (d,m)-set. Furthermore, we require that $1 \notin S$ if $d = m$, to rule out the complementary labeling.

There clearly is no (d,m)-set with $d > m$. But there are $2^{(m-1)/2}$ (m,m)-sets, because S must contain 1 or $m-1$, 2 or $m-2$, …, $\lfloor m/2 \rfloor$ or $\lceil m/2 \rceil$.

There's no (d,m)-set when $\lceil m/2 \rceil < d < m$. For S would have to contain m, $m-1$, …, $d+1$, and then there'd be no way to get edge $d-1$.

There's a unique $(\lceil m/2 \rceil, m)$-set, namely $S = \{m, m-1, \ldots, \lceil m/2 \rceil + 1\}$.

breadth-first vs depth-first
depth-first vs breadth-first
Adamaszek
Historical notes
Abrham
Kotzig
Kløve
McGill
Ollis
longest *cycle*

Finally, if $d < \lceil m/2 \rceil$, a (d,m)-set S must be $\{m, m-1, \ldots, m-d+1\} \cup S'$, where S' is a $(d, m-2d+2)$-set. (An interesting recursion!)

So the total number of solutions is $\sum_{d \backslash m} 2^{(d-1)/2} + \sum_{d \backslash n+1} 1 - 2^{n-1} - 1 - 2[n=1]$.

100. If $1 \longrightarrow m$, change each x_{ij} to $m - x_{ij}$. Then if $\min\{x_{11}, \ldots, x_{n1}\} > \min\{x_{1r}, \ldots, x_{nr}\}$, change each x_{ij} to $x_{i(r+1-j)}$. Finally, sort the rows so that $x_{11} < \cdots < x_{n1}$.

101. It appears in level 4, because that placement of vertex 2 creates not only edge 7 (the goal of level 3) but also edge 6. One can think of it as belonging to both levels.

102. (a) At level 5 we've created the $\binom{5}{2}$ edges $\{1, 2, 3, 4, m-6, m-4, m-3, m-2, m-1, m\}$; so the algorithm's next step is to create edge $m-5$. The possibilities are (i) $x_{21} = m-5$; (ii) $x_{51} = 1$; (iii) $x_{41} = m-3$; (iv) $x_{31} = 4$; (v) $x_{11} = 5$.

(b) Moves (i)–(v) of part (a) all work, and they nicely break left-right symmetry. There's also one more possibility, namely (vi) $x_{61} = 3$ and $x_{63} = m-2$; again this breaks reflection symmetry. [All these cases will take us through level 6 to level 7.]

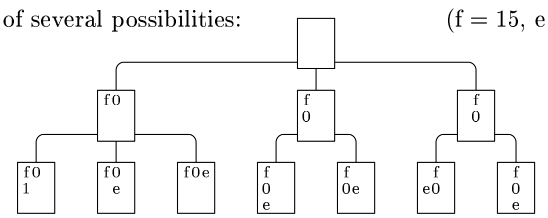
Historical notes: K. E. Petrie and B. M. Smith studied $K_n \square P_2$ for $n \le 5$, in order to test several strategies that exploit symmetry in instances of CSP [*Lecture Notes in Computer Science* **2833** (2003), 930–934]. Their methods were significantly improved by B. M. Smith and J.-F. Puget [*Constraints* **15** (2010), 64–92], who considered KP and KC graphs in general and discovered the unique labeling of $K_6 \square P_3$. However, the method illustrated in Fig. 107 is significantly faster than all of those approaches.

103. Instead of filling the matrix (x_{ij}) with explicit numbers, calculate *symbolically* with values of the form '$m-c$' or 'c' for small values of c. (See exercise 102, and imagine replacing (9, 8, 7, 6, 5) in levels 0 through 5 of Fig. 107 by (m, $m-1$, $m-2$, $m-3$, $m-4$). Notice that nodes on level $l+1$ involve only the values $\{0, 1, \ldots, l\}$ and $\{m, m-1, \ldots, m-l\}$, when $l < \lceil m/2 \rceil$.)

Hence the top $\lceil m/2 \rceil$ levels of this symbolic tree will be the same for all n, except for nodes that have too many rows. It turns out that this tree has only 8910 nodes, and its maximum level is 23. So we can't get an edge labeled $m-23$ when $m > 46$.

[The analogous trees for $K_n \square P_3$ and $K_n \square C_3$ have maximum level 52.]

104. Here is one of several possibilities: (f = 15, e = 14)



The nodes on level 2 have respectively (5, 6, 3, 3, 5, 6, 3) children; and they lead to respectively (60, 49, 29, 23, 47, 63, 13) solutions on level 16. (Left-right symmetry must still be broken below the rightmost node: Use column 1, not column 3, on level 3.)

105. For example, the numbers are 1, 177, 12754, 164273 for $n = 1, 2, 3, 4$; and an instance of $K_6 \square P_4$ is exhibited in Fig. 108. But by extending the method used for $r = 3$, it appears likely that $K_n \square P_4$ will be ungraceful for all sufficiently large n.

106. Applying Algorithm R to this 99-edge graph quickly yields many solutions(!), such as

$$\begin{pmatrix} 31 & 41 & 59 & 26 & 53 & 58 & 97 & 93 & 23 & 84 & 62 & 64 & 33 & 83 & 27 & 95 & 02 \\ 25 & 71 & 19 & 77 & 17 & 86 & 08 & 81 & 65 & 91 & 37 & 99 & 01 & 98 & 04 & 87 & 22 \\ 68 & 24 & 10 & 29 & 74 & 45 & 07 & 18 & 89 & 05 & 96 & 00 & 88 & 03 & 80 & 13 & 94 \end{pmatrix}.$$

recursion
breaks reflection symmetry
Historical notes
Petrie
Smith
symmetry
Smith
Puget
KP
KC
KC graphs

[Smaller examples are also of interest. Consider, for example,

$$\begin{pmatrix} 3 & 14 & 15 & 9 & 26 \\ 22 & 21 & 0 & 27 & 13 \\ 7 & 2 & 25 & 1 & 4 \end{pmatrix}, \quad \begin{pmatrix} 3 & 14 & 15 & 9 & 26 & 5 \\ 32 & 0 & 33 & 2 & 11 & 7 \\ 19 & 22 & 10 & 29 & 1 & 31 \end{pmatrix}, \quad \begin{pmatrix} 3 & 14 & 15 & 9 & 26 & 5 & 35 \\ 12 & 0 & 39 & 1 & 30 & 20 & 4 \\ 39 & 36 & 2 & 34 & 7 & 25 & 32 \end{pmatrix},$$

where there are respectively 1, 3, and 16 solutions having those top rows prescribed.]

107. Let $x_{1(2k+1)} = m - x_{2(2k+1)} = 4k$ and $x_{3(2k+1)} = m - 8k - 1$ for $0 \le k < \lceil r/2 \rceil$; $m - x_{1(2k)} = x_{3(2k)} = 4k - 2$ and $x_{2(2k)} = 8k - 3$ for $1 \le k \le \lfloor r/2 \rfloor$; here $m = 6r - 3$ is always odd. (These values are distinct; for example, the even numbers among them are $\{0, 2, \ldots, 2r-2\}$ together with about $1/4$ of the larger even numbers $\le 6r - 2$.)

The differences between rows 2 and 3 give the odd edges $\{1, 3, \ldots, 2r-1\}$. The other odd edges can be found in the differences between rows 1 and 2 or 3, and between adjacent columns of row 1. Finally, the even edges $\{m - (12k + \{1,3,5,7,9,11\})\}$ are all present too. [G. Suresh Singh, *National Academy Science Letters* **15** (1992), 193–194.]

108. Gracefulness is known, via Algorithm R, for $1 \le r \le 14$ at least (thanks to computations by the author and Filip Stappers).

109. $K_n \,\square\, C_r$ has $2rn!$ symmetries: We can reflect the corresponding matrix left↔right, and/or shift its columns cyclically, and/or permute its rows arbitrarily.

110. There are $\binom{n+1}{2} r$ edges; and $\binom{n+1}{2} \bmod 4 = (1, 2, 3, 0)$ when $n \bmod 8 = (1, 3, 5, 7)$. Thus $K_n \,\square\, C_r$ is ungraceful when $n \bmod 8 = 1$ and $r \bmod 4 \in \{1, 2\}$; when $n \bmod 8 = 3$ and $r \bmod 4 \in \{1, 3\}$; when $n \bmod 8 = 5$ and $r \bmod 4 \in \{2, 3\}$. (See Fig. 108 for the case $n = r = 5$. There's no restriction when $n \bmod 8 = 7$.)

111. The odd-degree vertices are those in the $r - 2$ "middle" cliques, if n is even; otherwise they're the ones in the two "extreme" cliques. This observation can sometimes be used to prune the search tree by ruling out partial solutions whose odd-degree vertices have all been labeled. For example, when proving that $K_6 \,\square\, P_3$ has a unique labeling, it decreases the tree size from 225 meganodes to less than 213 meganodes (about 95%).

112. The method of Fig. 107 shows, in fact, that $K_4 \,\square\, K_4$ has eleven different graceful labelings, one of which is shown here. (It needs only 3 Gμ to discover this, with a search tree of 12 million nodes. It needs 0.76 and 190 Tμ to prove that $K_5 \,\square\, K_4$ and $K_5 \,\square\, K_5$ are *not* graceful.)

$$\begin{pmatrix} 48 & 0 & 39 & 35 \\ 1 & 17 & 47 & 23 \\ 38 & 45 & 7 & 2 \\ 19 & 20 & 5 & 46 \end{pmatrix}$$

113. No; $K_2 \oplus K_2 \oplus K_2 \oplus K_2$ can't be graceful because it has 8 vertices. (But every graph with four edges and ≤ 5 vertices *is* graceful; see the list following Theorem S.)

115. (a) This is the mixed-radix representation $\pi = \left[\begin{smallmatrix} 3; & a_1, & a_2, & a_3, & \ldots \\ 1; & 2, & 3, & 4, & \ldots \end{smallmatrix}\right]$; see 4.1–(9). The recurrence $x_1 = \pi - 3$, $a_n = \lfloor (n+1)x_n \rfloor$, $x_{n+1} = (n+1)x_n - a_n$ yields $(a_1, \ldots, a_{20}) =$ (0, 0, 3, 1, 5, 6, 5, 0, 1, 4, 7, 8, 0, 6, 7, 10, 7, 10, 4, 10) [OEIS A075874].

(b) (0, 0, 0, 1, 0, 1, 1, 2, 2, 1, 2, 1, 1, 2, 1, 3, 2, 4, 3, 5) isolated; (1, 1, 1, 2, 2, 2, 3, 3, 3, 2, 3, 3, 3, 4, 3, 5, 3, 5, 4, 6) components. [These 20 graphs are all planar.]

(c) $\chi(G_m^\pi) = 2$ for $m \in \{1, 2, 3, 9, 10, 12, 15, 17\}$; $\chi(G_m^\pi) = 3$ for the other $m \le 20$.

[From this data we might be tempted to conjecture that a "random graceful labeling," with $m \to \infty$ edges, is a.s. planar, and 3-colorable. But F. Stappers has studied G_m^π for $m \le 10000$, and found them *nonplanar* for $m = 33$, 38, 41, 44, 46–49, 51–52, 54–56, 58–61, and all cases ≥ 63. On the other hand, they're all 3-colorable.]

116. While generating the 16! instances, as in the proof of Theorem S, we can maintain connectivity information, because the steps of union-find are easily undone (see Algorithm 2.3.3E). We get $\frac{\text{connected}}{\text{total}} = (\frac{864}{864}, \frac{1141312}{1141312}, \frac{159551124}{159601936}, \frac{6537511962}{6562523200}, \frac{106698003000}{108536168696},$

Suresh Singh
author
Stappers
mixed-radix representation
OEIS
planar
Stappers
union-find

$\frac{795992914532}{838037875584}$, $\frac{2869123162654}{3252044834968}$, $\frac{4974721374674}{6508147089024}$, $\frac{3859250594040}{6590461997960}$, $\frac{1104325114202}{3099651627904}$, $\frac{67540932632}{519187026552}$) for $7 \le n \le 17$. (Divide all numerators and denominators by 2 to avoid complement symmetry. Values for smaller m are tabulated in OEIS A329790.)

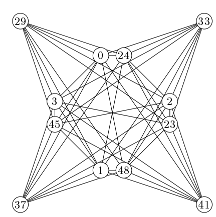
117. This goes faster, because the union-find algorithm can be modified to detect the creation of an odd cycle as soon as it occurs (see Section 7.4.1.1). The new counts are $\frac{8}{8}$, $\frac{22242}{22242}$, $\frac{6317382}{6318302}$, $\frac{427805408}{428781978}$, $\frac{10110694366}{10233657368}$, $\frac{99592576642}{103635506314}$, $\frac{432843270752}{479912612982}$, $\frac{796114433250}{1009922060716}$, $\frac{516439259812}{876211145722}$, $\frac{67540932632}{234013536424}$, for $8 \le n \le 17$; 2714363642056 altogether ($\approx 0.1297 \cdot 16!$).

Incidentally, there are 11932174 graphs with 16 edges and at most 17 vertices, of which 915503 (about 7.67%) are bipartite.

[When the labelings are also restricted to be α-graceful, in the sense of exercise 138, the results become $\frac{6}{6}$, $\frac{6840}{6840}$, $\frac{1855942}{1856280}$, $\frac{124467512}{124746754}$, $\frac{2945525928}{2980811422}$, $\frac{29277794448}{30452911120}$, $\frac{128904318498}{142798046522}$, $\frac{240333763962}{304499321272}$, $\frac{157722174046}{267381496426}$, $\frac{20772768256}{72154842584}$; 820394039226 altogether ($\approx .0392 \cdot 16!$).]

118. Such a graph must have $n = 2m/r$ vertices; so $2m/r$ must be an integer $> r$. We can proceed as in Theorem S and exercise 116, but prune the search by disallowing partial solutions with more than n nonisolated vertices, or with any vertex of degree $> r$.

Examples for small r are easy, and unique: K_3 when $r = 2$, K_4 when $r = 3$, and the octahedron $K_{2,2,2}$ when $r = 4$. There are six labelings when $(m, r) = (20, 5)$: Two of them give $\overline{C_8}$, the other four give $\overline{C_3 \oplus C_5}$. Similarly, $(m, r) = (27, 6)$ yields two graceful labelings of $\overline{C_9}$. A *unique* labeling appears for $(m, r) = (35, 7)$; its graph is $\overline{C_3 \oplus C_7}$.



When $r = 8$ we must go up to $m = 48$. Here there are 14 graceful labelings, for eight different graphs. The most symmetrical solution, shown here, has a graph with 384 automorphisms.

(All of these computations are short; but other methods are needed for $r > 8$. See E. Pegg Jr., `math.stackexchange.com/questions/3246000` (2019), and OEIS A308722. Pegg conjectures that the smallest instances for $r = 2k > 2$ occur when $m = 3k^2$.)

119. A 2-regular graph with m edges is a disjoint union of cycles, having a total of m vertices. The number of graceful labelings for $m = 3, 4, \ldots, 16$, with $0 — (m-1)$, is 1, 1, 0, 0, 7, 18, 0, 0, 175, 414, 0, 0, 7602, 20846. (Corollary E explains the zeros.)

It's easy to find the cyclic components of any given labeling; so we can identify isomorphic graphs among those labelings. There are $[z^m]\, 1/\prod_{n\ge 3}(1 - z^n)$ different 2-regular graphs with m edges; hence the potential numbers of graceful 2-regular graphs, for those values of m, are respectively 1, 1, 0, 0, 2, 3, 0, 0, 6, 9, 0, 0, 17, 21. The actual numbers turn out to be 1, 1, 0, 0, 2, 3, 0, 0, 5, 8, 0, 0, 14, 19. Missing are $2C_3 \oplus C_5$ (that is, $C_3 \oplus C_3 \oplus C_5$); $4C_3$; $5C_3$, $3C_3 \oplus C_6$, $3C_5$; $3C_3 \oplus C_7$, $2C_3 \oplus 2C_5$.

[In *Utilitas Mathematica* **7** (1975), 263–279, A. Kotzig proved that tC_5 is ungraceful for all $t \ge 1$. And in *Congressus Numerantium* **44** (1984), 197–219, he showed that a graceful 2-regular graph with t odd components must have at least $t(t+2)$ vertices. These results account for all of the missing cases listed above, except for $3C_3 \oplus C_6$. On the other hand he showed that $C_3 \oplus C_5 \oplus \cdots \oplus C_{2t+1}$ *is* graceful, for all $t \ge 1$. And with J. Abrham, he also proved that $C_p \oplus C_q$ is graceful if and only if $(p+q) \bmod 4 \in \{0, 3\}$; see *Discrete Mathematics* **150** (1996), 3–15.]

Incidentally, a gracefully labeled 2-regular graph always leaves one label $\in [0\,.\,.\,m]$ unused. The unused label was respectively (4, 5, ..., 12) in the case $m = 16$ exactly (311, 1547, 3208, 3510, 3651, 3532, 3241, 1554, 292) times.

complement symmetry
OEIS
α-graceful
unique
octahedron
C_n: cycle graph
symmetrical
Pegg Jr.
OEIS
cycles
Kotzig
Abrham

120. Now there are $m = 3t$ edges and $n = 2t$ nonisolated vertices, for $2 \le t \le 7$. The method of exercise 118 rapidly gives us graceful labelings galore, respectively (1, 5, 222, 22806, 2988280, 641731574) of them.

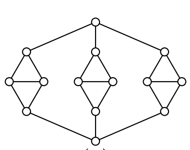
The main difficulty is to group them efficiently into equivalence classes of isomorphic graphs. One good way is to compute a "hash code" $h(G)$ for each graph G. Let r_1, r_2, ... be pseudorandom integers in the range $0 \le r_j < 2^{30}$, and let $r_0 = 0$. For each vertex v, compute $h(v)$ as follows: Let $V_k(v)$ be the set of vertices at distance k from v, and let $d(v)$ be the maximum k with $V_k(v) \ne \emptyset$. Set $t_w \leftarrow 2r_k + 1$ for each $w \in V_k(v)$. Then, for $k = d(v)$, $d(v) - 1$, ..., 0, compute $t'_w = t_w \prod_{u \text{—} w}(2r_{2d(v)+1-k} - t_u)$ for all $w \in V_k(v)$, and set $t_w \leftarrow t'_w$ for all such w. Let $h(v)$ be the product of all those values t'_w, mod 2^{32}. (Notice that $h(v)$ is always odd, and $h(v) = 1$ when v is isolated.)

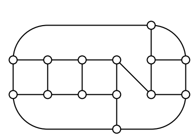
The hash code $h(G) = \bigl(\sum_v \lfloor h(v)/2 \rfloor\bigr) \bmod 2^{32}$, summed over all vertices v, now has the property that $h(G) = h(H)$ whenever graphs G and H are isomorphic. Furthermore, with trial and error we can find constants r_k for which $h(G) \ne h(H)$ whenever G and H are nonisomorphic cubic graphs with at most 14 nonisolated vertices.

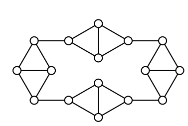
(The adjacency matrices for all connected cubic graphs with up to 24 vertices can be downloaded in a compact format from `hog.grinvin.org`, the "House of Graphs"; and the disconnected ones can be readily constructed from the connected ones. (See G. Brinkmann, K. Coolsaet, J. Goedgebeur, and H. Mélot, *Discrete Applied Math.* **161** (2013), 311–314.) For example, there are 509 connected cubic graphs with 14 vertices, and 540 altogether. In fact, the author's first try to choose random constants r_j actually was able to characterize uniquely every cubic graph with fewer than 20 vertices.)

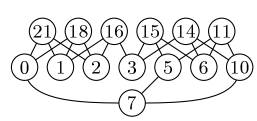
The bottom line is that *every cubic graph with at most* 14 *vertices is graceful, with only two exceptions:* $2K_4$ when $n = 8$ and $3K_4$ when $n = 12$. [A. Kotzig and J. Turgeon proved that the graph tK_n is graceful if and only if $t = 1$ and $n \le 4$; see *Colloquia Mathematica Societatis János Bolyai* **18** (1976), 697–703.] In fact, none of the *connected* cubic graphs are the least bit difficult to label; the two "least graceful" such graphs when $n = 14$ are graph $(*)$ below, with 9526 labelings and 96 automorphisms, and the Heawood graph 7–(57), with 10436 labelings and 336 automorphisms. (The disconnected graph $2K_4 \oplus (K_3 \square P_2)$, with 13824 automorphisms, has only 11 graceful labelings.) The "most graceful" of the 14-vertex cubics has 3762313 labelings(!) and only the identity automorphism; it's $(**)$ below.

Suppose we prespecify the labels $0 = l_0 < l_1 < \cdots < l_{n-1} = m$ that are to be used. Then a cubic graceful labeling is the solution to the MCC problem whose primary items are $\#1$, ..., $\#m$ and l_0, ..., l_{n-1}, where the l's have multiplicity 3; the options are simply '$\#k\ l_i\ l_j$' for $0 \le i < j < n$, where $k = l_j - l_i$. We can assume that $l_{n-2} = m-1$, and disallow '$\#(m-1)\ 1\ m$'. It turns out that only 27028 of the $\binom{19}{11} = 75582$ choices for the l's have solutions. The one for labels $\{0, 1, 2, 3, 5, 6, 7, 10, 11, 14, 15, 16, 18, 21\}$ is unique (see $(****)$ below); but $\{0, 1, 2, 3, 5, 6, 10, 11, 16, 17, 18, 19, 20, 21\}$ has 455698 solutions.


$(*)$


$(**)$


$(***)$

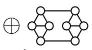


$(****)$

121. With considerably more computation, the results of exercise 120 can be extended to the 204,154,267,353 graceful labelings of cubic graphs on 16 vertices. There are 4207 such graphs, of which 4060 are connected. The evidence is overwhelming: Each of

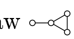
hash code
isomorphism clustering
adjacency matrices
House of Graphs
internet
Brinkmann
Coolsaet
Goedgebeur
Mélot
author
Kotzig
Turgeon
Heawood graph
MCC problem

the connected ones has at least 107,291 essentially different graceful labelings. (That "least graceful" example is (∗∗∗) above.) From this circumstantial evidence, the author conjectures confidently that every connected cubic graph is graceful.

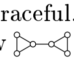
Furthermore, all 147 of the *disconnected* cubic graphs on 16 vertices are also graceful, except of course for $4K_4$. The closest to being ungraceful are $2K_4 \oplus$  (with 213 labelings) and $2K_4 \oplus P_2 \square P_2 \square P_2$ (with 1149). With only a bit of trepidation we may therefore conjecture that *every* cubic graph is graceful, except for $2K_4$, $3K_4$, ...

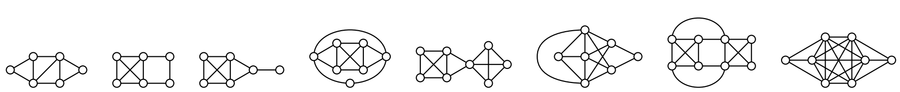
123. Backtracking via Theorem S, as in exercise 116, we can avoid most of the $m!/2$ cases by allowing at most 8 of the vertices $\{0, 1, \ldots, m\}$ to touch an edge. Thus we readily discover that the (1, 2, 7, 23, 122, 888, 11302) distinct graphs with $n =$ (2, 3, . . . , 8) nonisolated vertices have respectively (1, 2, 13, 157, 3292, 110578, 5903888) different graceful labelings. (Complementary labelings are not considered different.)

All graphs with at most 8 nonisolated vertices can be found in the House of Graphs. And the hash function in answer 120, but with different r_j, works for them.

One of the seven graphs with $n = 4$ nonisolated vertices, $2K_2$, doesn't have enough edges to be graceful. But the text points out that the other six work out fine (indeed, uniquely for $K_{1,3}$, P_4, C_4, and K_4, and up to 5 ways with the paw ).

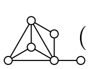
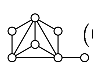
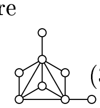
When $n = 5$, $K_2 \oplus P_3$ has too few edges; $K_2 \oplus K_3$ can't be labeled either; and Corollary E rules out C_5 and K_5, as well as $K_1 — 2K_2$. The other 18 are graceful: $K_{1,4}$ uniquely, and the "dart" $K_1 — (K_1 \oplus P_3)$ maximally (26 ways).

Cases $n =$ (6, 7, 8) lose respectively (4, 7, 19) graphs with too few edges, and (4, 20, 93) graphs that violate Corollary E. But they do include (109, 845, 11124) graceful graphs. Of course $K_{1,n-1}$ is always uniquely graceful. The other unique cases for $n = 6$ are $K_2 \oplus K_4$, $K_{3,3}$, $K_{2,2,2}$, and the double paw . The other unique cases for $n = 7$ are mostly disconnected: $P_2 \oplus L_{3,2}$, $P_3 \oplus C_4$, $C_3 \oplus P_4$, $C_3 \oplus C_4$, $C_3 \oplus L_{3,1}$, $P_3 \oplus K_4$, $K_3 \oplus K_{1,1,2}$, $K_2 \oplus K_5$; the connected one is $K_1 — (2K_1 — 2K_2)$. (Here $L_{m,n}$ denotes the "lollipop graph" on $m + n$ vertices, consisting of K_m and P_n joined by a bridge; $L_{3,1}$ is the paw.) There are 10 disconnected uniquely graceful graphs for $n = 8$: $K_2 \oplus C_6$, $2K_2 \oplus K_{1,1,2}$, $P_3 \oplus C_5$, $C_3 \oplus P_5$, $K_{1,3} \oplus L_{3,1}$, $2K_2 \oplus K_4$, $K_3 \oplus L_{4,1}$, $K_{1,3} \oplus K_4$, $P_3 \oplus K_5$, $K_3 \oplus (P_2 — P_3)$. And the 19 connected ones likewise have lots of symmetry: $K_{1,7}$, G_{14}, $2K_1 — 3K_2$, G_{16}, $4K_1 — 2K_2$, $2K_1 — (K_2 \oplus K_4)$, $K_2 — 2K_3$, $K_1 — G_{13}$, $K_1 — (2K_1 — (K_2 \oplus K_3))$, $2K_1 — K_{3,3}$, $K_3 — (K_1 \oplus 2K_2)$, $K_3 — (P_2 \oplus P_3)$, $K_3 — (2K_1 \oplus K_3)$, G_{21}, $K_1 — G'_{14}$, $2K_1 — G_9$, $K_2 — G'_9$, $K_2 — G''_9$, $K_3 — (K_1 \oplus C_4)$, where G_m or G'_m or G''_m denotes a special graph with m edges:



G_9 G'_9 G''_9 G_{13} G_{14} G'_{14} G_{16} G_{21}

The champions for gracefulness with 6, 7, and 8 vertices are

 (126 labelings),  (680 labelings), and  (3778 labelings).

125. (a) No, because edge 11 ($3 — 14$, `NC` — `SC`) doesn't touch edges 12–18 (see (33)).

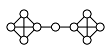
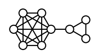
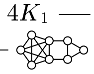
(b) 11067 (including the solution to Fig. 105(d)).

(c) A rooted labeling always defines a connected graph. We get n nonisolated vertices in respectively (864, 1122012, 148696974, 5469393230, 75003795230, 436515974020,

author
House of Graphs
paw
dart
K_n
$K_{m,n}$
P_n
C_n
$L_{m,n}$
lollipop graph
bridge
paw

1132397252122, 1296227076156, 605872421102, 94984144008, 2895168460) cases, for $7 \le n \le 17$. The total, 3649515044178, is approximately 17.4% of 16!.

(d) 1, 1, 1, 1, 2, 3, 1, 3, 3, 4, 5, 7, 3, 3, 15, 4. (See OEIS A338988 for further values. No pattern is evident. Does this sequence grow exponentially?)

126. (See exercise 123.) The *only* example with at most 8 vertices is $4K_1 \mathbin{—} 2K_2$. (And the only examples with 9 vertices are , , $K_1 \mathbin{—}$ , and $K_1 \mathbin{—} (2K_1 \mathbin{—} (K_2 \oplus (2K_1 \mathbin{—} K_2)))$; these are just four of the 259614 connected graceful graphs. The first of these is the only example with at most 14 edges.)

128. There are four with 0 at the Y: $\begin{smallmatrix}2&16\\8&15\end{smallmatrix}\,0\,1$; $\begin{smallmatrix}2&16\\14&1\end{smallmatrix}\,0\,15$, $\begin{smallmatrix}8&16\\11&7\end{smallmatrix}\,0\,15$; $\begin{smallmatrix}8&15\\9&1\end{smallmatrix}\,0\,16$. There's one with 15 at the Y: $\begin{smallmatrix}6&9\\12&3\end{smallmatrix}\,15\,0$. There are nine with 16 at the Y, such as $\begin{smallmatrix}3&6\\5&10\end{smallmatrix}\,16\,0$. And 33 with other elements at the Y, such as $\begin{smallmatrix}4&1\\2&3\end{smallmatrix}\,5\,0$ and $\begin{smallmatrix}8&4\\3&6\end{smallmatrix}\,12\,0$. Total 47.

129. (a) There are $k+1$ components and k residues.

(b) If r is bad and $x \bmod k = r$, then we clearly can't set `LO[k]` $\leftarrow x$. But if r is good, at least one such x is OK.

(c) Say that x is a big vertex if $x+k > m$. There are g big good vertices, lying in $\le g$ components. The largest good vertices in the other good components are OK.

(d) The vertices $\{r, r+k, \ldots, r+pk\}$ can't be connected by p edges of lengths $> k$.

(e) The $k+1-G$ bad components account for at least $2(k+1-G)$ bad residues, by (d). Hence $g \le k-2(k+1-G)$ and we have $G-g \ge k+2-G$. If $G \le \frac{2}{3}(k+2)$ we have $G-g \ge (k+2)/3$; otherwise either g or $G-g$ is $\ge G/2 > (k+2)/3$. Thus $\lceil (k+2)/3 \rceil = \lfloor (k+4)/3 \rfloor$ is a valid lower bound in all cases, by (b) and (c). [Experiments for $m \le 20$ suggest that $t_k = \lfloor (k+3)/2 \rfloor - [k \text{ odd and } k = \lceil m/2 \rceil - 2 > 1]$ may in fact be valid.]

(f) When $k \ge m/2$, all edges connect small to big. The hint follows because the cycle containing x and $x+k$ includes the edges $y \mathbin{—} (x+k) \mathbin{—} x \mathbin{—} z$.

Let there be c unusable vertices, in C components. A component that contains $q > 0$ unusable vertices $x_1 < \cdots < x_q$ therefore contains at least the $2q+2$ vertices $y_1 < x_1 < \cdots < x_q < x_1 + k < \cdots < x_q + k < z_q$, and it contains at least $2q+1$ of the $m-k$ edges. Consequently $m-k \ge 2c+C$; and the number of usable vertices is $m+1-k-c \ge (m-k)/2 + 1 + C/2 \ge 2 + \lfloor (m-k)/2 \rfloor$, unless $C = c = 0$.

[Altogether we get the superexponential lower bound $t_1 \ldots t_m = \Omega(m!/24^{m/2})$.]

130. (a) For example, when $n=4$ it's $\det\begin{pmatrix} x_1{+}x_1{+}x_2 & -x_1 & -x_2 \\ -x_1 & x_2{+}x_1{+}x_1 & -x_1 \\ -x_2 & -x_1 & x_3{+}x_2{+}x_1 \end{pmatrix}$.

(b) The sum of $s_2 \ldots s_{n-1} S(1, s_2, \ldots, s_{n-1})$ over all 2^{n-2} choices of $s_j = \pm 1$ is 2^{n-2} times the desired result. For example, when $n=4$ we have $[x_1x_2x_3]\, S(x_1,x_2,x_3) = (S(1,1,1) - S(1,1,-1) - S(1,-1,1) + S(1,-1,-1))/4$. [See OEIS A033472.]

131. Empirical investigations by D. Anick suggest that $\tau(n)/\tau(n-1)$ grows approximately as $a + bn + (-1)^n c/n$ for some constants a, b, c. If that is true, $\tau(n) = \exp\bigl(n \ln n - n \ln(e/b) + O(\log n)\bigr)$. The exact values for $n < 30$ suggest further that $a \approx 0.19$, $b \approx 0.636$, and $c \approx 0.42$. But rigorous proofs are unknown. (This function $\tau(n)$ was introduced by A. Kotzig, who computed it by hand for $n \le 6$ in 1984.)

132. Suppose $1 \le e < 2^n$, where $2^n + e = (e_n \ldots e_1 e_0)_2$. Then the edge labeled e is between $x = (x_{n-1} \ldots x_1 x_0)_2$ and $x \mathbin{\&} (x-1)$, if $e_k = 1$ and $e_{k-1} = \cdots = e_0 = 0$ and $x_j = e_j \oplus [j > k] e_{j+1}$ for $0 \le j < n$. (This is in fact an α-labeling. Notice that $l(x)$ is essentially a left-right reflection of inverse Gray code, 7.2.1.1–(8).)

