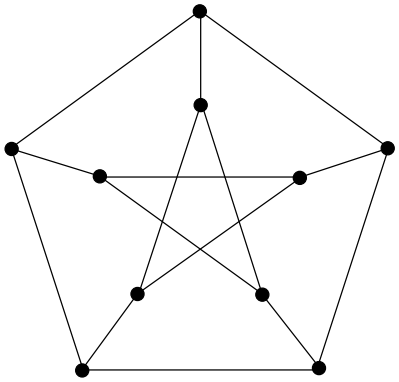
Problems

2008-01-18
07:37:15

Graphs: basic concepts

* 1. Find the adjacency matrix and the incidence matrix of the Petersen graph shown below.



** 2. The Petersen graph is a so called $(3,5)$-cage. In general a (k,n)-cage is the graph with the minimum number of vertices such that every vertex is of degree k and that the length of the shortest cycle in the graph is n. (a k-regular graph of girth n)

Show that a $(k,5)$-cage must have at least k^2+1 vertices.

Lexicographical order of k-subsets

* 3. (a) Find the 100th 3-subset, in lexicographical order, of $\{1,2,\ldots,10\}$.

(b) Find the lexicographical rank of $\{3,5,7\}$ among the 3-subsets of $\{1,2,\ldots,10\}$.

** 4. (a) Let $S=\{2,3,5,7,11,13\}$. What is the rank of $\{3,7,13\}$ among the 3-subsets of S?

(b) Show that $\mathsf{unrank}_{\text{lex}}(\mathsf{rank}_{\text{lex}}(\{3,7,13\}))=\{3,7,13\}$.

Minimum change order for subsets

* 5. (a) Find the rank of the codeword 00101011 in the binary reflected Gray code G^8.

(b) Find the 50th codeword in the binary reflected Gray code G^8.

* 6. Let $d \geq 2$ be an integer and $\mathcal{V} := \{0,1\}^d$, that is, $\mathcal{V}$ consists of $0-1$ strings of length d. Define the set of undirected edges $\mathcal{E}$ so that there is an edge between vertices $x, y \in \mathcal{V}$ if x and y differ at exactly one position. Considering the graph $(\mathcal{V}, \mathcal{E})$, find

(a) the degree of the vertices,

(b) the length of the shortest cycle (the girth of the graph), and

(c) the length of the longest cycle (the circumference of the graph).

*** 7. Let $1 \leq k \leq n$. Remove from the binary reflected Gray code G^n all codewords whose Hamming weight is not k. Show that the remaining codewords form a minimum change order of the k-subsets of an n-element set.

Finding a rank and unrank function

** 8. Find the lexicographical rank- and unrank-functions for car license plates of the form

$$\mathrm{X_1X_2X_3\text{-}Y_1Y_2Y_3}$$

where $\mathrm{X_1, X_2, X_3} \in \{\mathrm{A, B, C, D, \ldots, Z}\}$ and $\mathrm{Y_1, Y_2, Y_3} \in \{0, 1, 2, \ldots, 9\}$. What is the rank of the license plate IOI-010?

Cycle and list presentations of permutations

* 9. (a) Represent the permutation $\pi_1 = [2, 4, 6, 7, 5, 3, 1]$ as a product of disjoint cycles.

(b) Represent the permutation $\pi_2 = (1, 5, 6)(2, 4, 3)(7)$ as a list.

(c) Determine the inverse permutations π_1^{-1} ja π_2^{-1}.

(d) Find the products $\pi_1\pi_2$ and $\pi_2\pi_1$.

* 10. Suppose that $p, q, r_1, \ldots, r_k, s_1, \ldots, s_l$, where $k, l \geq 0$, are distinct elements. Simplify the following permutations, given as products of cycles, to products of disjoint cycles:

(a) $(p, q)(p, r_1, \ldots, r_k, q, s_1, \ldots, s_l)$

(b) $(p, q)(p, r_1, \ldots, r_k)(q, s_1, \ldots, s_l)$

** 11. A permutation π is *even*, if it can be expressed as the product of an even number of transpositions. Similarly π is *odd* if it can be expressed as the product of an odd number of transpositions.

Show that every permutation is either even or odd.

* 12. (a) Let $r \geq 1$. Present the r-sykli $(1, 2, \ldots, r)$ as a product of transpositions.

(b) Based on the previous, represent the permutation $[2, 4, 6, 7, 5, 3, 1]$ as a product of transpositions. Is the permutation even or odd?

Rank and successor of permutations

* 13. Find the rank and successor of $[2, 4, 6, 7, 5, 3, 1]$ both in lexicographical and Trotter–Johnson order.

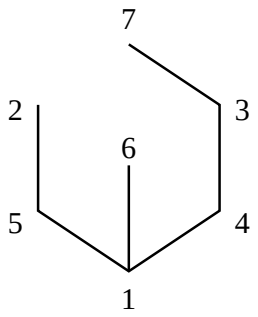
Integer partitions

* 14. Find the rsf-lex rank and successor of $1 + 3 + 4 + 6 + 6 + 8$.

** 15. How many ways of partitioning 20 people into 5 groups are there, if no two groups are allowed to be of the same size? A group may also be empty.

The Prüfer correspondence

* 16. (a) Find the Prüfer list presentation of the labeled tree below.



(b) Which labeled tree has the Prüfer list presentation $[5, 5, 4, 3, 2]$?

Catalan families

*** 17. The four-term product $abcd$ can be grouped into products of two terms in five different ways:

$$((ab)c)d, \quad (a(bc))d, \quad a((bc)d), \quad a(b(cd)), \quad (ab)(cd).$$

In general there are $n+1$ terms. Find a bijection between the groupings of terms and the Catalan family C_n. (Hint: every grouping can be presented as a binary tree, whose leaves correspond to the terms in the product.)

Backtrack search

* 18. Describe backtracking algorithms for the following problems:

(a) Find all ways of placing n queens on an $n \times n$ chess board so that no two queens threaten each other.

(b) Find the k-colorings of the graph $\mathcal{G}$.

(c) A (*self-avoiding walk*) starts at the origin and takes steps of length 1. Each step goes up, down, left, or right, and no point in the plane is visited twice. Find all self-avoiding walks of length n.

(d) A Steiner triple system $\mathrm{STS}(n)$ is the pair $(\mathcal{P}, \mathcal{B})$, where $\mathcal{P}$ is an n-element set and $\mathcal{B}$ is a set of $n(n-1)/6$ 3-subsets of $\mathcal{P}$, where each pair of points occurs in exactly one triplet. A Steiner triple system can only exist if $n \equiv 1 \mod 6$ or $n \equiv 3 \mod 6$. Find all Steiner triple systems $\mathrm{STS}(n)$.

*** (e) Given an $n \times n$ position in a minesweeper game, find the squares that

i. cannot contain a mine, and

ii. those that must contain a mine.

(The number of mines is assumed to be unknown.)

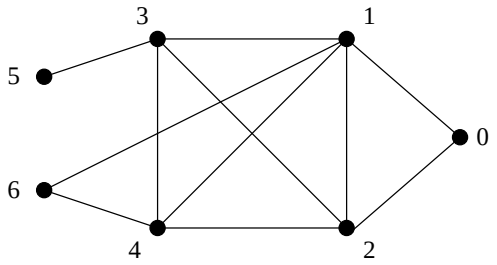
** 19. A Latin square of order n is an $n \times n$-array, whose every row and column contains each of the numbers $1, \ldots, n$ exactly once. A Latin square is said to be in reduced form, if the elements in the first row and column appear in the order $1, 2, \ldots, n$. Write a program that uses backtrack search to compute the number of reduced Latin squares of order n.

Improving performance of backtrack search

*** 20. Continuation of * 18. (e). Find methods of making the backtrack search to find consistent sets in minesweeper more efficient.

Maximum Clique

* 21. Find all maximal cliques of the graph below.



** 22. How could you use maximum clique problem in solving the n queens problem defined in * 18. (a)?

Exact Cover

*** 23. How could you use the exact cover problem to represent the problem of k-coloring of the graph $\mathcal{G}$, when $k = 2$? For $k > 2$ the problem is NP-complete, but for $k = 2$ it reduces into a polynomial problem. Can this be seen from the representation?

Graph coloring

** 24. The chromatic number of a graph $\mathcal{G}$ is defined by

$$\chi(\mathcal{G}) = \min\{k \mid \mathcal{G} \text{ has a vertex } k\text{-coloring}\}$$

and the clique number by

$$\omega(\mathcal{G}) = \max\{k \mid \mathcal{G} \text{ contains a } k\text{-vertex clique}\}.$$

According to Lemma 4.4 $\omega(\mathcal{G}) \leq \chi(\mathcal{G})$. Show that the inequality can be proper:find a graph for which $\omega(\mathcal{G}) < \chi(\mathcal{G})$.

** 25. Let $\mathcal{G}$ be a graph whose vertices are ordered. A greedy graph coloring algorithm colors the vertices in order, one at a time, so that each vertex in turn is colored with the least color (suppose that the colors are ordered) that does not appear among its already colored neighbors.

(a) Find an infinite family of graphs whose vertices are ordered so that the greedy algorithm does not produce the optimal coloring. (A coloring is optimal if the number of colors equals $\chi(\mathcal{G})$.)

(b) Does every graph have such an ordering of the vertices that the greedy algorithm produces an optimal coloring?

Cost function, neighborhood

* 26. Find a sensible cost function and neighborhood for the following hard combinatorial optimisation problems.

(a) The symmetric traveling salesman problem.

(b) Coloring a graph with n colors so that neighboring vertices are of different color.

(c) Graph partitioning: Partition the vertices of a graph (of which there are an even number) into two parts of equal size so that the number of edges between the parts is as small as possible.

(d) Number partitioning: Partition the set $A = \{a_1, a_2, \ldots, a_n\}$ into two parts A_1 and A_2 so that $A_1 \cup A_2 = A$, $A_1 \cap A_2 = \emptyset$ and

$$\sum_{a \in A_1} a = \sum_{a \in A_2} a.$$

** 27. Find a graph and a valid partitioning for its vertices $[X_1, X_2]$ ($|X_1| = |X_2|$), which is a local optimum (when the neighborhood is swapping one vertex from X_1 to X_2 and vice versa) but which is not a global optimum (there is a better feasible solution obtainable by swapping two vertices between X_1 and X_2).

Simulated annealing

** 28. Give an algorithm based on simulated annealing for the maximum clique problem.

Configuration graphs

Let us investigate the graphs defined by the neighborhoods in local search. The vertices of the configuration graphs are all elements of the search space $\mathcal{X}$, and two vertices are connected by an edge if and only if they belong to the neighborhood of each other. (Assume that the neighborhood $\mathsf{N} : \mathcal{X} \to 2^{\mathcal{X}}$ is symmetric, that is, that for all $x, y \in \mathcal{X}$ it holds that $x \in \mathsf{N}(y)$ if and only if $y \in \mathsf{N}(x)$.)

** 29. Give a sensible configuration graph for the following problems:

(a) Exact cover: from a collection of subsets of a given base set, find such a collection of subsets that every element of the base set appears in exactly one of them.

(b) Graph coloring with at most k colors.

(c) Traveling salesman problem (with the transposition neighborhood).

(d) Graph partitioning: Given a graph, partition its vertices (of which there are an even number) into two parts of equal size such that the number of edges between the parts is as small/large as possible.

(e) Find a k-clique in a given graph.

** 30. In the context of local search, some interesting properties of the configuration graph include

(a) the number of vertices (order of the graph)

(b) number of neighbors of vertices (vertex degree)

(c) whether the graph is connected, that is, whether from every node there is a path to every other node

(d) the length of the shortest cycle (girth)

(e) the length of the longest cycle (circumference)

(f) the maximum length of a shortest path between two vertices (the diameter of the graph)

Determine these properties for the configuration graphs in Problem ** 29..

Tabu search

** 31. Give some suitable tabu conditions for use with the following optimisation problems introduced in Problem * 26. (use the same neighborhood and objective function as in Problem * 26.).

(a) The symmetric traveling salesman problem.

(b) Coloring a graph with k colors so that neighboring vertices are always of different colors.

(c) Graph partitioning: Given a graph, partition the vertices (of which there are an even number) into two parts of the same size so that the number of edges between the two parts is as small as possible.

(d) Number partitioning: partition the set $A = \{a_{1,}a_{2,}\ldots,a_n\}$ into two parts A_1 ja A_2 such that $A_1 \cup A_2 = A$, $A_1 \cap A_2 = \emptyset$ and

$$\sum_{a \in A_1} a = \sum_{a \in A_2} a.$$

Groups

* 32. Let G be a nonempty set $(g_1, g_2) \mapsto g_1 \cdot g_2$ a mapping from $G \times G$ to G that satisfies (a)–(c).

(a) for all $g_1, g_2, g_3 \in G$ it holds that $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.

(b) there exists an element $1 \in G$, for which $g \cdot 1 = g$ for all $g \in G$.

(c) for all $g \in G$ there is an element $g^{-1} \in G$, for which $g^{-1} \cdot g = 1$.

Show that then the element $1 \in G$ may not be unique.

What if (b) is replaced with "there exists an element $1 \in G$, for which $1 \cdot g = g$ for all $g \in G$" ?

** 33. Let G be a finite group, H a subgroup of G, and K a subgroup of H. Let $\{g_1, \ldots, g_n\}$ be a left transversal of H with respect to G, and let $\{h_1, \ldots, h_m\}$ be a left transversal of K with respect to H. Show that any $g \in G$ can be represented in the form $g = g_i \cdot h_j \cdot k$, where $k \in K$ ja $1 \leq i \leq n, 1 \leq j \leq m$.

Schreier–Sims generators

** 34. Consider the permutation group G over $\{0, 1, \ldots, 9\}$ with the Schreier–Sims representation

$$\overrightarrow{G} = (\beta : [\mathcal{U}_0, \mathcal{U}_1, \ldots, \mathcal{U}_9]),$$

where $\beta = \mathbf{I}$ and

$$\mathcal{U}_0 = \left\{ \begin{array}{l} \mathbf{I},\ (0,1,3,6)\,(2,5,9,7)\,(4,8)\,,\ (0,2,5,9,6)\,(1,4,8,3,7)\,, \\ (0,3)\,(1,6)\,(2,9)\,(5,7)\,,\ (0,4,9,6)\,(1,2,5,8)\,(3,7)\,, \\ (0,5,6,2,9)\,(1,8,7,4,3)\,,\ (0,6,9,5,2)\,(1,7,3,8,4)\,, \\ (0,7,6)\,(1,2,8)\,(3,4,9)\,,\ (0,8,4,1,9)\,(2,5,3,6,7)\,, \\ (0,9,2,6,5)\,(1,3,4,7,8) \end{array} \right\},$$

$$\mathcal{U}_1 = \left\{ \begin{array}{l} \mathbf{I},\ (1,2)\,(3,4)\,(6,7)\,,\ (1,3,6)\,(2,4,7)\,(5,9,8)\,, \\ (1,4)\,(2,3)\,(6,7)\,(8,9)\,,\ (1,6)\,(2,7)\,(5,9)\,, \\ (1,7,3,2,6,4)\,(5,8,9) \end{array} \right\},$$

$$\mathcal{U}_2 = \{\mathbf{I}\}\,, \qquad \mathcal{U}_3 = \{\mathbf{I},\ (3,6)\,(4,7)\,(5,8)\}\,,$$

$$\mathcal{U}_4 = \mathcal{U}_5 = \mathcal{U}_6 = \mathcal{U}_7 = \mathcal{U}_8 = \mathcal{U}_9 = \{\mathbf{I}\}\,.$$

Which of the following permutations are in G?

(a) $\alpha = (0,1,2,3,4,5,6)$

(b) β=$(0,1,2,3,4)\,(5,6,7,8,9)$

(c) γ=$(0,3,5,8,7)\,(1,9,2,6,4)$

What is the order (number of elements) of G?

Counting permutations, partitions and colorings

** 35. How many different ways are there to label the six sides of a die, which each side labeled with a different number? (The die is assumed to be completely symmetrical and the orientation of the numbers on the sides is considered insignificant.) How about the 12 sides of a dodecahedron?

** 36. How many ways of partitioning a 19-set into one 4-set and three 5-sets are there? One such partitioning is, for example,

$$\{\{1,2,3,4,5\},\{6,7,8,9,10\},\{11,12,13,14,15\},\{16,17,18,19\}\}.$$

** 37. A square is split into 9 small squares, some of which are colored black. Two colorings are considered the same, if one can be obtained from the other by rotating and/or mirroring the whole square.

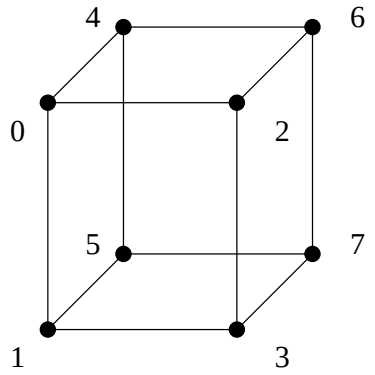
0	1	2
3	4	5
6	7	8

(a) How many different ways of coloring 5 small squares are there?

(b) Give all eight different ways of coloring 2 small squares.

Orbits, stabilizers

** 38. Let $\mathcal{G}$ be the graph given below.

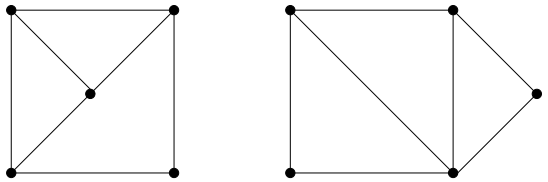


Find the orbits and stabilizers of the subsets $\{0, 7\}$ and $\{0, 1, 2, 3\}$ with respect to the group $\mathsf{Aut}(\mathcal{G})$. Additionally, for both stabilizer subgroups, find some left transversal with respect to $\mathsf{Aut}(\mathcal{G})$. (The elements of $\mathsf{Aut}(\mathcal{G})$ are listed in Table 6.1 of the book, and the Schreier–Sims presentation is given in Example 6.7.)

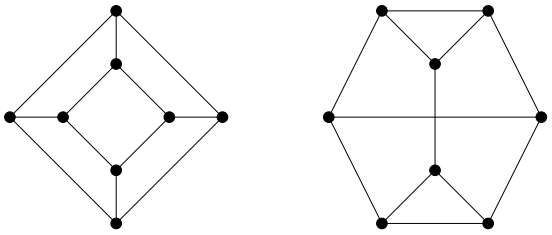
Invariants

* 39. Show, by using a suitable invariant, that the graphs given below are not isomorphic.

(a)



(b)

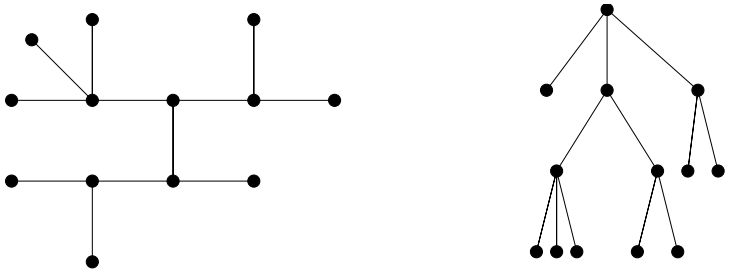


Ramsey numbers and edge-coloring complete graphs

** 40. The Ramsey number $R(k,l)$ is the least integer n, for which it holds that all edge 2-colorings of $\mathbf{K}_n$, the complete graph on n vertices, contain a $\mathbf{K}_k$ in the first or a $\mathbf{K}_l$ in the second color as a subgraph. Show that $R(3,4) > 8$ by constructing an edge 2-coloring of $\mathbf{K}_8$ that contains neither a $\mathbf{K}_3$ in the first color nor a $\mathbf{K}_4$ in the second color and which has cyclic symmetry.

Certificates

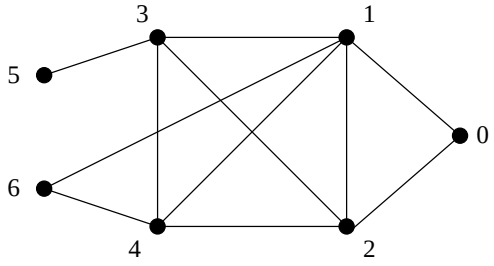
* 41. Check whether the trees given below are isomorphic by computing certificates for each of them.



* 42. Transform the certificate obtained as a result of Problem * 41. back into a tree.

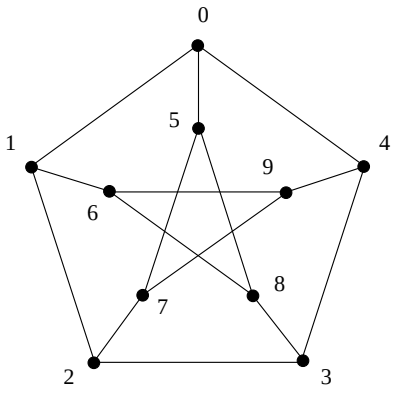
Coarsest equitable refinement of a partition

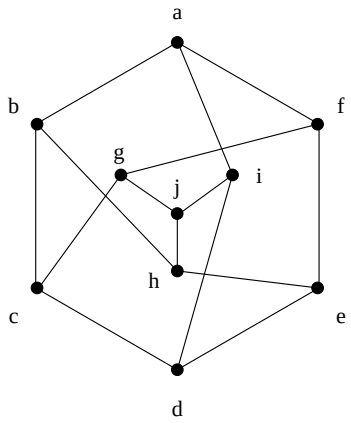
** 43. Partition the vertex set of the graph given below according to the degrees of the vertices, and find the coarsest equitable refinement of the partition.



Graph isomorphism

*** 44. Find an isomorphism between the given graphs, and find the order of the automorphism groups. (Hint: the automorphism groups are vertex transitive.)[1].)



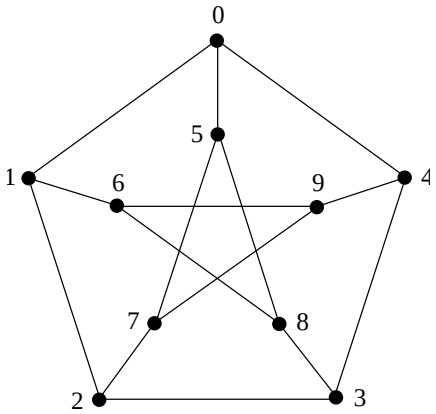


[1]A permutation group $G \subseteq \mathrm{Sym}(\{0, 1, \ldots, n-1\})$ is *transitive* if for all $x, y \in \{0, 1, \ldots, n-1\}$ there exists a $g \in G$ for which $y = g(x)$.

Solutions to problems

Problem * 1.

First label the vertices of the Petersen graph as below.



Then the adjacency matrix is

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	1	0	0	0	0
1	1	0	1	0	0	0	1	0	0	0
2	0	1	0	1	0	0	0	1	0	0
3	0	0	1	0	1	0	0	0	1	0
4	1	0	0	1	0	0	0	0	0	1
5	1	0	0	0	0	0	0	1	1	0
6	0	1	0	0	0	0	0	0	1	1
7	0	0	1	0	0	1	0	0	0	1
8	0	0	0	1	0	1	1	0	0	0
9	0	0	0	0	1	0	1	1	0	0

.

(In the adjacency matrix, the row corresponding to vertex i contains a 1 in the column corresponding to vertex j if there is an edge between i and j in the graph; otherwise the element in question is 0.)

With the same labeling, the incidence matrix is

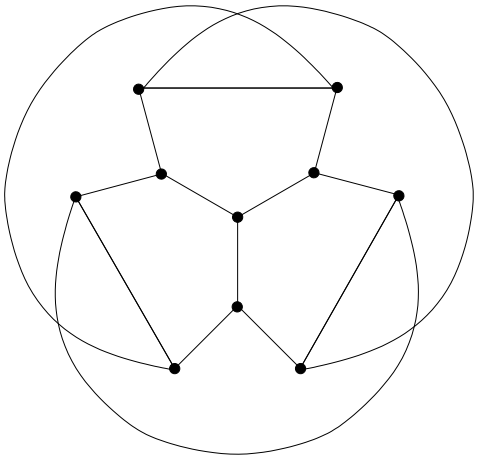
	01	12	23	34	40	05	16	27	38	49	57	58	68	69	79
0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1
8	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1

.

(In the incidence matrix, the row corresponding to vertex i has a 1 in the column corresponding to the edge $\{j, k\}$ if $i \in \{j, k\}$; otherwise the element in question is 0.)

Note that both the adjacency and the incidence matrix depend on the labeling of the vertices.

Extra problem. Below there is another presentation of the Petersen graph. Find a way of labeling the vertices of this graph so that you obtain the same adjacency and/or incidence matrix as above. (There are more than one solutions.)



Problem ** 2.

Choose one of the vertices of the graph, say v_0. Since every vertex of the graph is of degree k, there are k paths of length 1 with v_0 as one endpoint. There are

$k(k-1)$ paths of length 2 with v_0 as an endpoint: after choosing the first edge, the second edge can be chosen from the $k-1$ edges that do not lead back to v_0. Thus there are a total of k^2 distinct paths of length of length 1 or 2 with v_0 as one endpoint. If two of these paths would have another common endpoint in addition to v_0, there would be a cycle of length at most 4. So each of the k^2 paths must lead to a different vertex. Additionally the vertex v_0 is in the graph, so the graph must have at least k^2+1 vertices.

Problem * 3. (a)

We take advantage of the connection between the lexicographical and colexicographical ranks of k-subsets (Theorem 2.4 in the book): Let $T = \{t_1, t_2, \ldots, t_k\} \subseteq \{1, 2, \ldots, n\}$ be the k-subset under consideration and $T' \stackrel{\text{def}}{=} \{n+1-i \,:\, i \in T\}$. Then

$$\mathsf{rank}_{\text{lex}}(T) + \mathsf{rank}_{\text{colex}}(T') = \binom{n}{k} - 1.$$

In the problem we are given $n = 10$, $k = 3$ ja $\mathsf{rank}(T) = 99$, so

$$\mathsf{rank}_{\text{colex}}(T') = \binom{10}{3} - 1 - 99 = 120 - 1 - 99 = 20.$$

We use Algorithm 2.10 in the book to compute $T' = \{t'_1, t'_2, t'_3\} = \mathsf{unrank}_{\text{colex}}(20)$ with $n = 10$ and $k = 3$. The result is

$$20 = 20 + 0 + 0 = \binom{6}{3} + \binom{1}{2} + \binom{0}{1},$$

so

$$t'_1 = 6 + 1 = 7, \qquad t'_2 = 1 + 1 = 2, \qquad t'_3 = 0 + 1 = 1.$$

Thus we have $T' = \{7, 2, 1\}$, and $T = \{4, 9, 10\}$.

Problem * 3. (b)

We solve the problem as above. Now $T = \{3, 5, 7\}$, so $T' = \{8, 6, 4\}$. By definition of the colexicographical rank function

$$\mathsf{rank}_{\text{colex}}(T') \stackrel{\text{def}}{=} \sum_{i=1}^{k} \binom{t_i - 1}{k + 1 - i}$$

in the case $t'_1 = 8$, $t'_2 = 6$, $t'_3 = 4$ we obtain

$$\mathsf{rank}_{\text{colex}}(\{8,6,4\}) = \binom{7}{3} + \binom{5}{2} + \binom{3}{1} = 35 + 10 + 3 = 48,$$

so

$$\mathsf{rank}_{\text{lex}}(\{3,5,7\}) = \binom{10}{3} - 1 - \mathsf{rank}_{\text{colex}}(\{8,6,4\}) = 120 - 1 - 48 = 71.$$

Problem ** 4. (a)

Assume that the elements of S are ordered in the usual way. The problem can be solved by transforming the problem to the corresponding problem where the set is $\{1,2,3,4,5,6\}$ and using Algorithm 2.8 in the book, or Theorem 2.4 and Algorithm 2.9. The transformation can be carried out by a mapping $f : S \to \{1,2,3,4,5,6\}$, that preserves the order, that is, for all $x, y \in S$ we have $x < y$ if and only if $f(x) < f(y)$. The only such mapping is $f(2) = 1$, $f(3) = 2$, $f(5) = 3$, $f(7) = 4$, $f(11) = 5$, $f(13) = 6$. By applying the mapping f the set $\{3,7,13\} \subset S$ maps to the set $f(\{3,7,13\}) = \{2,4,6\}$. (Since f preserves the order, it can be shown that the lexicographical rank of $\{2,4,6\}$ among the 3-subsets of $\{1,2,3,4,5,6\}$ corresponds to the lexicographical rank of $\{3,7,13\}$ among the 3-subsets of S. We determine the lexicographical rank of $\{2,4,6\}$ by using Theorem 2.4 and Algorithm 2.9 (colex rank). By Theorem 2.4 the lexicographical rank of a k-subset T of $\{1,\ldots,n\}$ can be determined by determining the colexicographical rank of the k-osajoukon $T' = \{n+1-i : i \in T\}$ as follows:

$$\mathsf{rank}_{\text{lex}}(T) = \binom{n}{k} - 1 - \mathsf{rank}_{\text{colex}}(T').$$

Now $n = 6$ and $k = 3$, $T = \{2,4,6\}$ and $T' = \{6+1-2, 6+1-4, 6+1-6\} = \{5,3,1\}$. By Algorithm 2.9 we obtain

$$\mathsf{rank}_{\text{colex}}(\{5,3,1\}) = \binom{5-1}{3+1-1} + \binom{3-1}{3+1-2} + \binom{1-1}{3+1-3} = 4+1+0 = 5,$$

so by Theorem 2.4

$$\mathsf{rank}_{\text{lex}}(\{2,4,6\}) = \binom{6}{3} - 1 - 5 = 20 - 1 - 5 = 14.$$

Problem ** 4. (b)

From part (a) we know that $\mathsf{rank}_{\text{lex}}(\{2,4,6\}) = 14$. $\mathsf{unrank}_{\text{lex}}(14)$ can be computed either by Algorithm 2.8 or by Theorem 2.4 and Algorithm 2.10. We choose the latter; then $r = 5$, $n = 6$ and $k = 3$.