133. Notice that T_n, like P_n, has two automorphisms; so we divide the total number of graceful labelings by 4. This yields 30 and 988184 for T_3 and T_4; also approximately $4 \cdot 10^{18}$ and 10^{48} for T_5 and T_6, using ten million estimates with Algorithm 7.2.2E.

OEIS
OEIS
Anick
Kotzig
α-labeling
Gray code
automorphisms

136. (a) Suppose α has even parity and β has odd parity. Then $l(1\beta) - l(1\alpha) = l(0\beta) - l(0\alpha) - 2^{n-2} - 2a_{2^{n-2}}$, because $a_0 = 0$. Hence $L_1 = L_0 - 2^{n-2} - 2a_{2^{n-2}}$.

(b) Let $a_{2^k} = (k+2)2^{k-1}$. This choice makes $(a_0, a_1, \ldots) = (0, 1, 3, 4, 8, 9, \ldots)$, and we have $a_n = \sum_{k=1}^{n} 2^{\rho k}$ for all n. (It can be shown that $a_n = n + (e_1 2^{e_1} + \cdots + e_t 2^{e_t})/2$ when $n = 2^{e_1} + \cdots + 2^{e_t}$ with $e_1 > \cdots > e_t \geq 0$.) By part (a), $L_0 = L_1 + 2^{n-2} + 2a_{2^{n-2}} = L_1 + (n+1)2^{n-2}$. The other edges $0\alpha \longrightarrow 1\alpha$ have labels

$$\{m - k - a_k - a_{2^{n-1}-1-k} \mid 0 \leq k < 2^{n-1}\} = \{m - k - (n-1)2^{n-2} \mid 0 \leq k < 2^{n-1}\},$$

because $a_k + a_{2^{n-1}-1-k} = a_{2^{n-1}-1} = (n-1)2^{n-2}$. Thus $L_1 = \{1, 2, \ldots, (n-1)2^{n-2}\}$ by induction; and it all works, α-gracefully. [M. Maheo, *Discrete Mathematics* **29** (1980), 39–46; A. Kotzig, *Journal of Combinatorial Theory* **B31** (1981), 292–296.]

137. (a) $n = \sum_{k=0}^{r}(t_k - s_k + 1)$ vertices; $n - r - 1$ vertical plus $n - t_r - 1$ horizontal edges.

(b) Numbers in ovals don't change; in rectangles they're subtracted from 28.

(c) Use a rectangle for (x, y) when $x + y$ is odd. Label $(0, 0)$ with 0. For each edge, proceeding left to right and bottom to top, make the labels of its endpoints sum respectively to 0, 1, 2, ... (This will make the label in a rectangle equal to the one below it, and one less than the one above it, when those neighbors exist.)

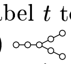
(d) Yes! In general let $\Sigma_0 = t_0$, $\delta_0 = 0$, and $\Sigma_{k+1} = \Sigma_k + t_k + t_{k+1} - 2s_{k+1} + 1$, $\delta_{k+1} = \Sigma_k - \delta_k - s_{k+1}$, for $0 \leq k < r$. Then the label of (x, y) corresponding to (i) is $\delta_x + \lfloor y/2 \rfloor$ when x is even, $\delta_x + \lceil y/2 \rceil$ when x is odd.

[This in fact is an instance of α-labeling as in exercise 138, where the u's are ovals and the v's are rectangles. A. Rosa presented a special case in Lemma 4.3 of his thesis.]

138. (a) We have $\overline{v_k} = m - v_k \geq m - (m - l) = l > u_j$. Hence all the complemented labels exceed all the uncomplemented ones, and $|u_k - \overline{v}_k| = m - v_k - u_k = m - k$ for all k.

(b) Since C_n has n edges, Corollary E tells us that $n \bmod 4$ must be 0 or 3. But a bipartite graph has no odd cycles; hence $n = 4k$. Conversely, the labels $0 \longrightarrow \overline{1} \longrightarrow 1 \longrightarrow \overline{2} \longrightarrow 2 \longrightarrow \overline{3} \longrightarrow 4 \longrightarrow \overline{4} \longrightarrow 5 \longrightarrow \overline{5} \longrightarrow 6 \longrightarrow \overline{0} \longrightarrow 0$ for C_{12} reveal a general pattern that works for all $k > 0$. (The similar non-α pattern $0 \longrightarrow 11 \longrightarrow 1 \longrightarrow 10 \longrightarrow 2 \longrightarrow 9 \longrightarrow 4 \longrightarrow 8 \longrightarrow 5 \longrightarrow 7 \longrightarrow 0$ for C_{11} shows that C_{4k+3} is at least graceful, for all $k \geq 0$.)

(c) Let $u_k^* = u_{m-1} - u_{m-1-k}$ and $v_k^* = v_{m-1} - v_{m-1-k}$. (Equivalently, change each vertex label t to $(l - 1 - t) \bmod (m+1)$. Notice that $u_{m-1} = l - 1$ and $v_{m-1} = m - l$.)

(d) . [A. Rosa introduced α-graceful graphs, and solved these problems as well as exercise 145(b), in his original paper that introduced graceful graphs. His thesis (1965) credited A. Kotzig for part (b).]

139. (a) This is obvious, because $u_k < l \leq \overline{v_k}$.

(b) We can assume that the central vertex is labeled 0. Then there are two solutions, both with leaves labeled $\{1, 2, 4\}$.

(c) Suppose the edges of G are $u_1 \longrightarrow v_1, \ldots, u_m \longrightarrow v_m$, and the vertices of $K_{m,m}$ are $\{a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1}\}$. We use a "cyclic" analog of the rainbow copies of K_{11} in Fig. 110(c): Let the edges of the k-th copy be $\bigl(l(u_j)+k\bigr) \bmod m \longrightarrow \bigl(l(v_j)+k\bigr) \bmod m$, for $1 \leq j, k \leq m$. (For example, the three copies of the path $0 \longrightarrow 3 \longrightarrow 1 \longrightarrow 2$ in $K_{3,3}$ are $a_1 \longrightarrow b_1 \longrightarrow a_2 \longrightarrow b_0$, $a_2 \longrightarrow b_2 \longrightarrow a_0 \longrightarrow b_1$, $a_0 \longrightarrow b_0 \longrightarrow a_1 \longrightarrow b_2$.)

(d) Simply let the ith copy v_i of vertex v have label $l(v) + (i-1)m$. (Consequently, by (c), we can perfectly pack $t^2 m$ copies of G into $K_{tm,tm}$; also $(2tm+1)t$ into K_{2tm+1}.)

Historical notes: I. Cahit proposed the concept of ordering labeling in an unpublished research report at the University of Waterloo [CORR 80-47 (December 1980), 6 pages]. It was introduced independently by S. I. El-Zanati, M. J. Kenig, and C. Vanden Eynden, in *Australasian J. Combinatorics* **21** (2000), 275–285, who

parity
ruler function ρ
α-gracefully
Maheo
Kotzig
Rosa
Rosa
historical notes
Kotzig
Historical notes
Cahit
El-Zanati
Kenig
Vanden Eynden

also showed that the graph $S_{n,2}$ with $2n$ edges $0 — a_j — b_j$ for $1 \le j \le n$ has an ordered graceful labeling; that graph isn't α-graceful when $n > 2$.

140. (a) $\sum_{l=1}^{m} \prod_{k=0}^{m-1} \bigl(\min(k+1,l) - \max(0,k+l-m)\bigr)$, since the choices for each k are independent and since $u_{m-1} = l-1$. (See OEIS A005193. Sheppard proved this when he introduced Theorem S). The values for $2 \le m \le 8$ are 2, 4, 10, 30, 106, 426, 1930.

(b) No simple formula is evident. The values are now 2, 4, 12, 40, 182, 906, 5404. When $m = 16$ there are 246,377,199,752, compared to 7,614,236,170 for (a).

Divide by 2 if complementary labelings are considered to be equivalent.

142. (a) If the elements of $K_{a,b}$ and $K_{c,d}$ are respectively u_i, v_j and x_k, y_l, the elements of $K_{a,b} \otimes K_{c,d}$ are $u_i x_k$, $u_i y_l$, $v_j x_k$, $v_j y_l$, for $1 \le i \le a$, $1 \le j \le b$, $1 \le k \le c$, $1 \le l \le d$. The edges are $u_i x_k — v_j y_l$, $u_i y_l — v_j x_k$, so the product is $K_{ac,bd} \oplus K_{ad,bc}$.

(b) (i) Think of the black or white squares of the $2m \times 2n$ chessboard, connected by bishop moves. Rotate by 90° to get either an $(m+n) \times (m+n-1)$ or $(m+n-1) \times (m+n)$ board, connected by rook moves, but with right triangles removed from the corners. These right triangles affect $m-1$ rows/columns at the upper left and lower right; they affect $n-1$ rows/columns at the lower left and upper right.

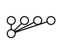
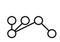
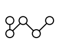
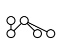
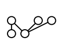
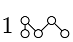
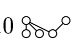
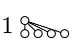
(ii) Now the board is $(m+n) \times (m+n)$, with n (not $n-1$) rows/columns affected at the upper left or lower right. Again the two graphs are isomorphic by transposition.

(iii) One of the graphs has $\lfloor (2m+1)(2n+1)/2 \rfloor$ vertices; it's an $(m+n) \times (m+n)$ board with $m-1$ and $n-1$ rows/columns affected at corners. The other, with one more vertex, is an $(m+n+1) \times (m+n+1)$ board with m and n rows/columns affected. [When $m = n$ these are the Aztec diamonds of orders n and $n + 1/2$.]

(iv) Both are generalized toruses (exercise 7–137), with offsets $(m, -m)$ and (n, n).

(v) The graph whose vertices are binary vectors $x_1 \ldots x_m y_1 \ldots y_n$ of given parity. Each vertex has mn neighbors: Complement one of the x's and one of the y's.

(c) Complementing labels interchanges parts; so we need only consider $(G \otimes H)'$. Let G's parts (U, V) have labels $l(u)$, $l(v)$, and let H's parts (X, Y) have labels $l(x)$, $l(y)$. The new labels $l(ux) = l(u) + ml(x)$, $l(vy) = l(v) + ml(y) - m$ work beautifully, where m is the number of edges in G. [H. S. Snevily, *Discrete Math.* **170** (1997), 185–194.]

145. (a) 000  001  010  011  100  101  110  111 

(b) Use labels 0, 1, ..., s for u_0 through u_s and $\overline{0}$, $\overline{1}$, ..., $\overline{t}$ for v_0 through v_t.

(c) There are m_{s+t+1} edges, where $m_0 = 0$ and $m_{i+1} = m_i + p_{s_i} + q_{t_i} + 1$.

(d) Assign the label $a_j + i$ to each element u_{ji} of U_j, and the label $\overline{b_k + i}$ to each element v_{ki} of V_k, where $a_0 = b_0 = 0$ and $a_{j+1} = a_j + p_j + q_{j'} + 1$, $b_{k+1} = b_k + p_{k''} + q_k + 1$; here $j' = \max\{i \mid u_j — v_i\}$ and $k'' = \max\{i \mid u_i — v_k\}$. The labels are distinct because $a_{j+1} > a_j + p_j$, $b_{k+1} > b_k + q_k$. These definitions ensure that $a_{s_i} + b_{t_i} = m_i$; hence the edges of the caterpillar between U_{s_i} and V_{t_i} receive the labels $m - m_i$, $m - m_i - 1$, ..., $m - m_{i+1} + 1$. When $i = s + t$ we have $s_i = s$, $t_i = t$; the final edge label is 1. In the example, $(a_0, a_1, a_2) = (0, 6, 11)$ and $(b_0, b_1, b_2) = (0, 4, 10)$; see below.

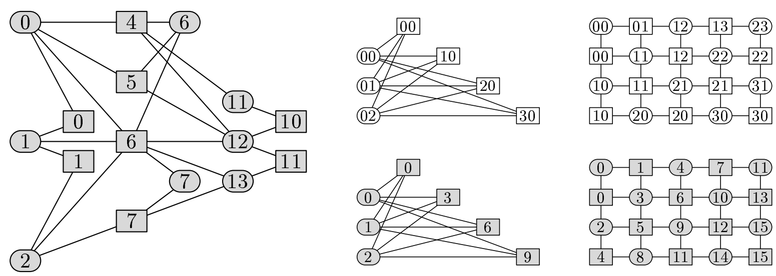
(e) Let $(s, t) = (0, r-1)$; this gives the caterpillar $K_{1,r}$, whose edges are $u_0 — v_0$, ..., $u_0 — v_{r-1}$. Then set $p_0 = n-1$ and $q_i = 0$ for $0 \le i < r$. (See the case $(3, 4)$ below.)

(f) Denote the grid points by (x, y) for $0 \le x < r$ and $0 \le y < n$. Let U_j be the points with $x + y = 2j$, and let V_k be the points with $x + y = 2k + 1$, as illustrated below for $n = 5$ and $r = 4$. The edges between $U_0 — V_0 — U_1 — V_1 — \cdots$ are staircase paths. (Hence this is a caterpillar net in which every caterpillar is simply a path. See B. D. Acharya and M. K. Gill, *Indian J. Math.* **23** (1981), 81–94.)

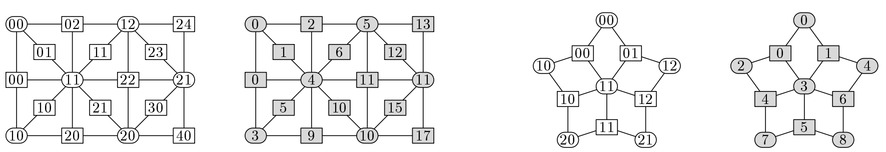
OEIS
Sheppard
chessboard
Aztec diamonds
generalized toruses
parity
Snevily
Acharya
Gill

In the following illustrations, digits ji in an oval signify u_{ji}; digits ki in a rectangle signify v_{ki}; shaded nodes show final vertex labels; shaded rectangles are complemented:

Acharya
roots of unity
factorization
intervals

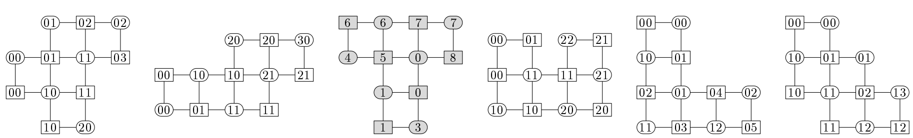


(g) Yes, they both are:



146. Exercise 137 applies to P, Q, V, W, and Z; but exercise 145 is stronger.

In fact, the skeletons of all but the T pentomino are caterpillar nets; the T does, however, have 1824 different α-graceful labelings. It's easy to decompose the others into small caterpillars, as in the decomposition of S below, thereby writing down a labeling quickly by hand — except that the (unique) decomposition of U is difficult to find. The R, V, and W also have surprising decompositions into rather large caterpillars:



[See B. D. Acharya, *Lecture Notes in Math.* **1073** (1984), 205–211.]

148. (a) $(\sum_{i=1}^{n} x^{a_i})(\sum_{j=1}^{r} x^{b_j}) = \sum_{k=0}^{m-1} x^k$ is an algebraic way to say that $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_r\}$ are nonnegative integers whose nr sums $a_i + b_j$ yield $\{0, \ldots, m-1\}$.

(b) Because the mth roots of unity are $e^{2\pi ik/m}$, the complete factorization of $(1-x^m)/(1-x)$ over the real numbers is $(1+x)^{[m\text{ even}]} \cdot \prod_{k=1}^{\lceil m/2\rceil -1}(1 - 2x\cos\frac{2\pi k}{m} + x^2)$. And any product of palindromials is a palindromial.

(c) Let $G(x) = g_0 + \cdots + g_c x^c$ and $H(x) = h_0 + \cdots + h_d x^d$. Clearly $g_0 = h_0 = 1$. Let k be minimal with $0 < g_k < 1$ or $0 < h_k < 1$; say $0 < g_k < 1$. Then $h_k = 0$, because $g_{c-k} = g_k$ and $g_{c-k}h_k + g_c h_0 \le 1$. But $g_k h_0 + g_{k-1}h_1 + \cdots + g_0 h_k = 1$, and all terms but $g_k h_0$ are 0 or 1. Contradiction.

(d) Since $g_1 + h_1 = 1$ we may assume that $g_1 = 1$. Then the nonzero coefficients of G can be written as a union of disjoint intervals $[a_0 \,.\,.\, a_0 + k_0) \cup [a_1 \,.\,.\, a_1 + k_1) \cup \cdots \cup [a_t \,.\,.\, a_t + k_t)$, where $a_0 = 0$ and $k_0 > 1$ and $a_{i+1} > a_i + k_i$. If we shift those intervals by s whenever h_s is 1, the union of all of the resulting disjoint sets is $[0 \,.\,.\, m)$.

Let $k = k_0$. Clearly $h_k = 1$. And we must have $k_i \le k$ for $0 \le i \le t$, to avoid overlap after shifting by k. Moreover, if $k_i < k$ for some i, where i is minimal, there will be a short gap between $a_i + k_i$ and $a_i + k$ that cannot be covered by any subsequent shift without overlap. Hence all $k_i = k$, and $T(x) = x^{a_0} + \cdots + x^{a_t}$.

(e) We have $G(1)=n$, $F_k(1)=k$, and $T(0)=H(0)=1$. So every nonzero term of T or H is a nonzero term of $T(x)H(x)=F_m(x)/F_k(x)=F_{m/k}(x^k)$.

(f) If $nr>1$, every factorization counted by $A(n,r)$ comes from one that's counted by $A(n/k,r)$ or by $A(n,r/l)$, for some $k\backslash n$ or some $l\backslash r$. In particular, $A(p^e,q^f)=A(p^{e-1},q^f)[e>0]+A(p^e,q^{f-1})[f>0]+[e=f=0]$. Hence $A(p^e,q^f)=\binom{e+f}{e}$.

(g) Let p_i denote the operation of dividing n by p_i, and let q_j denote the operation of dividing r by q_j. Then every permutation π of $\{p_1,p_2,q_1,q_2\}$ defines a factorization $F_m(x)=G_\pi(x)H_\pi(x)$, by the rules $G_{p_i\alpha}(x)=F_{p_i}(x)G_\alpha(x^{p_i}), H_{p_i\alpha}(x)=H_\alpha(x^{p_i})$; $G_{q_j\beta}(x)=G_\beta(x^{q_j}), H_{q_j\beta}(x)=F_{q_j}(x)H_\beta(x^{q_j})$; $G_\epsilon(x)=H_\epsilon(x)=1$. For example, $G_{p_1q_2p_2q_1}(x)=F_{p_1}(x)F_{p_2}(x^{p_1q_2})$, $H_{p_1q_2p_2q_1}(x)=F_{q_2}(x^{p_1})F_{q_1}(x^{p_1q_2p_2})$.

But we must avoid double-counting, because the operations $\{p_1,p_2\}$ and $\{q_1,q_2\}$ commute pairwise. There are 14 equivalence classes of permutations: $p_1p_2q_1q_2\equiv p_1p_2q_2q_1\equiv p_2p_1q_1q_2\equiv p_2p_1q_2q_1$, $p_1q_1p_2q_2$, $p_1q_1q_2p_2\equiv p_1q_2q_1p_2$, $p_1q_2p_2q_1$, $p_2q_1p_1q_2$, $p_2q_1q_2p_1\equiv p_2q_2q_1p_1$, $p_2q_2p_1q_1$, and seven more with $p\leftrightarrow q$. So $A(p_1p_2,q_1q_2)=14$.

(h) The Möbius polynomial for variables $\{p_1,\ldots,p_s,q_1,\ldots,q_t\}$, when the p's and q's commute pairwise, is $(1-p_1)\ldots(1-p_s)+(1-q_1)\ldots(1-q_t)-1$.

(i) $1/((1-q_1)\ldots(1-q_t)-p)=\sum_{e\ge0}p^e(1-q_1)^{-1-e}\ldots(1-q_t)^{-1-e}$.

[See M. Krasner and B. Ranulac, *Comptes Rendus Acad. Sci.* **204** (Paris, 1937), 397–399, as well as V. Senderov and A. Spivak, *Kvant* **29**, 1 (January–February 1998), 10–18, for comments on parts (b)–(d). N. Beluhov contributed to parts (a), (e), (f), (g), and (i). Beluhov has also discovered the amazing identity $A(p_1^ep_2^e,q_1^eq_2^e)=\sum_k(-1)^{e+k}\binom{2e}{k}^4$(!); see *Enumer. Combinatorics and Applic.* **2:1** (2022), #S2R6, 1–11.]

149. (a) Edges p through $2n$ are defined by the vertex labels already given. For the other $p-1$ edges we must choose the labels $2n-j$ or $2n-p+j$, for $1\le j\le\lfloor p/2\rfloor$; there are $2^{\lfloor p/2\rfloor}$ solutions. (For example, when $n=7$ there are two solutions with $\{14,11\}$ in the second part, and four with $\{14,9\}$. One of the former has $\{0,1,2,6,7,8,12\}$ in the first part; one of the latter has $\{0,1,2,3,4,11,13\}$.)

(b) The second part labels are $\{jn+k\mid 1\le j<r\}\cup\{nr\}$. For example, $K_{7,7}$ can be labeled with $\{0,1,2,3,4,47,48\}$ and $\{9,16,23,30,37,44,49\}$.

150. Not when $n,r\le 23$, according to calculations by F. Stappers. (Is $K_{n,n}$ *uniquely* graceful when $n=3k+2$ is prime?)

155. Primary items $\{1,\ldots,m\}$ for the arc labels, and m primary items vw for the arcs $v\longrightarrow w$. Also n secondary items v for the vertices, and $q=m+1$ secondary items $\{h_0,\ldots,h_m\}$ for the holders of arc labels. There are $(m+1)m^2$ options: '$((y-x)\bmod q)$ vw $v{:}x$ $w{:}y$ $h_x{:}v$ $h_y{:}w$', for each arc $v\longrightarrow w$ and each $x\ne y$ with $0\le x,y\le m$.

(We can greatly reduce the number of solutions by forcing some vertex v to be labeled 0, and forcing some other vertex w to be labeled with a divisor of q.)

156. $a=7$, $b=5$. (Subtract 3, then multiply by the inverse of $5-3$.)

157. Using exercise 155 we quickly (14 Mμ) discover exactly 48 solutions with $l(000)=0$ and $l(001)=1$. Each of them belongs to a set of 12 that are mutually equivalent, via automorphisms and antiautomorphisms followed by possible addition and multiplication, just as labelings (d) and (f) are obtained from (b) in Fig. 109. The four essentially different solutions are represented, lexicographically least, by $(l(000),\ldots,l(111))=(0,1,2,5,12,6,8,3)$, $(0,1,2,6,12,8,5,3)$, $(0,1,2,9,6,4,11,8)$, $(0,1,3,10,11,6,2,12)$.

160. (a) Let $d=\gcd(l(w)-l(v),q)$ and $q'=q/d$, so that $l(w)-l(v)=cd$ for some $c\perp q'$. There's a unique c' such that $0<c'<q'$ and $cc'\equiv1$ (modulo q').

permutation
Möbius polynomial
Krasner
Ranulac
Senderov
Spivak
Beluhov
amazing identity
Stappers
uniquely

There are d solutions to the simultaneous equations $\bigl(a\cdot l(v)+b\bigr) \bmod q = 0$ and $\bigl(a\cdot l(w)+b\bigr) \bmod q = d$, namely $a = a_k$ and $b = (-a_k \cdot l(v)) \bmod q$, where $a_k = c' + kq'$ and $0 \le k < d$. Hence we want to prove that $a_k \perp q$ for at least one value of k.

Say that the prime p is "in d" if $p\backslash q$ but $p\nmid q'$. (For example, if $d = 10$ and $q = 60$, then only 5 is in d.) We can write $d = rd'$, where the prime factors of r are in d but those of d' are not. If p divides $\gcd(a_k, q) = \gcd(c' + kq', q)$ it must be in d; otherwise it would divide q' but not c'. Therefore $\gcd(a_k, q) = \gcd(a_k, r)$. And the values of $a_k \bmod r$ for $0 \le k < d$ are d' copies of $\{0, 1, \ldots, r-1\}$, because $q' \perp r$.

(b) Exactly $d'\varphi(r) = d\prod_{p \text{ in } d}(1 - \frac{1}{p})$ graceful labelings are produced by that construction. Furthermore, different values of k give a different l': Let u and u' be the vertices for which $u \longrightarrow u'$ and $\bigl(l(u') - l(u)\bigr) \bmod q = 1$. Then $\bigl(l'_k(u') - l'_k(u)\bigr) \bmod q = a_k$.

(c) It suffices to find the essentially different cases that are *normalized*, in the sense that $l(v) = 0$ and $l(w)\backslash q$. Begin with the set of all normalized solutions (a), grouping the solutions for divisor d into equivalence classes of size $d\prod_{p \text{ in } d}(1 - \frac{1}{p})$ as in (b). Then, for each automorphism or antiautomorphism α, apply α to a representative of each class. If the result is in a different class, after normalization by an affine transformation, merge the classes. Repeat until no more merging is possible. (We need only consider enough α's to generate them all under composition.)

161. (a) Denote a labeling by the tuple $l(\mathrm{a})l(\mathrm{b})l(\mathrm{c})l(\mathrm{d})$. If we choose $v = \mathrm{b}$ and $w = \mathrm{c}$, the initial affine equivalence classes in answer 160(c) turn out to be $\{1024\}$, $\{4021\}$ for $d = 2$ and $\{1034, 5032\}$, $\{2035, 4031\}$ for $d = 3$, since there are no solutions for $d = 1$.

This digraph has two automorphisms, () and (b c). It also is self-converse, so it has two antiautomorphisms, one for each automorphism; they are (a d) and (a d)(b c).

Let $\alpha = (\mathrm{a\ d})$. Then $1024\alpha = 4021$ and $2035\alpha = 5032$; so the classes of equivalent labelings are $\{1024, 4021\}$ and $\{1034, 2035, 4031, 5032\}$ after the first step of merging.

Now let $\alpha = (\mathrm{b\ c})$. We have $1024\alpha = 1204$, which normalizes affinely to 1024. So no further merging occurs, and there are just two essentially distinct classes of equivalent solutions. (We needn't try $\alpha = (\mathrm{a\ d})(\mathrm{b\ c})$, which is generated by the others.)

(Alternatively, we could have chosen $v = \mathrm{a}$ and $w = \mathrm{b}$, say. Then the initial classes would have been $\{0143\}$, $\{0153\}$, $\{0243\}$, $\{0253\}$. The antiautomorphism (a d) would have merged them to $\{0143, 0253\}$ and $\{0153, 0243\}$. The automorphism (b c) would then have made no further change.)

(b) Choose $v = \mathrm{a}$ and $w = \mathrm{d}$, say, getting six initial classes $\{043125\}$, $\{015243\}$, $\{031245\}$, $\{034215\}$, $\{045213\}$, $\{053241\}$. The antiautomorphism (a f)(b e)(c d) merges them to $\{034215, 043125\}$, $\{015243\}$, $\{031245, 053241\}$, $\{045213\}$; four classes only.

164. Set `FIRST[`l`]` $\leftarrow -1$ for $0 \le l < q$. Then do the following steps for $l = 1, 2, \ldots, m$: Set $v \leftarrow$ `LO[`l`]`, $w \leftarrow (v + l) \bmod q$, $t \leftarrow$ `FIRST[`v`]`, `FIRST[`v`]` $\leftarrow w$, `NEXT[`l`]` $\leftarrow t$.

(A similar algorithm will create arrays `FIRSTP` and `NEXTP` with which all *predecessors* of any given vertex can be visited efficiently. We can also readily create `FIRST`, `NEXTL`, and `NEXTH` from the `LO` array of a graceful *undirected* graph.)

165. (a) Let $f(-1) = -1$, otherwise $f(x) = \bigl(a(x - b)\bigr) \bmod q$. Then `LO'[`$(al) \bmod q$`]` $=$ $f($`LO[`l`]`$)$; `FIRST'[`$(a(l - b)) \bmod q$`]` $= f($`FIRST[`l`]`$)$; `NEXT'[`$(al) \bmod q$`]` $= f($`NEXT[`l`]`$)$; `NAME'[`$(a(l - b)) \bmod q$`]` $=$ `NAME[`l`]`.

(b) `LO`, `FIRST`, and `NEXT` are unchanged; `NAME'[`l`]`α $=$ `NAME[`l`]`.

(c) `LO'[`$q - l$`]` $= \bigl($`LO[`l`]` $+ l\bigr) \bmod q$; `NAME'[`l`]`α $=$ `NAME[`l`]`; `FIRST'` and `NEXT'` must be computed from `LO'` using exercise 164.

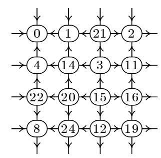
totient function
automorphisms
self-converse
antiautomorphisms
essentially distinct
NEXTL
graph representation

168. Now $q = 20$, and D^* has the same [anti]automorphisms as D. Choosing $v = 000$ and $w = 001$ in answer 160(c) yields respectively (46, 48, 14, 0, 0) affine classes for $d = (1, 2, 4, 5, 10)$; the classes for $d = 4$ are pairs of labelings, the others are singletons.

Automorphisms merge every class for $d > 1$ with at least one class for $d = 1$. So we can confine attention to the 23 labelings with $l(001) = 1$ and $l(010) < l(100)$.

An antiautomorphism finally leaves just seven classes: {`012acj5g`, `013cif28`, `016e745g`}, {`01649ehg`, `018aid54`, `0196edg4`}, {`0165icf8`, `01bec9ag`, `01becajg`}, {`0169ecjg`, `01bac568`, `01bac6f8`}, {`016acfb8`, `016j9ceg`, `0198e6b4`}, {`01358ife`, `014eb976`, `014hbje6`, `017fg56i`}, {`0135i8fe`, `01657fgi`, `01b9e476`, `01bjeh46`}. (Here the extended hexadecimal digits `0` through `j` encode the labels 0 through 19.)

169. Very much so, with millions and millions of labelings! Here's one of the 32 solutions for which $l(0000) = 0$, $l(0001) = 1$, $l(0010) = 2$, $l(0100) = 4$, $l(1000) = 8$, and $l(1111) = 15$, all found in 200 Gμ. By arranging the vertices of this interesting digraph as a Karnaugh map (see exercise 7.2.1.1–17), we can exhibit it as a "magical 4×4 torus."



172. (a) It suffices to consider tuples with $x_1 = 0$. Then $\mathcal{D}_2$ has two classes $\{00, 02\}$, $\{01\}^*$; and $\mathcal{D}_3$ has six: $\{000, 032\}$, $\{001, 011, 021, 031\}$, $\{002, 010, 022, 030\}^*$; $\{003, 033\}^*$; $\{012, 020\}$, $\{013, 023\}^*$. (Those marked with $*$ define a self-conjugate graceful digraph; the others define a conjugate pair. For example, $\{000, 032\}$ gives $K_1 \longrightarrow \overline{K_3}$, $\overline{K_3} \longrightarrow K_1$.)

(b) Use arithmetic mod q. If $a \perp q$ and $aa' = 1$, define $ax = y_1 \ldots y_m$ and $-ax^T = z_1 \ldots z_m$, where $y_l = a(x_{a'l} - x_{a'})$ and $z_l = 1 - l - y_l$. Reject x if $x > ax$ or $x > -ax^T$ lexicographically, for some $a \perp q$. The accepted tuples are inequivalent.

(c) The answer is $\sum_{a=1}^{m}[a \perp q] \sum_{b=0}^{m}\bigl(f(a,b,q)+g(a,b,q)\bigr)/(2q\varphi(q))$ by "Burnside's Lemma," where $f(a,b,q)$ and $g(a,b,q)$ are respectively the number of x with $D(x) = aD(x) + b$ and $D(x)^T = aD(x) + b$. Let $t(\alpha,\beta,q) = \gcd(\alpha,q)[\gcd(\alpha,q)\backslash\beta]$; this is the number of $x \in [0\,.\,.\,q)$ such that $\alpha x \equiv \beta$ (modulo q), when $\alpha, \beta \in [0\,.\,.\,q)$.

Let $f(l,a,b,q) = \bigl(a^s l < l?\ 1\colon t(a^s - 1, -b(a^{s-1} + \cdots + a + 1), q)\bigr)$, where $s > 0$ is minimum with $a^s l \le l$. (All arithmetic is mod q.) Then $f(a,b,q) = \prod_{l=1}^{q-1} f(l,a,b,q)$.

Let $g(l,a,b,q) = \bigl((-a)^s l < l?\ 1\colon t(a^s - 1, -b(a^{s-1} + \cdots + a + 1) - l(s \bmod 2), q)\bigr)$, where $s > 0$ is minimum with $(-a)^s l \le l$. Then $g(a,b,q) = \prod_{l=1}^{q-1} g(l,a,b,q)$.