First we find the greatest $x \leq 6$, such that $\binom{x}{3+1-1}$ is less than or equal to r. This must be $x = 4$, since $\binom{4}{3+1-1} = 4 \leq 5$) and $\binom{5}{3+1-1} = 10 > 5$). Thus we obtain $t_1 = x+1 = 5$. We continue the algorithm with the updated $r = 5 - \binom{x}{3+1-i} = 1$, the values obtained are tabulated below:

i	r	x s.t. $\binom{x}{k+1-i} \leq r$	t_i
1	5	4	5
2	1	2	3
3	0	0	1

Thus we obtain $T' = \{5,3,1\}$, and using Theorem 2.4 we find $T = \{2,4,6\}$. Finally we do the inverse transformation $f^{-1}(\{2,4,6\}) = \{3,7,13\}$ (the inverse transformation exists, since the function f is a bijection). Thus we find $\mathsf{unrank}_{\text{lex}}(\mathsf{rank}_{\text{lex}}(\{3,7,13\})) = \{3,7,13\}$.

Problem * 5. (a)

First we derive a recursive rank function for the binary reflected Gray code G^n. (The nonrecursive version is Algorithm 2.4 in the book.) In the base case $n = 1$ it clearly holds that

$$\mathsf{rank}(0) = 0, \qquad \mathsf{rank}(1) = 1.$$

When $n \geq 2$, by the recursive definition of G^n we have

$$\mathsf{rank}(b_n b_{n-1} \cdots b_1) = \begin{cases} \mathsf{rank}(b_{n-1} \cdots b_1) & \text{if } b_n = 0; \\ 2^n - 1 - \mathsf{rank}(b_{n-1} \cdots b_1) & \text{if } b_n = 1. \end{cases}$$

The rank of the codeword 00101011 can now be obtained by the recursion

$$\begin{aligned}
\mathsf{rank}(00101011) &= \mathsf{rank}(0101011) \\
&= \mathsf{rank}(101011) \\
&= 63 - \mathsf{rank}(01011) \\
&= 63 - \mathsf{rank}(1011) \\
&= 63 - (15 - \mathsf{rank}(011)) \\
&= 63 - (15 - \mathsf{rank}(11)) \\
&= 63 - (15 - (3 - \mathsf{rank}(1))) \\
&= 63 - (15 - (3 - 1))) \\
&= 50.
\end{aligned}$$

Problem * 5. (b)

Let the rank of the codeword $b_n b_{n-1} \cdots b_1$ in G^n be r_n. Now unranking can be performed by recursively investigating which half of G^n (codewords starting with 0 / codewords starting with 1) the codeword belongs to:

$$b_n = \begin{cases} 0 & \text{if } r_n < 2^{n-1}; \\ 1 & \text{if } r_n \geq 2^{n-1}, \end{cases} \qquad r_{n-1} = \begin{cases} r_n & \text{if } r_n < 2^{n-1}; \\ 2^n - 1 - r_n & \text{if } r_n \geq 2^{n-1}. \end{cases}$$

We were given $r_8 = 49$. By applying the previous, we find

n	r_n	2^{n-1}	b_n
8	49	128	0
7	49	64	0
6	49	32	1
5	14	16	0
4	14	8	1
3	1	4	0
2	1	2	0
1	1	1	1

.

Therefore $\mathsf{unrank}(49) = 00101001$. (Alternatively, the problem could be solved by Algorithm 2.5.)

Problem * 6.

The graph (V, E) is the so called d-dimensional hypercube.Let $x = (x_1, \ldots, x_d) \in \{0,1\}^d$ be a vertex in the graph. The vertex has d neighbors,

since there are clearly d vertices whose coordinates differ from the coordinates of x in exacly one coordinate.

The length of the shortest cycle is 4: the vertices $(0, 0, y_3, \ldots, y_d)$, $(0, 1, y_3, \ldots, y_d)$, $(1, 1, y_3, \ldots, y_d)$, $(1, 0, y_3, \ldots, y_d)$ form a cycle of length 4, and on the other hand the graph cannot contain a triangle, since in that case some vertex y would have two neighbors whose coordinates differ from each other in only one position. This impossible, for then y would have to be the same as one of the neighbors.

The longest cycle is of length 2^d, that is, it contains all vertices in the graph. When $d = 2$ the cycle is obviously $00, 01, 11, 10$. Suppose that for some d the cycle is $\nu_1, \nu_2, \nu_3, \ldots, \nu_{2^d}$. The cycle can be extended to a cycle in the $(d + 1)$-dimensional cube: $0\nu_1, 0\nu_2, \ldots, 0\nu_{2^d}, 1\nu_{2^d}, 1\nu_{2^d-1}, 1\nu_{2^d-2}, \ldots, 1\nu_1$. Thus for example the cycle in the case $d = 3$ could be obtained by extending the cycle $00, 01, 11, 10$ to 000, 001, 011, 010, 110, 111, 101, 100.

Problem *** 7.

Denote the sequence of codewords in the binary reflected Gray code on n bits by G^n, and the same sequence in reverse order by $\overline{\mathsf{G}^n}$. Now the binary reflected Gray code on $(n + 1)$ bits can be denoted by $\mathsf{G}^{n+1} = (0\mathsf{G}^n, 1\overline{\mathsf{G}^n})$.

Since G^n contains all 2^n binary codewords of length n, it is clear that G^n also contains all the $\binom{n}{k}$ binary codewords of length n and exactly k ones, that is, codewords of Hamming weight k. Denote the subsequence of G^n that consists of the codewords of weight k by G^n_k.

The sequence $\mathsf{G}^n_k = (x_1, \ldots, x_{\binom{n}{k}})$ is a minimum change order of k-subsets if the Hamming distance of two consecutive codewords (number of positions where they differ) always equals 2, that is, $d_H(x_j, x_{j+1}) = 2$ for all $j = 1, \ldots, \binom{n}{k} - 1$.

Claim. Let $n \geq 1$. Then for all $1 \leq k \leq n$ the sequence G^n_k is a minimum change order of the k-subsets of an n-element set.

Proof. When $k = n$, the claim is trivially true, since G^n_n consists of only one codeword. In the case $k = 1$ the sequence G^n_1 consists of the codewords $(10\cdots0, 010\cdots0, 0010\cdots0, \ldots, 0\cdots01)$ (not in this order), of which there are n. It is clear that any two codewords of this form are at Hamming distance 2 from each other.

We will perform an induction over n. The cases $n = 1, 2$ are clear from the above special cases and form the base of the induction. Suppose that the claim

holds for n and for all $1 \leq k \leq n$. Consider the sequence G^{n+1} and choose some k, $(2 \leq k \leq n)$. (The cases $k = 1$ and $k = n + 1$ were dealt with above.) By definition of the sequence $\mathsf{G}^{n+1} = (0\mathsf{G}^n, 1\overline{\mathsf{G}^n})$, the subsequence $\mathsf{G}^{n+1}_k = (x_1, \ldots, x_{\binom{n+1}{k}})$ of codewords of weight k can be split into consecutive subsequences $(x_1, \ldots, x_m)$ and $(x_{m+1}, \ldots, x_{\binom{n+1}{k}})$ such that $(x_1, \ldots, x_m)$ contains the codewords where the first coordinate is 0 and the second one contains the codewords where the first coordinate is 1.The sequence $(x_1, \ldots, x_m)$ is thus the subsequence of words of weight k in $0\mathsf{G}^n$, and $(x_{m+1}, \ldots, x_{\binom{n+1}{k}})$ the subsequence of $1\overline{\mathsf{G}^n}$ that consists of words where the weight of the last n coordinates is koordinaatin paino on $k-1$. Thus $(x_1, \ldots, x_m) = 0\mathsf{G}^n_k$ and $(x_{m+1}, \ldots, x_{\binom{n+1}{k}}) = 1\overline{\mathsf{G}^n_{k-1}}$, where $\overline{\mathsf{G}^n_{k-1}}$ denotes the sequence G^n_{k-1} in reverse order. By induction we therefore have $d_H(x_j, x_{j+1}) = 2$ for all $j = 1, \ldots, m-1$ and on the other hand $d_H(x_{j+1}, x_j) = d_H(x_j, x_{j+1}) = 2$ for all $j = m+1, \ldots, \binom{n+1}{k} - 1$. The case $j = m$ remains. Since x_m is the last codeword in $0\mathsf{G}^n_k$ and on the other hand x_{m+1} is the first codeword in $1\mathsf{G}^n_{k-1}$, in the case $j = m$ we have $d_H(x_j, x_{j+1}) = 2$ by the following lemma.

Lemma. Let $n \geq 1$. Then for all $1 \leq k \leq n$ the sequence G^n satisfies:

1. the Hamming distance of the first codeword of weight k and the first codeword of weight $(k-1)$ is 1, and
2. the Hamming distance of the last codeword of weight k and the last codeword of weight $(k-1)$ is 1.

Proof. In the case $k = 1$, the sequence G^n only contains one codeword, which has weight $k - 1 = 0$, that is, $00\cdots0$. All codewords of weight 1 are clearly at Hamming distance 1 from this codeword. The case $k = n$ is similar, and the cases $2 \leq k \leq n-1$ remain. We proceed by induction on$n$ as before; the cases $n = 1, 2$ are again true due to the above special cases. Suppose that the claim holds for some$n$ (and all $1 \leq k \leq n$). Consider the sequence G^{n+1} and choose some k, $(2 \leq k \leq n)$. By definition of the sequence $\mathsf{G}^{n+1} = (0\mathsf{G}^n, 1\overline{\mathsf{G}^n})$ and the choice of k we observe that the first codeword y_k of weight k and the first codeword y_{k-1} of weight $k-1$ are in the subsequence $0\mathsf{G}^n$. Since the zero in the first position of y_k and y_{k-1} does not affect their weight, we have $d_H(y_k, y_{k-1}) = 1$. Similarly the last codeword z_k of weight k and the last codeword z_{k-1} of weight $k-1$are in the subsequence $1\overline{\mathsf{G}^n}$, that is, z_k corresponds to the first codeword of weight k in $1\mathsf{G}^n$ and z_{k-1} to the first codeword of weight $k-1$. Now we must have $z_k = 1z'_{k-1}$ and $z_{k-1} = 1z'_{k-2}$, where z'_{k-1} is the first codeword of weight $k-1$ in G^n and z'_{k-2} the first codeword of weight $(k-2)$. Again we may use the induction assumption (with value $k - 1 \geq 1$), and we obtain $d_H(z'_k, z'_{k-1}) = d_H(z_k, z_{k-1}) = 1$. □

Problem ** 8.

First identify the alphabet A, B, C, ..., Z with the numbers $0, 1, \ldots, 25$:

$$\mathrm{A} \triangleq 0, \quad \mathrm{B} \triangleq 1, \quad \mathrm{C} \triangleq 2, \quad \cdots \quad \mathrm{Y} \triangleq 24, \quad \mathrm{Z} \triangleq 25.$$

View the license plates as lists of the form $[z_1, z_2, z_3, z_4, z_5, z_6]$, where $z_1, z_2, z_3 \in \{0, 1, \ldots, 25\}$ and $z_4, z_5, z_6 \in \{0, 1, \ldots, 10\}$. Then the license plate IOI-010 corresponds to the list $[8, 14, 8, 0, 1, 0]$.

The standard lexicographical order among license plates is now defined by $[z'_1, z'_2, z'_3, z'_4, z'_5, z'_6] < [z_1, z_2, z_3, z_4, z_5, z_6]$ if and only if there exists some $j \in \{1, 2, 3, 4, 5, 6\}$ s.t. $z'_j < z_j$ and $z'_i = z_i$ for all $i \in \{1, \ldots, j-1\}$.

We first construct the rankfunction. The lexicographical rank of license plate $[z_1, z_2, z_3, z_4, z_5, z_6]$ is the number of such license plates $[z'_1, z'_2, z'_3, z'_4, z'_5, z'_6]$ for which $[z'_1, z'_2, z'_3, z'_4, z'_5, z'_6] < [z_1, z_2, z_3, z_4, z_5, z_6]$. Such license plates can be grouped according to the value of j (see the definition of lexicographical order above). In particular, z'_j can be chosen in z_j ways so that $z'_j < z_j$. After this the values z'_k, where $k > j$, may be chosen arbitrarily. We thus get

$$\begin{aligned}\mathrm{rank}([z_1, z_2, z_3, z_4, z_5, z_6]) = {} & z_1 \cdot 26 \cdot 26 \cdot 10 \cdot 10 \cdot 10 + \\ & + z_2 \cdot 26 \cdot 10 \cdot 10 \cdot 10 + \\ & + z_3 \cdot 10 \cdot 10 \cdot 10 + \\ & + z_4 \cdot 10 \cdot 10 + \\ & + z_5 \cdot 10 + \\ & + z_6.\end{aligned}$$

Or shorter,

$$\mathrm{rank}([z_1, z_2, z_3, z_4, z_5, z_6]) = \sum_{j=1}^{6} \Big(z_j \prod_{k=j+1}^{6} B_k \Big)$$

if we let $B_1 = B_2 = B_3 = 26$ and $B_4 = B_5 = B_6 = 10$. (In the case $j = 6$ we additionally define $\prod_{k=j+1}^{6} B_k = 1$.)

For the license plate IOI-010 we now obtain

$$\begin{aligned}\mathsf{rank}([8,14,8,0,1,0]) &= 8\cdot 26\cdot 26\cdot 10\cdot 10\cdot 10+\\ &\quad +14\cdot 26\cdot 10\cdot 10\cdot 10+\\ &\quad +8\cdot 10\cdot 10\cdot 10+\\ &\quad +0\cdot 10\cdot 10+\\ &\quad +1\cdot 10+\\ &\quad +0\\ &= 5780010.\end{aligned}$$

Unranking can be carried out by the following algorithm:

1. Let $n_6 \stackrel{\text{def}}{=} \mathsf{rank}([z_1, z_2, z_3, z_4, z_5, z_6])$.
2. Repeat for each $i = 6, 5, \ldots, 1$.
   (a) Compute the remainder $z_i = n_i \mod B_i$.
   (b) Let $n_{i-1} = (n_i - z_i)/B_i$.
3. Return the result $\mathsf{unrank}(n_6) = [z_1, z_2, z_3, z_4, z_5, z_6]$.