(For example, it's 12502550 when $m = 9$; see OEIS A341884. The totient function $\varphi(n)$ is asymptotically not much less than n. In fact, $\liminf_{n\to\infty}(\ln\ln n)\varphi(n)/n = e^{-\gamma}$; see Hardy and Wright, *An Introduction to the Theory of Numbers*, Theorem 328.)

175. (a) Since $l_{2k+1} - l_{2k} = 2k + 1$ and $l_{2k+1} - l_{2k+2} = 2k + 2$, these labels are actually graceful for the nonoriented path P_n. Modulo $q = n$, the edge labels 2, 4, $\ldots$, $n - 2$ become arc labels $n - 2$, $n - 4$, $\ldots$, 2.

(b) Use the labels $l_{2k} = k$, $l_{2k+1} = r - 1 - k$ in the first half. Then define $l_{2r-1-k} = l_k + r + 1$. (This elegant construction is due to D. F. Hsu [*Lecture Notes in Math.* **824** (1980), 134–140], whose paper with G. S. Bloom [*Congressus Numerantium* **35** (1982), 91–103] introduced the notion of graceful digraphs and proved Theorem H.)

176. Let $l'(v) = (n+1)l(v)$ for $v \in D$, $l'(w_k) = k$ for the other vertices $\{w_1, \ldots, w_n\}$. [*SIAM Journal on Algebraic and Discrete Methods* **6** (1985), 519–536.]

177. $D = P_3^{\to}$; $l(v_0) = 2n + 1$, $l(v_1) = 0$, $l(v_2) = n + 1$, and $l(w_k) = k$ for $1 \le k \le n$.

178. Yes, because $K_{m,n}$ is α-graceful with labels $\{0, 1, \ldots, m - 1\}$ in one part.

180. (Answer left to the reader: Enjoy! Consider also the analogs of exercises 116–120, as well as the behavior of *random* graceful digraph labelings as $m \to \infty$. Many results have been reported by F. Stappers at `archive.org/details/graceful_digraphs_6`.)

digits, extended hexadecimal
Karnaugh map
magical 4×4 torus
self-conjugate
Burnside's Lemma
congruence enumeration
OEIS
totient function
γ
Hardy
Wright
Hsu
Bloom
Stappers

182. If D were graceful, its arc labels would sum to $1+\cdots+m=q(q-1)/2$. That sum is also congruent (modulo q) to $\sum_v(d^+(v)-d^-(v))l(v)$, which is even.

183. For each k with $1\le k\le m$, we can reverse the orientations on the arcs labeled k and $m+1-k$. [See the paper cited in answer 176, which introduced digracefulness.]

185. (a) A, E, C, F, B, D, G, H, I, J, K, L. (Note that A is the transitive tournament $K_5^{\to}$.)

(b) C, G, H, I, J, K are not graceful; the other six are uniquely graceful. (The lexicographically smallest `LO[1]LO[2]...LO[10]` tables for A, B, D, L are respectively 0040210442, 0010770742, 0010210742, 0017214742; E = B^T; F = D^T. Each labeling can be obtained from any of the others by reversing pairs as in exercise 183.)

(c) The four unlabeled tournaments for $n=4$ are A$'$, B$'$, C$'$, D$'$, obtained by removing the bottom vertices of A, B, C, D. The self-converse D$'$ is ungraceful; the others are uniquely graceful, with `LO` tables 002102 and 001042 for A$'$ and B$'$; C'^T = B. When $n=3$, A$''$ is uniquely graceful but B$''$ is the ungraceful $C_3^{\to}$.

(d) Let $v=q=\binom{n}{2}+1$. Given a graceful tournament on vertices $\{1,\ldots,n\}$, with labels $a_j=l(j)$, suppose arc $j\longrightarrow k$ is labeled l and arc $k'\longrightarrow j'$ is labeled $q-l$. Then $a_k\ominus a_j$ and $a_{k'}\ominus a_{j'}$ are two differences equal to l, so we have a cyclic $(v,n,2)$-difference set. (We'll have $j=k'$ and $k=j'$ when $l=q/2$, but never $j=j'$ and $k=k'$.) Conversely, by assigning labels from such a difference set, we get a graceful tournament if we define either ($j\longrightarrow k$ and $k'\longrightarrow j'$) or ($k\longrightarrow j$ and $j'\longrightarrow k'$) whenever $k\ominus j=k'\ominus j'$. [This connection was apparently first noted by Kumudakshi in her Ph.D. thesis (Mangalore: National Institute of Technology Karnataka, July 2016), Proposition 2.2.5.]

(e) These residues form a cycle (1 7 12 10 33 9 26 34 16) that defines a symmetrical graceful tournament, in which $u\longrightarrow v$ whenever v is one of the next four elements after u. (But the transitive tournament $K_9^{\to}$ is *not* graceful.) [In place of 7, R. D. Carmichael mentioned the equally good multiplier 16, on pages 437–438 of his *Introduction to the Theory of Groups of Finite Order* (1937); he probably learned about this remarkable difference set from someone else, so its origin is obscure. A computer search by L. J. Dickey has shown that no other cyclic difference sets with $\lambda=2$ exist for $n\le 5000$; see D. R. Hughes, *Lecture Notes in Mathematics* **686** (1976), 55–58.]

187. Say G is weakly digraceful with *tolerance* t if it can be gracefully oriented using just $m+t$ arcs. Calculations by Filip Stappers show that, for all 1044 graphs with up to 7 vertices, exactly (1013, 26, 4, 1) require tolerance t = (0, 1, 2, 3). (Only $3K_2$ needs tolerance 3; only $2K_2$, $L_{3,4}$, K_6, and K_7 need tolerance 2. For K_7 we can use the vertex labels $\{0,1,2,4,7,15,19\}$, mod 24, with all arcs $u\longrightarrow v$ going from $\min(u,v)$ to $\max(u,v)$ *except* that $2\longrightarrow 1$, $4\longrightarrow 2$, $7\longrightarrow 4$, $19\longrightarrow 15$, $15\longrightarrow 0$; the "tolerant" arcs $15\longrightarrow 7$ and $15\longrightarrow 1$ also pair up with their reversals $7\longrightarrow 15$ and $1\longrightarrow 15$.)

It seems likely that all connected graphs are weakly digraceful with *bounded* tolerance, because each modulus $q=m+t+1$ gives a "fresh start" for achieving gracefulness.

190. The arc labels between k and $k+1$ are $\pm(2k+1)$ (modulo q), where $q=2n+1$, except for two values of k. The exceptional values are $k=\lfloor(n-1)/2\rfloor$, when the labels are ±1, and $k=n$, when they are $\pm(n-(-1)^n)$. Altogether, they are therefore $\Delta(n)=\{\pm1,\pm3,\ldots,\pm(2n-1)\}$, because the "missing" case $\pm(2k+1)$ for $k=\lfloor(n-1)/2\rfloor$ turns out to be $\pm(n-(-1)^n)$. Finally, $\Delta(n)$ is the same as $\{\pm1,\pm2,\ldots,\pm n\}$ (modulo q). [*Discrete Mathematics* **261** (2003), 116.]

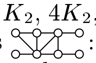
191. Regard T as rooted at v, with subtrees T_1, ..., T_d where $|T_1|\ge\cdots\ge|T_d|$, and number the vertices v_0, v_1, ..., v_m in preorder. Let $l(v_0)=0$; and for $k=1$, ..., $\lceil m/3\rceil$ let $l(v_k)$ be the least positive integer such that $l(v_k)\ne l(v_j)$ and $|l(v_k)-l(\mathrm{parent}(v_k))|\ne$

historical notes
transitive tournament
Kumudakshi
Carmichael
Dickey
Hughes
tolerance
Stappers
lollipop $L_{m,n}$
preorder

$|l(v_j) - l(\text{parent}(v_j))|$ for $1 \le j < k$. At most $3(k-1)$ values are excluded, hence $l(v_k) \le m$. Let $C = \{|l(v_k) - l(\text{parent}(v_k))| \mid 1 \le k \le \lceil m/3 \rceil\}$ be the "colors" used.

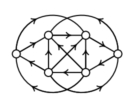
The remaining $m - \lceil m/3 \rceil \le 2m/3$ vertices are leaves adjacent to v, by hypothesis. So we can label them with the *negatives* of the unused colors, $-(\{1,\ldots,m\} \setminus C)$.

192. The labels $\{0,1,2,5,12,23,29\}$ give all differences $\{\pm1,\pm2,\ldots,\pm18\}$ (modulo 37), with ±1, ±10, ±11 occurring twice. For (i), let $0 \not- 1$, $2 \not- 29$, $12 \not- 23$; for (ii), let $1 \not- 2 \not- 12 \not- 1$. For (iii), work modulo 41 and let $0 \not- 36 \not- 18 \not- 31 \not- 0$, $1 \not- 2 \not- 22 \not- 28 \not- 1$. [Each of these labelings is essentially unique. The other graphs on 7 and 8 vertices that are uniquely rainbow graceful are $\overline{3K_1 \oplus 2K_2}$, $\overline{4K_1 \oplus K_{1,3}}$, $\overline{3K_1 \oplus K_{1,4}}$, $\overline{3K_1 \oplus K_2 \oplus C_3}$, $\overline{3K_1 \oplus K_2 \oplus P_3}$, K_8.]

193. P. Adams and J. Appleton (see S. I. El-Zanati and C. Vanden Eynden, *Mathematica Slovaca* **59** (2009), 1–18) found that $G^{\leftrightarrow}$ is graceful except in the following 18 cases: For $n = 6$ vertices, $\overline{4K_1 \oplus K_2}$, the complement of K_2. For $n = 7$, the complements of $K_{3,3}$, $K_{1,5}$, K_2, and K_1. For $n = 8$, the complements of $K_{4,4}$, $K_{3,4}$, $K_{2,6}$, $K_{1,6}$, $K_{1,5}$, $K_{2,2}$, $K_3 \oplus K_2$, $4K_2$, K_3, $3K_2$, $2K_2$, $K_{1,2}$, and K_2. (The "most rainbow graceful" 8-vertex graph is : There are 41,636 essentially different ways to label it!)

[It turns out that 43 copies of K_7 *can* be packed perfectly into K_{43}, but not cyclically. On the other hand, 29 copies of $\overline{4K_1 \oplus K_2}$ *cannot* be packed perfectly into K_{29}, cyclically or otherwise. It's the smallest example of an m-edge graph whose copies can't exactly cover K_{2m+1}. See S. Hartke, P. R. J. Östergård, D. Bryant, and S. I. El-Zanati, *Journal of Combinatorial Designs* **18** (2010), 94–104.]

194. No; $\overline{4K_1 \oplus K_2}$ is digraceful (answer 187), yet not rainbow graceful (answer 193). (It has 156 essentially distinct graceful orientations, 18 of which are self-converse. The most graceful of these, with 5 labelings, is shown.)



196. (a) There are $n^3 - 1$ nonzero triples, in equivalence classes of size $n-1$, hence $(n^3-1)/(n-1)$ classes. Each class has a unique element whose first nonzero component is 1; thus $a_1 = 1$ in n^2 classes, $(a_1,a_2) = (0,1)$ in n, and $(a_1,a_2,a_3) = (0,0,1)$ in 1.

(b) $a_1 + 2a_3 \equiv 0$ (modulo 3) $\iff a_1 \equiv a_3$. So the answer is $\{(0,1,0), (1,0,1), (1,1,1), (1,2,1)\}$. (In general $a_1b_1 + a_2b_2 + a_3b_3 = 0$ has $n^2 - 1$ nonzero solutions (a_1,a_2,a_3) in F, belonging to $(n^2-1)/(n-1)$ classes, when $[b_1,b_2,b_3]$ is nonzero.)

(c) The nonzero vectors $[b_1,b_2,b_3]$, $[b_1',b_2',b_3']$ are linearly independent when one isn't a multiple of the other. In that case the homogeneous equations $a_1b_1 + a_2b_2 + a_3b_3 = a_1b_1' + a_2b_2' + a_3b_3' = 0$ have $n-1$ nonzero solutions (a_1,a_2,a_3), all equivalent.

[See 7–(57) for the case $n = 2$; see also exercise 7–19.]

197. (a) It's an immediate consequence of the definitions; there are $m = \binom{n+1}{2}$ edges.

(b) If $\pi^3 = c_1\pi^2 + c_2\pi + c_3$, then $\pi^{3p} = c_1\pi^{2p} + c_2\pi^p + c_3$. Hence the other roots are π^p and π^{p^2}. [And $c_1 = \pi + \pi^p + \pi^{p^2}$, $-c_2 = \pi^{1+p} + \pi^{1+p^2} + \pi^{p+p^2}$, $c_3 = \pi^{1+p+p^2}$.]

(c) Since $\pi^{k+1} = c_1\pi^k + c_2\pi^{k-1} + c_3\pi^{k-2}$, $a_1' = a_2 + c_1a_1$, $a_2' = a_3 + c_2a_1$, $a_3' = c_3a_1$.

(d) $b_1' = b_2$, $b_2' = b_3$, $b_3' = (b_1 - c_1b_2 - c_2b_3)/c_3$.

(e) Eschewing parentheses and commas, they are 001, 010, 100, 403, 132, 223, 031, 310, 304, 244, 241, 211, 411, 212, 421, 312, 324, 444, 042, 420, 302, 224, 041, 410, 202, 321, 414, 242, 221, 011, 110, 003. Since $\pi^{31} = 3$, we have $\pi^{31+k} = 3\pi^k$.

(f) Let $v = p^2 + p + 1$. Then $\{1, \pi^v, \pi^{2v}, \ldots, \pi^{(p-2)v}\} = \{1, 2, \ldots, p-1\}$, so the triples for $\{1, \pi, \ldots, \pi^{v-1}\}$ are all the points. The given labels L are the points of the line $[1,0,0]$. Hence the points of the line $[1,0,0]\alpha^k$ are $(L+k) \bmod v$, and we have a cyclic $(v, p+1, 1)$-difference set. (For example, $L = \{0,1,6,18,22,29\}$ when $p = 5$.)

(g) Let F be the field of p^{3e} elements, specified by a primitive polynomial modulo p, and let π be a root of f in F. Then the subfield F_0 of p^e elements is $\{0, 1, \pi^v, \ldots,$

unique
Adams
Appleton
El-Zanati
Vanden Eynden
complement
exact cover
Hartke
Östergård
Bryant
El-Zanati
self-converse
linearly independent
homogeneous equations
primitive polynomial

$\pi^{(p^3-2)v}\}$, where $v = p^{2e} + p^e + 1$. The polynomial $f_0(x) = (x-\pi)(x-\pi^{p^e})(x-\pi^{p^{2e}}) = x^3 - c_1x^2 - c_2x - c_3$ is primitive for F and has coefficients in F_0. Proceed as before.

When $n = 8$ we can use $f(x) = x^9 - x^5 - 1$. Then $\omega = \pi^v = \pi^{73} = \pi^8 + \pi^7 + \pi^4 + \pi + 1$ is a primitive root for F_0, and we have $f_0(x) = x^3 - (\omega^2+1)x^2 - x - \omega$. Using octal notation with $\mathit{0} = 0$, $\mathit{1} = 1$, $\mathit{2} = \omega$, …, $\mathit{7} = \omega^2 + \omega + 1$, the points 1, π, π^2, …, π^{72} are *001*, *010*, *100*, *512*, *777*, *603*, *451*, *655*, *131*, *602*, *441*, *755*, *423*, *175*, *242*, *304*, *276*, *044*, …, *151*, and they yield the graceful rainbow labels $\{0, 1, 17, 39, 41, 44, 48, 54, 62\}$ for K_9.

[*Transactions of the Amer. Math. Soc.* **43** (1938), 377–385. T. P. Kirkman had discovered cyclic difference sets "by accident" for the projective planes of orders 2, 3, 4, 5, and 8, in *Trans. Hist. Soc. Lancashire and Cheshire* **9** (1857), 127–142. A famous conjecture that K_{n+1} is rainbow graceful if *and only if* n is a prime power has been verified for all $n \le 2000000$; see D. M. Gordon, *Electronic J. Combin.* **1** (1994), #R6, 1–7.]

199. (a) It suffices to consider tuples with $x_1 = 0$. Then $\mathcal{R}_2$ has two classes {00, 01, 03, 04}, {02}, and $\mathcal{R}_3$ has eleven: {000, 011, 015, 050, 054, 065}, {001, 002, 024, 041, 063, 064}, {003, 026, 031, 034, 046, 062}, {004, 061}, {005, 013, 021, 044, 052, 060}, {006, 014, 030, 035, 051, 066}, {010, 055}, {012, 020, 022, 043, 045, 053}, {016, 025, 032, 033, 040, 056}, {023, 042}, {036}. For example, the first and fourth classes give $K_{1,3}$.

(b) Use arithmetic mod q. Reject x if $x > ax$ lexicographically for some $a \perp q$, where $ax = y_1 \ldots y_m$ is defined by first setting $z_{al} \leftarrow ax_l$ if $al \le m$, otherwise $z_{q-al} \leftarrow a(x_l + l)$; then $y_l = z_l - z_1$. The accepted tuples are inequivalent.

(c) It's $\sum_{a=1}^{2m}[a \perp q]\sum_{b=0}^{2m} f(a,b,q)/(q\varphi(q))$, where $f(a,b,q) = \prod_{l=1}^{m} f(l,a,b,q)$ and $f(l,a,b,q) = \bigl(a^s l = l?\ t(a^s-1, -b(a^{s-1}+\cdots+a+1), q)\colon q - a^s l = l?\ t(a^s - 1, l - b(a^{s-1}+\cdots+a+1), q)\colon 1\bigr)$, where $s > 0$ is minimum with $a^s l \le l$ or $q - a^s l \le l$. (Compare with answer 172(c). We get 943532049 when $m = 9$; see OEIS A342357.)

200. This conjecture was introduced by S. I. El-Zanati, C. Vanden Eynden, and N. Punnim, *Australasian J. Combinatorics* **24** (2001), 209–219. In fact, they conjectured that *every bipartite graph G* with no isolated vertices has an "ordered graceful rainbow labeling," in which the smaller endpoint of every edge belongs to one part and the larger endpoint belongs to the other. (One such labeling for C_6 is (041327).)

203. True and true.

204. The unique answer is `chord` — `chore` — `chose` — `chase` — `chasm` — `charm` — `chard` — `chord`. (But one might argue that an induced cycle is always "chordless.")

205. Yes. One must check that $d($`cords`$,$`costs`$) = 3$ and $d($`colts`$,$`carts`$) = 3$ in `WORDS`(5757): The first is true because `corts` and `cosds` are nonwords, according to the Stanford GraphBase; the second is true because `corts` and `calts` are nonwords.

207. (a) $\binom{n_1}{2} + \binom{n_2}{2} + \cdots + \binom{n_r}{2}$.

(b) $n_1!\,n_2!\ldots n_r!\,t_2!\,t_3!\,t_4!\ldots$, when t_q of the n_k are equal to q.

(c) 4. (This question is too easy. Hamming distance is defined in exercise 7–23.)

(d) Suppose $x_1 \ldots x_r$ — $y_1 \ldots y_r$ because $x_j \ne y_j$, and $x_1 \ldots x_r$ — $z_1 \ldots z_r$ because $x_k \ne z_k$. Then $y_1 \ldots y_r$ — $z_1 \ldots z_r$ if and only if $j = k$.

(e) $K_{2,1,1}$. It contains two triangles that share an edge; hence the images of both triangles vary in only one constituent, by (d). But then all vertices are adjacent.

(f) Suppose we change coordinates k_0, k_1, …, k_4 as we go around the cycle. Then $k_0 \ne k_1 \ne \cdots \ne k_4 \ne k_0$, by (d). And each k_i must equal some k_j for $j \ne i$.

208. Every induced C_7 of a Hamming graph is equivalent to 000 — 100 — 110 — 111 — 121 — 021 — 001 — 000. So we can start by dividing `WORDS`(5757) into $\binom{5}{2} =$

octal notation
Kirkman
Gordon
Burnside's lemma
OEIS
El-Zanati
Vanden Eynden
Punnim
bipartite graph
ordered graceful rainbow labeling
chordless
joke
Stanford GraphBase
Hamming distance
$K_{2,1,1}$

unsupported
backtrack
strict embedding
tied-path graphs

10 families of subgraphs in which two of the coordinates are constant. (The largest such subgraphs are `*a*e*`, `*a**s`, `*o**s`, and `***es`, with sizes 305, 316, 329, and 371.)

To find all solutions within each subgraph, count the frequency of each letter in each coordinate position. Choose the coordinates (i, j, k) that will contain respectively $(3, 2, 2)$ letters in the solution, with $j < k$. A word is "unsupported" if any of its letters in positions (i, j, k) have frequencies less than $(2, 3, 3)$. There must also be at least one letter, in each of coordinates (i, j, k), whose frequency *exceeds* $(2, 3, 3)$. Discard unsupported words (and update the frequencies) until all words are supported and all frequencies are satisfactory. Then visit the solutions, of which there are 69457.

A solution is isometric if and only if three specific five-letter strings, found as in answer 205, are nonwords. Exactly 6879 solutions survive this test — including just one that belongs to WORDS(1000), namely `beams` — `seams` — `seems` — `seeds` — `sends` — `bends` — `beads` — `beams`. (Furthermore, exactly (2628, 2088) of the 5757 words participate in at least one (induced, isometric) cycle; (225, 298) in only one of them. The champion words are `pares`, in 2543 induced cycles; `later`, in 233 isometric cycles.)

209. (a) To satisfy (i), permute the elements with coordinate k. To satisfy (ii), permute the coordinates according to their first use.

(b) Straightforward backtrack suffices, branching on the possible $f(v_i)$ adjacent to $f(v_{i'})$. Also ensure that, for all $0 \le j < i$ and $j \ne i'$, the Hamming distance d_H satisfies $[v_j \mathrel{-\!\!\!/\!\!\!-} v_i] < d_H\bigl(f(v_i), f(v_j)\bigr) \le d(v_i, v_j)$.

210. (a) Yes. Any strict embedding of G also strictly embeds all G's induced subgraphs.

(b) If not connected, one of its components is nonembeddable (and induced).

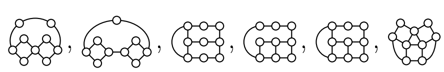
(c) True. Suppose $G \setminus v$ is disconnected, with induced components G' and G'', where G'' isn't embeddable. Then $G \setminus v'$ is connected for some v' in G'; it contains G''.

211. Let $(\mathcal{C}_n,\ \mathcal{H}_n,\ \mathcal{M}_n)$ be the n-vertex graphs that are respectively (connected, connected and Hamming embeddable, MNH). Clearly $\mathcal{H}_3 = \mathcal{C}_3$. Given lists of $\mathcal{C}_n$ for $4 \le n \le 9$, exercise 210 tells us that we can compute $\mathcal{H}_n$ and $\mathcal{M}_n$ as follows: Start with $\mathcal{H}_n$ and $\mathcal{M}_n$ empty. For each $G \in \mathcal{C}_n$, test if all n of its subgraphs $G \setminus v$ are either disconnected or in $\mathcal{H}_{n-1}$. If not, do nothing. Otherwise use exercise 209(b) to test if G has a Hamming embedding. If so, put G into $\mathcal{H}_n$; otherwise put G into $\mathcal{M}_n$.

The resulting sizes $(|\mathcal{H}_4|/|\mathcal{C}_4|, \ldots, |\mathcal{H}_9|/|\mathcal{C}_9|)$ turn out to be (5/6, 11/21, 36/112, 117/853, 469/11117, 2023/261080); and $(|\mathcal{M}_4|, \ldots, |\mathcal{M}_9|) = (1, 2, 0, 1, 1, 6)$.

The MNH graphs for $n \le 8$ all turn out to be "tied-path graphs," namely the graphs $P(n_1, \ldots, n_k)$ with $2+n_1+\cdots+n_k$ vertices and $k+n_1+\cdots+n_k$ edges that are obtained by tying together the endpoints of paths $P_{n_1+2}, \ldots, P_{n_k+2}$: $\mathcal{M}_4 = \{P(0,1,1)\}$; $\mathcal{M}_5 = \{P(1,2),\ P(1,1,1)\}$; $\mathcal{M}_6 = \emptyset$; $\mathcal{M}_7 = \{P(1,1,3)\}$; $\mathcal{M}_8 = \{P(2,2,2)\}$.

If we knew only these results, we'd be tempted to conjecture falsely that $P(1,1,5)$ and $P(3,3,3)$ are MNH. But all such hopes are shattered by

$\mathcal{M}_9 = \Bigl\{$  $\Bigr\};$

we might still conjecture tentatively, however, that all MNH graphs are planar.

212. In a normalized embedding, say that i is "k's pioneer for c" if $i = \min\{j \mid x_{jk} = c\}$. Then 0 is every coordinate's pioneer for 0. But a positive i cannot be a *double* pioneer; v_i can't be breaking records in two different coordinates, because it differs from its parent in only one place. Let $p(k, c)$ be k's pioneer for c, if it exists.

We shall prove, by induction on $i > 0$, that at most one normalized label $l(v_i)$ is isometrically consistent with $l(v_0), \ldots, l(v_{i-1})$. Suppose we could legitimately set

$\nu(x)$
sideways addition
nearest common ancestor
median labels

either $x_{ik} = a$ or $x_{ik} = b$, where $a < b$, and let $j = p(k, a)$. Then $j < i$, and $d(v_j, v_i)$ takes on two different values when we set $x_{ik} = a$ and $x_{ik} = b$. Contradiction.

Now suppose moves are legitimate in two different coordinates, $k < l$, so that if $(x_{i'k}, x_{i'l}) = (a, b)$ we could set (x_{ik}, x_{il}) to either (a', b) or (a, b'). If $a' > a$, let $j = p(k, a)$ and $t = x_{jl}$. Then $d(v_i, v_j) = \Delta + [a \neq a'] + [t \neq b] = \Delta + [a \neq a] + [t \neq b']$ for some Δ; consequently $1 + [t \neq b] = [t \neq b']$, and we must have $t = b$. Let $h = p(l, b)$ and $t' = x_{ih}$. Then $d(v_i, v_h) = \Delta' + [t' \neq a'] + [b \neq b] = \Delta' + [t' \neq a] + [b \neq b']$; consequently $[t' \neq a'] = [t' \neq a] + 1$ and $t' = a$. Hence $h = p(k, a) = j$, and j is a double pioneer! So $a = b = 0$. Finally let $g = p(k, 1)$. Then $x_{gl} = 0$; and $d(v_i, v_g) = \Delta'' + [1 \neq a'] + [0 \neq 0] = \Delta'' + [1 \neq 0] + [0 \neq b']$, a contradiction. A similar contradiction arises when $a' < a$.

So the desired algorithm is simplicity itself: To find $l(v_i)$, there are fewer than $2i$ candidates; for $0 \le j < i$ we need $O(i)$ operations to test that $d(l(v_i), l(v_j))$ is correct. If a candidate succeeds, we know $l(v_i)$, and no other candidates need be examined. If no candidate succeeds, there's no isometric embedding. Total time is $O(n^4)$, usually less.

[See *Discrete Applied Mathematics* **7** (1984), 221–225, also for exercise 213.]

213. (a) There are three kinds of vertices: corner (C, with degree 2); interior (I, with degree 4); other (O, with degree 3). There are four types of edges, which we may call CO, II, IO, OO. The relations OO $\bowtie$ OO, OO $\bowtie$ II, II $\bowtie$ II always hold. Each CO or IO is related to itself and to three others "parallel" to it.

(b) True. For example, $(0 \mathbin{—} 1) \bowtie (1 \mathbin{—} 2) \bowtie (2 \mathbin{—} 3) \not\bowtie (0 \mathbin{—} 1)$.

(c) Clearly $\bowtie$ is reflexive and symmetric. If $(u \mathbin{—} v) \bowtie (u' \mathbin{—} v') \bowtie (u'' \mathbin{—} v'')$ in *any* isometric Hamming embedding, and if $u_k \neq v_k$, $u'_{k'} \neq v'_{k'}$, $u''_{k''} \neq v''_{k''}$, where u_k denotes the kth coordinate of $l(u)$, then $k = k' = k''$. And if the embedding is ternary, we must also have $\{u, v\} \cup \{u'', v''\} \neq \emptyset$, hence $(u \mathbin{—} v) \bowtie (u'' \mathbin{—} v'')$.

(d) Let there be r equivalence classes, and let $u^{(k)} \mathbin{—} v^{(k)}$ represent class k. Assign label $l(w) = w_1 \ldots w_r$ to vertex w, where $w_k = \bigl(d(w, u^{(k)}) - d(w, v^{(k)})\bigr) \bmod 3$.

214. The graph with labels $\{00, 10, 20, 11, 21, 31\}$ answers (i); for (ii), add a seventh vertex labeled 30. Example (ii) shows that induced "minimal nonisometrically embeddable" subgraphs should *not* be used to prune the search for embeddable ones. But we still can exclude graphs with an induced MNH. Totals for $1 \le n \le 9$ are (1/1, 1/1, 2/2, 4/5, 9/11, 28/35, 86/111, 318/427, 1265/1742), where the denominators show *every* isometric embedding and the numerators show only the ternary ones.

216. (a) $\nu\bigl((b \oplus b')\ \&\ {\sim}(a \mid a')\bigr)$, the number of non-$*$ bits that differ.

(b) There are essentially only two other possibilities:

$$l(0) = 0000,\ l(1) = 1000,\ l(2) = 110{*},\ l(3) = {*}{*}11,\ l(4) = 0001;$$
$$l(0) = 000{*},\ l(1) = 100{*},\ l(2) = 1{*}10,\ l(3) = {*}111,\ l(4) = 010{*}.$$

(c) Let 5 be the top vertex, and let 6 and 7 be the two vertices inside the induced five-cycle. Use the labels $l(5) = 1{*}01$, $l(6) = 01{*}0$, $l(7) = {*}010$.

(d) If $v \neq r$, let $v' = \text{parent}(v)$. We want to prove that $\nu\bigl(l(u) \oplus l(v)\bigr) = d(u, v)$ for all u and v. If w is their nearest common ancestor, coordinates $(u, u', \ldots, u^{(d(u,w)-1)}, v^{(d(w,v)-1)}, \ldots, v', v)$ of $l(u)$ and $l(v)$ are respectively $(1, 1, \ldots, 1, 0, \ldots, 0, 0)$ and $(0, 0, \ldots, 0, 1, \ldots, 1, 1)$; other coordinates match. So there are $d(u, v)$ mismatches. (This construction is a special case of median labels; see 7.1.1–(63).)

(e) For example, suppose $d(u, w) = 4$ and $d(w, v) = 2$. Coordinates (u, u', u'', u''') are 1 in $l(u)$, non-1 in $l(v)$; coordinates (v', v) are non-1 in $l(u)$, 1 in $l(v)$; other coordinates are either both 1 or both non-1, so they contribute nothing to the "distance."

Notice that coordinates (v', v) contribute $\frac{1}{2}(1+d(u,v)-d(u,v'))+\frac{1}{2}(1+d(u,v')-d(u,w)) = \frac{1}{2}(d(w,v)+d(u,v)-d(u,w))$. Similarly, coordinates (u, u', u'', u''') contribute $\frac{1}{2}(d(u,w)+d(u,v)-d(w,v))$. So the total "distance" is indeed $d(u,v)$.

For the Petersen graph, with vertices ij for $0 \le i < j < 5$ and root 01, we have

$$\begin{array}{r@{\;}ccccccccc}
 & 23 & 04 & 14 & 24 & 03 & 13 & 34 & 02 & 12\\
l(01) = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
l(23) = & 1 & 0 & 0 & 0 & ? & ? & 0 & ? & ?\\
l(04) = & 1 & 1 & 0 & ? & ? & * & ? & ? & *\\
l(14) = & 1 & 0 & 1 & ? & * & ? & ? & * & ?\\
l(24) = & 0 & ? & ? & 1 & 0 & 0 & 0 & ? & ?\\
l(03) = & ? & ? & * & 1 & 1 & 0 & ? & ? & *\\
l(13) = & ? & * & ? & 1 & 0 & 1 & ? & * & ?\\
l(34) = & 0 & ? & ? & 0 & ? & ? & 1 & 0 & 0\\
l(02) = & ? & ? & * & ? & ? & * & 1 & 1 & 0\\
l(12) = & ? & * & ? & ? & * & ? & 1 & 0 & 1
\end{array}
\quad\Longrightarrow\quad
\begin{array}{r@{\;}ccccccccc}
 & 23 & 04 & 14 & 24 & 03 & 13 & 34 & 02 & 12\\
l(01) = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
l(23) = & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
l(04) = & 1 & 1 & 0 & 0 & * & * & 0 & * & *\\
l(14) = & 1 & 0 & 1 & 0 & * & * & 0 & * & *\\
l(24) = & 0 & * & * & 1 & 0 & 0 & 0 & 0 & 0\\
l(03) = & * & 0 & * & 1 & 1 & 0 & 0 & * & *\\
l(13) = & * & * & 0 & 1 & 0 & 1 & 0 & * & *\\
l(34) = & 0 & * & * & 0 & * & * & 1 & 0 & 0\\
l(02) = & * & 0 & * & * & 0 & * & 1 & 1 & 0\\
l(12) = & * & * & 0 & * & * & 0 & 1 & 0 & 1
\end{array}\;.$$

(f) Change '?' to ('$*$', '0') in u_v when $[u<v]+f(u,v)$ is (even, odd), where '$<$' is preorder and $f(u,v) = d(u,v)+d(u,r)+d(v,r)$. *Proof:* Let $p = d(u,w)$ and $q = d(w,v)$, and assume that $u < v$. Then the ancestors of u satisfy $u^{(k)} < v$ for $0 \le k < p$; similarly, $u < v^{(k)}$ for $0 \le k < q$. Define $x_k = f(u^{(k)}, v) \bmod 2$ for $0 \le k \le p$, and $x_{p+q-k} = f(u, v^{(k)}) \bmod 2$ for $0 \le k \le q$. Notice that $x_p = 0$, and $x_0 = x_{p+q}$. In $l(u)$ and $l(v)$ we have $u_{u^{(k)}} = 1$ and $v_{u^{(k)}} = {?}$ if and only if $x_k \ne x_{k+1}$, for $0 \le k < p$; similarly $u_{v^{(k)}} = {?}$ and $v_{v^{(k)}} = 1$ if and only if $x_{p+q-k} \ne x_{p+q-k-1}$, for $0 \le k < q$. So the number of ?s is the number of substrings '01' and '10' within x, say $2m$. If there are m' transitions '10' before the 0 at x_p, there are $m - m'$ transitions '01' after it.