For example

i	n_i	B_i	z_i
6	5780010	10	0
5	578001	10	1
4	57800	10	0
3	5780	26	8
2	222	26	14
1	8	26	8

,

so $\mathsf{unrank}(5780010) = [8, 14, 8, 0, 1, 0]$ as it should.

Some definitions on permutations

Let us review some definitions on permutations. Let X be a finite nonempty set.

- The permutation σ on the set X is an *r-cycle* if there exist elements $a_1, \ldots, a_r \in X$ such that

$$\sigma(a_1) = a_2, \quad \sigma(a_2) = a_3, \quad \ldots, \sigma(a_{r-1}) = a_r, \quad \sigma(a_r) = a_1,$$

and $\sigma(a) = a$ for all $a \in X \setminus \{a_1, \ldots, a_r\}$. Such an r-cycle can be denoted by $(a_1, a_2, \ldots, a_r)$. The same cycle can be represented in several ways, as the starting point is arbitrary; e.g. $(a_2, a_3, \ldots, a_r, a_1) = (a_1, a_2, \ldots, a_r)$.

- A *transposition* is a 2-cycle on X.
- The permutations σ_1, σ_2 on X are *disjoint* if there is no $x \in X$ for which both $\sigma_1(x) \neq x$ and $\sigma_2(x) \neq x$.
- The product $\pi_1\pi_2$ of two permutations π_1, π_2 over X is defined by $\pi_1\pi_2(x) = \pi_1 \circ \pi_2(x) = \pi_1(\pi_2(x))$.
- The inverse permutation π^{-1} of a permutation π over X is the permutation over X for which $\pi\pi^{-1} = \pi^{-1}\pi = \iota$, where ι is the identity, that is, $\iota(x) = x$ for all $x \in X$.
- Every permutation on X can be represented as a unique (up to order of cycles) product of disjoint permutations, where each $x \in X$ appears on exactly one cycle.

Problem * 9. (a)

From the list presentation $[2, 4, 6, 7, 5, 3, 1]$ of π_1 we see that

$$\pi_1(1) = 2, \quad \pi_1(2) = 4, \quad \pi_1(4) = 7, \quad \pi_1(7) = 1,$$

so the cycle that contains the element 1 is $(1, 2, 4, 7)$. The least element of $\{1, 2, 3, 4, 5, 6, 7\}$ that is not on this cycle is 3. The cycle that contains 3 is $(3, 6)$, as

$$\pi_1(3) = 6, \quad \pi_1(6) = 3.$$

The least element that is not on the previous cycles is 5, for which $\pi_1(5) = 5$. Now we have examined all cycles of the elements in $\{1, 2, 3, 4, 5, 6, 7\}$, so we obtain

$$\pi_1 = (1, 2, 4, 7)(3, 6)(5).$$

Problem * 9. (b)

From the cycle presentation $\pi_2 = (1, 5, 6)(2, 4, 3)(7)$ we see that

$$\pi_2(1) = 5, \quad \pi_2(2) = 4, \quad \pi_2(3) = 2, \quad \pi_2(4) = 3,$$
$$\pi_2(5) = 6, \quad \pi_2(6) = 1, \quad \pi_2(7) = 7,$$

so $\pi_2 = [5, 4, 2, 3, 6, 1, 7]$.

Problem * 9. (c)

The inverse of a permutation can be easily formed from the cycle representation by reversing both the order of cycles and the elements in each cycle. For example

$$\pi_1^{-1} = (5)(6,3)(7,4,2,1), \qquad \pi_2^{-1} = (7)(3,4,2)(6,5,1).$$

Problem * 9. (d)

The product $\pi_1\pi_2$, expressed with cycle notations, is

$$\pi_1\pi_2 = (1,2,4,7)(3,6)(5)(1,5,6)(2,4,3)(7),$$

and can be simplified to

$$\pi_1\pi_2 = (1,5,3,4,6,2,7)$$

by computing the image of each element. When the image of each element is computed, the cycles are read from *right to left*. For example in the previous product the cycle (7) keeps the element 1 fixed, as does (2,4,3). The cycle $(1,5,6)$ maps 1 onto 5. The cycle (5) keeps 5 fixed, as do the cycles $(3,6)$ and $(1,2,4,7)$. Thus $\pi_1\pi_2(1) = 5$.

Similarly, the product $\pi_2\pi_1$, using cycle notations, is

$$\pi_2\pi_1 = (1,5,6)(2,4,3)(7)(1,2,4,7)(3,6)(5),$$

which can be simplified to

$$\pi_2\pi_1 = (1,4,7,5,6,2,3)$$

by computing the image of each element. Note that combining permutations (taking their product) is not a commutative operation, that is, $\pi_1\pi_2 \neq \pi_2\pi_1$.

Problem * 10.

The products given can be simplified to

$$(p,q)(p,r_1,\ldots,r_k,q,s_1,\ldots,s_l) = (p,r_1,\ldots,r_k)(q,s_1,\ldots,s_l)$$

and

$$(p,q)(p,r_1,\ldots,r_k)(q,s_1,\ldots,s_l) = (p,r_1,\ldots,r_k,q,s_1,\ldots,s_l).$$

Thus multiplying a permutation from the left by a transposition either "cuts" a cycle into two or "glues" two disjoint cycles.

We observe that if a permutation π is multiplied on the left by a transposition (p, q), then the permutation $(p, q)\pi$ either has one cycle more (part a) or one cycle less (part b) than the permutation π.

Problem ** 11.

Every permutation π over the set $\{1, \ldots, n\}$ can be uniquely (up to order of cycles) presented as a product of disjoint cycles:

$$\pi = (a_{1,1}, a_{1,2}, \ldots, a_{1,m_1}) \cdots (a_{k,1}, a_{k,2}, \ldots, a_{k,m_k}),$$

where $k \geq 1$ and $m_j \geq 1$ for all $j = 1, \ldots k$. The cycle $(a_{j,1}, \ldots, a_{j,m_j})$ can be presented as a product of $m_j - 1$ transpositions, e.g.

$$(a_{j,1}, a_{j,2})(a_{j,2}, a_{j,3}) \cdots (a_{j,m_j-2}, a_{j,m_j-1})(a_{j,m_j-1}, a_{j,m_j}).$$

Let $\lambda(\pi)$ be the number of transpositions when the permutation π is presented as a product of transpositions. For an arbitrary permutation π over $\{1, \ldots, n\}$ it then holds that

$$\lambda(\pi) = \sum_{j=1}^{k} (m_j - 1),$$

where k is the number of disjoint cycles in the permutation and $m_1, \ldots, m_k$ are their lengths. The function λ is well defined, since the value of $\lambda(\pi)$ does not depend on the order of the cycles.

We examine how multiplying π by an arbitrary transposition (x, y) affects the value of λ. We again have two cases:

(i) x and y are on the same cycle in π $(a_{j,1}, \ldots, a_{j,m_j})$ or $x = a_{j,p}$ and $y = a_{j,q}$ for some $p < q$. Then the cycle $(a_{j,1}, \ldots, a_{j,m_j})$ is cut into two disjoint cycles in the permutation $(x, y)\pi$
$(a_{j,1}, \ldots, a_{j,p-1}, a_{j,q}, \ldots, a_{j,m_j})$ and $(a_{j,p}, \ldots, a_{j,q-1})$, while the remaining cycles in π are unaffected. Thus $\lambda((x, y)\pi) = \lambda(\pi) - (m_j - 1) + (m_j - (q - p) - 1) + (q - p - 1) = \lambda(\pi) - 1$, since the length of the new cycles are $m_j - (q - p)$ and $q - p$.

(ii) x and y are on disjoint cycles $(a_{j,1}, \ldots, a_{j,m_j})$ and $(a_{l,1}, \ldots, a_{l,m_l})$ and we assume that $x = a_{j,1}$ and $y = a_{l,1}$. Then instead of the above permutations

$(x, y)\pi$ will contain the cycle $(a_{j,1}, \ldots, a_{j,m_j}, a_{l,1}, \ldots, a_{l,m_l})$ while other cycles remain unchanged. Thus $\lambda((x, y)\pi) = \lambda(\pi) - (m_j - 1) - (m_l - 1) + (m_j + m_l - 1) = \lambda(\pi) + 1$.

Thus we have $\lambda((x, y)\pi) \equiv (\lambda(\pi) + 1) \mod 2$.

Finally, suppose that an arbitrary permutation τ could be presented in two different ways as a product of transpositions: $\tau = (x_1, y_1) \cdots (x_m, y_m)$ and $\tau = (x'_1, y'_1) \cdots (x'_{m'}, y'_{m'})$. Since

$$\begin{aligned} \lambda(\tau) &= \lambda((x_2, y_2) \cdots (x_m, y_m)) + 1 = \ldots \\ &= m - 1 + \lambda((x_m, y_m)) = m \pmod 2 \end{aligned}$$

and on the other hand

$$\lambda(\tau) = \lambda((x'_2, y'_2) \cdots (x'_{m'}, y'_{m'})) + 1 = \ldots = m' \pmod 2,$$

either m and m' are both even or both odd.

Problem * 12.

The r-cycle in part (a) can be represented in several ways as a product of transpositions. E.g.

$$(1, 2, \ldots, r) = (1, 2)(2, 3)(3, 4) \cdots (r - 2, r - 1)(r - 1, r)$$

or

$$(1, 2, \ldots, r) = (1, r)(1, r - 1)(1, r - 2) \cdots (1, 3)(1, 2)$$

are both valid representations with $r - 1$ transpositions.

In part (b) the permutation can be represented as a product of transpositions e.g. by first representing the permutation $[2, 4, 6, 7, 5, 3, 1]$ as a product of disjoint cycles and then applying the solution to part (a) to each cycle. Then we obtain

$$\begin{aligned} [2, 4, 6, 7, 5, 3, 1] &= (1, 2, 4, 7)(3, 6)(5) \\ &= (1, 2)(2, 4)(4, 7)(3, 6). \end{aligned}$$

Therefore the permutation $[2, 4, 6, 7, 5, 3, 1]$ is even.

In general, if there are c cycles of lengths $r_1, \ldots, r_c$ in the representation of the permutation π over X, the permutation can be expressed as a product of $\lambda(\pi) = \sum_{i=1}^{c}(r_i - 1)$ transpositions. Since $\sum_{i=1}^{c} r_i = |X|$, this can be simplified to $\lambda(\pi) = |X| - c$. A permutation π is thus even (odd) exactly when $\lambda(\pi)$ is even (odd).

Problem * 13.

Consider first the lexicographical order. The lexicographical successor of

$$\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$$

is defined as follows:

1. Find the largest $j \in \{1, \ldots, n-1\}$, for which $\pi(j) < \pi(j+1)$. If no such j exists, we have the lexicographically last permutation $[n, n-1, \ldots, 1]$, which has no successor.

   For the permutation $[2, 4, 6, 7, 5, 3, 1]$ we find $j = 3$.

2. Among the elements $\pi(j+1), \ldots, \pi(n)$ we find the least element that is greater than $\pi(j)$.

   For $[2, 4, 6, 7, 5, 3, 1]$ this is 7.

3. We swap $\pi(j)$ with the element chosen in the previous step to obtain π'.

   For $[2, 4, 6, 7, 5, 3, 1]$ we get $\pi' = [2, 4, 7, 6, 5, 3, 1]$.

4. We order the elements $\pi'(j+1), \ldots, \pi'(n)$ in ascending order (since the elements are in decreasing order, it suffices to reverse their order). The result is the lexicographical successor of π.

   The successor of $[2, 4, 6, 7, 5, 3, 1]$ is thus $[2, 4, 7, 1, 3, 5, 6]$.

The lexicographical rank of the permutation $\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$ is defined recursively:

1. Base case $n = 1$: $\mathsf{rank}(\pi) = 0$.

2. General case $n \geq 2$:

   $$\mathsf{rank}(\pi) = (\pi(1) - 1)(n-1)! + \mathsf{rank}(\pi'),$$

   where

   $$\pi'(j) = \begin{cases} \pi(j+1) & \text{if } \pi(j+1) < \pi(1); \\ \pi(j+1) - 1 & \text{if } \pi(j+1) > \pi(1). \end{cases}$$

Based on this, the rank of $[2, 4, 6, 7, 5, 3, 1]$ is

$$\begin{aligned}
&\mathsf{rank}([2, 4, 6, 7, 5, 3, 1]) = \\
&= (2-1)6! + \mathsf{rank}([3, 5, 6, 4, 2, 1]) \\
&= (2-1)6! + (3-1)5! + \mathsf{rank}([4, 5, 3, 2, 1]) \\
&= (2-1)6! + (3-1)5! + (4-1)4! + \mathsf{rank}([4, 3, 2, 1]) \\
&= (2-1)6! + (3-1)5! + (4-1)4! + (4-1)3! + \mathsf{rank}([3, 2, 1]) \\
&= (2-1)6! + (3-1)5! + (4-1)4! + (4-1)3! + (3-1)2! + \mathsf{rank}([2, 1]) \\
&= (2-1)6! + (3-1)5! + (4-1)4! + (4-1)3! + (3-1)2! + (2-1)1! \\
&= 1 \cdot 6! + 2 \cdot 5! + 3 \cdot 4! + 3 \cdot 3! + 2 \cdot 2! + 1 \cdot 1! \\
&= 1055.
\end{aligned}$$

Consider now the Trotter–Johnson order. In the Trotter–Johnson order the successor of

$$\pi = [\pi(1), \pi(2), \ldots, \pi(n)]$$

is defined by

1. Find $j \in \{1, \ldots, n\}$ such that $\pi(j) = n$.

   For $[2, 4, 6, 7, 5, 3, 1]$ we find $\pi(4) = 7$.

2. Determine whether the permutation $\pi' = [\pi(1), \ldots, \pi(j-1), \pi(j+1), \ldots, \pi(n)]$ is even or odd.

   In the case of permutation $[2, 4, 6, 7, 5, 3, 1]$

   $$\pi' = [2, 4, 6, 5, 3, 1] = (1, 2, 4, 5, 3, 6) = (1, 2)(2, 4)(4, 5)(5, 3)(3, 6)$$

   is odd.

3. If π' is even and $j > 1$: the successor of π is obtained by swapping $\pi(j-1)$ and $\pi(j)$.

4. If π' is even and $j = 1$: the successor of π is

   $$[n, \pi''(1), \pi''(2), \ldots, \pi''(n-1)],$$

   where π'' is the Trotter–Johnson successor of π'.

5. If π' is odd and $j < n$: the successor of π is obtained by swapping the elements $\pi(j)$ and $\pi(j+1)$.

   The Trotter–Johnson successor of $[2, 4, 6, 7, 5, 3, 1]$ is therefore $[2, 4, 6, 5, 7, 3, 1]$.

6. If π' is odd and $j = n$: the successor of π is

$$[\pi''(1), \pi''(2), \ldots, \pi''(n-1), n],$$

where π'' is the Trotter–Johnson successor of π'.

The Trotter–Johnson rank of π is computed recursively:

1. Base case $n = 1$: $\mathsf{rank}(\pi) = 0$.
2. For $n \geq 2$: Find $j \in \{1, \ldots, n\}$ such that $\pi(j) = n$.
3. Determine recursively the rank of permutation

$$\pi' = [\pi(1), \ldots, \pi(j-1), \pi(j+1), \ldots, \pi(n)].$$

4. If $\mathsf{rank}(\pi')$ is even,

$$\mathsf{rank}(\pi) = n\mathsf{rank}(\pi') + n - j.$$

5. If $\mathsf{rank}(\pi')$ is odd, let

$$\mathsf{rank}(\pi) = n\mathsf{rank}(\pi') + j - 1.$$

For $[2, 4, 6, 7, 5, 3, 1]$ we get:

n	π	j	$\mathsf{rank}(\pi)$
1	$[1]$	1	0
2	$[2, 1]$	1	$2 \cdot 0 + (2 - 1) = 1$
3	$[2, 3, 1]$	2	$3 \cdot 1 + (2 - 1) = 4$
4	$[2, 4, 3, 1]$	2	$4 \cdot 4 + (4 - 2) = 18$
5	$[2, 4, 5, 3, 1]$	3	$5 \cdot 18 + (5 - 3) = 92$
6	$[2, 4, 6, 5, 3, 1]$	3	$6 \cdot 92 + (6 - 3) = 555$
7	$[2, 4, 6, 7, 5, 3, 1]$	4	$7 \cdot 555 + (4 - 1) = 3888$

Problem * 14.

The partition $1 + 3 + 4 + 6 + 6 + 8$ is in reverse standard form; the standard form is $8 + 6 + 6 + 4 + 3 + 1$. The rsf-lex rank is the lexicographical rank of the list presentations of the partitions in reverse standard form. The rsf-rank of a partition can be computed by the following recursion (p. 76 in the book): Suppose that the

partition to be ranked is in the standard form $[a_1, \ldots, a_n]$, where $a_1 \geq \cdots \geq a_n$. In the base case of the recursion $n = 1$ always $\mathsf{rank}([a_1]) = 0$. When $n > 1$ the recursion step is

$$\mathsf{rank}([a_1, \ldots, a_n]) = \begin{cases} \mathsf{rank}([a_1, \ldots, a_{n-1}]) & \text{jos } a_n = 1 \\ \mathsf{rank}([a'_1, \ldots, a'_n]) + P(m-1, n-1) & \text{if } a_n > 1, \end{cases}$$

where $a'_j = a_j - 1$ for all $1 \leq j \leq n$ and $m = \sum_{i=1}^{n} a_i$. By applying the recursion equation we find

$$\mathsf{rank}([8,6,6,4,3,1]) = \mathsf{rank}([8,6,6,4,3]) = \mathsf{rank}([7,5,5,3,2]) + P(27-1, 5-1),$$

and further

$$\begin{aligned} \mathsf{rank}([7,5,5,3,2]) &= \mathsf{rank}([6,4,4,2,1]) + P(22-1, 5-1) \\ &= \mathsf{rank}([6,4,4,2]) + P(21,4), \\ \mathsf{rank}([6,4,4,2]) &= \mathsf{rank}([5,3,3,1]) + P(16-1, 4-1) \\ &= \mathsf{rank}([5,3,3]) + P(15,3), \\ \mathsf{rank}([5,3,3]) &= \mathsf{rank}([4,2,2]) + P(11-1, 3-1), \\ \mathsf{rank}([4,2,2]) &= \mathsf{rank}([3,1,1]) + P(8-1, 3-1) \\ &= \mathsf{rank}([3]) + P(7,2) = P(7,2). \end{aligned}$$

By combining we obtain:

$$\begin{aligned} \mathsf{rank}([8,6,6,4,3,1]) &= P(26,4) + P(21,4) + P(15,3) + P(10,2) + P(7,2) \\ &= 136 + 72 + 19 + 5 + 3 = 235. \end{aligned}$$

The successor of a partition in standard form:

1. Find the first sublist that is not equally split, that is, the least i, for which $a_1 > a_i + 1$,
2. increment a_i by one and set $a_2, \ldots, a_{i-1}$ to their minimum values ($= a_i$),
3. justify the sum by setting $a_1 = m - \sum_{i=2}^{n} a_i$.

For $[8,6,6,4,3,1]$ we find $i = 2$. Set $a_2 = a_2 + 1 = 7$. Finally $a_1 = 28 - 7 - 6 - 4 - 3 - 1 = 7$, and we obtain $[7,7,6,4,3,1]$.

Problem ** 15.

Suppose first that people can be split into groups arbitrarily. If we leave $0 \leq k \leq 4$ groups empty out of the five, the remaining persons are split into $5-k$ groups. We number the nonempty groups $1, \ldots, 5-k$ and use a_j to denote the number of people in group j. Now clearly $a_j \geq 1$ for $j = 1, \ldots, 5-k$, and on the other hand $\sum_{j=1}^{5-k} a_j = 20$. Therefore $\{(a_1, \ldots, a_{5-k}) \ : \ a_j \geq 1 \text{ and } \sum_{j=1}^{5-k} a_j = 20\}$ contains all possibilities of dividing 20 people into *labeled* groups. However, since the groups are not labeled, many of the partitions are counted more than once; for example $(1,1,1,1,16)$ and $(16,1,1,1,1)$ in the case $k=0$ are alike. This duplication can be avoided by requiring additionally that $a_1 \leq \cdots \leq a_{5-k}$. Then we observe that the elements of the set

$$\mathcal{P}(20, 5-k) := \{(a_1, \ldots, a_{5-k}) \ : \ 1 \leq a_1 \leq \cdots \leq a_{5-k} \text{ and } \sum_{j=1}^{5-k} a_j = 20\}$$

actually correspond to the partitions of 20 into $5-k$ parts, so there are $P(20, 5-k)$ distinct partitions. Thus there are a total of

$$\sum_{k=0}^{4} P(20, 5-k) = P(20,5) + P(20,4) + \cdots + P(20,1) = 192$$

distinct partitions.

Now we forbid partitions with two equal-sized parts. Now clearly at most one group can be left empty. Let $0 \leq k \leq 1$ be the number of empty groups. The remaining $5-k$ groups should be filled so that no two groups have the same number of members. This can be achieved by requiring that $1 \leq a_1 < a_2 < \cdots < a_{5-k}$ and $\sum_{j=1}^{5-k} a_j = 20$. Therefore the number of such partitions equals the number of elements in

$$\mathcal{P}'(20, 5-k) = \{(a_1, \ldots, a_{5-k}) \ : \ 1 \leq a_1 < \cdots < a_{5-k} \text{ ja } \sum_{j=1}^{5-k} a_j = 20\}.$$

Now we observe that the mapping

$$(a_1, \ldots, a_{5-k}) \mapsto (a_1, a_2 - 1, a_3 - 2, \ldots, a_{5-k} - (5-k-1))$$

from $\mathcal{P}'(20, 5-k)$ onto $\mathcal{P}(20 - 1 - 2 - \cdots - (5-k-1), 5-k)$ is a bijection, so

$$|\mathcal{P}'(20, 5-k)| = P(20 - \frac{(5-k-1)(5-k)}{2}, 5-k),$$

and the number of partitions of the required kind is

$$P(20-10, 5) + P(20-6, 4) = P(10,5) + P(14,4) = 7 + 23 = 30.$$

Problem * 16. (a)

Given the edge set $\mathcal{E}$ of a labeled tree the Prüfer list representation can be computed as follows:

1. $i = 1$
2. Let v be the highest-numbered vertex of degree 1. Find the edge $\{v, v'\} \in \mathcal{E}$ and set $L[i] = v'$. Remove the edge $\{v, v'\}$ from the graph.
3. Let $i = i + 1$ and if $i < n - 1$ go to 2.
4. The Prüfer list representation is $[L[1], \ldots, L[n-2]]$.

The execution of the algorithm is given in the array:

Degrees of vertices	i	v	$L[i]$	Edge removed
$3,1,2,2,2,1,1$	1	7	3	$\{7,3\}$
$3,1,1,2,2,1,0$	2	6	1	$\{6,1\}$
$2,1,1,2,2,0,0$	3	3	4	$\{3,4\}$
$2,1,0,1,2,0,0$	4	4	1	$\{4,1\}$
$1,1,0,0,2,0,0$	5	2	5	$\{2,5\}$

The Prüfer list representation is thus $[3,1,4,1,5]$.

Problem * 16. (b)

Follow algorithm 3.11 of the book. The input is the list $[L[1], L[2], \ldots, L[n-2]]$.

1. The number of vertices n = length of list + 2.
2. Degree of vertex = number of occurrences in list + 1.
3. Append $L[n-1] = 1$.
4. For $i = 1, \ldots, n-1$ do:
   (a) Find the greatest vertex x of degree 1.
   (b) Add the edge $\{x, L[i]\}$ to the tree.
   (c) Decrement the degree of the vertices x and $L[i]$ by one.

The length of the given list $[5, 5, 4, 3, 2]$ is 5, so there are $n = 7$ vertices in the tree. Proceed according to the algorithm.

		asteluvut							
i	$L[i]$	1	2	3	4	5	6	7	$x, L[i]$
1	5	1	2	2	2	3	1	$\underline{1}$	$7, 5$
2	5	1	2	2	2	2	$\underline{1}$	0	$6, 5$
3	4	1	2	2	2	$\underline{1}$	0	0	$5, 4$
4	3	1	2	2	$\underline{1}$	0	0	0	$4, 3$
5	2	1	2	$\underline{1}$	0	2	0	0	$3, 2$
6	1	1	$\underline{1}$	0	0	0	0	0	$2, 1$

As the result we obtain the graph below.

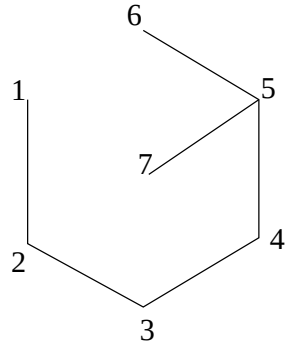


Figure 1: The tree that corresponds to Prüfer list representation $[5, 5, 4, 3, 2]$.

Problem *** 17.

Every way of grouping an $n+1$ term product to two term products corresponds to a rooted ordered binary tree, whose branches corresponds to multiplications and whose leaves correspond to the terms.

All ways of grouping a four term product, and the corresponding parse trees, are presented in Figure 2.

A full binary tree is a rooted tree, where each vertex is either a leaf or has two children. Suppose additionally that the children of each node are ordered so that we can speak of a "left" and "right" child node. Now clearly every parse tree of the product of $n + 1$ terms is a full binary tree with $n + 1$ leaves and conversely. By induction we can show the following lemma:

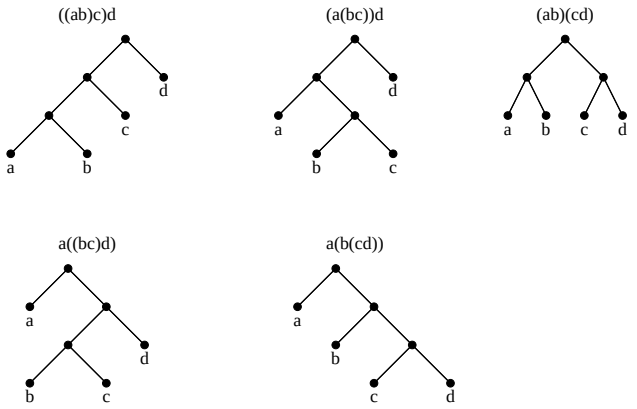


Figure 2: Ways of grouping a four term product and the corresponding parse trees.

Lemma. In a full binary tree with $n + 1$ leaves there are exactly n vertices with two children.

Proof. The base case $n = 0$ is obvious. Suppose that the lemma holds for $0 \leq k \leq n$ and consider the full binary tree with $(n + 1) + 1 \geq 2$ leaves. Now clearly the root must have two children. Let l_1 be the number of leaves in the subtree rooted at the left child of the root, and l_2 similarly the number of leaves in the right subtree. Since the root is not a leaf node, we must have $n+2 = l_1+l_2$ and $l_1, l_2 \geq 1$, so $l_1 \leq n+1$ and $l_2 \leq n+1$. Thus the induction assumption is applicable to the subtrees rooted at the left and right child of the root, which clearly must be full binary trees. The whole tree thus contains $1+(l_1-1)+(l_2-1) = l_1+l_2-1 = n+1$ vertices, including the root, that have two children. □

An immediate corollary of the lemma is that a full binary tree with $n + 1$ leaves has exactly $2n$ edges.

The Catalan family C_n consists of all $2n$-bit binary strings $a_1a_2 \cdots a_{2n}$, for which

(a) the string contains exactly n zeroes and n ones, and

(b) for all $1 \leq i \leq 2n$, the substring $a_1a_2 \cdots a_i$ contains at least as many zeroes as ones.

It is now easy to find the correspondenge between full binary trees with $n + 1$ leaves and the Catalan family:

Given a full binary tree with $n + 1$ leaves, label each edge that leads to the left child of a vertex by 0 and each edge to the right child by 1. Now traverse the tree

in preorder (first the vertex, then recursively the left child, then recursively the right child) so than upon arriving at a vertex we output the label on the edge that leads to the parent of the vertex we arrived at.

Clearly each of the labels on the $2n$ edges is output once. The output sequence satisfies (a), since each vertex with a left child also has a right child; condition (b) is satisfied, since for each vertex, the label of the edge leading to its left child is output before the label on the edge to the right child.

Conversely, given a binary string $a_1 a_2 \cdots a_{2n}$ in a Catalan family, the corresponding full binary tree can be constructed as follows:

1. First the tree consists of only the root, which is a leaf. Set the root as the current vertex.
2. For $i = 1, 2, \ldots, 2n$ do:
   (i) If $a_i = 0$, create a left child for the current vertex, and set the child as the current vertex.
   (ii) If $a_i = 1$, backtrack from the current node towards the root until we find a node with no right child. Create a right child for this node and set it as the current vertex.

This construction is well defined, since according to the balance condition (b) in step (ii) a vertex to which a right child can be added always exists. On the other hand, the balance condition (a) gives us that each node for which a left child has been added according to (i) will also receive a right child according to (ii). The result of the construction is a full binary tree with $2n + 1$ vertices $2n$ edges. Now we can again show by induction that such a tree must have $n + 1$ leaves. Thus the construction always results in a full binary tree with $(n + 1)$ leaves.