Notes: If we shrink each subcube to a point, we get a "squashed cube." The subcube labels define an isometric embedding into a squashed cube — we can't get shorter paths by going outside the image and coming back again. (However, the computation of shortest distances between *unused* points of the squashed cube isn't easy.) The existence of a subcube representation with $n-1$ coordinates was conjectured by R. L. Graham and H. O. Pollak [*Bell System Tech. J.* **50** (1971), 2495–2519] and proved by P. M. Winkler [*Combinatorica* **3** (1983), 135–139].

squashed cube
isometric embedding
Graham
Pollak
Winkler
3-PARTITION
strongly NP-complete
Garey
Johnson

217. Because $G \sqsubseteq G'$ implies $G \subseteq G'$, (i), (iii), (v), and (vii) are obviously true. And (viii) clearly holds. But (ii), (iv), (vi) fail either when $G = G'$ or when $G' = G''$.

218. False. (Maybe $G_1 = G_2 = K_2$, $H_1 = C_4$, $H_2 = K_1$.)

219. True. Suppose f embeds G into H, $u \not\relbar\!\!\relbar v$, $f(u) \relbar\!\!\relbar f(v)$, and $u = u_0 \relbar\!\!\relbar u_1 \relbar\!\!\relbar \cdots \relbar\!\!\relbar u_k = v$. Then $k > 1$, and $f(u_0) \relbar\!\!\relbar f(u_1) \relbar\!\!\relbar \cdots \relbar\!\!\relbar f(u_k) \relbar\!\!\relbar f(u)$ is a cycle.

220. The vertex of degree m must map to r; its neighbors must map to $\{x_{10}, \ldots, x_{m0}\}$. So each path P_{a_i} must be mapped to a_i vertices of $\{x_{j1}, \ldots, x_{jn}\}$ for some j. Those with the same j form a submultiset of sum $\le n$. So we get a suitable partition.

Conversely, such a partition yields an embedding. (And if $a_1 + \cdots + a_t = mn$ and $t = 3m$, we've solved the 3-PARTITION problem, which is strongly NP-complete. See M. R. Garey and D. S. Johnson, *Computers and Intractability* (1979), §4.2.2.)

221. (a) $5n$ vertices and $6n + \binom{n}{2}$ edges.

(b) If $v \mapsto f(v)$ is a strict embedding from G to H, then $(v,k) \mapsto (f(v),k)$ is easily seen to be an embedding from $q(G)$ to $q(H)$, by considering the three kinds of edges.

Conversely, assume that $(v,k) \mapsto (f(v,k), g(v,k))$ is an embedding. For fixed v, let $w_k = f(v,k)$ and $r_k = g(v,k)$. If $w_k \ne w_{k+1}$ we must have $r_k = 0$ or $r_{k+1} = 0$. And if, say, r_0 is the only 0, we have $w_0 \ne w_1 = \cdots = w_4$ and $\{r_1, r_4\} = \{1, 3\}$, implying both $w_0 \relbar\!\!\relbar w_1$ and $w_0 \not\relbar\!\!\relbar w_1$. A similar contradiction arises if r_k is the only 0. So

Solnon
automorphisms
complete graphs
directed graphs
complete binary tree

$(r_0,\ldots,r_4)$ must be a cyclic permutation of $(0,1,0,3,4)$ or $(0,1,0,4,3)$; but none of those is compatible with (w_2,r_2) — (w_4,r_4). Hence $f(v,k)=f(v)$ is independent of k. Now the image of $q(G)$ has $5n$ vertices and $6n+\binom{n}{2}$ edges; it must be an isomorphic copy.

222. If G has n vertices V, let $s(G)$ have n^2 vertices (v,w), where $(v,v) — (v,w) — (w,w)$ for all $v\neq w$, and $(v,w) — (w,v)$ when $v — w$. Let $t(G)$ be $s(G)$ together with additional vertices $\{v,w\}$ whenever $v — w$; we have $(v,v) — \{v,w\} — (w,w)$ when $\{v,w\}$ exists. One can now prove that $G\subseteq H$ if and only if $s(G)\sqsubseteq t(H)$.

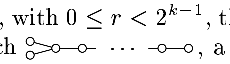
For example, if f is a strict embedding of $s(G)$, $f(v,v)$ must be a vertex of the form $(f(v),f(v))$, at least when $n>2$, because (v,v) has degree $2n-2$ in $s(G)$ and the other vertices of $t(H)$ have degree ≤ 3. Then $f(v,w)$ and $f(w,v)$ when $v — w$ in G must be $(f(v),f(w))$ and $(f(w),f(v))$, since those are the only adjacent vertices in $t(H)$ that are neighbors of both $(f(v),f(v))$ and $(f(w),f(w))$. But when $v\neq w$ and $v \not— w$, $\{f(v,w), f(w,v)\}$ can be any two of $(f(v),f(w))$, $(f(w),f(v))$, and possibly $\{f(v),f(w)\}$.

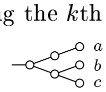
(Christine Solnon noticed that $s(G)$ has a huge number of automorphisms, because one can independently swap (v,w) with (w,v) when $v\neq w$. To avoid this problem she uses *directed* arcs $(v,v)\longrightarrow(v,w)\longrightarrow(w,w)$.)

224. (a) Suppose the given ISIP has edge labels L_j for $0\le j<J$. Define a labeled SIP on $\widehat{G}$ and $\widehat{H}$, the complete graphs on the vertices of G and H, giving their vertices the labels l_i and compatibilities they have in G and H. Also give their edges the existing labels L_j on existing edges, with the existing compatibilities; and let $L_j(u,v)=\Lambda$ when $u\not— v$, where Λ is always compatible with Λ. Finally — and this is the key point — introduce a new edge label L_J, where $L_J(v,w)=[v — w]$, compatible if and only if equal.

(b) Suppose the given SIP has labels l_i for $0\le i<I$ and L_j for $0\le j<J$. Introduce a new vertex label l_I, where $l_I(v)=[v — u]$ for $v\in G\setminus u$ and $l_I(\hat v)=[\hat v — \hat u]$ for $\hat v\in H\setminus\hat u$; these labels are compatible if and only if $l_I(v)\le l_I(\hat v)$. Also introduce new vertex labels l_{I+1+j} for $0\le j<J$, where $l_{I+1+j}(v)=L_j(u,v)$ if $u — v$, otherwise $l_{I+1+j}(v)=\Lambda$, using the compatibility relation of L_j and letting Λ be self-compatible.

(For directed graphs, however, we need more. Arc labels $L_j(v,w)$ are given when $v\longrightarrow w$. In part (a) let $L_J(v,w)=2[v\longrightarrow w]+[v\longleftarrow w]$. In part (b), let $l_I(v)=[v\longrightarrow u]$, $l_{I+1}(v)=[v\longleftarrow u]$, $l_{I+2+j}(v)=L_j(v,u)$ or Λ, $l_{I+2+J+j}(v)=L_j(u,v)$ or Λ.)

226. Given a 3SAT problem with m clauses, where every literal occurs exactly twice (exercise 7.2.2.2–208), construct G and H as follows: Start with the complete binary tree B_m with m leaves; if $m=2^k-r$, with $0\le r<2^{k-1}$, there are r leaves on level $k-1$ and $m-r$ leaves on level k. Attach , a path of length $10m$ together with a 'Y' at one end, to the root of B_m, and call the result B_m^+. Then G is obtained from B_m^+ by replacing each leaf $-\!\circ$ by a path $-\!\circ\!-\!\circ\!-\!\circ$. Similarly, H is obtained from B_m^+ by replacing the kth leaf $-\!\circ$ by the graph



where $\{a,b,c\}$ are the literals of clause k;

we also add nontree edges, two from each of a, b, c, to the vertices called respectively $\bar a$, $\bar b$, $\bar c$ in the other clauses. (These labels define the nontree edges, but don't appear in H.) Notice that G has $14m+1$ vertices, $14m$ edges; H has $17m+1$ vertices, $20m$ edges.

If the clauses are satisfiable, then $G\sqsubseteq H$, because we can match the "tip" of leaf k to a literal a, b, or c that satisfies clause k. Conversely, if $G\sqsubseteq H$, the 'Y' of G must correspond to the 'Y' of H, because the path of length $10m$ can't originate within B_m. Also the embedding of levels 0 through k must properly match up the r leaves on level $k-1$ and the $m-r$ leaves on level k. Thus the embedding will specify literals that satisfy each clause, never choosing both l and $\bar l$.

[This construction is based on an idea of C. Papadimitriou. On the other hand, E. Luks [*J. Computer and System Sciences* **25** (1982), 42–65] gave a polynomial-time algorithm to test *full* isomorphism between graphs of bounded degree. J. Matoušek and R. Thomas [*Discrete Math.* **108** (1992), 343–364] have shown how to solve $G \subseteq H$ and $G \sqsubseteq H$ in polynomial time if G has bounded degree and H has bounded treewidth.]

228. (a, b) Both equivalences are easily proved. Notice that all vertices of $\widehat{G}$ either have in-degree 0 (the original vertices of G) or in-degree 2 (the original edges of G); the embeddings must distinguish them too. (See T. Werth, M. Wörlein, A. Dreweke, I. Fischer, and M. Philippsen, in *Data Mining for Business Applications* (2009), 213.)

229. No. If $M = 2k+1$, use breadth-first searches to test if H contains a vertex u and two vertices $v \,—\, w$ at distance k from u. A similar method works when $M = 2k+2$.

231. Yes, by including additional items in the option for v and V, namely

$$\{e \cdot E \mid e = (u \not\!\!— v) \text{ and } E = (U \,—\, V) \text{ for some } u \text{ and } U\}.$$

232. For SIP, there's a secondary item $uv \cdot UV$ for every arc $u \longrightarrow v$ in the pattern and every nonarc $U \not\longrightarrow V$ in the target; this item is inserted into the option for '`u U`' and the option for '`v V`' (and no other options). For ISIP, those options also get a secondary item $uv \cdot UV$ for every nonarc $u \not\longrightarrow v$ in the pattern and every arc $U \longrightarrow V$ in the target.

233. (a) If there are W_{mn} strict embeddings from $\mathcal{G}_m$ to $\mathcal{G}_n$, then $\mathrm{E}(W_{mn}) = n^{\underline{m}}/2^{\binom{m}{2}}$, because each of the $n^{\underline{m}}$ embedding functions f succeeds with probability $1/2^{\binom{m}{2}}$. When $m = 2\lg n + 1 + \delta$ we have $\binom{m}{2} \geq \binom{2\lg n}{2} + 2(1+\delta)\lg n = (m+\delta)\lg n$. Hence, by the first moment principle (MPR–(21)), $\Pr(\mathcal{G}_m \sqsubseteq \mathcal{G}_n) = \Pr(W_{mn} > 0) \leq \mathrm{E}(W_{mn}) \leq n^{-\delta}$.

(b) Clearly $\mathrm{E}(W_{mn}) \geq n^m 2^{-\binom{m}{2}}(1 - m^2/n)$; and when $m = 2\lg n + 1 - \delta$, one can show that $\mathrm{E}(W_{mn}^2) \leq \bigl(n^m/2^{\binom{m}{2}}\bigr)^2\bigl(1 + O(n^{-2\delta/3})\bigr)$. Hence, by the second moment principle (MPR–(22)), $\Pr(\mathcal{G}_m \sqsubseteq \mathcal{G}_n) \geq 1 - O(n^{-2\delta/3})$. [To appear.]

235. In general, assume that G and H are connected graphs with $G \subseteq H$, and that H can be disconnected into components H_1 and H_2 by cutting k edges. Then there must be a way to cut k edges from G in such a way that each resulting component can be embedded in either H_1 or H_2. (But (52) remains connected when any two edges are cut.)

236. `BRAIN83`(600) suffices for this, with $0 \mapsto 53$, $0{+} \mapsto 56$, $1{-} \mapsto 15$, $1 \mapsto 36$, $1{+} \mapsto 38$, $2{-} \mapsto 79$, $2 \mapsto 76$, $2{+} \mapsto 55$, $3{-} \mapsto 35$, $3 \mapsto 39$, $3{+} \mapsto 14$, $0{-} \mapsto 77$.

237. Yes: $12 \cdot 3$ ways in `BRAIN83`(370), found in 3.7 Tμ (but none in `BRAIN83`(360)).

238. (J_5 is 3-regular.) Not into `BRAIN83`(600); but $20 \cdot 86$ ways into `BRAIN83`(700).

239. Require $f(0{+}) < f(v)$ for $v \in \{1{-}, 1{+}, 2{-}, 2{+}, 3{-}, 3{+}, 0{-}\}$. ("Pairwise ordering," exercise 7.2.2.1–20, makes options still longer but cuts the solving time to 1.5 Gμ.)

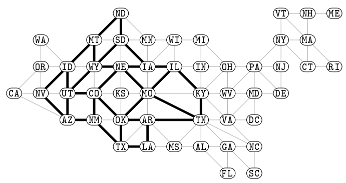
242. (a) The same result holds if 'deg' is replaced by 'd' in the definition of s, where d is *any* supplemental labeling function. Proof: Let v's neighbors in G be $v_1, \ldots, v_p$, where $d(v_1) \geq \cdots \geq d(v_p)$; similarly, let $f(v)$'s neighbors in H be $w_1, \ldots, w_q$, where $d(w_1) \geq \cdots \geq d(w_q)$ and $q \geq p$. Given $k \leq p$, there are indices $1 \leq i_1 < \cdots < i_k \leq q$, depending on k, such that $\{f(v_1), \ldots, f(v_k)\} = \{w_{i_1}, \ldots, w_{i_k}\}$. Let j be the index with $w_{i_k} = f(v_j)$; then $d(v_k) \leq d(v_j) \leq d(w_{i_k}) \leq d(w_k)$. [See S. Zampelli, Y. Deville, and C. Solnon, *Constraints* **15** (2010), 327–353.]

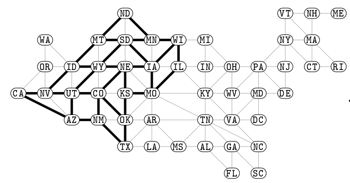
(b) (Solution by C. Solnon.) For $1 \leq k \leq p$, let v have a_k neighbors of degree k; also let w have b_k neighbors of degree k, or of degree $\geq k$ when $k = p$. Then check whether or not $b_p \ldots b_1$ majorizes $a_p \ldots a_1$, namely whether or not $b_p + \cdots + b_k \geq a_p + \cdots + a_k$ for $p \geq k \geq 1$. (Compare with Algorithm 5.2D and exercise 7.2.1.4–54.)

Papadimitriou
Luks
isomorphism between graphs
bounded degree
Matoušek
Thomas
treewidth
Werth
Wörlein
Dreweke
Fischer
Philippsen
breadth-first searches
first moment principle
second moment principle
cutting
Pairwise ordering
supplemental labeling function
Zampelli
Deville
Solnon
majorizes

[This partial ordering of multisets is a distributive lattice. When restricted to multisets of at most r positive integers, all $\leq s$, it's the lattice $L(r,s)$ of partitions into at most r parts $\leq s$, of which there are $[q^k]\binom{r+s}{r}_q$ partitions of k by 7.2.1.4–(51).]

243. `02` $\mapsto$ `MS` would force `01` $\mapsto$ `AL` and `03` $\mapsto$ `AL`. `02` $\mapsto$ `TX` would force `01` $\mapsto$ `NM` and `03` $\mapsto$ `NM`. Now `02` $\mapsto$ `LA` limits the domains of `01` and `03` to $\{$`MS`$,$ `TX`$\}$; and that forces both `00` $\mapsto$ `NM` and `04` $\mapsto$ `NM`. (`AL` has no neighbors in `h`, so we can't map `00` $\mapsto$ `AL`.)

244. 

and 

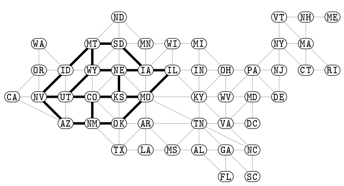
.

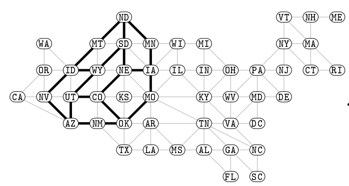
245. (a) One of $4\cdot 12$ embeddings for $P_2 \mathbin{\square} P_{12}$ is $\begin{pmatrix}\texttt{CA OR ID WY NE IA WI MI OH WV PA NJ}\\ \texttt{AZ NV UT CO KS MO IL IN KY VA MD DE}\end{pmatrix}$.

(b) And $\begin{pmatrix}\texttt{OR ID WY NE IA IL}\\ \texttt{NV UT CO KS MO KY}\\ \texttt{CA AZ NM OK AR TN}\end{pmatrix}$ is one of $4\cdot 9$ for $P_3 \mathbin{\square} P_6$.

246. $P_2 \mathbin{\square} P_2$ (in just $8\cdot 3$ ways, including $\{$`CO`$,$ `NE`$,$ `MO`$,$ `OK`$\}$); $P_3 \mathbin{\square} P_0$.

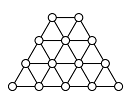
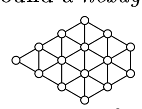
247. There are $10\cdot 7$ ways, including for instance



and 

.

(And there are $12\cdot 19$ ways to embed six pentagons that surround a *hexagon*.)

250. There are unique embeddings  $\sqsubseteq$ `USA` and  $\sqsubseteq$ `USA` of *simplex*$(4,4,4,3,0,0,0)$ and *simplex*$(5,5,3,3,0,0,0)$. (Put `NV` in the left corner.)

253. Let $M(v)$ be the mate of vertex v in the given matching, so that $M(x_i)=y_{j_i}$ and $M(y_{j_i})=x_i$. Also let $M(y_j)=\perp$ if $j\notin\{j_1,\ldots,j_m\}$. Suppose there's also *another* feasible matching, with mate function m, in which $m(x_i)=y_j$ (hence j isn't removable).

Let $u_0=x_i$, $v_0=y_j$, and $u_1=M(v_0)$. If $u_k\neq\perp$, let $v_k=m(u_k)$ and $u_{k+1}=M(v_k)$. If $u_k=u_0$, this sequence will be periodic, and $u_k \longrightarrow v_{k-1}\longrightarrow u_{k-1}\longrightarrow\cdots\longrightarrow v_0\longrightarrow u_0$ will be a path in T; hence x_i and y_j will be in the same strong component.

But if $u_k=\perp$, let $v_{-1}=M(u_0)$ and $u_{-1}=m(v_{-1})$. If $u_{-l}\neq\perp$, let $v_{-l-1}=M(u_{-l})$ and $u_{-l-1}=m(v_{-l-1})$. Eventually we'll have $u_{-l}=\perp$, and a path $u_k\longrightarrow v_{k-1}\longrightarrow u_{k-1}\longrightarrow\cdots\longrightarrow v_{-l}\longrightarrow u_{-l}$; so y_j and $\perp$ will be in the same strong component.

Conversely, if there's an oriented path $x_i\longrightarrow\cdots\longrightarrow y_j\longrightarrow x_i$ or $\perp\longrightarrow\cdots\longrightarrow y_j\longrightarrow x_i$ in T, we can convert the given matching to a feasible matching with x_i — y_j by reversing each edge of that path. Hence j isn't removable.

254. (a) [This is Philip Hall's theorem, *J. London Math. Soc.* **10** (1935), 26–30, where Hall sets are featured. When x_1 — y_{j_1}, $\ldots$, x_m — y_{j_m} is such a matching, the sequence $j_1\ldots j_m$ is called a "system of distinct representatives." Group theorists use the term "Hall set" for quite a different concept — also due to Philip Hall.] The condition is certainly necessary. If the algorithm fails, its final dag supplies an I with $|D(I)|<|I|$.

partial ordering of multisets
distributive lattice
lattice
$L(r,s)$
partitions
q-nomial coeffs
Philip Hall
historical notes
system of distinct representatives
distinct representatives
Hall set
pigeonhole principle

(b, c) If j is removable from x_i's domain, there's no matching in the subgraph with x_i and y_j deleted. So there's a subset $I \subseteq \{1,\ldots,m\} \setminus i$ with $|D'(I)| < |I|$, where D' is the subdomain in the subgraph. Thus $|D(I)| \le |I|$; by feasibility, $|D(I)| = |I|$.

(d) Because all values in $D(I)$ must be used as the images of I's variables.

(e) Let A, B, C be disjoint subsets of $\{1,\ldots,n\}$, with $a = |A|$, $b = |B|$, $c = |C|$, $a' = |D(A)|$, $b' = |D(B) \setminus D(A)|$, $c' = |D(C) \setminus D(A)|$, $a' + b' = a + b$, $a' + c' = a + c$. By feasibility we have $a' \ge a$ and $a' + b' + c' \ge |D(A \cup B \cup C)| \ge a + b + c$. Therefore $2a' + b' + c' \ge 2a + b + c = 2a' + b' + c'$, hence $a' = a$ and $|D(A \cup B \cup C)| = a + b + c$.

(f) This structure is a consequence of parts (b) and (d); I_1 through I_r are the minimal nonempty Hall sets. (Consequently the problem now has $r+1$ independent sets of variables $\{x_i \mid i \in I_j\}$, each of which has the all-different constraint only within its subdomain $D(I_j)$; moreover, perfect matchings are required, except between I_0 and $D(I_0)$.

(g) Each I_j is the set of x's belonging to some strong component, with $j = 0$ when that component also contains $\perp$. (Notice that I_0 might be $\emptyset$. There might be more than $r + 1$ strong components, but only because $\{y_j\}$ is a singleton strong component when $D(i) = \{j\}$ is a singleton domain.)

Historical notes: Chapter 7 of C. Berge's book *Graphs and Hypergraphs* (1973) surveys the theory of alternating paths, which allows us to understand the family of all maximum matchings. Minimal nonempty Hall sets correspond to connected bipartite graphs for which every edge is part of a perfect matching. Such graphs are called "elementary bipartite" by L. Lovász and M. D. Plummer [*Matching Theory* (1986), Chapter 4], who have traced the concept back to D. Kőnig [*Mathematikai és Természettudományi Értesítő* **33** (1915), 221–229]. One of many interesting properties of such graphs, noted in their exercise 4.1.5, can be paraphrased as follows: "Let F be a loopfree digraph on vertices $\{x_1,\ldots,x_n\}$, and let G be the bigraph on $\{x_1,\ldots,x_n\}$, $\{y_1,\ldots,y_n\}$ whose edges are $x_i \longrule y_i$ for $1 \le i \le n$ and $x_i \longrule y_j$ whenever $x_i \longrightarrow y_j$ in F. Then F is strongly connected if and only if G is elementary."

J.-C. Régin [*Proc. Nat. Conf. on Artificial Intelligence* **12** (1994), 362–367] developed the algorithm of exercise 253 after discovering that every removable element of an all-different constraint can be identified from a single computation of strong components. Subsequent refinements of his algorithm were surveyed carefully and investigated empirically by I. P. Gent, I. Miguel, and P. Nightingale [*Artificial Intelligence* **172** (2008), 1973–2000], who noted gains in efficiency after strong components I_j have been identified as in (f) and used for GAD filtering on the smaller domains $D(I_j)$.

255. Given a matching, let T simply be the digraph on vertices $\{y_1,\ldots,y_n\}$ with arcs $\{y_{j_i} \longrightarrow y_k \mid x_i \longrightarrow y_k \text{ and } k \ne j_i\}$. Then $k \ne j_i$ is removable from D_i if and only if y_k and y_{j_i} belong to different strong components. (We're essentially identifying x_i with y_{j_i}.)

256. (a) The domain D_a of `a` must be a target vertex with a predecessor of out-degree ≥ 4; so $D_\mathrm{a} = \{6, 8, 13, 14, 15, 16\}$. And D_d is the set of targets with a predecessor having out-degree ≥ 1, in-degree ≥ 1, and at least two neighbors, namely $\{3, 6, 8, 12, 13, 14, 15, 16\}$. But $D_\mathrm{b} = \{3\}$ is the only target with a predecessor of bi-degree ≥ 1, where "bi-degree" is the number of two-way ($\leftrightarrow$) edges; $D_\mathrm{c} = \{8\}$ is the only target with bi-degree ≥ 1 and out-degree ≥ 2. Similarly, $D_\mathrm{e} = D_\mathrm{a}$; $D_\mathrm{f} = \{0\}$; $D_\mathrm{g} = \{14\}$.

After the forced assignments `b` $\mapsto 3$, `c` $\mapsto 8$, `f` $\mapsto 0$, `g` $\mapsto 14$, the remaining domains reduce to $D_\mathrm{a} = D_\mathrm{e} = \{6, 13, 15, 16\}$; $D_\mathrm{d} = D_\mathrm{a} \cup \{12\}$. LAD filtering now tells us that `e` $\not\mapsto 16$, because D_d doesn't contain 16's successor (3). Similarly, `d` $\not\mapsto 13$; `d` $\not\mapsto 16$.

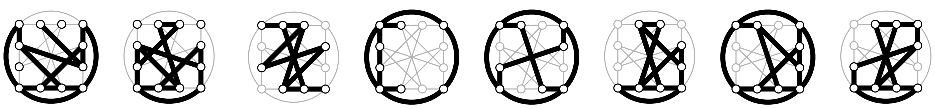
perfect matchings
Historical notes
Berge
perfect matching
elementary bipartite
Lovász
Plummer
Kőnig
bigraph
strongly connected
Régin
Gent
Miguel
Nightingale
bi-degree

line graph
automorphisms
domain

So we branch on e, and there are three cases: If $\mathtt{e} \mapsto 6$, then $\mathtt{d} \mapsto 15$ and we discover two solutions, $\mathtt{a} \mapsto 13$ or 16. If $\mathtt{e} \mapsto 13$, then $\mathtt{d} \mapsto 12$ and $\mathtt{a} \mapsto 6$, 15, or 16. If $\mathtt{e} \mapsto 15$, then $\mathtt{d} \mapsto 6$ and $\mathtt{a} \mapsto 13$ or 16.

(b) With strict embedding the initial domain D_{e} is reduced to $\{8, 13, 16\}$. Only two of the previous solutions survive: $(\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}) \mapsto (6$ or $15, 3, 8, 12, 13, 0, 14)$.

259. Yes; but there are three essentially different ways to delete two edges. If the edges are adjacent — at distance 1 in the line graph — there are $32 \cdot 4$ embeddings, such as (P1) below. If at distance 2, (P2) is one of $16 \cdot 7$ embeddings. At distance 3 there are none.



(P1) (P2) (i) (ii) (iv) (v) (vi) (vii)

260. Respectively $8 \cdot 1$, $32 \cdot 1$, 0, $16 \cdot 1$, $16 \cdot 3$, $64 \cdot 1$, $8 \cdot 1$ strict embeddings. (Notice that in case (iv), the pattern has 8 automorphisms, the target has 8, and the image has 4. So we get $(8 \cdot 8)/4 = 16$ different embedding functions f.)

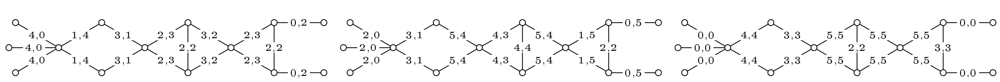
263. Spectacularly false. For example, if $H = G \mathbin{—\!—} K_1$ then $H^{\le 2}$ is a complete graph.

264. The degree of 12 in $G^{\le 2}$ is 11. So we can exclude 22 vertices whose degree in $H^{\le 2}$ is 10 or less: {AZ, CA, CT, DC, DE, FL, GA, LA, MA, ME, MI, MN, ND, NH, NJ, NV, OR, RI, SC, TX, VT, WA}. (The text's original method didn't exclude AZ or TX; its supplemental edge labels ℓ_G, ℓ_H did exclude all of these except MN and NV, and picked off also NM and WI.)

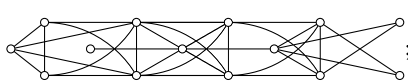
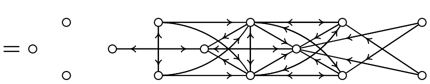
265. $d_G^{P_{k+1}}(v)$ is the number of simple paths of length exactly k that begin at v. (Thus when $k = 1$, $d_G^{P_2}(v) = \deg(v)$.) Consequently v's degree in $G^{\le 2}$ is $d_G^{P_2}(v) + [d_G^{P_3}(v) > 0]$.

266. Symmetrically equivalent vertices have the same label. Left to right, they are: (i) (4, 2, 7, 6, 8, 8, 5, 2); (ii) (0, 20, 2, 12, 6, 12, 6, 0); (iii) (2, 6, 6, 16, 12, 10, 7, 5); (iv) (0, 8, 7, 16, 12, 20, 8, 0); (v) (0, 0, 22, 16, 24, 34, 4, 0); (vi) (0, 0, 0, 2, 4, 4, 2, 2); (vii) (0, 2, 2, 4, 2, 2, 0, 0); (viii) (0, 0, 0, 0, 2, 0, 0, 0); (ix) (0, 0, 0, 2, 0, 2, 0, 0).

267. Here 'a, b' stands for the label left-to-right, then right-to-left:



268. Graph (i) is undirected, because s and t are symmetrically placed.

(i) =  ; (ii) = 

269. Indeed, if $v \mapsto$ MO and v' is diagonally adjacent to v, we can't have $v' \mapsto$ AL, GA, LA, MN, NC, NM, OH, WV, or WY, even though those states are at distance 2 from MO, because no appropriate 4-cycle connects them to MO.

270. Only 15 vertices v of $H =$ USA have at least 4 neighbors in $H^{S,2}$, namely {AR, CO, IA, IL, KS, KY, MO, NE, NV, OK, SD, TN, UT, WV, WY}. Furthermore, if say $11 \mapsto$ NV, then $12 \mapsto$ AZ, CA, ID, OR, or UT; hence $12 \mapsto$ UT. Similarly $11 \mapsto$ NV implies $21 \mapsto$ UT, a contradiction. An analogous contradiction rules out $11 \mapsto$ WV.

273. One part has the neighbors u' of u in G (either $u \longrightarrow u'$ or $u \longleftarrow u'$ or both). The other part has the neighbors v' of v. There's a potential match between u' and v' if and only if all of the following conditions hold: (i) v' is in the current domain of u'. (ii) If $u \longrightarrow u'$ in G then $v \longrightarrow v'$ in H. (iii) If $u \longleftarrow u'$ in G then $v \longleftarrow v'$ in H. (iv) For each supplemental pair label that we've computed, satisfying (64), $\ell_G(u,u') \le \ell_H(v,v')$ and $\ell_G(u',u) \le \ell_H(v',v)$. And if G is to be *strictly* embedded into H, we also have two more conditions: (v) If $u \not\rightarrow u'$ in G then $v \not\rightarrow v'$ in H. (vi) If $u \not\leftarrow u'$ in G then $v \not\leftarrow v'$ in H.

Condition (i) implies that $d_G(u') \le d_H(v')$ for every supplemental label that we've computed, because we used those labels to initialize the domains.

This bipartite matching problem arises not only for the original pattern graph G and the original target graph H, but also (and independently) for every pair of supplemental graphs G^Σ and H^Σ that we know are solutions to (65).

274. Count the number of *strict* embeddings $S \sqsubseteq G$ that map $v \mapsto s$ and possibly $w \mapsto s$, in a motif S with designated vertices s and possibly t. (In particular, when S is $\overline{K}_2$ on the vertices s and t, the complementary graph $G^\Sigma = \overline{G}$ is supplementary.)

277. (a) Choose a vertex p in each connected component, and use breadth-first search to list the elements $p^{(1)}p^{(2)}\ldots$ reachable from p in increasing order of distance, starting with p itself. Concatenate those lists. (Some choices are much better than others.)

(b) (This data structure is a special case of a sparse-set representation.) Maintain also the inverse permutation $u_1 \ldots u_n$ so that, if the target vertices are $\{1,\ldots,n\}$, we have $t_j = k$ if and only if $u_k = j$. Initially $t_j = u_j = j$ for $1 \le j \le n$. When assigning $f(p_{l+1}) = k$, first set $j \leftarrow u_k$, $l \leftarrow l+1$, $k' \leftarrow t_l$, $t_l \leftarrow k$, $t_j \leftarrow k'$, $u_k \leftarrow l$, $u_{k'} \leftarrow j$. Then for each neighbor k'' of k, set $j \leftarrow u_{k''}$ and, if $j > s_l$, set $s_l \leftarrow s_l+1$, $k' \leftarrow t_{s_l}$, $t_{s_l} \leftarrow k''$, $t_j \leftarrow k'$, $u_{k'} \leftarrow j$, $u_{k''} \leftarrow s_l$. Finally, if $s_l < r_l$, set $l \leftarrow l-1$. (That assignment to k cannot be part of a solution, so we must backtrack. No changes to the t and u arrays need to be made when backtracking.)

(c) Yes; this condition is weaker than LAD filtering. (Notice that $q = q_l$ is fixed and can be computed in advance; also a target vertex k is near if and only if $u_k \le s_l$.)

(d) Yes, in the ISIP (strict embedding); again $q = q'_l$ is fixed. But no, in the SIP.

[These heuristics are used by the SIP and ISIP solvers VF2 and VF3 to prune the backtrack tree. See V. Carletti, L. P. Cordella, P. Foggia, A. Saggese, C. Sansone, and M. Vento, *IEEE Trans.* **PAMI-26** (2004), 1367–1372; **PAMI-40** (2018), 804–818.]

278. At step j, H is a Hall set, based on domains different from D_j. [See C. McCreesh and P. Prosser, *LNCS* **9255** (2015), 300–301.]

279. First assume for convenience that the target graph has $n \le 64$ vertices, that all graphs are undirected, and that there are at most 7 supplemental graphs (thus at most 8 altogether). Represent the pattern by an $m \times m$ matrix A_{uv} of bytes; the individual bits of A_{uv} tell us which of the 8 pattern graphs have $u \mathbin{—} v$. Each target graph H^S is represented by n octabytes $H^S_{v'}$; bit u' of $H^S_{v'}$ is 1 if and only if $u' \mathbin{—} v'$ in H^S.

To assign $v \mapsto v'$, first set $D_v \leftarrow \{v'\}$, and mark it "final" so that it won't participate at deeper levels of the search. Then, for every pattern vertex $u \ne v$, we must set $D_u \leftarrow D_u \mathbin{\&} H^S_{v'}$ whenever A_{uv} tells us that $u \mathbin{—} v$ in H^S; we simply set $D_u \leftarrow D_u \setminus \{v'\}$ if $A_{uv} = 0$. (For strict embedding, also set $D_u \leftarrow D_u \mathbin{\&} {\sim}H^S_{v'}$.)

The resulting domains should now be refined further as in exercise 278. That algorithm is readily extended to recognize quickly whether or not at least one nonfinal domain has been reduced to size 1; if so, we repeat the process with a new v and v'.

If the target graph has $n > 64$ vertices, a similar procedure can be carried out with $\lceil n/64 \rceil$ octabytes per domain and with $\lceil n/64 \rceil$ octabytes in place of each $H^S_{v'}$. If the graphs are directed, byte A_{uv} should represent $u \longrightarrow v$ in the pattern graphs, and bit u' of $H^S_{v'}$ should represent $u' \longrightarrow v'$ in H^S. The transposed target graphs should also be represented separately, so that bit u' of $H^{ST}_{v'}$ represents $v' \longrightarrow u'$ in H^S. If A_{vu} tells us that $v \longrightarrow u$ in H^S, we should set $D_u \leftarrow D_u \mathbin{\&} H^{ST}_{v'}$.

initialize the domains
complementary graph
breadth-first search
data structure
sparse-set representation
LAD filtering
VF2
VF3
Carletti
Cordella
Foggia
Saggese
Sansone
Vento
Hall set
McCreesh
Prosser
strict embedding

[*Historical notes:* Bitwise domain reduction was recommended by J. R. Ullmann in one of the first papers about SIP solving, *JACM* **23** (1976), 31–42. See also J. J. McGregor, *Information Sciences* **19** (1979), 229–250, as well as Ullmann's subsequent paper in *ACM J. Experimental Algorithmics* **15** (2011), 1.6:1–1.6:64. C. McCreesh has reported (unpublished) that the state-of-the-art Glasgow solver, c. 2020, spends roughly 1/3 of its time doing bitwise propagation, 1/4 doing relaxed GAD filtering, 1/6 copying domains from one level to the next, and 1/10 choosing the variable on which to branch.]

280. In the following code, D_k is the octabyte in address `dom` $+\,8k$. Sorting is achieved by making byte `START[`i`]` point to the first domain of size i; `NEXT[`k`]` points to the next domain of the same size. The assembler code '`start GREG @ ;next GREG @+64 ;dom GREG @+128`' appears somewhere in the `Data_Segment`, so that we can address those arrays conveniently. Bucket m receives all domains of size $\geq m$, because they can be treated in any order. Symbols `t`, `u`, `h`, `i`, `j`, `k`, `kk` denote registers \$255, \$0, \$1, \$2, \$3, \$4, \$5.

```
Sort SET   j,0         j ← 0.
     SET   i,56        i ← 56.
1H   STOU  j,start,i   START[i..i+7] ← 0.
     SUB   i,i,8       i ← i − 8.
     PBNN  i,1B        Repeat while i ≥ 0.
     CMP   t,i,0       t ← −1.
     STB   t,m,start   START[m] ← −1.
     SET   k,m         k ← m.
1H   8ADDU kk,k,0      kk ← 8k.
     LDOU  t,kk,dom    t ← D_k.
     SADD  t,t,0       t ← |D_k|.
     CMP   i,t,m
     CSP   t,i,m       If t > m set t ← m.
     LDB   i,start,t
     STB   i,next,k    NEXT[k] ← START[t].
     STB   k,start,t   START[t] ← k.
     SUB   k,k,1       k ← k − 1.
     PBP   k,1B        Loop while k > 0.
DoIt SET   u,0         u ← ∅.
     SET   h,0         h ← ∅.
     SET   i,0         i ← 0. (j = 0)
     LDB   k,start,0   k ← START[0].
     PBZ   k,2F        No domain empty?
1H   INCL  j,1         j ← j + 1.
     8ADDU kk,k,0
     LDOU  t,kk,dom    t ← D_k.
     OR    u,u,t       U ← U ∪ t.
     ANDN  t,t,h       t ← t \ H.
     BZ    t,Unfeas    To Unfeas if t = ∅.
     STOU  t,kk,dom    D_k ← t.
     SADD  t,u,0       t ← |U|.
     CMP   t,t,j       t ← sign(t − j).
     BN    t,Unfeas    To Unfeas if |U| < j.
     CSZ   h,t,u       If |U| = j set H ← U.
     LDB   k,k,next    k ← NEXT[k].
     BP    k,1B        Repeat loop if k > 0.
     BN    k,Feas      We're done if k < 0.
2H   INCL  i,1         i ← i + 1.
     LDB   k,i,start   k ← START[i].
     PBP   k,1B        Repeat loop if k > 0.
     PBZ   k,2B        Increase size if k = 0. ■
Feas ...
```

The total time is approximately $(8\mu + 30\upsilon)m + 10\mu + 38\upsilon$.

Complete GAD filtering can also be done with bitwise manipulation, but the algorithms are considerably more complicated and time-consuming. See P. Van Kessel and C.-G. Quimper, *Proceedings of the AAAI Conference* **26** (2012), 577–583.

283. (a) The problem is to find knight paths $p_1 \ldots p_m$ and $q_1 \ldots q_n$ so that the mn cells $p_i + q_j$ lie in a chessboard and are distinct. There are respectively (2, 13, 16, 3) essentially different solutions for $(m,n) = (2,22), (3,12), (4,7), (6,6)$; examples appear in (i)–(vi) of Fig. A–14. The symmetrical constructions (iv) and (v) show that $P_{n-2} \mathbin{\square} P_{n-2} \subseteq N_n$ for all $n \geq 4$, indeed in at least two different ways when n is even. Case (vi) is delightfully "symmetrical" although it has no nontrivial automorphism: It arises from 64 different embedding functions f, while cases (iv) and (v) arise from only 16 each.

(b) Every extremal solution is shown in (vii)–(xiv) of Fig. A–14.

(c) Case (xv) is one of three essentially different solutions for $n = 20$.

(d) Case (xvi) is the essentially unique solution for $n = 8$.

(e) Case (xvii) is one of two essentially different solutions for $n = 8$.

(f) Case (xviii) is the essentially unique embedding for $n = 2$, and it's strict.

[Incidentally, the 4-cube $P_2 \mathbin{\square} P_2 \mathbin{\square} P_2 \mathbin{\square} P_2$, which is also $C_4 \mathbin{\square} C_4$, is *uniquely* embeddable in N_n for all $n \geq 7$, and that embedding is in fact strict.]

Historical notes
Ullmann
McGregor
McCreesh
copying domains
`OR`
`ANDN`
Van Kessel
Quimper
geek art
automorphism
4-cube

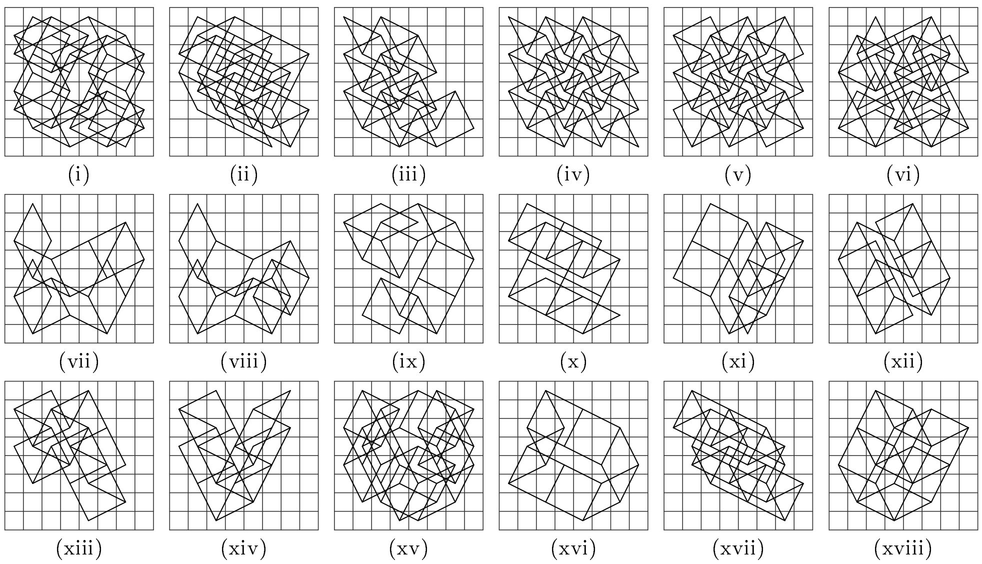


Fig. A–14. A gallery of knight's grids in a chessboard.

284. Although SIP solvers use sophisticated techniques like filtering and supplemental labels, the special geometry of these problems means that a specially tuned backtrack search can be significantly faster. For example, suppose t is given, as well as a fixed knight path $p_1 \ldots p_m$. Instead of mapping a pattern vertex into a fixed vertex of the target graph N_t, we can map q_1 to the origin and backtrack over all knight paths $q_1q_2\ldots$ for which the points $p_i + q_j$ are distinct and fit into a $t \times t$ region of the plane. That avoids $\Theta(t^2)$ near-similar branches at the top levels of the search tree.

We have $\bigl(f_2(3), \ldots, f_2(11)\bigr) = (1, 2, 7, 10, 15, 22, 29, 36, 46)$; and $f_2(12) \geq 57$, because of the knight path $q_1 \ldots q_{57}$ in Fig. A–15. Using somewhat similar paths one can prove that $f_2(t) = t^2/2 - O(t)$, with most of the cells q_j on "even" rows.

When $m = 3$ we can compute exact results a bit further: $\bigl(f_3(3), \ldots, f_3(14)\bigr) = (1, 1, 3, 5, 9, 12, 16, 20, 27, 33, 39, 48)$; and $f_3(15) \geq 55$ because of a knight path $q_1 \ldots q_{55}$ that sticks to cells (i, j) with $(i + j) \bmod 3$ fixed. Using such paths together with a "crooked path" $p_1p_2p_3$ one can show that $f_3(t) = t^2/3 - O(t)$. However, $f_3(14) = 48$ is obtained with a "straight" $p_1p_2p_3$ and a completely mysterious path $q_1 \ldots q_{48}$.

When $m = 4$ we have $\bigl(f_4(3), \ldots, f_4(17)\bigr) = (1, 1, 2, 4, 5, 7, 10, 15, 18, 22, 25, 34, 37, 43, 52)$, and $f_4(18) \geq 61$. In this case the optimum solutions for $13 \leq n \leq 16$ all occur when $p_1p_2p_3p_4$ is the zigzag path shown as '**1** 1 1 1'; such solutions prove that $f_4(t) = t^2/4 - O(t)$. However, the zigzag path yields only $f_4(17) \geq 49$. Hence the straight path wins when $t = 17$, and the sequence $f_4(t)$ remains mysterious.

Turning now to induced subgraphs, $\bigl(\overline{f}_2(3), \ldots, \overline{f}_2(18)\bigr) = (1, 2, 5, 8, 8, 10, 12, 15, 19, 24, 28, 32, 36, 40, 46, 52)$; also $\bigl(\overline{f}_3(3), \ldots, \overline{f}_3(24)\bigr) = (1, 1, 3, 4, 5, 6, 7, 10, 11, 12, 14, 16, 20, 21, 25, 28, 32, 34, 41, 44, 49, 53)$; furthermore $\bigl(\overline{f}_4(3), \ldots, \overline{f}_4(36)\bigr) = (1, 1, 2, 4, 4, 5, 6, 8, 8, 10, 12, 12, 14, 15, 17, 18, 20, 20, 22, 24, 25, 26, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44)$. It appears that $\lim_{t\to\infty} \overline{f}_2(t)/t^2 = \alpha_2$ and $\lim_{t\to\infty} \overline{f}_3(t)/t^2 = \alpha_3$ for some (unknown) positive constants α_2 and α_3. But $\overline{f}_4(t) = O(t)$, because none of the paths $p_1p_2p_3p_4$ allow us to "turn a corner."

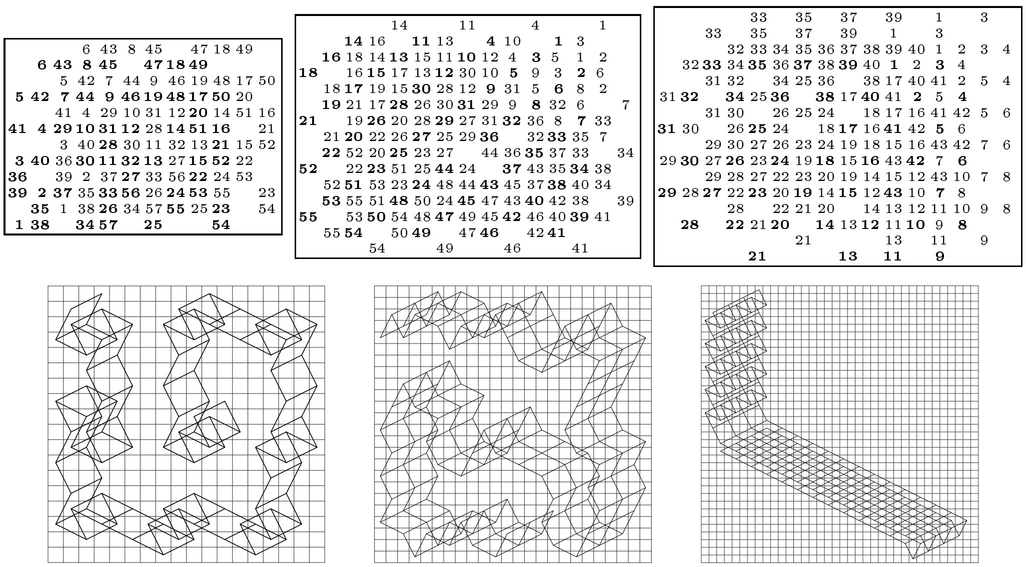


Fig. A–15. Champion knight's grids on larger boards.

McCreesh
Glasgow solver
restarts
symmetrical solutions
Solnon
Rokicki
horizontal and vertical symmetry
axial symmetry
4-fold symmetry
90-degree rotation
central symmetry
Beluhov
Lo Shu
magic square
Dürer
axial symmetry

285. C. McCreesh found the first solution below in 2019, in 446 seconds using 160 parallel threads, on the Glasgow solver (which incorporates random restarts). It seems probable that millions of solutions exist, but a person has to be lucky to find them.

The problem is essentially to label each cell ij of a chessboard with the name of another cell xy, so that when two cells are a knight move apart their labels are a queen move apart. (For example, the knight-move neighbors of the cell labeled 36 in the first solution are labeled 06, 63, 47, and 66.) In problems such as this it's often easier (and fun) to look for *symmetrical solutions*, because such solutions have many fewer variables. For example, we can impose further constraints: (i) if $ij \mapsto xy$ then $ji \mapsto yx$; (ii) if $ij \mapsto xy$ then $i\bar{j} \mapsto x\bar{y}$, where $\bar{y} = 7 - y$; (iii) if $ij \mapsto xy$ then $\bar{i}j \mapsto \bar{x}y$. C. Solnon discovered in 2021 that condition (i) cannot be satisfied. But T. Rokicki found that there are exactly $8 \cdot 4$ ways to satisfy both (ii) and (iii), as in the second solution below, thus achieving "axial symmetry" (see exercise 7.2.2.1–386). He showed furthermore that exactly $8 \cdot 14$ solutions have the other kind of 4-fold symmetry, under 90-degree rotation, as in the third solution; the constraint in this case is (iv) if $ij \mapsto xy$ then $j\bar{i} \mapsto y\bar{x}$. Also exactly $4 \cdot 23$ solutions, like the fourth, satisfy (ii) but not (iii). And $32 \cdot 991$ have central symmetry: (v) if $ij \mapsto xy$ then $\bar{i}\bar{j} \mapsto \bar{x}\bar{y}$, but not (ii) or (iii).

60 30 36 44 74 15 64 14	12 11 06 05 02 01 16 15	67 50 32 41 30 07 34 71	10 40 71 61 66 76 47 17
06 41 65 33 66 17 75 24	51 00 13 41 46 14 07 56	33 63 60 57 12 74 31 02	11 72 60 43 44 67 75 16
10 63 00 47 35 34 11 67	10 42 55 04 03 52 45 17	00 37 23 52 61 35 01 24	62 21 41 77 70 46 26 65
05 56 32 61 77 57 13 31	50 24 30 43 44 37 23 57	73 51 66 15 56 22 72 13	63 00 73 24 23 74 07 64
62 70 27 55 43 71 37 02	20 54 40 33 34 47 53 27	64 05 55 21 62 11 26 04	51 42 27 30 37 20 45 56
45 50 52 76 07 22 53 73	60 32 25 74 73 22 35 67	53 76 42 16 25 54 40 77	05 03 33 53 54 34 04 02
72 26 25 23 51 46 01 04	21 70 63 31 36 64 77 26	75 46 03 65 20 17 14 44	22 57 32 36 31 35 50 25
20 12 54 16 21 03 40 42	62 61 76 75 72 71 66 65	06 43 70 47 36 45 27 10	12 06 13 55 52 14 01 15

286. N. Beluhov notes that the 3×3 "Lo Shu" magic square may actually be regarded as an embedding of N_3 into $P_3 \,\square\, P_3$; and the famous magic square in Albrecht Dürer's *Melencolia I* is an embedding of N_4 into Q_4, with axial symmetry!

Without reducing for symmetry, there are 44176 embeddings for $n = 3$, 171569126 for $n = 4$, and zillions for $5 \le n \le 7$. Restricting to solutions with central symmetry, these counts become (80, 66624, 69200, 1599680, 48560, 32000), for $3 \le n \le 8$.

Surprisingly, no 4-way symmetry is possible for $9 \le n \le 30$. In fact Rokicki found that there are only $32 \cdot 2$ symmetrical solutions for $n = 9$, all with central symmetry.

[But Solnon has discovered that *unsymmetrical* solutions can be obtained quite quickly, with a dynamically weighted improvement of the MRV heuristic, at least for $9 \le n \le 12$! I shall be reporting on that at length, in a future version of these notes.]

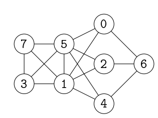
287. (Solution by N. Beluhov.) Of course N_n has very few edges when n is small, so the task is easy; (1, 24, 1296, 69120) embeddings solve the problem when $n = (1, 2, 3, 4)$.

When $n = 5$, there are exactly 28800 embeddings. In fact, they are the mappings $ij \mapsto p((2i+j) \bmod 5)q((3i+j) \bmod 5)$ and their transposes, when p and q are arbitrary permutations of $\{0, 1, 2, 3, 4\}$. Those maps also embed the toroidal 5×5 knight moves.

But it's impossible when $n > 5$, because knight edges of the same slope must map onto rook edges of the same slope. (This is true in each "knight rhombus," and we can connect moves of the same slope by chains of such rhombuses.) And without loss of generality, knight edges of at least two distinct slopes map onto horizontal rook edges.

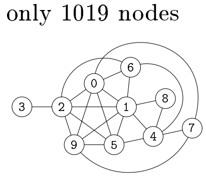
(And in general, the $n \times n$ graph of every skew free (p, q)-leaper is embeddable in the $n \times n$ rook graph for $n = p^2 + q^2$, but not for larger n.)

288. If solvable, there would be headline news: We could name 75 American collegiate football teams who played each other in 1990 if and only if 75 corresponding characters encountered each other in the first half of Victor Hugo's *Les Misérables* (1862)! But unfortunately this one is *not* solvable. Indeed, 95 of the target teams belong to one of eleven "conferences"; and they play almost everybody in their own conference. So the largest independent set among those teams has at most 1+1+1+1+1+1+1+1+2+2+2 members. Since at most 8 of the remaining 25 teams are independent, the target graph has at most 23 independent vertices. But the pattern graph has 27 *isolated* vertices.



290. (a) The unique solution is nicely symmetric. One interesting way to find it is to consider a Boolean function on $\binom{8}{2} = 28$ variables x_{uv}, one for each potential edge u — v. The function that characterizes 4-universal graphs H is $\bigwedge_{G \in \mathcal{G}_4} S(G)$, where $\mathcal{G}_4$ is the set of all 4-vertex graphs and $S(G) = [G \sqsubseteq H]$. For example, when $G = L(3,1)$ we have $S(\;) = \bigvee_{tuvw} x_{tu}x_{tv}x_{tw}x_{uv}\bar{x}_{uw}\bar{x}_{vw}$, which is an OR taken over all $8 \cdot 7 \cdot 6 \cdot 5 = 1680$ ordered quadruples of vertices $tuvw$.

Many simplifications are possible, because H must contain a 4-vertex clique C as well as an independent set I of size 4, having just one vertex in common with C. The eighth vertex must not be adjacent to all of $C \setminus I$, but adjacent to at least one of $I \setminus C$. That leaves only 11 unspecified variables x_{uv}; the resulting BDD has only 1019 nodes and can be computed in only 4 megamems.



(b) It turns out that exactly 90 distinct 4-universal 8-vertex graphs can be strictly embedded in a 5-universal 10-vertex graph — but *not* the graph of (a). This example becomes 4-universal when we delete vertices 8 and 9; further deletion of $\{5, 6, 7\}$ gives the bull.

The Boolean function for all 5-universal graphs in $\mathcal{G}_{10}$, analogous to the one in part (a), has $\binom{10}{2} - 22 = 23$ variables and a BDD of size 3803(!), computed in 2.5 Gμ.

[*Historical notes:* J. W. Moon introduced n-universal graphs in *Proc. Glasgow Math. Assoc.* **7** (1965), 32–33. He defined $\lambda(n)$ as the minimum number of vertices in such a graph, and showed that $2^{(n-1)/2} < \lambda(n) < 1.1n2^{(n-1)/2}$. N. Alon sharpened this

Rokicki
Solnon
MRV heuristic
Beluhov
(p, q)-leaper
leaper
football teams
Hugo
Les Misérables
independent set
isolated
Boolean function
lollipop
paw
clique
BDD
Historical notes
Moon
Alon

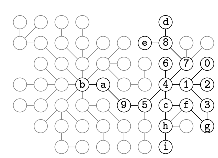
to $\lambda(n) = 2^{(n-1)/2}(1 + O(n^{-1/2}(\log n)^{3/2}))$ in *Geometric and Functional Analysis* **27** (2017), 1–32. Exact values for small n were computed by J. Trimble [arXiv:2109.00075 [math.CO] (2021), 22 pages], who found $(\lambda(1), \ldots, \lambda(6)) = (1, 3, 5, 8, 10, 14)$ and $16 \le \lambda(7) \le 18$. The minimum number of edges in an n-universal graph is (0, 1, 4, 11, 21) for $1 \le n \le 5$; Trimble's examples for $n = 6$ and 7 have respectively 45 and 77 edges.]

291. In fact we can obtain each V_{j+1} by "promoting" a vertex of V_j: $V_1 = 0125$ ($K_{1,1,2}$); $V_2 = 0126$ (C_4); $V_3 = 0136$ (P_4); $V_4 = 0236$ ($P_3 \oplus K_1$); $V_5 = 0237$ ($K_2 \oplus 2K_1$); $V_6 = 0247$ ($\overline{K_4}$); $V_7 = 1247$ ($K_{1,3}$); $V_8 = 1347$ ($L(3,1)$); $V_9 = 1357$ (K_4); $V_{10} = 1367$ ($C_3 \oplus K_1$); $V_{11} = 2367$ ($2K_2$). *Exercise:* Make and post an animated video of this. [See suggestions by Filip Stappers (`https://archive.org/details/gray-4-universal/`) and Ho Boon Suan (`https://www.youtube.com/watch?v=KelZOGPr3Zw`).]

An interesting CSP now suggests itself: Given a digraph in which each vertex v has a given color $c(v) \in \{1, \ldots, d\}$, we seek an oriented path $v_1 \longrightarrow v_2 \longrightarrow \cdots \longrightarrow v_d$ such that each color occurs once in $\{c(v_1), c(v_2), \ldots, c(v_d)\}$. Let's call this the *rainbow path problem.* There's a nice way to formulate it as an XCC: Let there be $3d$ primary items x, $x+$, $x-$ for $1 \le x \le d$, together with a secondary item v for each vertex v; we also have two special primary items $\perp$ and $\top$. If the vertices colored x are $v_1, \ldots, v_t$, there are $3t$ options '$x\ v_1{:}\delta_{1s}\ \ldots\ v_t{:}\delta_{ts}$', '$\perp\ v_s{:}1\ x-$', '$x+\ v_s{:}1\ \top$', for $1 \le s \le t$. Also, for each arc $v \longrightarrow v'$ with $c(v) \ne c(v')$, there's an option '$c(v)+\ v{:}1\ v'{:}1\ c(v')-$'.

This exercise is the special case where each v is a 4-element subset of $\{0, \ldots, 7\}$ and $c(v)$ is the corresponding induced subgraph; $v \longrightarrow v'$ if and only if v' increases an element of v by 1. The associated XCC has 105 items, 341 options, and 22 solutions, found in 3 megamems. (But we were lucky, because there are $8! = 40320$ ways to label the vertices of H and only 4224 of them yield solutions.)

293. (a) The answer is unique, except for permutation of $\{\mathtt{0}, \mathtt{2}, \mathtt{3}\}$:



(b) Yes. Subtree S_r has nodes $\{\mathtt{7}, \mathtt{8}, \mathtt{d}, \mathtt{e}\}$; subtree T_e has nodes $\{\mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{A}, \mathtt{B}\}$; map $\mathtt{7} \mapsto \mathtt{w}$, $\mathtt{8} \mapsto \mathtt{x}$, $\mathtt{d} \mapsto \mathtt{y}$, $\mathtt{e} \mapsto \mathtt{z}$. (This example uses an extended hexadecimal code in which the letters $[\mathtt{a}\,..\,\mathtt{z}]$ denote $[10\,..\,35]$ and the letters $[\mathtt{A}\,..\,\mathtt{Z}]$ denote $[36\,..\,61]$.)

(c) Let $e_j = {}^{\,v}_{w_j}$, for $1 \le j \le l$. The stated embedding is possible if and only if there are analogous embeddings of $S_{r_1}, \ldots, S_{r_k}$ into some k distinct subtrees T_{e_j}.

(d) The condition in (c) is that there's a matching of size k in the graph with k boys, l girls, and $b_i \mathbin{—} g_j \iff \mathtt{sol}[r_i][e_j]$.

(e) Let v's neighbors be $\{w_0, \ldots, w_l\}$; define e_j as in (c), but for $0 \le j \le l$. Now consider the graph of (d), but with $l + 1$ girls. The embedding for $e = {}^{u}_{v}$ is possible when $u = w_j \iff$ there's a matching of size k with g_j *unmatched.* And Algorithm 7.5.1H has the beautiful property that such a matching exists $\iff g_j \in \{\mathtt{QUEUE}[0], \ldots, \mathtt{QUEUE}[q-1]\}$ when that algorithm terminates with no free boys. (This brilliant idea saves us a factor of n. See Theorem 3.4 in Matula's paper, *Annals of Discrete Math.* **2** (1978), 91–106.)

(f) Assign integers $[0\,..\,2n-2)$ to the arcs e of T so that (i) all arcs $e = {}^{u}_{v}$ with the same value of v are consecutive, and (ii) if $\deg(e) < \deg(e')$ then $e < e'$. (Here $\deg(e)$ means $\deg(v)$ when $e = {}^{u}_{v}$.) For $1 \le d \le n$, set $\mathtt{THRESH}[d]$ to the number of arcs with $\deg(e) < d$. If $e = {}^{u}_{v}$ and $e' = {}^{v}_{u}$, set $\mathtt{UERT}[e] \leftarrow u$, $\mathtt{VERT}[e] \leftarrow v$, $\mathtt{DUAL}[e] \leftarrow e'$.

The heart of the computation is *solve*(r), a recursive procedure to set $\mathtt{sol}[q][e]$ for all arcs e and all descendants q of r, where r is a node of S. Here's how it works: If r is a leaf, simply set $\mathtt{sol}[r][e] \leftarrow 1$ for $0 \le e < 2n-2$. Otherwise suppose r_1, $\ldots$, r_k are r's children, and *solve*(r_i) for $1 \le i \le k$. We start with $\mathtt{sol}[r][e] \leftarrow 0$ for

Trimble
internet
video
Stappers
Ho Boon Suan
CSP
rainbow path problem
XCC
hexadecimal code, extended
unmatched girl in maximum matching
recursive procedure

author
downloadable programs

$0 \le e < 2n-2$. Then we set $d \leftarrow k+1$, $e \leftarrow$ `THRESH[`d`]`, and do the following while $e < 2n-2$: While $e =$ `THRESH[`$d+1$`]` set $d \leftarrow d+1$; set up a bipartite matching problem (see below), and use its solution to fix `sol[`r`][`$e+j$`]` for $0 \le j < d$; then set $e \leftarrow e+d$.

(One can abort, concluding that $S \not\subseteq T$, if $solve(r)$ never sets any `sol[`r`][`e`]` $\leftarrow 1$.)

The bipartite graph for r and $[e\,.\,.\,e+d)$ has k boys $b_i = r_i$ (the children of r) and $d > k$ girls $g_j = e+j$, with b_i —— g_j if and only if `sol[`b_i`][DUAL[`g_j`]]` $= 1$. However, several special cases are important: If b_i —— g_j for *no* j, there's no perfect matching and we don't bother to look for one. If b_i —— g_j for *all* j, we omit boy b_i from the graph. (That happens often, for example whenever r_i is a leaf.) So we're left with $k' \le k$ boys, where k' is at most the inner degree of r. If $k' = 0$ (every boy matches every girl), we set `sol[`r`][`$e+j$`]` $\leftarrow 1$ for $0 \le j < d$. Otherwise if Algorithm 7.5.1H terminates with $f = d - k'$, and with q girls in its queue, we set `sol[`r`][QUEUE[`j`]]` $\leftarrow 1$ for $0 \le j < q$.