The mappings constructed from trees to binary strings and vice versa can be verified to be inverse mappings of each other.

Problem * 18. (a)

Assume the rules of chess are know. Examine each row $1, \ldots, n$ of the board in turn. The choice set would be the set of squares on that line that are not yet threatened by a queen on the board. Try each of them in turn, recursively, and go to the next line.

It may not be best to consider the lines in order. It may be more efficient to organize the search so that at each step we consider the line with the fewest unthreatened squares at the time. Then the branching factor of the search can be smaller near the root, so the number of search nodes is likely smaller, even though all alternatives are examined here too.

Problem * 18. (b)

Consider the vertices in order. For each vertex the choice set is the set of colors $\{1, 2, \ldots, k\}$, from which we however must remove those colors that have been used to color a neighbor of the current vertex.

Problem * 18. (c)

The choice set can be the set of those neighbors of the current vertex that have not yet been visited. Try each of them in turn and continue recursively until n steps have been taken.

Problem * 18. (d)

If $n \not\equiv 1 \mod 6$ and $n \not\equiv 3 \mod 6$, there is no Steiner triple system STS(n). Otherwise, generate all 3-subsets of an n-element set and order them. Construct the collection of 3-subsets by adding one 3-subset at a time. The choice set is the set of compatible 3-subsets (the set of all 3-subsets except those that contain some pair that occurs in one of the 3-subsets already in the set). A Steiner triple system is found when we have $n(n-1)/6$ subsets.

Alternatively this could be treated as an exact cover problem: For the choice set, choose some pair that is not covered by the 3-subsets already chosen. Let the choice set be the set of those subsets that cover that particular pair and are compatible with the previously chosen 3-subsets.

Problem * 18. (e)

In Minesweeper, the player tries to find all mines in an $n \times n$ array. The player has a detector that shows the number of mines in the squares surrounding the square in question. A position in the game consists of a set of known squares, where the reading of the mine detector is known. The game proceeds from position to

position so that the player chooses an unknown square; if that square contains a mine, the player loses the game, while if there is no mine, the player finds out the reading of the mine detector in that square, and the game continues. The player wins if all unknown squares contain a mine. (In this version the number of the mines is assumed to be unknown.)

Let the set of squares in the array be $D_n = \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$. The *surroundings* of the square (x, y) is the set

$$N(x, y) = \{(x, y) + (\delta_1, \delta_2) \ : \ -1 \leq \delta_1, \delta_2 \leq 1, \ (\delta_1, \delta_2) \neq (0, 0)\} \cap D_n.$$

Note that $(x, y) \notin N(x, y)$.

Let $\mathsf{mines} \subseteq D_n$ be the set of squares that contain a mine. (This set is of course unknown to the player.) The reading of the mine detector in square (x, y) is then

$$d(x, y) = |\mathsf{mines} \cap N(x, y)|.$$

Let $\mathsf{known} \subseteq D_n$ be the set of squares selected by the player. Supposing that the game is not over, we may assume that $\mathsf{known} \cap \mathsf{mines} = \emptyset$.

Since the total number of mines is unknown, the player can only obtain information about the number and placement of mines via readings of the mine detector in chosen squares. By definition the reading of the mine detector in square (x, y) only gives information about the mines in $N(x, y)$, so the player can make deductions about the location of the mines at the *edge*

$$\mathsf{boundary} = \Big(\bigcup_{(x,y) \in \mathsf{known}} N(x, y) \Big) \setminus \mathsf{known}$$

of the known area. The player cannot deduce anything about mines outside the edge. (This may not hold if the total number of mines is known in advance!)

We limit the consideration to the set $\mathsf{boundary}$. We call a set

$$E \subseteq \mathsf{boundary}$$

consistent, if for all $(x, y) \in \mathsf{known}$ it holds that $d(x, y) = |E \cap N(x, y)|$. Thus a set E is consistent if and only if it is possible that the mines would actually be in the squares of E. In particular, the set $\mathsf{mines} \cap \mathsf{boundary}$ is consistent.

Let $\mathscr{E}$ be the set of consistent sets. Define the sets

$$\mathsf{safe} = \mathsf{boundary} \setminus \bigcup_{E \in \mathscr{E}} E, \quad \text{and} \quad \mathsf{flag} = \bigcap_{E \in \mathscr{E}} E.$$

First consider the set safe. Now safe $\cap$ mines $= \emptyset$, since mines $\cap$ boundary $\in \mathscr{E}$. The set safe is thus a set of safe squares, that definitely do not contain a mine. On the other hand there are no such squares outside safe, since all squares $x \in$ boundary $\setminus$ safe are members of some consistent set E, and the mine detector readings cannot be used to decisively dismiss the possibility that mines $\cap$ boundary $= E$.

Consider next the set flag. Since flag $\subseteq E$ for all $E \in \mathscr{E}$, this also holds for the set $E =$ mines $\cap$ boundary. Therefore flag $\subseteq$ mines. The set flag thus only contains squares with a mine. On the other hand for all squares $x \in$ boundary $\setminus$ flag there exists a consistent set E for which $x \notin E$. Therefore flag is the maximum set of squares that definitely contain a mine.

The sets safe and flag are thus the sets asked for in subproblems (i) and (ii).

Backtrack search could be used for determining the sets safe and flag for example as follows:

- Design a backtrack algorithm that outputs all consistent sets.
- First initialize safe $\leftarrow$ boundary and flag $\leftarrow$ boundary.
- Use backtrack search to find all consistent sets E. For each consistent set found, set
$$\text{safe} \leftarrow \text{safe} \setminus E \quad \text{and} \quad \text{flag} \leftarrow \text{flag} \cap E.$$
- When all consistent sets have been examined, safe and flag are the desired sets. (Note that both of them can be empty. This happens for example at the beginning of the game, when known $= \emptyset$.)

The consistent sets can be produced by backtrack search for example as follows.

- Let there be N elements $P_1, \ldots, P_N$ in boundary, where $N \geq 1$. (We ignore the special case $N = 0$.)
- We model the subsets $E \subseteq$ boundary as binary lists $\vec{E} = [x_1, \ldots, x_N]$, for which $x_i = 1 \Leftrightarrow P_i \in E$ and $x_i = 0 \Leftrightarrow P_i \notin E$.
- A partial solution is a list $[x_1, \ldots, x_{k-1}]$; we start with the empty list.
- At depth k in the search tree we append to the list, in turn, the elements $x_k = 0$ and $x_k = 1$, and proceed recursively.

- When there are N elements in the list, we test if the corresponding set is consistent. (If it is, we update the sets safe and flag as above.) After this we return to the previous level in the search tree.
- This search can be made *significantly* more effective by eliminating partial solutions that clearly cannot be completed to consistent sets (see problem *** 20.).

Problem ** 19.

The program below solves the problem in a simple manner. Since the first row and column of a reduced Latin square are defined, to construct an $n \times n$ Latin square it suffices to place $(n-1)^2$ numbers; this is the maximum depth of the search. The problem does not actually maintain the Latin square being constructed in memory; rather it uses the Boolean variables `n_in_row_i` ja `n_in_col_j`, to indicate whether the number n already appears on row i or column j.

```
/* Find reduced Latin squares of order N */
#define N 4

int n_in_row_i[N+1][N+1]; /* has number n been placed in row i */
int n_in_col_j[N+1][N+1]; /* has number n been placed in column j */

int place(int depth) {
  int i,j,n,sum;

  if (depth==(N-1)*(N-1)) /* found a square, as we have */
    return 1;             /* placed (N-1)**2 numbers */

  i=2+depth%(N-1); /* compute from depth, where to place */
  j=2+depth/(N-1); /* next number         */

  sum=0;
  for (n=1; n<=N; n++) {
    if(!n_in_row_i[n][i] && !n_in_col_j[n][j]) {
      n_in_row_i[n][i]=1;  /* for each n in turn:         */
      n_in_col_j[n][j]=1;  /* if n has not been placed in row i */
      sum+=place(depth+1); /* or column j, place it at their */
      n_in_row_i[n][i]=0;  /* intersection and continue search; */
      n_in_col_j[n][j]=0;  /* finally remove the number */
```

```
    }
  }
  return sum;
}

int main(int argc, char **argv) {
  int i,j;

  for(i=1;i<=N;i++) {      /* zero arrays */
    for(j=1;j<=N;j++) {
      n_in_row_i[i][j]=n_in_col_j[i][j]=0;
    }
  }

  for(i=1;i<=N;i++) {   /* place 1...n     */
    n_in_row_i[i][i]=1; /* in first column */
    n_in_col_j[i][i]=1; /* and row         */
  }
  printf("%d\n", place(0));
  return 0;
}
```

Problem *** 20.

We use the concepts defined for problem * 18. (e).

The problem is to determine the consistent subsets of boundary, that is, all ways of placing mines in the squares of boundary so that the allocation of mines agrees with the readings of the mine detector.

The naive solution works by a generate and test principle: it produces all subsets of boundary and then tests if each subset is consistent. This is of course not very efficient, since the readings of the mine detector around boundary often limit the number of consistent subsets to a very small fraction of all subsets.

In particular, the mine detector readings $d(x, y)$ in the squares of

$$\mathsf{constraints} = \{(x, y) \in \mathsf{known} \ : \ N(x, y) \cap \mathsf{boundary} \neq \emptyset\}$$

limit the structure of consistent sets.

We now define a backtrack algorithm so that the restrictions posed by constraints

are verified already for partial solutions, so that the search space will be significantly smaller than with the naive solution. The pseudocode is given below.

```
CONSISTENT_SETS(E, R)
    if R = ∅
        check consistency of E, output solution if consistent.
        return
    for (x, y) ∈ constraints
        if d(x, y) = |E ∩ N(x, y)| and R ∩ N(x, y) ≠ ∅                    (i)
            CONSISTENT_SETS(E, R \ N(x, y))
            return
        if d(x, y) = |E ∩ N(x, y)| + |R ∩ N(x, y)| and R ∩ N(x, y) ≠ ∅    (ii)
            CONSISTENT_SETS(E ∪ (R ∩ N(x, y))), R \ N(x, y))
            return
        if d(x, y) > |E ∩ N(x, y)| and R ∩ N(x, y) = ∅                    (iii)
            return
        if d(x, y) < |E ∩ N(x, y)|                                        (iv)
            return
    P ← any point from R
    CONSISTENT_SETS(E, R \ {P})
    CONSISTENT_SETS(E ∪ {P}, R \ {P})
    return
```

The algorithm CONSISTENT_SETS takes two arguments E, R, where

- $E \subseteq$ boundary is a set that contains the squares (x, y), where the algorithm has placed a mine.
- $R \subseteq$ boundary is a set that contains the squares (x, y), of which the algorithm has not yet decided whether to place a mine there or not.

To determine all consistent sets the algorithm is called with the arguments

$$E = \emptyset \quad \text{and} \quad R = \text{boundary}.$$

For each recursive step of the algorithm, the R becomes smaller, until finally $R = \emptyset$, when the consistency of E is checked. After this the algorithm backtracks to the previous level in the search tree. The aim of the "for" loop is to prevent branching in the search (conditions (i) and (ii) and on the other hand to prune the search, if the partial solution E cannot possibly be completed to a consistent set (conditions (iii) and (iv)). The algorithm would work (like the abovementioned naive algorithm) even if the "for" loop were completely omitted.

Consider the conditions (i)–(iv) one at a time. The conditions (i) and (ii) detect "forced situations", in which the partial solution E must be extended in a particular way so that the end result could be consistent. In particular,

- condition (i) detects a situation, when the minesweeper reading $d(x, y)$ in square $(x, y) \in$ constraints corresponds exactly to the number of mines $|E \cap N(x, y)|$ already placed in the neighborhood $N(x, y)$. Then if there are squares in $N(x, y)$ that have not been decided yet, it is clear that these squares cannot contain mines, or the known mine detector reading would be $d(x, y)$ exceeded; on the other hand
- condition (ii) detects a situation, where the mine detector reading $d(x, y)$ in square $(x, y) \in$ constraints matches exactly the number of mines $|E \cap N(x, y)|$ already placed in $N(x, y)$ plus the number of squares in $N(x, y)$ that have not yet been fixed. Then clearly all squares in $R \cap N(x, y)$ must contain a mine in order to satisfy the reading $d(x, y)$ in the final solution.

Conditions (iii) and (iv) eliminate partial solutions that cannot possibly be completed to a consistent set.

- Condition (iii) detects a situation when too few mines have been placed in the neighborhood of (x, y), and the situation cannot be fixed, since all squares in $N(x, y)$ have already been fixed.
- Condition (iv) detects a situation when there are too many mines in the neighborhood of (x, y).

Problem * 21.

We use backtracking search to find all cliques. Solution (a set of vertices) x is feasible (i.e. maximal clique), if its vertices do not have common neighbours.

Let us first set $C_0 = V$. Then going through all elements of C_l in alphabetical order, the algorithm first successively chooses vertices 0, 1 and 2, which is a maximal clique. After that it backtracks to 0 and chooses 2. 0 and 2 is not maximal clique, so it backtracks in the beginning and starts with 1. The log of whole calculation is presented below:

0, 1, 2 (maximal!) (2 backtracks)
0, 2 (2 backtracks)

1, 2, 3, 4 (maximal!) (2 backtracks)
1, 2, 4 (2 backtracks)
1, 3, 4 (2 backtracks)
1, 4, 6 (maximal!) (3 backtracks)
2, 3, 4 (3 backtrack)
3, 4 (1 backtrack)
3, 5 (maximal!) (2 backtrack)
4, 6 (2 backtracks)
5 (backtrack)
6 (backtrack)

Problem ** 22.

Form a graph of n^2 vertices v_{ij}, each corresponding a one square. Add edges between each square, where 2 queens can be without threatening each other. A clique of size n of vertices $\{v_{ij}\}$ corresponds to n places in coordinates (i, j) where you can put queens safely.

Problem *** 23.

Let there be a set $S = \{b_{ij}|\{i,j\} \in E\} \cup \{w_{ij}|\{i,j\} \in E\}$. The b is for color black and w for color white. The sets used are $U_k^b = \{b_{ij}|i = k\}$ and $U_k^w = \{b_{kj}|\{k,j\} \in E\}$, which are used to denote the used color, and $L_{ij}^b = \{b_{ij}, w_{ji}|\{i,j\} \in E\}$ and $L_{ij}^w = \{w_{ij}, b_{ji}|\{i,j\} \in E\}$, to denote the two possible coloring of the ends of all edges. If in the correct coloring node i is colored black, then in the exact cover you can choose sets U_k^b and L_{ij}^b $\forall i, j \in E$ into the exact cover. This implies that in adjacent vertices you must use color white.

Because of symmetry of coloring, you can always choose first vertex to be colored in white. In a backtracking search let us go through all vertex colorings in a fixed order, such that the next vertex is adjacent to one or more which have been already colored. It can be easily seen that there is only one option left (if graph is connected), so if there is even one backtrack, it will backtrack all the way to the beginning, and the result will be that no 2-coloring exists. The time is then at most n for each choice of maximum n vertices, and then n for backtracking all of them. Similarly, since each element in S is in exactly two subsets, then when always choosing elements according to the fixed order induced by adjacent vertices as before, there is only one option, one subset to cover it.

Problem ** 24.

The 5-vertex cycle $\mathcal{C}_5$ (the pentagon) has chromatic number 3, but its largest clique has 2 vertices.

Problem ** 25. (a)

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{W} \subseteq \mathcal{V}$. The vertex set $\mathcal{W}$ is *independent*, if no two vertices in it are connected by an edge. A graph $\mathcal{G}$ is *bipartite)* if its vertex set $\mathcal{V}$ can be partitioned into two nonempty independent sets $\mathcal{W}$ and $\mathcal{V} - \mathcal{W}$.

A bipartite graph can always be colored by two colors, since one color for each independent set is enough.

A complete bipartite graph $(\mathcal{V}, \mathcal{E})$ contains all possible edges between its two independent sets, that is, $\mathcal{E} = \{\{u, v\} \mid u \in \mathcal{W}, v \in \mathcal{V} - \mathcal{W}\}$.

Construct a family of graphs as follows: Take the complete bipartite graph with n vertices in each independent set, and label the vertices such that the first independent set contains the odd-numbered vertices $\{1, 3, \ldots, 2n-1\}$, and the second one of the even-numbered vertices $\{2, 4, \ldots, 2n\}$. Remove from the complete bipartite graph the edges $\{1, 2\}, \{3, 4\}, \{5, 6\}, \ldots, \{2n-1, 2n\}$.

The greedy algorithm will color the graph as follows: Vertex 1 is colored with color 1. Vertex 2 is colored with color 1, since there is no edge $\{1, 2\}$ in the graph. Vertex 3 is colored with color 2, since there is the edge $\{2, 3\}$, so color 1 cannot be used. Vertex 4 is colored with color 2, since because of the edge $\{1, 4\}$ color 1 cannot be used. Vertex 5 is colored with color 3, since colors 1 and 2 cannot be used because of edges $\{2, 5\}$ and $\{4, 5\}$. Vertex 6 is colored with 3, since colors 1 and 2 cannot be used because of edges $\{1, 6\}$ and $\{3, 6\}$, etc.

In the general case the odd-numbered vertex $2k-1$ ($k \geq 2$) is colored with color k, as the edges $\{2j, 2k-1\}$, $1 \leq j \leq k-1$ prevent using colors less than k, as vertex $2j$ has already been colored with color j for all $1 \leq j \leq k-1$. On the other hand the even-numbered vertex $2k$ ($k \geq 2$) is colored with color k, as the edges $\{2j-1, 2k\}$, $1 \leq j \leq k-1$ prevent the use of colors less that k, as vertex $2j-1$ has been colored with color j for all $1 \leq j \leq k-1$.

The greedy coloring of this graph thus requires n colors in the end, even though the graph is 2-colorable.

Problem ** 25. (b)

It is clear that irrespective of the coloring, the vertices of a given color form an independent set. Suppose that we have an optimal coloring available. If we then list the vertices of the graph one color class at a time (say, first all red vertices, then blue ones, etc.), we get an ordering with which the greedy algorithm will produce an optimal coloring. The greedy coloring may not be identical to the coloring we started with, but the number of colors is the same.

Problem * 26. (a)

The most natural choice for the objective function is the length of the route. One possible neighborhood would be defined by using 2-OPT moves: replace two edges from the cycle that represents the current route by two other ones.

Problem * 26. (b)

Here we describe a so called fixed k method. A solution is any partitioning of the vertices into k parts (with empty parts allowed). Two colorings are neighbors of each other, if one can be obtained from the other by moving one vertex from one part to another. As the cost function we could take for example the number of edges with both endpoints in the same part of the partition. The neighborhood could be limited by only allowing moving vertices that appear in at least one edge with both endpoints of the same color; however, then the neighborhood is no longer symmetric.

Problem * 26. (c)

A simple approach: the solution space consists of all partitionings of the vertices into two equal-sized sets. A move is swapping two vertices between parts. The cost function is the number of edges between the parts.

Better results have been obtained with simulated annealing by allowing arbitrary partitions into two parts V_1 and V_2. Now the neighborhood would consist of moving one vertex from one part to the other, and the cost function would include the penalty term

$$\alpha(|V_1| - |V_2|)^2,$$

which increases the cost whenever the parts are not of the same size. Despite the

penalty term the optimization algorithm might end up at an unbalanced solution, at which point we can either increase α or use some simple heuristic to balance the solution by moving a few vertices.

Problem * 26. (d)

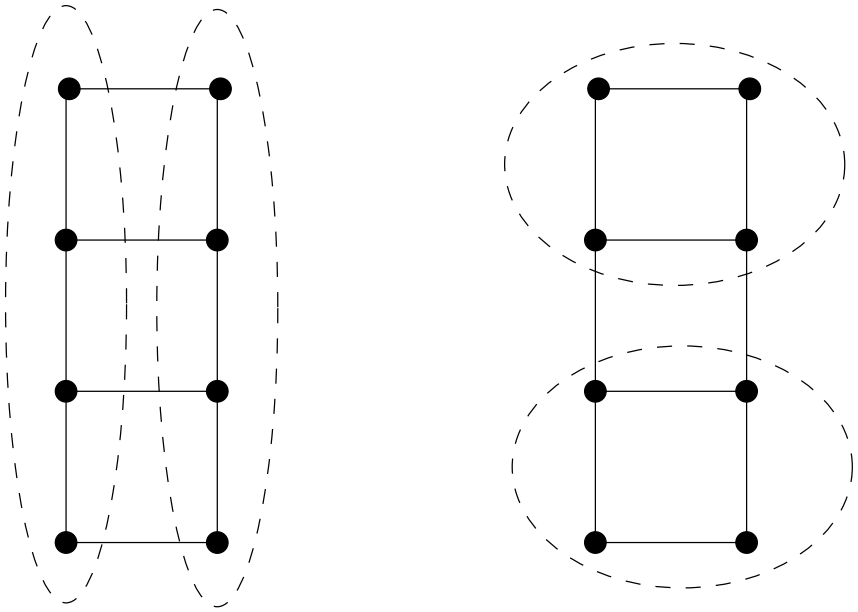
Take as the solution space all partitionings of the vertices into two parts. Define the k-neighborhood so that in one move at most k elements can be moved from one part to the other. As the cost function we may choose

$$\left| \sum_{a \in A_1} a - \sum_{a \in A_2} a \right| .$$

This problem is quite difficult in that it may have an extremely large number of local optima. Simulated annealing with the proposed neighborhood cannot cope with the Karmarkar–Karp heuristic devised for this problem, but better local search methods have been developed.

Problem ** 27.

On the left a graph partitioning that is a local optimum which we cannot improve by moving one vertex; on the right the global optimum, which is reachable by moving two vertices from one part to the other.



Problem ** 28.

The following is pseudocode for a simulated annealing algorithm for finding the maximum clique:

```
Find_Max_Clique(G, c_max, T_0, α)
    c = 0
    T = T_0
    Choose feasible X ∈ 𝒳
    X_best = X
    while c ≤ c_max
        Y = h_N(X)
        if Y ≠ Fail
            if |Y| > |X|
                X = Y
                if |X| > |X_best|
                    X_best = X
            else
                r = random(0, 1)
                if r < e^((|Y|−|X|)/T)
                    X = Y
        c = c + 1
        T = αT
    return X_best
```

Cooling down is linear, and α, c_{max} and T_0 must be suitably chosen. As the set $\mathcal{X}$ we have the set of subgraphs of G, and $X \in \mathcal{X}$ is feasible, if it is a clique (for all $x, y \in X, x \neq y$ there is an edge between x and y in X). As the objective function we have the size of the clique (the subgraph). As the neighborhood $N(X)$ of X we may choose (for example) those subgraphs of G that can be obtained by adding to or removing from X one vertex and its incident edges. As the heuristic h_N we could use for example "some feasible $Y \in N(X)$".

Problem ** 29. (a,b)

For part (a) we suppose that the given subset collection contains exactly n subsets $E_1, E_2, \ldots, E_n \subseteq F$, where F is the base set. The local search should now find $\{j_1, \ldots, j_w\} \subseteq \{1, \ldots, n\}$ so that $E_{j_1} \cup \cdots \cup E_{j_w} = F$ and $E_{j_s} \cap E_{j_t} = \emptyset$ always when $s \neq t$. In other words the collection $E_{j_1}, \ldots, E_{j_w}$ partitions F.

Choose as the solution space $\mathcal{X}$ all subsets of the set $\{1, \ldots, n\}$, and define the neighborhood N by the minimum change principle: a subset $x \in \mathcal{X}$ is a neighbor of $y \in \mathcal{X}$ if and only if y can be obtained from x by adding or removing one element. The elements of the search space can be identified with n-bit binary words in the obvious way.

For part (b), suppose that the graph to be colored has exactly n vertices, labeled $v_1, \ldots, v_n$. Take as the solution space all ways of choosing for each vertex one of the q colors. Note that in the general case such a choice of colors is not a feasible q-coloring. Then the color choices can be considered to be words of length n over a q-element alphabet: the ith character in the word indicates the color of v_i for all $i = 1, 2, \ldots, n$. The neighborhood can again be defined by the minimum change principle: colorings x and y are neighbors of each other if and only if they differ from each other in exactly one position.

The configuration graph corresponding to the neighborhoods in (a) and (b) is the following: The vertex set of the Hamming graph $H(n, q)$ consists of all words of length n over a q-element alphabet $\Sigma_q = \{0, 1, 2, \ldots, q-1\}$, that is, $\mathcal{V} = \Sigma_q^n$. Between the vertices $x, y \in \mathcal{V}$ there is an edge if and onlyif the words x and y differ at exactly one position.

The Hamming graphs $H(n, 2)$ for $n = 1, 2, 3, 4$ are given in Figure 3.

Problem ** 29. (c)

Suppose that there are n cities. We model the possible tours as permutations of the set $\{0, 1, 2, \ldots, n-1\}$. Thus the list representation of the permutation $\pi \in S_n = \mathcal{X}$, that is, $[\pi(0), \pi(1), \ldots, \pi(n-1)]$, tells the order in which the cities are visited: we start at city $\pi(0)$, then go to city $\pi(1)$, ..., until we return from city $\pi(n-1)$ back to city $\pi(0)$. We define the neighborhood so that two permutations $\pi_1, \pi_2 \in \mathcal{X}$ are neighbors of each other if and only if one is obtained from the other by one transposition (swapping the positions of two cities in the list representation of the permutation).

The configuration graph thus formed is the Cayley graph of the symmetric group generated by transpositions. The vertices of the Cayley graph $\Gamma(S_n, T_n)$ are all permutations $\pi \in S_n$ of the set $\{0, 1, 2, \ldots, n-1\}$. There is an edge between permutations $\pi_1, \pi_2 \in S_n$ if and only if $\pi_1^{-1}\pi_2 \in T_n$, where T_n is the set of all transpositions over the set $\{0, 1, 2, \ldots, n-1\}$. The Cayley graphs $\Gamma(S_n, T_n)$ for $n = 2, 3, 4$ are given in Figure 4.

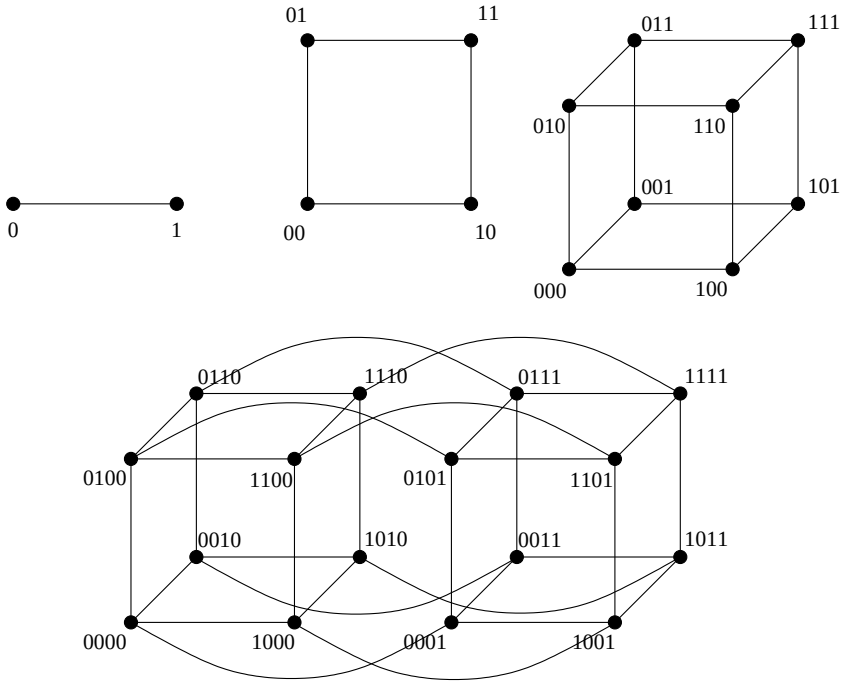


Figure 3: Hamming graphs $H(n, 2)$ for $n = 1, 2, 3, 4$.

Problem ** 29. (d,e)

Suppose for part (d) that the graph contains $2n$ vertices. We could take as the solution space $\mathcal{X}$ all n-subsets of the vertex set. Then the current partition is given by the n-subset and its complement. Similarly, for (e) we can choose as the search space $\mathcal{X}$ the k-subsets of the vertex set. The neighborhood can again be defined by the minimum change principle: two subsets are neighbors of each other if and only if one can be obtained from the other by adding one and removing one element.

The configuration graph formed is the Johnson graph $J(v, k)$: the vertices are all k-subsets of $\{1, 2, \ldots, v\}$; two subsets E_1, E_2 are connected by an edge if and only if $|E_1 \cap E_2| = k - 1$. Johnson graphs $J(4, 2)$ and $J(5, 3)$ are given in Figure 5.

Problem ** 30.

We examine the properties one graph class at a time.