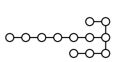
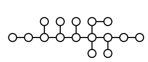
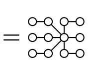
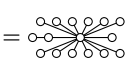
Finally, $S \subseteq T$ if and only if `sol[1][`e`]` $= 1$ for at least one arc e.

Here's the `sol` matrix for the trees of (a):

```
u 11796cclfhnuxxwwCCHOONpRpML1435686aclkmfifo5sjtCQuGF0tVpRJSqMrUqT032sNuwzyGWP5Lr5b975ked5nhguBAxuHED5KJItFCwv4qpjfc6
v 028abdemgiovyzABDEQPWVISKUT33447799kkllhhnnjjssHHFFGGNNJJRRrrMMLL111tttxxx000qqq6666ccccffffwwwwCCCCppppuuuuu5555555
1 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001110000000000000000
2 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
3 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
4 00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001010000000
5 00000000000000000000000000010110101010101011111011101010101010101101111011011111011111110111011101111111111111111111
6 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
7 00000000000000000000000000001100101010001010110010110010100010001000110011000110111011101111111011101111111111111111
8 00000000000000000000000000000000000000000000000000000000000000000111111111111111111111111111111111111111111111111111
9 00000000000000000000000000011101011100101011111011111011101110101101111011011111111111111111111111111111111111111111
a 00000000000000000000000000011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
b 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
c 00000000000000000000000000000000000000000000000000000000000000000000111000000111111100111111011000110101111111111111
d 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
e 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
f 00000000000000000000000000011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
g 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
h 00000000000000000000000000011111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
i 11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

The author's online program MATULA includes additional matrices `solx` and `soly`, which record the `MATE` and `QUEUE` information of subproblem solutions, so that an actual embedding of S into T can be exhibited when one exists.

294. (a) The average running time is less than 2 kilomems, and the standard deviation is very small. There are exactly 516399 pairs with $S \subseteq T$ (4.85%). Tree S_1 is embeddable the most (2016 Ts); tree $S = K_{1,12}$ is embeddable the least (31 Ts); tree T_1 has the most embedded subtrees (74 Ss); trees $T = P_{16}$ and $K_{1,15}$ have the fewest (1 S); trees S_2 and T_2 lead to the largest bipartite matching problem (5 boys, 14 girls).

$S_1 =$  $\subseteq$ max; $T_1 =$  $\supseteq$ max; $S_2 =$  $\not\subseteq$ $T_2 =$ .

(b) These tests, which take about 250 kilomems, find $S \subseteq T$ slightly more than half of the time, and rarely need bipartite matching with more than 3 boys.

(c) These tests take about 10 megamems, and find a unique embedding about 30% of the time. (About 5% of the time there are five or more.) The matching problems usually all have fewer than 5 boys and fewer than 9 girls.

295. If D has n vertices, there's a solution if and only if $\vec{K}_{n-k} \sqsubseteq D$.

297. Let there be a primary item v for each variable v, and let D_v be v's domain. Let there be a secondary item u_iv_j for all elements of $N = \{(u,v,i,j) \mid u < v,\ i \in D_u,\ j \in D_v,\ (u,v) = (i,j) \text{ disallowed}\}$. There's one option for each v and each $j \in D_v$:

$$`v\ \ \bigvee_{(u,v,i,j)\in N} u_iv_j\ \ \ \bigvee_{(v,u,j,i)\in N} v_ju_i\text{'}.$$

Prestwich
Klavžar
ternary-to-binary encodings
unit clauses
weakened encoding

For radio coloring we can in fact do better. Let $D_v = [0\,..\,d)$ for all v, and $N = \{(u,v,i,j) \mid u \mathrel{—} v,\ u < v,\ |i-j| < 2\}$; introduce also a secondary item v_j for each v and j, meaning that v has a neighbor colored j. The option for v and j is then

$$\text{‘}v \ \ \textstyle\bigvee_{u—v} u_j \ \ \bigvee_{(u,v,i,j)\in N} u_i v_j \ \ \bigvee_{(v,u,j,i)\in N} v_j u_i\text{’}.$$

298. (a) 608 ($d=7$); (b) 95520 ($d=10$); (c) 3311464 ($d=12$); (d) 401800 ($d=11$).

300. (a) Yes. For example, $(\bar u_2 \vee \bar v_2)$ says that we don't have $u=v=2$. (Like the binary encoding, it allows all pairs of binary bits in each variable's representation. But it's better, because it lumps 11 together with 10 instead of with 00.)

(b) $(000, 001, 01{*}, 1{*}{*})$. (See also the variable-length example in 6.2.2–(33).)

(c) Omit 000. (S. Prestwich introduced this alternative in order to study encodings that have many bit patterns assigned to a single value.)

301. $v_2v_1 = (00, 01, 10, 11)$ means $v = (0, 1, 2, 1 \text{ or } 2)$. The allowable $u_2u_1v_2v_1$ are 0001, 0010, 0011, 0100, 0110, 1000, 1001, 1100; hence $u \ne v$. (See also answer 332.)

302. Direct: (vars, clauses, totlits) $= (3V, 4V+3E, 9V+6E)$. Multivalued: $(3V, V+3E, 3V+6E)$. Log or Ordered: $(2V, V+3E, 2V+8E)$. Binary: $(2V, 6E, 24E)$. Support: $(3V, 4V+6E, 9V+18E)$. Weakened: $(3V, V+3E, 3V+12E)$. Reduced: $(2V, 3E, 8E)$. Prefix: $(2V, 3E, 10E)$. Curiously, the multivalued encoding has fewer total literals than the reduced encoding when $E > \frac{3}{2}V$, although it has more variables and more clauses.

303. By induction on n, the colors at the corners are uniquely determined: Given the colors of vertices $01\ldots1$ and $02\ldots2$, there are two ways to 3-color each of the subgaskets $1{*}\ldots{*}$ and $2{*}\ldots{*}$; but three of those four possibilities fail to hook up. [S. Klavžar, *Taiwanese Journal of Mathematics* **12** (2008), 513–522.]

304. True. There are two 3-colorings when $n=1$. And any 3-coloring $\subseteq S_n^{(3)}$ with equal colors at two corners can be extended to a 3-coloring $\subseteq S_{n+1}^{(3)}$, in one or two ways.

306. The hint follows by induction. Consider three ternary-to-binary encodings

$$\begin{array}{lll} 0\rho=\bar1; & 0\sigma=1; & 0\tau=\bar1;\\ 1\rho=\bar1; & 1\sigma=\bar1; & 1\tau=1;\\ 2\rho=1; & 2\sigma=\bar1; & 2\tau=\bar1; \end{array} \qquad \text{and let } a_1\ldots a_{n-1} \mapsto \begin{pmatrix} ((a_1\rho)\ldots(a_{n-1}\rho))_2,\\ ((1a_1\sigma)\ldots(a_{n-1}\sigma))_2,\\ ((a_1\tau)\ldots(a_{n-1}\tau))_2 \end{pmatrix}.$$

For example, $1202 \mapsto \bigl((\bar11\bar11)_2, (1\bar11\bar11\bar1)_2, (1\bar1\bar1\bar1)_2\bigr) = (-5, 5, 1)$. It's easy to verify that $\alpha \mapsto (x,y,z)$ implies that x, y, and z are odd numbers with $x+y+z=1$. Conversely, one can go back from such (x,y,z) to α, but only if α is a ternary vector $a_1\ldots a_{n-1}$.

To go from the representation of triangle α to its three vertices $\alpha0$, $\alpha1$, or $\alpha2$, add respectively $(-1,-1,1)$, $(-1,1,-1)$, or $(1,-1,-1)$. The corner points are $0\ldots00 \mapsto (-2^n, 2^{n+1}, -2^n)$; $1\ldots11 \mapsto (-2^n, 0, 2^n)$; and $2\ldots22 \mapsto (2^n, 0, -2^n)$.

307. Assert the unit clauses $(\bar u_1)$, $(\bar u_2)$, (v_1), $(\bar v_2)$, $(\bar w_1)$, (w_2). In the weakened encoding, also assert $(\bar v_0)$ and $(\bar w_0)$.

309. Here are typical running times for Algorithm 7.2.2.2C, in units of 10^n mems:

	without clique hints								*with clique hints*								
	3	4	5	6	7	8	9	n	3	4	5	6	7	8	9	10	11
Dir	1.0	4.3	4.7	4.1	5.3	7.6	12.0		1.2	4.0	4.3	3.7	3.5	4.5	7.9		
Mul	0.9	4.6	4.3	4.1	4.8	7.1	13.1		1.2	4.1	3.4	3.8	4.3	4.9	7.5		
Log	0.9	3.1	3.9	3.9	3.2	4.9	8.1		1.1	4.0	3.6	3.0	2.8	3.4	5.8	10.7	23.5

Bin	28.9	48.2	70.3	78.5	89.7											
Sup	2.4	6.4	6.9	5.8	5.9	6.9	13.3	2.8	7.8	7.0	6.2	6.0	6.8	9.6		
Wea	9.1	14.9	14.5	14.7	16.9	23.6	27.0									
Red	0.8	3.9	3.3	3.4	3.2	3.7	8.6	1.0	3.8	3.8	2.7	2.6	3.1	6.5	12.9	24.0
Pre	3.4	7.7	7.5	5.7	6.4	7.5	11.2	1.4	2.8	3.4	3.0	2.3	3.0	4.8	7.3	14.6

Tests were omitted when there was little chance of success. Since runtime depends on random choices, each experiment was done nine times; the median is shown. (For example, the values for Pre when $n = 11$ ranged from 11.2 to 16.1.) Algorithm 7.2.2.2L gives very similar results when $n \le 5$. But for $n \ge 6$ it often loses focus and takes forever.

311. For $n > 1$ let X_n be the search tree size for 3-coloring $S_n^{(3)}$ when distinct colors have already been specified for vertices $0\ldots00$ and $0\ldots01$; also let Y_n be similar, but for vertices $0\ldots00$ and $1\ldots11$; and let Z_n be similar, but for $\widehat{S}_n^{(3)}$ instead of $S_n^{(3)}$. Then $X_2 = Y_2 = 5$ and $Z_2 = 2$. Also $X_{n+1} = X_n + (X_n + Y_n - 1) + (X_n + 2Z_n)$; $Y_{n+1} = 1 + (X_n + Y_n - 1 + Y_n - 1) + (X_n + 2Z_n)$; $Z_{n+1} = X_n + Y_n - 1 + 2Z_n$. Via generating functions we find $X_{n+1} = Y_{n+1} = 2^{n-1}L_{2n+1} + 1$ and $Z_{n+1} = 2^{n-1}L_{2n} - 1$, where $L_m = F_{m+1} + F_{m-1}$ is a Lucas number. The answer, if we decide to save a factor of 6 by prespecifying the colors of $0\ldots00$ and $0\ldots01$, is $Z_n = \lfloor (3+\sqrt{5}\,)^{n-1}/2 \rfloor$.

313. For all $n > 2$, Filip Stappers has constructed 2^{n-1} sets of $2^{n-1}+1$ removable edges as follows: Each set includes the two "tip" edges that touch vertex $2\ldots2$, plus $2^{n-1}-1$ non-tip edges. When $n = 2$ the non-tip edge is $00 — 01$, which can also be written $11 — 01$; these two forms have different progeny. When $n > 2$, replace each non-tip edge $\alpha a — \alpha b$ for $n-1$ by two non-tip edges $\alpha aa — \alpha ac$, $\alpha bb — \alpha bc$, where $c \notin \{a, b\}$; also add either $2\ldots200 — 2\ldots201$ or $2\ldots211 — 2\ldots210$ as a further non-tip edge.

For example, when $n = 4$ we get eight sets of nine removable edges, such as: $0000 — 0001$, $0022 — 0021$, $0111 — 0110$, $0122 — 0120$, $2000 — 2002$, $2011 — 2012$, $2200 — 2201$, $2202 — 2222$, $2212 — 2222$. (This construction actually produces *all* of the largest removable sets when $n \le 4$. Is that conjecture actually true for all n?)

315. (a, c) $(u_1 \vee v_1 \vee w_1) \wedge (u_2 \vee v_2 \vee w_2)$.
(b) $(u_0 \vee v_0 \vee w_0) \wedge (u_1 \vee v_1 \vee w_1) \wedge (\bar{u}_1 \vee \bar{v}_1 \vee \bar{w}_1) \wedge (u_2 \vee v_2 \vee w_2)$.
(d) $(u_1 \vee v_1 \vee w_1) \wedge (\bar{u}_1 \vee \bar{v}_1 \vee \bar{w}_1) \wedge (u_2 \vee v_2 \vee w_2)$.

316. If $(\neg 0202_2)$ and $(\neg 0222_2)$, the clique hint $(0202_2 \vee 0212_2 \vee 0222_2)$ implies (0212_2). Hence $(\neg 0201_2)$; and $(0022_2 \vee 0201_2 \vee 0202_2)$ implies (0022_2), contradicting (0002_2).

318. The construction of exercise 7.2.2.1–117(b) nicely sets up an exact cover problem with $\lceil 3^n/2 \rceil$ primary items, 3^n secondary items, and $\lceil 3^{n+1}/2 \rceil - 5$ options, each of size at most 3. (The colors of the first three vertices are forced.) And Algorithm 7.2.2.1X nicely proves uncolorability, with a search tree of size $Z_n = O(5.24^n)$ (see exercise 311). Indeed, only a few links need to dance at every node of that tree.

But there's a catch! The author's implementation of step X3, which was used for many of the experiments in Section 7.2.2.1, looks at the `LEN` fields of *every* uncovered primary item, when choosing the item for branching by MRV (see exercise 7.2.2.1–9). So his implementation incurs a cost of $\Omega(3^n/2 - l)$ for each node at level l. That's foolish — because in this problem all the `LEN` fields are at most 3!

A better implementation of step X3 solves the problem with only $O(n)$ steps of computation at each node, while making precisely the same choices: We maintain heap-ordered lists of all uncovered primary items that have a given length. (See Algorithm 5.2.3H.) Then we simply choose the smallest item in the nonempty list of least `LEN`. With that improvement the running time for $\widehat{S}_n^{(3)}$ is $\approx (1.7n + 12)(3+\sqrt{5}\,)^n\mu$.

focus
recurrence relations
generating functions
Lucas number
Fibonacci numbers
Stappers
dance
author
`LEN` fields
primary item
MRV
heap-ordered lists

In this particular problem it turns out to be very important to choose the smallest item each time; otherwise the algorithm gets lost and exercise 311 does not apply. Notice that the secondary items (three per clique) could actually be made primary; surprisingly, however, that changes the order of exploration and messes everything up.

319. The same question can of course be asked for $\widehat{S}_n^{(d)}$ and $\overline{S}_n^{(d)}$.

320. We observed in Section 7.2.2.2 that Algorithm 7.2.2.2L is hopelessly slow for this problem. Runtimes shown here are in units of q^2 kilomems. One of several surprises in this experiment is that the weakened encoding performs much better than expected, especially when q is small.

q	29	49	99	199	399	799	1599	3199
Dir	58	44	46	40	52	56	40	
Mul	96	60	53	36	57	99	62	
Log	35	24	24	28	29	33	19	17
Sup	148	169	111	99	109	152	105	
Red	53	29	31	27	34	37	29	30
Wea	32	35	41	55	66	51	39	58
Pre	47	31	35	35	77	77	35	38

321. (The author hopes that some reader will supply a good answer. His best so far is to remove nine edges, such as these: $a_0 — b_1$, $e_0 — d_1$, $f_0 — c_1$, $a_1 — b_1$, $b_1 — c_1$, $b_1 — d_1$, $c_1 — d_1$, $c_1 — e_1$, $d_1 — f_1$.)

323. $013 = 031$; $113 = 131$; $123 = 132$; $133 = 311$; $213 = 231$; $312 = 321$; $313 = 331$.

324. A path of length 2^{n-1}. (And $\overline{S}_n^{(2)}$ is a $(2^{n-1}+1)$-cycle.)

325. True: Consider the vertices $a_1 \ldots a_n$ with $a_j < d$ for all j. (And we can independently remap the coordinates of those vertices in $d'^{\underline{d}} = d'(d'-1)\ldots(d'-d+1)$ ways.)

326. The pure vertices $j \ldots j$ for $0 \le j < d$ ($S_n^{(d)}$), $2 \le j < d$ ($\widehat{S}_n^{(d)}$), $1 \le j < d$ ($\overline{S}_n^{(d)}$).

328. (a) $d^n(d-1)/2$ clique edges; $(d^n - d)/2$ nonclique edges.

(b) Contract all the nonclique edges.

(c) Add a loop to each pure vertex $j \ldots j$, then take the line graph.

[The graphs $s_n^{(d)}$ for arbitrary d were introduced by S. Klavžar and U. Milutinović, in *Czechoslovak Mathematical Journal* **47** (1997), 95–104; a few years later, M. Jakovac, in *Ars Combinatoria* **116** (2014), 395–405, introduced $S_n^{(d)}$. For a comprehensive survey of graph-theoretical properties satisfied by these and similar graphs, see A. M. Hinz, S. Klavžar, and S. S. Zemljič, *Discrete Applied Mathematics* **217** (2017), 565–600.]

330. Each of d^n vertex labels receives a color, and each color c appears d^{n-1} times — once in every clique. And c appears an even number of times on the impure labels, since they're paired up. So its pure appearances are congruent to d^{n-1} (modulo 2).

Incidentally, a d-coloring of $S_2^{(d)}$ is essentially a self-transpose $d \times d$ latin square.

332. Each variable v must be represented individually. Direct and Support: d Boolean variables $v_j = [v = j]$, with the at-least-one clause $(v_0 \vee \cdots \vee v_{d-1})$ and $\binom{d}{2}$ at-most-one clauses $\bar{v}_i \vee \bar{v}_j$. Multivalued and Weakened: Omit those at-most-one clauses. (If $v_j = 1$ and $v_k = 0$ for $j < k < d$ in the weakened encoding, $v = j$.) Log: $l = \lceil \lg d \rceil$ variables $v_1, v_2, v_4, \ldots,$ denoting $v = (\ldots v_4 v_2 v_1)_2$. Assert clauses of length l to exclude the cases $d \le v < 2^l$. (Those clauses can often be shortened; for example, to exclude $v > 4$ when $d = 5$ it suffices to assert $(\bar{v}_4 \vee \bar{v}_2)$ and $(\bar{v}_4 \vee \bar{v}_1)$.) Prefix: Again $\lceil \lg d \rceil$ variables, but there are no constraints; $v = j$ is represented by the path to the jth leaf in the complete binary tree with j external nodes. For example, the five values when $d = 5$ are represented by $v_4 v_2 v_1 = 000$, 001, $01{*}$, $10{*}$, $11{*}$, effectively lumping together the binary values $\{2,3\}$, $\{4,5\}$, $\{6,7\}$. Reduced: $d-1$ variables $v_j = [v = j]$ for $0 < j < d$. Order: $d-1$ variables $v^j = [v \ge j]$ for $0 < j < d$; assert $(\bar{v}^j \vee v^{j-1})$ for $1 < j < d$.

We also must assert clauses to prohibit $u = j$ and $v = j$. Direct, Multivalued: $(\bar{u}_j \vee \bar{v}_j)$. Reduced: Same, but assert $(u_1 \vee \cdots \vee u_{d-1} \vee v_1 \vee \cdots \vee v_{d-1})$ when $j = 0$.

weakened encoding
author
pure vertices
contraction of a graph
line graph
Klavžar
Milutinović
Jakovac
Zemljič
self-transpose
latin square
complete binary tree

Log: Assert a clause of length $2l$ from the binary representation of j; for example, when $l = 3$ and $j = 4$, assert $(\bar u_4 \vee u_2 \vee u_1 \vee \bar v_4 \vee v_2 \vee v_1)$. (However, that clause can be shortened to $(\bar u_4 \vee u_1 \vee \bar v_4 \vee v_1)$ when $d = 6$, and to $(\bar u_4 \vee \bar v_4)$ when $d = 5$.) Support: Assert $(\bar u_j \vee v_1 \vee \cdots \vee v_{j-1} \vee v_{j+1} \vee \cdots \vee v_{d-1})$, and the same with $u \leftrightarrow v$. Weakened: Assert $(\bar u_j \vee u_{j+1} \vee \cdots \vee u_{d-1} \vee \bar v_j \vee v_{j+1} \vee \cdots \vee v_{d-1})$. Prefix: Assert a clause of length $2l$ or $2l-2$ based on the path to leaf j. For example, when $d = 5$ and j corresponds to $\{4,5\}$, assert $(\bar u_4 \vee u_2 \vee \bar v_4 \vee v_2)$. (See exercise 7.2.2.2–391(c).) Order: Assert $(\bar u^j \vee u^{j+1} \vee \bar v^j \vee v^{j+1})$; but omit $\bar u^0$, $\bar v^0$, u^d, v^d (which are always false). SAT

333. We assume that all domain sizes are d, and that we want to assert all possible hints when the underlying constraint graph has a d-clique $\{v^{(1)}, \ldots, v^{(d)}\}$. Let v_k be one of the Boolean variables representing vertex v. If we know that $v_k = 1$ for at least one v in any c-clique, where $3 \le c \le d$, we can assert the positive clause $(v^{(i_1)} \vee \cdots \vee v^{(i_c)})$ for all $\binom{d}{c}$ subsets $\{i_1, \ldots, i_c\} \subseteq \{1, \ldots, d\}$. Similarly, if we know that $v_k = 0$ for at least one such v, we can assert the negative clause $(\bar v^{(i_1)} \vee \cdots \vee \bar v^{(i_c)})$ for all such subsets.

Let's assume, for example, that the vertices $\{u, v, w, x, y\}$ form a clique when $d = 5$. Direct, Multivalued, Support, and Reduced have positive hints $(u_j \vee v_j \vee w_j \vee x_j \vee y_j)$ for $0 \le j < d$; we must, however, omit $j = 0$ in the reduced encoding, where v_0 doesn't exist. Log encoding, likewise, has $(u_4 \vee v_4 \vee w_4 \vee x_4 \vee y_4)$; and when $j \in \{1, 2\}$ it also has five positive clauses for $c = 4$, namely $(u_j \vee v_j \vee w_j \vee x_j)$, …, $(v_j \vee w_j \vee x_j \vee y_j)$, as well as ten negative clauses for $c = 3$, such as $(\bar u_j \vee \bar v_j \vee \bar w_j)$. Thus, Log has $1+5+5+10+10 = 31$ hints altogether, for every 5-clique(!). Order has even more: Positive for $cj \in \{32, 43, 54\}$ and negative for $cj \in \{33, 42, 51\}$, totalling $10+5+1+10+5+1 = 32$. (Examples are the hints $(\bar u^1 \vee \bar v^1 \vee \bar w^1 \vee \bar x^1 \vee \bar y^1)$, $(\bar u^2 \vee \bar v^2 \vee \bar w^2 \vee \bar x^2)$, and $(u^3 \vee v^3 \vee w^3)$.) And Prefix has positive hints for $cj \in \{42, 44, 51\}$, negative hints for $cj \in \{33, 34, 51\}$, also totalling 32. Finally, Weakened has positive hints for $cj \in \{50, 51, 52, 53, 54\}$, negative hints for $cj \in \{33, 42\}$.

335. Suppose we have a 4-coloring h, with $h(a_1 \ldots a_n) \in \{0, 1, 2, 3\}$ for all vertices $a_1 \ldots a_n$. If π is any permutation of $\{1, 2, 3\}$, let $0\pi = 0$. Then $h'(a_1 \ldots a_n) = h\bigl((a_1\pi)\ldots(a_n\pi)\bigr)\pi^-$ is a 4-coloring; and $h'(0\ldots0j) = h(0\ldots0(j\pi))\pi^- = j\pi\pi^- = j$.

Consequently we can assume without loss of generality that $h(0\ldots011) = 0$. Let v_k be the vertex $a_1 \ldots a_n$ such that $k = (a_1 \ldots a_n)_2$. Then the sequence $h(v_1)$, $h(v_3)$, $h(v_5)$, …, $h(v_{2^n-1})$ begins 1, 0, and ends with 2 or 3. So there's a first odd index j with $h(v_j) > 1$, and we can assume without loss of generality that $h(v_j) = 2$.

We could exploit this when backtracking to save a factor of at least 3. But if we are using SAT, the assertions $(0\ldots011_0)$ and $(\neg 0\ldots101_3)$ *don't* actually give any speedup.

336. (Prefix = Log when $n = 4$.) To avoid decimal points in the table below, the running times are given in units of 10^{2n-4} mems, rounded to two significant digits.

	$\overline{S}_3^{(4)}$	$\overline{S}_4^{(4)}$	$\overline{S}_5^{(4)}$	$\overline{S}_6^{(4)}$	$\overline{S}_7^{(4)}$
Dir	580	200	24	12	13
Mul	580	130	33	18	17
Log	5900	2600	440	62	48
Sup	2100	800	250	85	24
Wea	140000	8900	1700	290	450
Red	9100	5000	200	20	24
Ord	680	130	26	11	16

	$\widehat{S}_3^{(5)}$	$\widehat{S}_4^{(5)}$	$\widehat{S}_5^{(5)}$	$\widehat{S}_6^{(5)}$
Dir	460	650	160	25
Mul	22000	2500	250	30
Log	8400	6700	1600	
Sup	8700	3900	480	
Pre	6100	4700	1200	1600
Red	16000	1800	180	17
Ord	2500	1700	320	150

337. Similarly, the table entries above are in units of 10^{2n-2} mems. Reasons for the sterling performance of the direct encoding when $n = 3$, and for the poor performance of the prefix encoding when $n = 6$, are unknown.

338. Here are the clique-hint runtimes for Kissat 2022-light on an Intel Xeon computer, model E5-2620 v4 2.1GHz, reported by Armin Biere. (The units for $\widehat{S}_n^{(3)}$, $\overline{S}_n^{(4)}$, $\widehat{S}_n^{(5)}$ are respectively 10^{n-11}, 10^{n-7}, and 10^{n-5} sec; the units for $L_q = L(J_q)$ are q^2 μsec. Algorithm 7.2.2.2C is totally eclipsed on the $\widehat{S}_n^{(3)}$ and $\overline{S}_n^{(4)}$ benchmarks!)

	$\widehat{S}_9^{(3)}$	$\widehat{S}_{10}^{(3)}$	$\widehat{S}_{11}^{(3)}$	$\widehat{S}_{12}^{(3)}$	$\overline{S}_6^{(4)}$	$\overline{S}_7^{(4)}$	$\overline{S}_8^{(4)}$	$\widehat{S}_5^{(5)}$	$\widehat{S}_6^{(5)}$	$\widehat{S}_7^{(5)}$	$\widehat{S}_8^{(5)}$	L_{1023}	L_{2047}	L_{4095}	L_{8191}
Dir	170	84	45	26	6	7	9	8	6	28	31	62	57	49	15
Mul	150	81	45	24	10	11	19	34	40	37		28	23	17	13
Log	110	60	32	18	54	38	27	40	39	38		23	18	14	10
Sup	180	73	54	28	25	31	33	47	57	59		20	15	10	6
Red	100	59	35	18	17	100	46	13	11	11	14	21	20	14	10
Wea	120	85	50	29	170	210	340					32	19	14	11
Pre	60	44	33	21	54	38	27	55	74	76		30	24	27	19
Ord	110	60	32	18	7	14	51	22	26	30	25	23	18	14	10

341. Changing the notation to gain symmetry, let's encode '$u+v \ge 2^n-1+t$', where $u = (u_{n-1}\ldots u_0)_2$ and $v = (v_{n-1}\ldots v_0)_2$. It's the same problem, since $\bar u = (\bar u_{n-1}\ldots\bar u_0)_2 = 2^n-1-u$. There are no constraints if $t \le 1-2^n$; there are no solutions if $t \ge 2^n$.

For all $n>0$ and $1-2^n < t < 2^n$, let $a_{n,t}$ be an auxiliary variable and construct the following clauses: (i) $(\bar a_{n,t} \vee u_{n-1} \vee v_{n-1})$ if $0 \le t < 2^{n-1}$; (i$'$) $(\bar a_{n,t} \vee u_{n-1} \vee v_{n-1} \vee a_{n-1,t+2^{n-1}})$ if $t<0$; (ii) $(\bar a_{n,t} \vee u_{n-1} \vee a_{n-1,t})$; (iii) $(\bar a_{n,t} \vee v_{n-1} \vee a_{n-1,t})$; (iv) $(\bar a_{n,t} \vee a_{n-1,t-2^{n-1}})$, if $t>1$ and $n>1$. (In cases (ii) and (iii), omit $a_{n-1,t}$ if $t \ge 2^{n-1}$.) Then $u+v \ge 2^n-1+t$ if and only if u and v satisfy these clauses with $a_{n,t}=1$, for some values of the other auxiliary variables.

(We can remove $\bar a_{n,t}$, and all clauses that contain pure literals of the form $\bar a_{n',t'}$.)

For instance, $t=-1$ encodes '$u \le v+1$': $(\bar u_8 \vee v_8 \vee a_{3,7})$, $(\bar u_8 \vee a_{3,-1})$, $(v_8 \vee a_{3,-1})$, $(\bar a_{3,7} \vee \bar u_4)$, $(\bar a_{3,7} \vee v_4)$, $(\bar a_{3,7} \vee a_{2,3})$, $(\bar a_{3,-1} \vee \bar u_4 \vee v_4 \vee a_{2,3})$, $(\bar a_{3,-1} \vee \bar u_4 \vee a_{2,-1})$, $(\bar a_{3,-1} \vee v_4 \vee a_{2,-1})$, $(\bar a_{2,3} \vee \bar u_2)$, $(\bar a_{2,3} \vee v_2)$, $(\bar a_{2,3} \vee a_{1,1})$, $(\bar a_{2,-1} \vee \bar u_2 \vee v_2 \vee a_{1,1})$, $(\bar a_{1,1} \vee \bar u_1)$, $(\bar a_{1,1} \vee v_1)$. And '$u \le v-2$' is $(\bar u_8 \vee v_8)$, $(\bar u_8 \vee a_{3,2})$, $(v_8 \vee a_{3,2})$, $(a_{3,-6})$, $(\bar a_{3,2} \vee \bar u_4 \vee v_4)$, $(\bar a_{3,2} \vee \bar u_4 \vee a_{2,2})$, $(\bar a_{3,2} \vee v_4 \vee a_{2,2})$, $(\bar a_{3,2} \vee a_{2,-2})$, $(\bar a_{3,-6} \vee \bar u_4 \vee v_4 \vee a_{2,-2})$, $(\bar a_{2,2} \vee \bar u_2)$, $(\bar a_{2,2} \vee v_2)$, $(\bar a_{2,2} \vee a_{1,0})$, $(\bar a_{2,-2} \vee \bar u_2 \vee v_2 \vee a_{1,0})$, $(\bar a_{1,0} \vee \bar u_1 \vee v_1)$.

342. The shortest "covering" is $(\bar u_0 \vee v_0 \vee \bar w_1) \wedge (u_0 \vee w_1) \wedge (\bar u_1 \vee v_2) \wedge (\bar u_2 \vee v_1) \wedge (v_1 \vee \bar w_2)$.

343. Besides the at-least-one and at-most-one clauses, the direct encoding has preclusion clauses $(\bar u_0 \vee \bar v_2) \wedge (\bar u_1 \vee \bar v_0) \wedge (\bar u_1 \vee \bar v_1) \wedge (\bar u_2 \vee \bar v_1) \wedge (\bar u_2 \vee \bar v_2)$, while the support encoding has $(\bar u_0 \vee v_0 \vee v_1) \wedge (\bar u_1 \vee v_2) \wedge (\bar u_2 \vee v_0) \wedge (\bar v_0 \vee u_0 \vee u_2) \wedge (\bar v_1 \vee u_0) \wedge (\bar v_2 \vee u_1)$.

346. $(R_{00} \vee R_{01} \vee R_{12} \vee R_{20}) \wedge (\overline{R}_{00} \vee u_0) \wedge (\overline{R}_{00} \vee v_0) \wedge (\overline{R}_{01} \vee u_0) \wedge (\overline{R}_{01} \vee v_1) \wedge (\overline{R}_{12} \vee u_1) \wedge (\overline{R}_{12} \vee v_2) \wedge (\overline{R}_{20} \vee u_2) \wedge (\overline{R}_{20} \vee v_0) \wedge (\bar u_0 \vee R_{00} \vee R_{01}) \wedge (\bar u_1 \vee R_{12}) \wedge (\bar u_2 \vee R_{20}) \wedge (\bar v_0 \vee R_{00} \vee R_{20}) \wedge (\bar v_1 \vee R_{01}) \wedge (\bar v_2 \vee R_{12})$ and the at-least-one, at-most-one clauses for u and v.

347. After deducing u_0, v_0, w_0, we have (for example) $\bar w_1$; hence $\overline{R}_{001}$.

350. (a) There are $N = d_1 \ldots d_k - G$ clauses of length k, hence Nk literals altogether.

(b) The clause exemplified by (80) has G literals; the Gk clauses like the left of (81) each have 2; the $d_1+\cdots+d_k$ clauses like the right of (81) have a total of $d_1+\cdots+d_k+Gk$. So the grand total is $(3k+1)G + d_1+\cdots+d_k$.

351. Consider a general relation R as in exercise 350, with Boolean variables v_{ja} for $1 \le j \le k$ and $0 \le a < d_j$. Then $R(a_1,\ldots,a_k)$ is true if and only if every preclusion clause is satisfied with v_{ja_j} true for $1 \le j \le k$ and the other Boolean variables arbitrary. (The reduced encoding without at-most-one is the "multivalued encoding"; see Table 2.)

352. Let C_a be the clause for $a \in D_u$, and let $C = \bigvee\{u_a \mid a \in D_u\}$ be u's at-least-one clause. Given $b \in D_v$, resolve C with each C_a for which $ab \notin R = R(u,v)$; this gives $C' = U_b \vee V_b$, where $U_b = \bigvee\{u_a \mid ab \in R\}$, $V_b = \bigvee\{v_c \mid c \in R'_b\}$, and $R'_b = \{c \mid ac \in R$

Kissat
Intel Xeon computer
Biere
auxiliary variable
pure literals
covering
preclusion clauses
multivalued encoding

for some a with $ab \notin R\}$. If $R'_b \neq \emptyset$, we get the desired clause $(\bar v_b \vee U_b)$ by resolving C' with $(\bar u_c \vee \bar u_b)$ for each $c \in R'_b$. Otherwise the desired clause is subsumed by U_b, which can be obtained by resolving C with C_a for all $a \in D_u$ that have no support in R.

(The other half of the clauses are, however, important for *unit* resolution.)

353. Form the 27-bit vectors for the set of all 2^9 truth tables a_i on (x_1, x_2, x_3) that define binary relations on (x_1, x_2); also similar vectors b_j and c_k for (x_1, x_3) and (x_2, x_3). The number of distinct $a_i \mathbin{\&} b_j \mathbin{\&} c_k$ is 1614530, which is $\approx 1.2\%$ of 2^{27}. (The answer to the analogous question for domain size 2 is 166, by exercise 7.2.2.2–191.)

354. There are 111618 classes; they form 55809 pairs, because the complements of equivalent relations are equivalent. One of the pairs has classes of size 1 (the empty relation and the full relation). Another pair has classes of size 9 (for example, '$x = 0$' and '$x \neq 0$'). Another has classes of size 12 ('$(x \pm y \pm z) \bmod 3 = \text{const}$' or '$(x \pm y \pm z) \bmod 3 \neq \text{const}$', analogous to the parity relations mod 2). Then there's size 18 (like '$x = y$' or '$x \neq y$'). The class containing '$x = y = z = 0$' is one of 20 classes of size 27. The class containing 'x, y, and z are distinct' is one of 4 classes of size 36; so is the class containing '$x = y = z$'. There are 12722 classes of size 648, and 96726 of the maximum size, 1296.

The 1614530 decomposable relations in answer 353 form 1841 equivalence classes. Those classes are *not* closed under complementation; for example, '$\langle xyz \rangle = 1$', whose class has size 108, is decomposable; but '$\langle xyz \rangle \neq 1$' differs in six places (x, y, z) from the intersection of its projections onto $\{x, y\}$, $\{x, z\}$, $\{y, z\}$. Altogether 6034 of the classes, and 6496994 of the relations ($\approx 4.8\%$), are within 1 of that intersection; 65623736 relations are within 5. Only the class that contains '$(x+y+z) \bmod 3 = 0$' is at distance 18.

356. Yes; it's not difficult to prove that $R(u, v, w) = P(u, v) \wedge P(u, w) \wedge P(v, w)$, where $P(u, v) = (\max(u, v) \geq c) \wedge (\min(u, v) \leq c)$.

357. `hells`, `shart`, and `trice`. (But `hells` and `trice` are in `WORDS(3500)`.)

358. (a) If $a_1 \ldots a_k \in R$ there's a solution with $v_{1a_1} = \cdots = v_{ka_k} = 1$.

(b) All clauses are satisfied, and the value of every literal has been unambiguously forced. Furthermore exactly one v_{ja_j} is true for each j.

(c) If v_a becomes false, D_v loses the value a. If v_a becomes true, all $v_{a'}$ for $a' \neq a$ become false; we're left with the support clauses for a relation on the variables $\neq v$.

(d) The current relation R' has at least two elements in D_v.

(e, f) The arguments in (a), (b), (c) remain valid.

Historical note: F. Rossi, C. J. Petrie, and V. Dhar [*ECAI* **9** (1990), 550–556] described the "hidden variable" trick as part of the CSP folklore; U. Montanari had alluded to it on page 105 of his paper of 1974.

360. (a) Each of the n clusters $\{x_{ij} \mid 0 \leq j < n\}$ is an n-clique, so their values must be a permutation of the domain. If $i > 0$ and $j > 0$, $x_{i0} < 2$ implies $x_{ij} \geq 2$; hence $x_{i0} \geq 2$. So the $n-1$ variables $\{x_{01}, \ldots, x_{0(n-1)}\}$ have only $n-2$ available values.

(b) Since there really are only $n^2 - n + 1$ variables, by (iii), we can identify x_{i0} with x_{0i}. Let there be $2n^2 - n + 1$ primary items x_{ij} and v_{ij} for $0 \leq i, j < n$, omitting x_{0j} when $j > 0$. Introduce $2(n-1)^2$ secondary items a_{ij} and b_{ij} for $0 < i, j < n$, in order to forbid $(x_{i0}, x_{ij}) = (0, 1)$ and $(1, 0)$. There's an option containing x_{ij} and v_{ik} for each $0 \leq i, j, k < n$ except when $i = 0$ and $j > 0$. If $i > 0$ and $j = 0$ that option contains also v_{0k}, as well as $a_{ij'}$ for $0 < j' < n$ when $k = 0$, and $b_{ij'}$ for $0 < j' < n$ when $k = 1$. If $i > 0$ and $j > 0$ it contains also b_{ij} when $k = 0$ or a_{ij} when $k = 1$.

The running time for Algorithm 7.2.2.1X is approximately proportional to $(n-1)!$, if the primary items have their natural order; for example, it's $105\mathrm{M}\mu$ when $n = 8$ and

truth tables
bitwise AND
parity
all-different
median of three
projections
Historical note
Rossi
Petrie
Dhar
hidden variable
Montanari
clique
permutation
pigeonhole principle

90Gμ when $n=12$. But the time is much, much longer when they're randomly ordered (e.g., 1880Gμ when $n=7$). On the other hand, Algorithm 7.2.2.1P quickly proves unsatisfiability in $\Theta(n^4)$ steps, because the domains of x_{ij} and v_{ij} are inconsistent. For example, it needs only 22Mμ to remove all options when $n=32$.

(c) Use, for instance, the direct representation, with $x_{ijk}=[x_{ij}=k]$; identify x_{i0k} with x_{0ik}. The clauses for clique i are $A_i \wedge B_i \wedge C_i \wedge D_i$ for $0 \le i < n$, where

$$A_i = \textstyle\bigwedge_{j=0}^{(n-1)[i\neq 0]}\Bigl(\bigl(\bigvee_{k=0}^{n-1} x_{ijk}\bigr) \wedge \bigwedge_{0\le k<k'<n}(\bar x_{ijk} \vee \bar x_{ijk'})\Bigr) \qquad \text{[domain constraints]};$$

$$B_i = \textstyle\bigwedge_{0\le j<j'<n}\bigwedge_{k=0}^{n-1}(i>0?\ (\bar x_{ijk}\vee \bar x_{ij'k}) \colon (\bar x_{j0k}\vee \bar x_{j'0k})) \qquad \text{[clique constraints]};$$

$$C_i = \textstyle\bigwedge_{k=0}^{n-1}(i>0?\ (\bigvee_{j=0}^{n-1} x_{ijk}) \colon (\bigvee_{j=0}^{n-1} x_{j0k})) \qquad \text{[clique hints]};$$

$$D_i = (i>0?\ \textstyle\bigwedge_{j=1}^{n-1}((\bar x_{i00}\vee \bar x_{ij1}) \wedge (\bar x_{i01}\vee \bar x_{ij0})) \colon \wp) \qquad \text{[constraint (ii)]}.$$

Thanks to the clique hints, classical SAT solvers handle this problem quite well. For example, in nine runs for $n=32$ with different random seeds, the median time for Algorithm 7.2.2.2L was 59Mμ, and Algorithm 7.2.2.2C needed only 2.4Mμ. But without the clique hints the runtime is exponential—for example 270Gμ with 7.2.2.2C for $n=11$. The multivalued encoding does poorly too (280Gμ), even with clique hints.

[This problem was introduced by M. R. C. van Dongen as one of the benchmarks for the 2nd international CSP solver competition in 2006. In the competition, of course, only the variables, domains, and constraints were given, and variable names were randomized. A mechanical solver wouldn't be able to deduce unsatisfiability efficiently without somehow understanding the clique structure, and introducing something like the v_{ij} items of (b) or the hints of (c).]

365. (a) Let $q=1-p$. In Pass 1, r_{ij} is examined if and only if $r_{ik}=0$ for $0\le k<j$, hence with probability q^j. So the expected total cost is $\sum_{i=0}^{d-1}\sum_{j=0}^{d'-1} q^j = (1-q^{d'})d/p$.

Pass 2 examines r_{ij} if and only if we have (i) $r_{kj}=0$ for $0\le k<i$; (ii) $r_{ik}=1$ for some $k<j$; and (iii) either $r_{k0}\ldots r_{k(j-1)} \ne 0\ldots0$ or $r_{k0}\ldots r_{kj} = 0\ldots00$, for $i<k<d$. So the probability is $q^i(1-q^j)(1-pq^j)^{d-1-i}$.

Summing this geometric series over i, we find that the total expected cost of Pass 2 is $(1-q^d)d'/p-S$, where $S=\sum_{j=0}^{d'-1}(1-pq^j)^d/p$ is the expected number of unnecessary probes made by the naïve algorithm. [This analysis was first carried out by M. R. C. van Dongen, A. B. Dieker, and A. Sapozhnikov, who also derived a complicated formula for the variance. See *Constraint Programming Letters* **2** (2008), 55–77.]

(b) Do the inner loop only for values of j with $s'_j=0$. Then, if that loop ends with $s_i=0$, do another loop on j, but only for values of j with $m_{ij}=0$. (This algorithm is due to M. R. C. van Dongen; see Fig. 7.3 in his Ph.D. thesis (Cork: National Univ. of Ireland, 2002). The expected number of probes in Pass 1 remains the same; but the expected number of column supports found on that pass is increased. No simple formula is known for the expected number of probes in the subsequent Pass 2.)

[When $d'=2$, the expected cost of this improved algorithm can be shown to equal $(1+q)d+q^{d-1}-q^{2d-1}$. And the expected cost when $d'=3$ turns out to be $(1+q+q^2)d+q^{d-2}+2q^{d-1}-q^{2d-3}(1+q+q^2)+q^{3d-3}(1-q^2)$.]

(c)	$p=.01$	$p=.02$	$p=.03$	$p=.04$	$p=.05$	$p=.10$	$p=.50$	$p=.90$
mean cost (naïve)	12700	8700	6300	4900	4000	2000	400	220
mean cost (a)	8100	6000	4600	3700	3100	1700	390	210
mean cost (b)	7500	5200	3700	2800	2200	1100	200	110
dev (b)	310	300	280	260	200	130	18	3

domain inconsistency
direct representation
$\wp$, tautology
clique hints
multivalued encoding
van Dongen
benchmarks
competition
geometric series
van Dongen
Dieker
Sapozhnikov
variance
van Dongen

366. Although that algorithm treats rows and columns in dramatically different ways, its expected cost does appear to be symmetrical in d and d' (at least when $d \le 4$ or $d' \le 4$). That's nicely consistent with being optimum. Furthermore, the author has proved optimality when $d = 2$, as well as when $(d, d') = (3, 3)$ and $(3, 4)$.

Marc van Dongen observes that an optimum algorithm queries r_{ij} only when either (i) both s_i and s'_j are unknown, or (ii) one of them is known but not the other. Every optimum algorithm can be assumed to make all of its type (i) queries first, because (ii) followed by (i) is never better than (i) followed by (ii).

370. (a) Every x_i has the value of some source, by induction on i. But x_n doesn't.

(b) Let R_i be primary and x_i be secondary for $1 \le i \le n$. Also let $x_{i,j}$ be secondary for $1 \le i \le m$ and $j \in \{0, 1, 2\}$, together with $3m$ options 'R_i $x_i{:}j$ $x_{i,j}$'. Add another primary item #, with three options '# $x_n{:}j$ $x_{1,j}$ $\ldots x_{m,j}$' for $j \in \{0, 1, 2\}$; that takes care of the binary constraints. Finally, introduce $15(n-m)$ options 'R_i $x_i{:}a$ $x_{j(i)}{:}b$ $x_{k(i)}{:}c$' for $m < i \le n$ and for all $a, b, c \in \{0, 1, 2\}$ with ($a = b$ or $a = c$).

(c) Define $j(i)$ and $k(i)$ in $\binom{m}{2}\binom{m+1}{2}\ldots\binom{n-1}{2} = 2^{m-n}(n-1)^{\underline{n-m}}(n-2)^{\underline{n-m}}$ ways.

(d) These problems are tough for Algorithm 7.2.2.1C; for instance, the first random example tried for $m = 24$ and $n = 64$ took 1.4 teramems. But it became much more tractable, only 31 *giga*mems, when each item R_i for $m < i \le n$ was renamed $\#R_i$, and the sharp preference heuristic of exercise 7.2.2.1–10 was used. (That trick also polished off nine other random instances, with a median run time of 1.5 *mega*mems.)

(e) To support $x_i = a$ in a binary constraint, set the other variable to $(a+1) \bmod 3$. To support $x_i = a$ in a ternary constraint, set the other two variables to a.

But after $x_n \leftarrow a$, Algorithm D will remove a from the domains of x_1, x_2, $\ldots$.

[This family of problems was introduced by J. Hwang and D. G. Mitchell, *LNCS* **3709** (2005), 343–357, who showed that with suitable choices of $j(i)$ and $k(i)$ it can be solved via backtracking only with an exponentially large search tree, if every node of that tree is a d-way branch on the value of some variable (or on the options that can cover an item), assuming that the algorithm prunes domains (or removes options) only via forward checking. They devised a Prover–Delayer game, as in Theorem 7.2.2.2R.

On the other hand, a polynomial-size search tree *can* be constructed with *binary* branching, where every search tree node chooses either to include an option or not: For each value a tentatively assigned to x_n, try to include an option for R_i that specifies either $x_{j(i)}{:}a$ or $x_{k(i)}{:}a$, where i is as small as possible. That option leads to an immediate contradiction. So we can remove it, and continue until $x_n = a$ is contradicted.

We can obviously generalize the chain CSP by allowing *arbitrary* ternary constraints R_i for $m < i \le n$, perhaps different for each i. Many such generalizations are likely to be instructive.]

371. (a) Let $X_i = [x_i \text{ is a sink}]$. Then $\mathrm{E}\,X_i = \Pr(X_i = 1) = q_{\max(i,m)+1} \ldots q_n$, where $q_l = \Pr(i \notin \{j(l), k(l)\}) = \binom{l-2}{2}/\binom{l-1}{2}$. Hence $\mathrm{E}\,X_i = \binom{\max(i,m)-1}{2}/\binom{n-1}{2}$; and $S_{m,n} = \sum_{i=1}^{n} \mathrm{E}\,X_i = \frac{n}{3}(1 + 2m^3/n^3)$.

(b) Set $d \leftarrow n - m - 1$, $a_{0,0} \leftarrow 1$. Then for $1 \le i \le d$ and $0 \le j \le i$, set

$$a_{i,j} \leftarrow \binom{m+i-j-2}{2}[j \ne i]\, a_{i-1,j} \;+\; \left(\binom{m+i-1}{2} - \binom{m+i-j-1}{2}\right)[j \ne 0]\, a_{i-1,j-1}.$$

Then there are $a_{i,j}$ cases in which x_i is connected to exactly j of the variables $\{x_{m+1}, \ldots, x_{m+i}\}$. Consequently the number of cases in which x_1 is *not* connected to x_n is $b_{m,n} \leftarrow \sum_{j=0}^{d} \binom{n-j-2}{2} a_{d,j}$; and the probability that a source is connected to x_n is $p_{m,n} = 1 - b_{m,n}/q_{m,n}$, where $q_{m,n}$ is defined in exercise 370(c). Finally, $C_{m,n} = mp_{m,n} + \sum_{i=m+1}^{n} p_{i,n}$. We have $C_{24,64} \approx 8.4023$ and $C_{24,500} \approx 41.08$.

author
van Dongen
sharp preference heuristic
Hwang
Mitchell
backtracking
exponentially large
d-way branch
forward checking
Prover–Delayer game
Delayer
binary branching

(c) Let $f(s,t) = 0$ if $s < 0$ or $t < 0$ and $f(s,t) = [s\!=\!0]$ if $s + t = m$; also $f(s,t) = \binom{t+1}{2} f(s-2,t+1) + (s-1)tf(s-1,t) + \binom{s}{2} f(s,t-1)$ when $s + t > m$. Then $f(s,t)$ is the number of cases with $s+t$ variables, m of which are sources, and t sinks. Hence $c_{m,n} = f(n-1,1)/q_{m,n}$. We have $c_{24,64} \approx 1.7522 \times 10^{-25}$.

Incidentally, $c_{m,n} = 0$ for $n < 2m-1$; and $c_{m,2m-1} = m!(m-1)!^2(m-2)!/((2m-2)!(2m-3)!) = 32 \cdot 16^{-m} m^2 \pi\,(1 + O(1/m))$.)

372. A similar random model, but with each arc chosen independently with replacement, was studied by L. Devroye and S. Janson in *Arkiv för Matematik* **49** (2011), 61–77. Janson observes that $C_{m,n} \le C_{2,n} \le C_{m,n} + m$; thus, when m is fixed, its value doesn't affect the asymptotics of $C_{m,n}$ as $n \to \infty$, and he conjectures that the limit for fixed m is $\sim \sqrt{2\pi n}$. He also conjectures that $c_{m,n}$ approaches $(\rho + o(1))^n$ for some constant $\rho < 1$.

380. There are exactly d ways to insert 'n' into such a permutation of $\{1,\ldots,n-1\}$, namely at the beginning or after one of $\{n-1, n-2, \ldots, n-d+1\}$. So the answer is $d!\,d^{n-d}$, by induction. [The number of permutations with exactly k instances of $p_{j+1} \ge p_j + d$ was investigated by J. Riordan in the final chapter of *An Introduction to Combinatorial Analysis* (1958), thereby generalizing the Eulerian numbers. See OEIS A120434 for the case $d = 2$.]

381. $p_j = n$ if and only if $j \ge 1$ is minimum with $j \notin S$. Remove $p_1 \ldots p_j$ and recurse.

[Richard Stanley, in *Enumerative Combinatorics* **1**, second edition (2012), exercise 1.114(b), discovered another interesting family of permutations with this uniqueness property, namely those $p_1 \ldots p_n$ such that $\{p_1,\ldots,p_k\}$ is an *interval*, for $1 \le k \le n$; a typical example for $n = 7$ is 4325617. Such permutations are readily generated via backtracking, but not so easy to set up as a CSP; the condition is that if $i < j < k$ we don't have $p_i < p_k < p_j$ or $p_j < p_k < p_i$, In other words, they're '$(132, 312)$-avoiding'. Their left-to-right reversals, the $(213, 231)$-avoiding permutations, are precisely those that produce degenerate binary search trees; see exercise 6.2.2–5(b).]

382. True. In fact, the set S corresponding to the inverse is $S^R = \{n - j \mid j \in S\}$.

383. Let p_j and q_j be primary items for $1 \le j \le n$; also introduce $(n-1)(n-2)$ secondary items $j.k$ for $1 \le j < n$ and $1 \le k < m = n-1$, which correspond to items y_k of the pairwise encoding trick that enforces $p_{j+1} \le p_j + 1$. For each $1 \le j,k \le n$ there's an option containing p_j, q_k, and perhaps other items: If $j > 1$ and $k > 2$, set $t \leftarrow k-2$; then while $t > 0$ include $(j-1).t$ and set $t \leftarrow t \,\&\, (t-1)$. (This contributes α_{k-2}.) If $j < n$ and $k \le m$, set $t \leftarrow -k$; then while $t > -m$ include $j.(-t)$ and set $t \leftarrow t \,\&\, (t-1)$. (This contributes β_{k-1}.) For example, the "diagonal" options when $n = 5$ are 'p_1 q_1 1.1 1.2', 'p_2 q_2 2.2', 'p_3 q_3 2.1 3.3', 'p_4 q_4 3.2', 'p_5 q_5 4.3 4.2'.

384. We can also require $q_{k+1} \le q_k + 1$: Introduce new secondary items $j{,}k$ for $1 \le j < n$ and $1 \le k < m$. Whenever answer 383 put $r.s$ in the option for p_j and q_k, also put $r{,}s$ in the option for p_k and q_j. (Thus one option for $n = 5$ is 'p_2 q_3 1.1 2.3 3,2'.)

390. (a)

$t =$	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	26	27	28	29
$l_t =$	0	1	2	1	0	1	2	3	2	1	2	3	4	3	4	5	4	3	2	3	2	1	2	3	2	3	4	3	2	1
$s_t =$	0	0	0	0	0	1	1	0	1	1	2	2	0	2	4	0	4	2	2	3	2	1	2	2	2	3	2	3	2	1

(b) Initially $p_0 \leftarrow 0$, $q_0 \leftarrow \infty$, $r \leftarrow 0$. For $t = 0, 1, \ldots$, do this: Set $k \leftarrow r$ and, while $l_t < p_k$ or $l_t > q_k$, set $k \leftarrow k-1$; then $s_t \leftarrow p_k$. If $l_{t+1} < l_t$ (a backward step), update the intervals as follows: If $l_{t-1} < l_t$ (a "valley"), first set $r \leftarrow r+1$ and $q_r \leftarrow l_t$; then set $p_r \leftarrow l_t$; then if $p_{r-1} = p_r$, set $r \leftarrow r-1$ and, if $q_{r+1} > q_r$, also set $q_r \leftarrow q_{r+1}$.

Devroye
Janson
Riordan
Eulerian numbers
OEIS
Stanley
interval
backtracking
modeling as CSP
patterns in permutations
$(132, 312)$-avoiding
reversals
$(213, 231)$-avoiding
degenerate
binary search trees
valley

For example, after finding $s_7 = 0$ in (a), the current intervals are updated to $[0\,..\,\infty]$, $[1\,..\,2]$, $[3\,..\,3]$ because $t = 7$ is a valley at level 3. They're next updated to $[0\,..\,\infty]$, $[1\,..\,2]$, $[2\,..\,3]$. When eventually $t = 25$ they're $[0\,..\,\infty]$, $[1\,..\,2]$, $[2\,..\,5]$, $[3\,..\,3]$.

(c) The shortest such sequence goes from 0 down to 8, then up to 1, down to 6, up to 3, down to 5, up to 4, down to 15, up to 9, down to 12, up to 10, down to 14, up to 13 at time $8 + 7 + 5 + 3 + \cdots + 1 = 53 = 2\sum_{j=1}^{r}(q_j - p_j + 1) + p_r - 1$. (In general the shortest goes from 0 down to q_1, then up to $p_1 - 1$, ..., down to q_r, up to $p_r - 1$.)

(d)

	$t=$	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	26	27	28	29
either	$l_t=$	0	1	2	3	4	5	4	5	6	5	6	7	6	7	8	7	8	7	8	7	8	7	8	7	6	5	4	3	2	1
	$s_t=$	0	0	0	0	0	0	0	5	0	5	6	0	6	7	0	7	8	7	8	7	8	7	8	7	6	5	0	0	0	0
or	$l_t=$	0	1	2	3	4	5	6	5	6	7	6	7	8	7	8	7	8	7	8	7	8	7	8	7	6	5	4	3	2	1
	$s_t=$	0	0	0	0	0	0	0	0	6	0	6	7	0	7	8	7	8	7	8	7	8	7	8	7	6	0	0	0	0	0

(e) $389533569/9694845 \approx 40.2$. [In general, if we consider $X = \sum_{t=0}^{2m-1} s_t$ over all $\binom{2m}{m}/(m+1)$ level sequences that have m forward steps and m backward steps, empirical results for small m suggest that $\max X \approx .54m^2$.]

(f) Empirically, the average of $l_t - s_t$ for $m = 10^6$ is only about 6.8, although the standard deviation turns out to be about 52; and the empirical average value of r at step t is, incidentally, 441 ± 200. By exercise 2.3.4.5–5, the average of l_t is exactly $\bigl((m+1)4^m - (2m+1)\binom{2m}{m}\bigr)/\bigl(2m\binom{2m}{m}\bigr) = \frac{1}{2}\sqrt{\pi m} + O(1)$; this is ≈ 885 when $m = 1000000$. So backmarking can be expected to save considerable recomputation. [Can the asymptotics of $\sum(l_t - s_t)$ and $\sum r_t$ as $m \to \infty$ be determined analytically?]

(g) Initially $M_{ja} = 0$ for all j and a. We let $a_j = 0$ before x_j is assigned.

M1. [Begin step t.] Set $j \leftarrow l_t + 1$ and $a \leftarrow a_j$. (We want to assign a new value to x_j.)

M2. [Advance a.] Set $a \leftarrow a + 1$. Go to M7 if $a > d_j$. Otherwise repeat this step if $M_{ja} < s_t$. (We've already seen that $x_j = a$ isn't consistent with the currently assigned values $\{x_i = a_i \mid 1 \le i \le M_{ja}\}$.)

M3. [Begin further tests.] Set $k \leftarrow M_{ja}$.

M4. [Check relations with x_k.] If $(a_1, \ldots, a_k, a)$ doesn't satisfy all constraints between $(x_1, \ldots, x_k)$ and x_j that involve x_k, set $M_{ja} \leftarrow k$ and return to M2. (If $k = 0$, we simply test the unary constraint on x_j.)

M5. [Loop on k.] If $k < l_t$, set $k \leftarrow k + 1$ and return to M4.

M6. [Finish forward step.] Set $M_{ja} \leftarrow j$, $a_j \leftarrow a$, $l_{t+1} \leftarrow l_t + 1$; step t is done.

M7. [Finish backward step.] Set $a_j \leftarrow 0$ and $l_{t+1} \leftarrow l_t - 1$; step t is done. ▌

(Here s_t is evaluated as in (b). Of course the values of l_t and s_t need not be stored in memory. This technique was introduced by J. Gaschnig for binary constraints [*Proc. Int. Joint Conference on Artificial Intelligence* **5** (1977), 457], and extended to $(k+1)$-ary constraints by R. Dechter in §5.2.2 of her book *Constraint Processing* (2003).)

400. In each CSA below, $I = \{q_0\}$ and Q is implicit.

(a) $\Omega = \{q_1, q_3\}$; $q_r \mapsto (* \leftarrow a^*?\ q_{(r+a) \bmod 5}\colon q_r)$.

(b) Clearly $n \le 2d$. Let $e_a = 0^{a-1}10^{d-a}$ be the unit d-vector with 1 in coordinate a. Let $\Omega = \{q(a,p)\}$, over all $a \in [0\,..\,d)$ and ternary d-vectors p with $p_0 + \cdots + p_{d-1} = n$. Use the transitions $q_0 \mapsto (* \leftarrow a^*?\ q(a, e_a)\colon q_0)$; $q(a,p) \mapsto R_{a,p}, (* \leftarrow b^*?\ q(b, p + e_a)\colon q(a,p))$, where $R_{a,p} = \emptyset$ if $p_a \ne 2$; otherwise $R_{a,p} = \{v_1 \setminus a, \ldots, v_n \setminus a\}$ excludes a from all unassigned domains.

(c) Same as (b), but with Ω restricted to the $\binom{d}{n/2}$ vectors p with no 1s.

(d) Let $\Omega = \{q(n,a,b) \mid 0 \le a < d,\ 0 \le b < 2\}$. Use transitions $q_0 \mapsto R, (v_1 \leftarrow 0?\ q(1,0,0)\colon q_0)$ and $q(j,a,b) \mapsto R(j,a,b), (v_j \leftarrow a'^*?\ (a' = a{+}1?\ q(j{+}1, a{+}1, 0)\colon q(j{+}1,$

Catalan numbers
unary constraint
Gaschnig
Dechter
unit d-vector

$a, 1))$: $q(j, a, b))$, where $R = \{v_j \setminus a \mid 1 \le j \le n,\ j \le a < d\}$; $R(j, a, 0) = \emptyset$; and $R(j, a, 1) = \{v_{j+1} \setminus a+2,\ v_{j+2} \setminus a+3,\ \ldots,\ v_n \setminus a+(n-j)+1\}$.

(e) Let $S_j = (-1)^{v_1} + \cdots + (-1)^{v_j}$. A necessary and sufficient condition is that $S_j \ge 0$ for $1 \le j < n$, and $S_n = 0$. One solution is therefore to enter state $q(j, S_j)$ after assigning v_1 through v_j, with $\Omega = \{q(n, 0)\}$: $q_0 \mapsto v_1 \setminus 1,\ v_n \setminus 0,\ (v_1 \leftarrow 0?\ q(1,1)\colon q_0)$; $q(j, s) \mapsto (v_{j+1} \leftarrow a^*?\ q(j+1, s+(-1)^a)\colon q(j, s))$, for $1 \le j < n$ and $0 \le s \le j$.

[These constructions can often be significantly improved by reducing the domains further. For example, if $d = 10$ and $n = 12$ in (c), $R_{0,1111120000}$ could exclude $\{6, 7, 8, 9\}$ from the domains of all five unassigned variables. In part (d) the underlying CSP might find it much better to assign variables in a different order; if then a is in the domain D_k of v_k, we must have $a - 1 \in D_1 \cup \cdots \cup D_{k-1}$. In part (e) we could assign values successively to v_1, v_n, v_2, v_{n-1}, and so on. We could even allow the CSP to assign variables in order of smallest domain; a partial assignment in which s variables have been assigned to 0 and t variables to 1 is then feasible if and only if we get a nested vector by assigning the leftmost $n/2 - s$ unassigned variables to 0 and the others to 1.]

401. Let Q be the set of all states $q(j, a)$ or $q'(j, a)$, for $j \ge 1$ and $0 \le a < d$, together with the special state $\perp$ that is reached when "symmetry is broken." Let $I = \{q(1, 0)\}$. Use the transitions $q(j, a) \mapsto (v_j \leftarrow a?\ q'(j, a)\colon q(j, a+1))$; $q'(j, a) \mapsto v_{n+1-j} \setminus a-1$, $(v_{n+1-j} \leftarrow a?\ q(j+1, 0)\colon \perp)$; $\perp \mapsto (* \leftarrow *?\ \perp\colon \perp)$. (The restriction '$v_{n+1-j} \setminus a-1$' is omitted when $a = 0$.) We can let $\Omega = Q$; but the actual final states are $q(n/2+1, 0)$ or $q'((n+1)/2, a)$ for palindromic solutions, $\perp$ for the others.

402. The following CSA uses Duval's algorithm (see answer 7.2.1.1–106) to produce only the solutions that are powers of a prime string: Let $I = \{q_0\}$, $Q = I \cup \{q(j, k) \mid 1 \le j < k \le n+1\} \cup \{\#\}$, and $\Omega = \{q(j, n+1) \mid j \text{ divides } n\}$. (State # is "dead.") Use the transitions $q_0 \mapsto (v_1 \leftarrow *?\ q(1, 2)\colon q_0)$ and

$$q(j, k) \mapsto (v_k \leftarrow a^*?\ (a < v_{k-j}?\ \#\colon a = v_{k-j}?\ q(j, k+1)\colon q(k, k+1))).$$

[This method is attractive, but additional pruning is often possible. For example, if D_k is the kth domain, we can remove from D_1 any element $> \max D_k$, for any $k > 1$.]

403. (a) Yes, but only in special cases. The middle row, when $i = \bar{\imath}$ (hence $i = (n-1)/2$) is special; that's the only time we can have $a_i = \bar{c}_i$. And we clearly have $a_i = \bar{a}_i$ if and only if $a_i = (n-1)/2$. Also $a_i = \bar{d}_i$ if and only if $R_i = C_i$; that can happen without attacking queens if and only if $R_i = i = C_i$. Similarly, $a_i = \bar{b}_i$ occurs if and only if $R_i = \bar{\imath}$ and $C_{\bar{\imath}} = i$. Cyclic symmetry dispenses with the other cases, like $b_i = \bar{c}_i$.

(b) For example, transposition $(i, j) \leftrightarrow (j, i)$ swaps $R_i \leftrightarrow C_i$; thus $(a_i, b_i, c_i, d_i) \leftrightarrow (\bar{d}_i, \bar{c}_i, \bar{b}_i, \bar{a}_i)$. In general, reflection complements the set $\{a_i, b_i, c_i, d_i\}$.

(c) Each tuple spawns seven others: $(b_{n'}, c_{n'}, d_{n'}, a_{n'}; \ldots; b_{n-1}, c_{n-1}, d_{n-1}, a_{n-1})$; $(c_{n'}, d_{n'}, a_{n'}, b_{n'}; \ldots; c_{n-1}, d_{n-1}, a_{n-1}, b_{n-1})$; $(d_{n'}, a_{n'}, b_{n'}, c_{n'}; \ldots;\ d_{n-1},\ a_{n-1},\ b_{n-1},\ c_{n-1})$; $(\bar{d}_{n'}, \bar{c}_{n'}, \bar{b}_{n'}, \bar{a}_{n'}; \ldots; \bar{d}_{n-1}, \bar{c}_{n-1}, \bar{b}_{n-1}, \bar{a}_{n-1})$; and so on. Thus the eight tuples for the first solution are (2, 7, 7, 2; 4, 4, 2, 5; 6, 0, 0, 1; 3, 3, 6, 6); (7, 7, 2, 2; 4, 2, 5, 4; 0, 0, 1, 6; 3, 6, 6, 3); (7, 2, 2, 7; 2, 5, 4, 4; 0, 1, 6, 0; 6, 6, 3, 3); (2, 2, 7, 7; 5, 4, 4, 2; 1, 6, 0, 0; 6, 3, 3, 6); (5, 0, 0, 5; 2, 5, 3, 3; 6, 7, 7, 1; 1, 1, 4, 4); (0, 0, 5, 5; 5, 3, 3, 2; 7, 7, 1, 6; 1, 4, 4, 1); (0, 5, 5, 0; 3, 3, 2, 5; 7, 1, 6, 7; 4, 4, 1, 1); (5, 5, 0, 0; 3, 2, 5, 3; 1, 6, 7, 7; 4, 1, 1, 4).

The second solution has central symmetry, so it has only four distinct tuples: (1, 7, 1, 7; 7, 1, 7, 1; 5, 4, 5, 4; 3, 2, 3, 2); (7, 1, 7, 1; 1, 7, 1, 7; 4, 5, 4, 5; 2, 3, 2, 3); (0, 6, 0, 6; 6, 0, 6, 0; 3, 2, 3, 2; 5, 4, 5, 4); (6, 0, 6, 0; 0, 6, 0, 6; 2, 3, 2, 3; 4, 5, 4, 5).

The other two solutions each have eight tuples, of which the lexicographically least turn out to be (0, 1, 8, 7; 6, 3, 5, 1; 1, 8, 1, 3; 5, 6, 4, 6; 2, 4, 0, 8) and (5, 5, 5, 5; 1, 7, 1, 7; 4, 1, 4, 2; 7, 10, 10, 10; 10, 6, 7, 6; 2, 2, 2, 1).

partial assignment
$\perp$
palindromes
Duval
dead
transposition
central symmetry

one-to-one function
Kotěšovec
leaper
Historical note
Sakinaga
recurrences

(d) Indeed, if $f(x)$ is *any* one-to-one function that maps every solution x of some combinatorial problem into a tuple, the x's for which $f(x)$ is lexicographically least, over all solutions equivalent to x by any definition of equivalence, are canonical.

(e) True: $\min(a_{n'}, \bar{a}_{n'}) < n'$; and we can't have $a_{n'} = b_{n'} = c_{n'} = d_{n'} = n' - 1$.

(f) In the following, 'ij?' is shorthand for '$R_i \leftarrow j$?' or '$C_j \leftarrow i$?' in exercise 400; it means that we either place a queen in cell (i, j) or forbid that cell. Similarly, 'not ij' is shorthand for '$R_i \neq j$, $C_j \neq i$'; this restriction is vacuous unless $0 \leq i, j < n$. States q_k arise when we potentially have 4-fold symmetry; states r_k arise when we potentially have 2-fold symmetry; and states s_k are intermediary. After symmetry has been broken we reach state $\perp$, which is the wildcard state '$\perp \mapsto (x^* \leftarrow a^*?\ \perp\colon \perp)$' as in answer 401.

$$
\begin{array}{ll}
q_1(i,j) \mapsto R(i,j), (ij?\ q_2(i,j)\colon q_1(i,j{+}1)); & r_1(i,j) \mapsto \text{not } \overline{\bar{\imath} j{-}1}, (ij?\ r_2(i,j)\colon r_1(i,j{+}1)); \\
q_2(i,j) \mapsto (j\bar{\imath}?\ q_3(i,j)\colon s_2(i,j)); & r_2(i,j) \mapsto (\bar{\imath}\bar{\jmath}?\ r_3(i,0)\colon \perp); \\
q_3(i,j) \mapsto (\bar{\imath}\bar{\jmath}?\ q_4(i,j)\colon s_4(i,j)); & r_3(i,j) \mapsto \text{not } \overline{j{-}1}i, (j\bar{\imath}?\ r_4(i,j)\colon r_3(i,j{+}1)); \\
q_4(i,j) \mapsto (\bar{\jmath}i?\ q_1(i{+}1,0)\colon \perp); & r_4(i,j) \mapsto (\bar{\jmath}i?\ r_1(i{+}1,0)\colon \perp);
\end{array}
$$

$s_2(i,j) \mapsto$ not $\bar{\jmath}i, (\bar{\imath}\bar{\jmath}?\ s_3(i,j{+}1)\colon \perp)$; $s_3(i,j) \mapsto$ not $\overline{j{-}1}i, (j\bar{\imath}?\ r_4(i,j)\colon s_3(i,j{+}1))$; $s_4(i,j) \mapsto$ not $\bar{\jmath}i, \perp$; $q_1(i,n) = r_1(i,n) = r_3(i,n) = s_3(i,n) = \perp$. Here $R(i,j)$ stands for the restrictions 'not $(j{-}1)\bar{\imath}$, not $\overline{\bar{\imath}j{-}1}$, not $\overline{j{-}1}i$', as well as four more when $i = \lceil n/2 \rceil$: 'not $\bar{\imath}j$, not ji, not $i\bar{\jmath}$, not $\bar{\jmath}\bar{\imath}$'. These rules suffice when $n = 2n'$ and $I = \{q_1(n', 0)\}$.

If $n = 2n' + 1$, let $I = \{s_1(0)\}$ and introduce $n' + 1$ new states $s_1(j)$, where we have $s_1(0) \mapsto$ not $n'j$ and not jn' for $n' < j < n$, $(n'0?\ \perp\colon s_1(1))$; $s_1(j) \mapsto$ not $\overline{j{-}1}n'$, $(n'j?\ \perp\colon s_1(j{+}1))$ for $0 < j < n'$; and $s_1(n') \mapsto (n'n'?\ q_1(n'{+}1, 0)\colon \perp)$.

The final state is $(q_1(n,0), r_1(n,0), \perp)$ for solutions with $(4,2,1)$-fold symmetry.

404. (14, 14, 14, 14; 16, 16, 16, 16; 31, 31, 31, 31; 29, 29, 29, 29; 26, 27, 27, 27; 24, 24, 6, 24; 3, 1, 1, 6; 27, 3, 3, 3; 10, 30, 10, 10; 22, 6, 25, 21; 5, 26, 30, 11; 11, 11, 11, 8; 23, 22, 23, 23; 12, 12, 12, 12; 7, 5, 22, 22; 13, 13, 13, 13). [Place twelve queens in extreme positions, then start the CSA of answer 403 in state $q_1(20, 26)$; only six canonical solutions remain. More than 32 queens could, of course, be treated similarly.]

405. With $\widehat{Q}(n) = 8\widehat{Q}_{\rm a}(n) + 4\widehat{Q}_{\rm s}(n) + 2\widehat{Q}_{\rm d}(n)$ solutions (see answer 7.2.2.1–24), we have

$n =$	10	11	12	13	14	15	16	17	18	19	20
$\widehat{Q}_{\rm a}(n) =$	0	5	18	231	642	4040	25320	166201	1115373	8060958	61981118
$\widehat{Q}_{\rm s}(n) =$	1	1	2	6	11	49	79	245	498	1192	3798
$\widehat{Q}_{\rm d}(n) =$	0	0	2	2	0	0	12	17	0	0	60
$\widehat{Q}(n) =$	4	44	156	1876	5180	32516	202900	1330622	8924976	64492432	495864256

[See §12.2 of V. Kotěšovec's book *Non-Attacking Chess Pieces* (online since 2011) for detailed information about pieces that combine a queen with a leaper.]

450.

2443	2177	5144	2141	2143	2144
2433	2772	5544	2442	2443	2644
1422 ;	1372 ;	1552 ;	1412 ;	1413 ;	1662 .
3331	3377	3332	3331	3331	6662
(1 of 3)	(1 of 32)	(1 of 3)	(1 of 1)	(1 of 12)	(1 of 24)

[*Historical note:* Fillomino was invented by Waku Sakinaga; see *Puzzle Communication Nikoli* **47** (February 1994).]

451. If $t_n = t_n(1) + t_n(2) + \cdots$ is the desired number, where $t_n(m)$ is the number of patterns with m in the upper right corner, we have the recurrences $t_n(m) = a_n(m,m) + \sum_{m'=1}^{2n-m} b_n(m,m;m',m')$; $a_n(l,m) = \bigl(l < 2?\ 0\colon l > 2n?\ 0\colon l = 2n?\ 1\colon l = 2?\ t_{n-1} - 2t_{n-1}(m) + a_{n-1}(m,m)\colon a_{n-1}(l-2,m) + 2\sum_{m'=1}^{2n-l} b_{n-1}(l-2,m;m',m')\bigr)$;

$b_n(l,m;l',m') = (m = m'?\ 0{:}\ l < 1?\ 0{:}\ l' < 1?\ 0{:}\ l+l' > 2n?\ 0{:}\ l+l' = 2n?\ 1{:}\ l = l'?\ t_{n-l} - t_{n-l}(m) - t_{n-l}(m') + b_{n-l}(m,m;m',m'){:}\ l < l'?\ a_{n-l}(l'-l,m') + \sum_{m''=1}^{2n-l-l'} b_{n-l}(m'',m'';l'-l,m')[m'' \neq m]{:}\ b_n(l',m';l,m))$. Here $a_n(l,m)$ is the number of length n prefixes of $2\times\infty$ fillomino patterns that end with two m's at the right, where those m's are part of an l-omino; $b_n(l,m;l',m')$ is similar, but ending with $m \neq m'$ at the right, respectively parts of an l-omino and an l'-omino. Hence $(t_1, t_2, \ldots, t_{145}) = (1$, 5, 33, 138, 715, 3524, ..., 51376 52565 68766 30928 69800 54061 86098 15559 89493 34784 20112 85272 12992 22603 93822 34860 83493 24519 70607 50508). (The ratio t_{n+1}/t_n converges rapidly to 4.91867 12250 37424 13083 06703 91572 28440 ....)

452. Suppose the given shape S has N cells. There are N primary items ij, one for each cell. (If S is an $m \times n$ grid, for instance, we have $N = mn$ and $0 \le i < m$, $0 \le j < n$.) A *potential d-omino* is a set $P \subseteq S$ of rookwise connected cells for which every $ij \in P$ is either blank or labeled d, but not adjacent to any cell $\notin P$ that's labeled d. (All such P can be found by using an interesting variant of the algorithm in exercise 7.2.2–75; see, for example, the author's online program FILLOMINO-DLX.) There are secondary items e_d, one for each edge between two unlabeled cells and each possible d. And there's one option for each potential d-omino P, containing (i) all $ij \in P$ and (ii) all e_d for which e is an edge between a blank cell $\in P$ and a blank cell $\notin P$.

For example, the puzzle of exercise 450 has $m = n = 4$, $N = 16$, and exactly (8, 5, 11, 11) potential (1, 2, 3, 4)-ominoes, hence 35 options. Two of the potential tetromino options are '02 03 12 13 h_{224} v_{124}' and '11 21 22 32 h_{224} v_{124} v_{334}', where h_{ij} and v_{ij} denote the horizontal and vertical edges that connect $(i-1)j$ and $i(j-1)$ with ij.

How large can d be? Suppose c_d of the given cells are labeled d, for a total of $C = c_1 + c_2 + \cdots + c_s$ "clues," where s is the maximum label. Then every potential d-omino has $d \le \max(N-C, s)$. And there's a sharper bound $\max(N-C^+, s)$, where $C^+ = \sum_{d=1}^{s} c_d^+$ and c_d^+ is a lower bound on the number of d labels that are known to exist. For example, we may take $c_1^+ = c_1$; $c_2^+ = 2(c_2 -$ the number of pairs of adjacent 2s); and for $d \ge 3$, $c_d^+ = d\lceil c_d/d\rceil + [c_d \bmod d = 0$ and the ds aren't disjoint d-ominoes$]$.

453.

(a)
```
221221
1ee312
e1e332
eeeeee
233e1e
213ee1
122122
```
; (b)
```
335555666
314159266
444999236
555899932
535897932
338877773
888772643
338462643
344466644
```
; (c)
```
6244221244
6244888244
6688866688
2684446888
2684226882
6448446482
6248441448
6248862248
6688666688
6882268888
```
; (d)
```
1334141224
2344i44434
2143i22334
1223i33i14
21i3i3ii22
2iiiiii444
34444i3324
33221i3421
1444ii4433
3334224223
```
; (e)
```
2244446633
5153366313
5553616345
4444646345
5566643145
3565543345
3552542215
3442513334
2344544414
2332242244
```
.

[Puzzle (c), by Chris Green, was posted at `puzzleparade.blogspot.com` (19 July 2013), #60; puzzle (d), which totally defeats the construction in answer 452 because it requires a humongous number of options, was posted at `gmpuzzles.com` by Tapio Saarinen (7 October 2014); puzzle (e) was posted on Twitter by `@kobouzu17` (31 December 2022).

454. We save a factor of roughly 8 by removing symmetry. Each potential puzzle with k clues leads to k potential puzzles with one fewer clue, until we reach invalid cases that are matched by more than one of the given 59951. Potential puzzles without redundancies must still be screened to ensure that they can't be solved with labels greater than 5.

All told we obtain 938484 nonisomorphic minimal 4×4 puzzles whose clues don't exceed 5, of which (937236, 1240, 8, 0) have (1, 2, 4, 8)-fold symmetry. Exactly (1124, 56253, 374643, 377611, 104436, 20410, 3520, 430, 57) of them have (4, 5, ..., 12) clues.

author
downloadable programs
Green
Saarinen
`@kobouzu17`
symmetry

4␣␣␣	4␣␣␣	4␣␣␣	4␣␣␣	␣␣␣1	␣␣␣3	␣␣5␣	1␣33	␣1␣1	␣3␣2	␣␣2␣	␣␣41
␣␣␣4	␣␣␣4	␣␣␣4	␣␣␣4	␣15␣	␣␣3␣	␣5␣5	55␣␣	2␣4␣	4␣4␣	3␣␣1	2␣␣␣
1␣␣␣	1␣␣␣	2␣␣␣	2␣␣␣	␣51␣	␣3␣␣	5␣5␣	␣544	␣5␣4	␣2␣4	␣␣␣4	␣3␣␣
␣␣␣3	␣␣␣2	␣␣␣2	␣␣␣5	1␣␣␣	3␣3␣	5␣5␣	5544	5␣5␣	4␣4␣	␣2␣␣	3␣␣2
(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)	(x)	(xi)	(xii)

Fig. A–16. A gallery of interesting 4×4 fillomino puzzles.

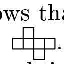
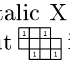
Puzzles (i)–(iv) in Fig. A–16, which have just 4 clues each, make a nice sequence by which we can introduce newbies to the wonders of fillomino. One of the cutest examples with 4-fold symmetry is puzzle (v). And (vi) and (vii) are among the 15 with "pure" clues (all the same). Puzzle (viii) is interesting not because it's hard to solve, but because all twelve of its clues are necessary. Similarly, none of the eight clues in (ix) and (x), which appear in the cells of odd parity like a checkerboard, are redundant. The most difficult 4×4 fillomino puzzles, rated by the search tree size (16) when full domain consistency is maintained, are probably those in (xi) and (xii). (See Appendix E.)

455. (Solution by N. Beluhov.) Let P be a maximal rookwise connected subset of the solution, having no labels $\leq s$. Every element of P must have the same label $d = |P|$, because the solution is unique. Let T be a spanning tree of P. Every edge u — v of T partitions T, hence P, into two polyominoes $P_u \cup P_v$ when that edge is removed from T, where $u \in P_u$ and $v \in P_v$. By uniqueness, we cannot have $s < |P_u| < |P_v|$.

Case 1, T has one centroid, v. When T is rooted at v, let v's children be u_1, …, u_k, with corresponding subtree sizes $s_1 \geq \cdots \geq s_k$. Then $s_1 \leq s_2 + \cdots + s_k$ by Eq. 2.3.4.4–(7). Hence we have $|P_v| = s_2 + \cdots + s_k + 1 > |P_{u_1}| = s_1$ in the decomposition $P = P_{u_1} \cup P_v$. It follows that $s_1 \leq s$; and $d \leq ks + 1 \leq 4s + 1$.

Case 2, T has two centroids, u — v. We may suppose that $u = i(j-1)$ and $v = ij$, coordinatewise. If T also contains both of the edges $u' = (i-1)(j-1)$ — u and $v' = (i-1)j$ — v (or, similarly, if T contains both $u' = (i+1)(j-1)$ — u and $v' = (i+1)j$ — v), we get a decomposition $P = P_{u'} \cup P_{v'} \cup P_{uv}$ by deleting those edges, where $|P_{u'}| \leq s$ and $|P_{v'}| \leq s$. On the other hand, if in the original tree T we delete only edge u — v and replace it with edge u' — v', we get a new tree for which u' and v' are the two centroids, as well as a new polyomino $P_{u'v'} = P_{u'} \cup P_{v'}$. By symmetry between u — v and u' — v', therefore, $d = |P_{uv}| + |P_{u'v'}| \leq 2|P_{u'v'}| \leq 2|P_{u'}| + 2|P_{v'}| \leq 4s$.

Finally, if T doesn't contain such u' and v', we can regard u and v as co-roots of T; and their subtrees (at most four total) must each have size $\leq s$. Hence $d \leq 4s + 2$.

This proof shows that we can obtain $d = 4s + 2$ for $s = 1$ only when the P is the "italic X hexomino" . That's impossible if the overall shape is a rectangular grid; but  is a valid puzzle in a grid minus two corners, and $d = 5$ is possible in a 3×3 grid.

Here's $s = 2$:

```
␣␣␣1␣1␣␣
␣1␣2␣2␣␣
1␣␣␣␣␣␣1 ;
␣␣2␣2␣1␣
␣␣1␣1␣␣␣
```

for $s = 3$, see exercise 453(a); and here's $s \geq 4$, shown for $s = 5$:

```
14␣␣␣1␣5␣␣34
43212␣␣54␣2␣
␣␣␣3␣␣␣5␣31␣
␣␣4␣␣␣␣5␣␣2␣
␣55555␣5␣␣␣1
␣␣␣␣␣␣␣␣␣␣␣␣ .
1␣␣␣5␣55555␣
␣2␣␣5␣␣␣␣4␣␣
␣13␣5␣␣␣3␣␣␣
␣2␣45␣␣21234
43␣␣5␣1␣␣␣41
```

456. (Solution by N. Beluhov and P. Mebane.) When $m = 1$ they are ␣ and

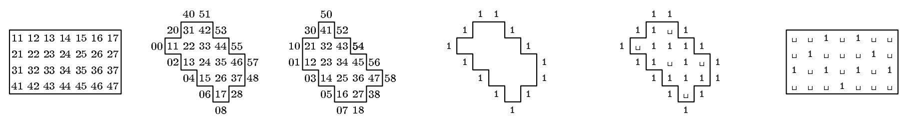
$$\mathtt{1}^a \sqcup^{b_1} \mathtt{1} \sqcup^{b_2} \mathtt{1} \ldots \mathtt{1} \sqcup^{b_r} \mathtt{1}^c \quad \text{for } r \geq 1,\ 0 \leq a, c \leq 1,\ 2 \leq b_j \leq 4,\ (1-a)b_1 \leq 2,\ (1-c)b_r \leq 2.$$

parity
checkerboard
domain consistency
Beluhov
spanning tree
centroid of a free tree
hexomino
Beluhov
Mebane

When $m = 2$ they're $\alpha\genfrac{}{}{0pt}{}{\sqcup}{\sqcup}\beta_1\genfrac{}{}{0pt}{}{\sqcup}{\sqcup}\ldots\genfrac{}{}{0pt}{}{\sqcup}{\sqcup}\beta_r\genfrac{}{}{0pt}{}{\sqcup}{\sqcup}\gamma$ for $r \geq 0$, $\alpha, \gamma \in \{\epsilon, \genfrac{}{}{0pt}{}{1}{\sqcup}, \genfrac{}{}{0pt}{}{\sqcup}{1}\}$, $\beta_j \in \{\genfrac{}{}{0pt}{}{1\sqcup}{\sqcup 1}, \genfrac{}{}{0pt}{}{\sqcup 1}{1\sqcup}\}$.

Now suppose $3 \leq m \leq n$. By answer 455 with $s = 1$, this problem is equivalent to tiling an $m \times n$ rectangle with monominoes and X pentominoes, together with tee tetrominoes at the edges and bent trominoes at the corners.

Notice that the 1s all occur in cells ij with the same parity $(i+j) \bmod 2$. Therefore we can transform the $m \times n$ grid into two "Aztec rectangles," rotating 45° and mapping $ij \mapsto (m - i + j + \delta, i + j - \delta)/2$ for $1 \leq i \leq m$, $1 \leq j \leq n$, and $\delta \in \{0, 1\}$. If we put a border around the Aztec rectangles by appending the $m + n + 2$ images of ij for $i = 0$ or $i = m{+}1$ or $j = 0$ or $j = n{+}1$, and if we fill those border cells with 1s, we get an equivalent problem that's easy to visualize: (i) There must be no 2×2 array of 1s; (ii) no two ␣s can be kingwise adjacent. For example, if $m = 4$, $n = 7$, and $\delta = 0$ we have



matrix 4×7 | even Aztec 4×7 | odd Aztec 4×7 | even Aztec 4×7 problem | even Aztec 4×7 solution | even matrix 4×7 solution

The two Aztec rectangles are isomorphic when mn is even. The solutions when m is even and $\delta = 0$ are the first n columns of A_m when $n \bmod (m{+}1) \in \{1, m\}$, and of B_m when $n \bmod (m{+}1) \in \{m{-}2, m\}$, where A_m and B_m are infinite matrices

A_6 =
```
␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣
␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1
1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣
␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣
␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣
␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1
```
…, B_6 =
```
1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣
␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1
␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣
␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1
1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣
␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ ␣
```
…

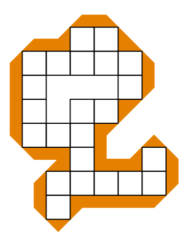
whose columns have period length $2m + 2$, illustrated here for $m = 6$.

There are no solutions for odd $m \leq n$ unless n is odd and $\delta = 0$. Two of the solutions are easy: The even-numbered rows are ␣1␣1␣1␣1␣1…, and the odd-numbered rows alternate between 1␣␣␣1␣␣␣1␣␣␣1… and ␣␣1␣␣␣1␣␣␣1␣␣….

Finally, $(n - 1)/2$ additional solutions arise when $m = n$ is odd, such as these:

```
1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1    ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣    1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1    ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣    1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1
␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣    ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣    ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣
1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣    1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1    ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣    1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1    ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1
␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣    ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣    ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣
␣ ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1    1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣    1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1    ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1    1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ ␣
␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣
1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ ␣    ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1    1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ 1    1 ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣    ␣ ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1
␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣
␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1    1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1    ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣    1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1    1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣
␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣    ␣ 1 ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣    ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣    ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣    ␣ ␣ ␣ 1 ␣ 1 ␣ 1 ␣ 1 ␣
1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1    ␣ ␣ 1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣    1 ␣ ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1    ␣ ␣ 1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣    1 ␣ 1 ␣ ␣ ␣ 1 ␣ ␣ ␣ 1
```

457. Given a polyomino P with $d = d(P)$ cells, consider the path that goes through the centers of all cells that lie just outside of P's perimeter (its outer boundary). The region between this path and the perimeter is called P's "halo," and we shall call its area $h = h(P)$. The formula $h = (p - v)/2$ is not difficult to prove, where p is the perimeter's length and v is the number of maximal vertical line segments that it has. (A 23-omino and its halo are pictured.) Furthermore, N. Beluhov and S. Gerdjikov [*Középiskolai Matematikai és Fizikai Lapok [KöMaL]* (3) **70**, 7 (October 2020), 419, problem A. 783] have shown that $h \leq t/2$ implies $d \leq b_t = \lfloor (t^2 + 4)/8 \rfloor$. This bound is in fact sharp for all $t \geq 4$, with the maximum d attained by a $\lfloor t/2 \rfloor \times \lceil t/2 \rceil$ Aztec rectangle with $\delta = 0$ (see exercise 456).



$h = \frac{36-11}{2}$

monominoes
X pentominoes
pentominoes
tee tetrominoes
tetrominoes
bent trominoes
trominoes
parity
Aztec rectangles
rotating 45°
kingwise adjacent
halo
Beluhov
Gerdjikov
Aztec rectangle

Toroidal
Friedman
Aztec rectangles

Polyominoes without common edges have disjoint halos. Therefore if an $n \times n$ fillomino pattern Φ_n contains k d-ominoes, we must have $k(d+h_d) \le n^2 + O(n)$, where $h_d = \lceil \sqrt{8d-4}\,\rceil/2$ is the smallest possible halo area of a d-omino. Consequently $\#_d(\Phi_n)/n^2 \le d/(d{+}h_d){+}O(1/n)$, and we have $\delta_d \le d/(d{+}h_d)$. These upper bounds are:

$d =$	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
$\frac{d}{d+h_d} =$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{6}{11}$	$\frac{4}{7}$	$\frac{5}{8}$	$\frac{12}{19}$	$\frac{7}{11}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{20}{29}$	$\frac{11}{16}$	$\frac{12}{17}$	$\frac{13}{18}$	$\frac{28}{39}$	$\frac{30}{41}$	$\frac{8}{11}$	$\frac{17}{23}$	$\frac{3}{4}$	$\frac{38}{51}$	$\frac{40}{53}$	$\frac{42}{55}$	$\frac{22}{29}$	$\frac{23}{30}$	$\frac{24}{31}$	$\frac{25}{32}$

Toroidal constructions (with appropriate offsets) give lower bounds for small d:

(d=1)	(d=2)	(d=3)	(d=4)	(d=9)	(d=11)	(d=16)

`15`
`555`
`151` ;
	`122`
	`244`
	`244` ;
		`13344`
		`313443`
		`331332`
		`443132`
		`443313`
		`332233` ;
			`144`
			`4144`
			`4413`
			`1433` ;
				`199`
				`9999922`
				`2299` ;
					`bbb1`
					`bbbbb333`
					`bbb1` ;
						`ggg1`
						`ggggg5`
						`ggggg5555`
						`ggg1` .

(See `https://erich-friedman.github.io/mathmagic/0316.html` for generalizations.) Since Aztec rectangles and their halos can be packed efficiently, we also have the lower bounds $\delta_d \ge d/(b_{2k-1}+k)$ when $1 < b_{2k-2} < d \le b_{2k-1}$; $\delta_d \ge d/(b_{2k}+k)$ when $b_{2k-1} < d \le b_{2k}$; they're optimum when $d = b_{2k}$. So we have the following partial results:

$d =$	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
$\delta_d \ge$	$\frac{3}{8}$	$\frac{4}{9}$	$\frac{17}{35}$	$\frac{8}{15}$	$\frac{5}{8}$	$\frac{3}{5}$	$\frac{7}{12}$	$\frac{2}{3}$	$\frac{9}{14}$	$\frac{2}{3}$	$\frac{11}{16}$	$\frac{2}{3}$	$\frac{13}{18}$	$\frac{2}{3}$	$\frac{5}{7}$	$\frac{16}{23}$	$\frac{17}{24}$	$\frac{3}{4}$	$\frac{19}{28}$	$\frac{5}{7}$	$\frac{3}{4}$	$\frac{11}{16}$	$\frac{23}{32}$	$\frac{3}{4}$	$\frac{25}{32}$

Are these the true values of δ_d, or can some of them be increased?

500. True. It will set $R_{i'j} \leftarrow O$ when $k = i$, and $R_{ij'} \leftarrow O$ when $k = j$; then $R_{i'j'} \leftarrow O$.

501. (a) Let $X = \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$. When $k=1$, $R_{52} \leftarrow I$. Then when $k=2$, $R_{13} \leftarrow I$, $R_{53} \leftarrow X$. Then when $k=3$, we set $R_{14} \leftarrow X$, $R_{24} \leftarrow I$, $R_{54} \leftarrow I$. Then when $k=4$, $R_{15} \leftarrow I$, $R_{25} \leftarrow X$, $R_{35} \leftarrow I$, $R_{55} \leftarrow O$. Soon all are O, by exercise 400.

(b) These relations say that $x_{j+1} = (x_j + 1) \bmod 3$ for $1 \le j < 5$; but x_5x_1 can be not only 01, 12, or 20, but also 02. However, it's peculiar because (for example) $R_{21} = \left(\begin{smallmatrix}1&1&1\\1&1&1\\1&1&1\end{smallmatrix}\right)$ puts *no* constraint on x_1x_2! The constraints begin to propagate as in (a): When $k=1$, $R_{52} \leftarrow \left(\begin{smallmatrix}1&0&1\\1&0&0\\0&1&0\end{smallmatrix}\right)$; then when $k=2$, $R_{13} \leftarrow \left(\begin{smallmatrix}0&0&1\\1&0&0\\0&1&0\end{smallmatrix}\right)$, $R_{53} \leftarrow \left(\begin{smallmatrix}1&1&0\\0&1&0\\0&0&1\end{smallmatrix}\right)$; etc.; $R_{55} = \left(\begin{smallmatrix}1&0&0\\0&0&0\\0&0&0\end{smallmatrix}\right)$ after $k=4$, meaning that x_5 must be 0. A lot happens when $k=5$, basically forcing $x_1x_2x_3x_4x_5 = 20120$ and allowing only one possibility for x_ix_j when $i \le j$.

The first round ends with $R_{21} = \left(\begin{smallmatrix}0&1&1\\0&0&0\\0&0&0\end{smallmatrix}\right)$, $R_{31} = \left(\begin{smallmatrix}0&0&0\\0&1&1\\0&0&0\end{smallmatrix}\right)$, etc. But round 2 "purifies" each R_{ij} with $i > j$ so that only one 1 remains, thus achieving a perfectly stable state.

(c) Nothing changes. In fact, there will be no change in any scenario where $R_{kk} = I$ for all k and $R_{ik}R_{kj}$ is all 1s whenever $i \ne k \ne j$.

(d) When $k=2$, $R_{13} \leftarrow \left(\begin{smallmatrix}1&0&0\\0&0&1\end{smallmatrix}\right)$ and $R_{33} \leftarrow \left(\begin{smallmatrix}1&0&0\\0&0&0\\0&0&1\end{smallmatrix}\right)$. Then $R_{31} \leftarrow \left(\begin{smallmatrix}1&1\\0&0\\1&1\end{smallmatrix}\right)$, $R_{32} \leftarrow \left(\begin{smallmatrix}1&1\\0&0\\1&1\end{smallmatrix}\right)$ when $k=3$. No further changes occur; hence the final state is curiously asymmetric, with $R_{13} \ne R_{31}^T$, $R_{23} \ne R_{32}^T$, and R_{21} still equal to $\left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right)$. (Don't ask what that means.)

502. It's true for $r=1$; so assume that $r > 1$. Let $k = \max\{k_1, \ldots, k_{r-1}\}$, and let (p,q) be minimum and maximum with $k_p = k_q = k$. (Possibly $p = q$.) By (200) there's a value x such that $(s,x) \in R_{ik}$, $(x,x) \in R_{kk}$, $(x,t) \in R_{kj}$. Hence by induction on r we can find suitable $x_0 \ldots x_p$ with $x_0 = s$ and $x_p = x$, suitable $x_p \ldots x_q$ with $x_p = x_q = x$, and suitable $x_q \ldots x_r$ with $x_q = x$ and $x_r = t$.

999. ...

ANSWERS TO PUZZLES IN THE ANSWERS

4144	4424	4441	4144	3122	4441	
4414	4424	4224	4454	3441	2433	
1434	1344	2444	2454	3144	2313	(see answer 454)
2233	3322	2122	2555	1221	3322	
(i)	(ii)	(iii)	(iv)	(xi)	(xii)	

No intuitive answers, please.
— LIFE INTERNATIONAL (17 December 1962)

INDEX AND GLOSSARY

Index-making has been held to be the driest
as well as lowest species of writing.
We shall not dispute the humbleness of it;
but the task need not be so very dry.
— LEIGH HUNT, in *The Indicator* (1819)

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.