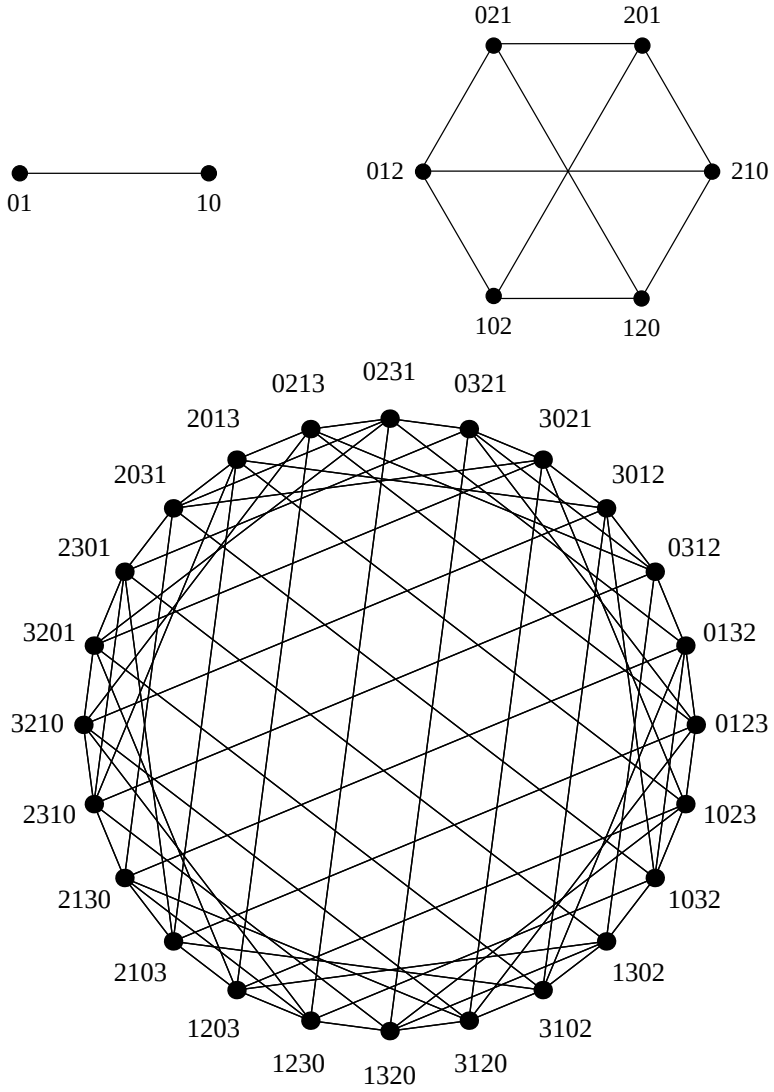


Figure 4: Cayley graphs $\Gamma(S_n, T_n)$ for $n = 2, 3, 4$.

1. The Hamming graph $H(n, q)$

   (a) The number of vertices is q^n.

   (b) Each vertex has $n(q-1)$ neighbors, as the coordinate to be changed can be chosen in n ways and the value for the coordinate in $q-1$ ways.

   (c) The Hamming graph is connected. From an arbitrary vertex we can reach any other vertex via the edges by changing the coordinate values

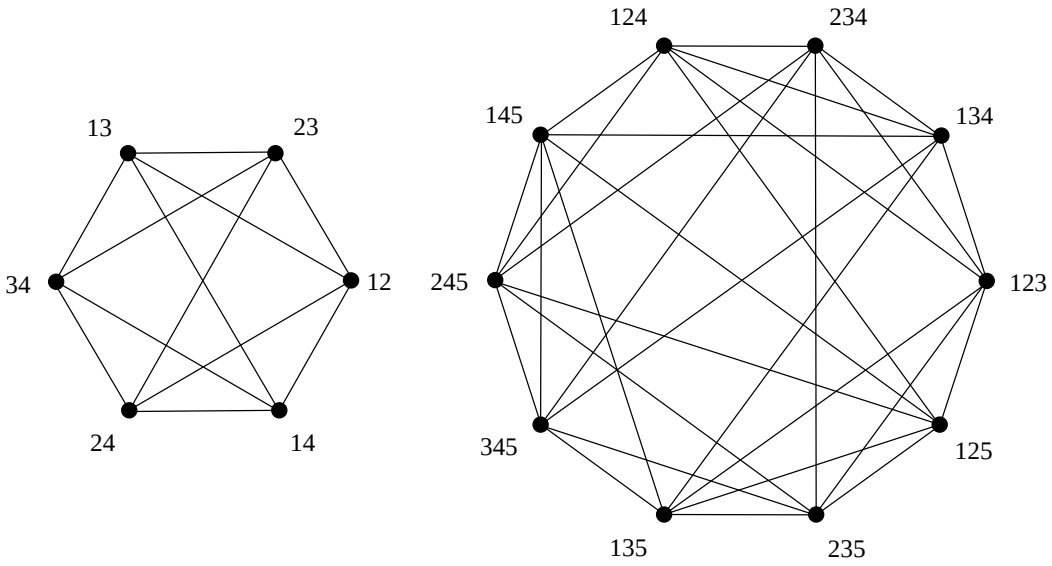


Figure 5: Johnson graphs $J(4,2)$ and $J(5,3)$.

of the first vertex one by one to the coordinate values of the second vertex.

(d) The length of the shortest cycle is 4 when $q = 2$, and 3 when $q > 2$. For $q = 2$ there exists a 4-cycle since two coordinates can be varied as 00, 01, 11, 10, 00; no 3-cycle can exist since graph is bipartite — the number of ones changes between even and odd on each move, so all cycles are of even length. For $q > 2$ a single coordinate can be varied as 0, 1, 2, 0.

(e) The length of the longest cycle is q^n, that is, there is a cycle that includes all vertices. For $q = 2$ the binary reflected Gray code G^n is an example of such a cycle. The construction of the binary reflected Gray code is easily generalized to other cases with even q. For odd q we need to slightly modify the construction: let $x_1, \ldots, x_N$ be the code of length n. Form the code of length $n + 1$ as follows:

- Repeat for all $0 \leq p \leq q - 3$:
  If p is even, output the partial code $px_1, px_2, \ldots, px_N$.
  If p is odd, output the partial code $px_N, px_{N-1}, \ldots, px_1$.
- Output the partial code $(q-2)x_N, (q-2)x_{N-1}, \ldots, (q-2)x_2$.
- Output the partial code $(q-1)x_2, (q-1)x_3, \ldots, (q-1)x_N$.
- Output the partial code $(q-1)x_1, (q-2)x_1$.

(f) The diameter of the graph is clearly n, since from any vertex we can

reach any other vertex by changing at most n coordinates. On the other hand n changes are necessary to get from $00\cdots0$ to $11\cdots1$.

2. The Cayley graph $\Gamma(S_n, T_n)$

   (a) Since there are $n!$ permutations of $\{0, 1, 2, \ldots, n-1\}$, there are $n!$ vertices.

   (b) There are $\binom{n}{2}$ transpositions of $\{0, 1, 2, \ldots, n-1\}$, so every vertex has that number of neighbors.

   (c) The graph is connected, since the transpositions generate the symmetric group.

   (d) For $n = 2$ there are no cycles. Otherwise the length of the shortest cycle is at most 4: the permutations

   $$\pi, \quad \pi(0,1), \quad \pi(0,1)(1,2), \quad \pi(0,1)(1,2)(0,1)$$

   form a 4-cycle, since $\pi(0,1)(1,2)(0,1)(0,2) = \pi$. On the other hand no 3-cycle can exist, or the three transpositions corresponding to the edges would combine to the identity permutation, which is impossible, as the identity is an even permutation.

   (e) The length of the longest cycle is $n!$, which can be obtained for example by the Trotter–Johnson minimum change ordering.

   (f) The diameter of the graph is $n-1$, since every permutation can be presented as the product of at most $n-1$ transpositions. On the other to get from the identity to the n cycle $(0, 1, 2, \ldots, n-1)$ takes at least $n-1$ transpositions.

3. The Johnson graph $J(v, k)$

   (a) The number of vertices equals the number of k-subsets of a v-set, that is, $\binom{v}{k}$.

   (b) There are k ways of removing an element from a given k-subset of a v-set, and there are $v-k$ ways of adding one. Thus every vertex has $k(v-k)$ neighbors.

   (c) The graph is connected, as it is obvious that by adding and removing one element at a time any k-subset can be changed to any other k-subset.

   (d) When $v = k$ or $v = 2$ there are no cycles. The length of the shortest cycle is clearly 3, as $\{x, y\}$, $\{x, z\}$, $\{y, z\}$ are neighbors of each other when $k \geq 2$. The case $k = 1$ is straightforward.

(e) The length of the longest cycle is $\binom{v}{k}$, which can be obtained e.g. by the revolving door minimum change order (pp. 48–52 in the book).

(f) The diameter of the graph is at most k, which is achieved when we can find two vertices with no elements in common. This is true when $v \geq 2k$. When $v < 2k$, any two k-subsets intersect in at least $2k - v$ points. Thus the diameter of the graph is $k - \max(2k - v, 0) = \min(k, v - k)$ in the general case.

Problem ** 31.

As in problem * 26., here too the solutions presented may not be the best ones possible in all cases. One should experiment with different approaches and take the one that works best with the problem instances being considered.

(a) For example: an added edge must not be removed during the next n iterations. A removed edge may not be replaced during the next m iterations. The parameters n and m need to be tuned depending on the problem instances.

(b) For example: if the color of a vertex has been changed from color v_i to color v_j, it must not be changed back to color v_i within the next n iterations. Alternatively: if the color of vertex v has been changed, it may not be changed again within the next n iterations. This limits the neighborhood selection more than the first condition.

(c) If vertices v_1 and v_2 have been swapped, then this pair must not be changed again within the next n iterations. Alternatively: if v has been moved from one part to another, then it may not be moved again within the next n iterations.

(d) When $k = 1$, we can use the tabu conditions of the previous point.

If $k > 1$, the situation is more complicated: we could use the same tabulist as above, but it might restrict the search too much. One possibility would be to consider the sum $S_1 = \sum_{a \in A_1} a$. If $S_1 = x$ after some move, we could consider such moves tabu for the next n iterations that would lead to $S_1 = x$. This will not work if S_1 can only take a few different values.

Problem * 32.

Let G be an arbitrary nonempty set, and let us define the binary operation $(g_1, g_2) \mapsto g_1$. The operation is clearly associative (condition (a)), as

$$(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot g_2 = g_1 = g_1 \cdot g_2 = g_1 \cdot (g_2 \cdot g_3).$$

Now take any arbitrary element of G as 1. Condition (b) holds, since clearly $g \cdot 1 = g$ for all $g \in G$. On the other hand condition (c) holds, as we may choose $g^{-1} := 1$ for all $g \in G$. Thus if there are at least 2 elements in G, the unity element would not be unique.

If condition (b) is changed to "there exists an element $1 \in G$, for which $1 \cdot g = g$ for all $g \in G$" the unity element can be shown to be unique: Choose an arbitrary $g \in G$ and use the notational short cuts $g_1 g_2 := g_1 \cdot g_2$, $g' := g^{-1}$ and $g'' := (g^{-1})^{-1}$. Now based on (a)–(c) we obtain

$$g1 = 1(g1) = (1g)1 = ((g''g')g)1 = (g''(g'g))1 = (g''1)1 =$$

$$= g''(11) = g''1 = g''(g'g) = (g''g')g = 1g = g,$$

so $1g = g1 = g$ for all $g \in G$. Thus, if there were some elements $1 \in G$ and $1' \in G$, both of which would satisfy (b) and (c), we could deduce that $1' = 11' = 1$. After the unity element has been shown to be unique, we can show that the left inverse of an element is also its right inverse by using the axioms (a)–(c):

$$gg' = g(1g') = (g1)g' = (g(g'g))g' = ((gg')g)g' = (gg')(gg'),$$

so we must have $gg' = 1$. Now the uniqueness of the inverse element can be shown as follows:

$$g' = g'1 = g'(g\hat{g}') = (g'g)\hat{g}' = 1\hat{g}' = \hat{g}'.$$

Problem ** 33.

If H is a subgroup of the finite group G and $\{g_1, \ldots, g_n\}$ a left transversal of H in G, then by definition the set $\{g_1H, \ldots, g_nH\}$, where $g_iH := \{g_ih \mid h \in H\}$ for all $i = 1, \ldots, n$, is a partitioning of G into nonempty parts. Now choose some $g \in G$. Since the sets g_iH partition G, there exists a unique i, for which $g \in g_iH$. Further there exists a unique $h \in H$ for which $g = g_ih$, since if $g_ih = g = g_ih'$, we can obtain

$$h = 1h = (g_i^{-1}g_i)h = g_i^{-1}(g_ih) = g_i^{-1}(g_ih') = (g_i^{-1}g_i)h' = 1h' = h'.$$

Therefore every $g \in G$ can be uniquely represented in the form $g_i h$, with $h \in H$. Similarly we can obtain that the $h \in H$ in the previous representation can be uniquely representedin the form $h = h_j k$, where $k \in K$ and $h_j \in \{h_1, \ldots, h_m\}$. Clearly we could continue in this manner if K would have a subgroup, etc.

The previous idea is useful for example when we want to manipulate finite groups on a computer. Then a group may be considered as a sequence of transversals with respect to a certain subgroup chain. For example the subgroup chain in the Schreier–Sims representation (Section 6.2.3 of the book) consists of nested point stabilizer subgroups.

Problem ** 34. (a)

Since α maps 0 onto 1, we look for an $h_0 \in \mathcal{U}_0$, that maps 0 onto 1. One is found: $h_0 = (0,1,3,6)(2,5,9,7)(4,8)$. By multiplying α with its inverse we obtain $h_0^{-1}\alpha = (0)(1,7,9,5,3,8,4,2)(6)$. Since here 1 maps onto 7, we look for an $h_1 \in \mathcal{U}_1$, that maps 1 onto 7. One is found: $h_1 = (1,7,3,2,6,4)(5,8,9)$. Now $h_1^{-1}h_0^{-1}\alpha = (0)(1)(2,4,3,5,7,8,6)(9)$. Since here 2 maps onto 4, we look for an $h_2 \in \mathcal{U}_2$, that maps 2 onto 4. Such a h_2 does not exist, so $\alpha \notin G$.

Problem ** 34. (b)

Since β maps 0 onto 1, we look for an $h_0 \in \mathcal{U}_0$, that maps 0 onto 1. One is found: $h_0 = (0,1,3,6)(2,5,9,7)(4,8)$. By multiplying β with its inverse we obtain $h_0^{-1}\beta = (0)(1,7,4,6,9,2)(3,8,5)$. Since here 1 maps onto 7, we look for an $h_1 \in \mathcal{U}_1$ that maps 1 onto 7. One is found: $h_1 = (1,7,3,2,6,4)(5,8,9)$. Now $h_1^{-1}h_0^{-1}\beta = (0)(1)(2,4)(3,5,7,6,8,9)$. Since here 2 maps onto 4, we look for an $h_2 \in \mathcal{U}_2$ that maps 2 onto 4. No such h_2 is found, so $\beta \notin G$.

Problem ** 34. (c)

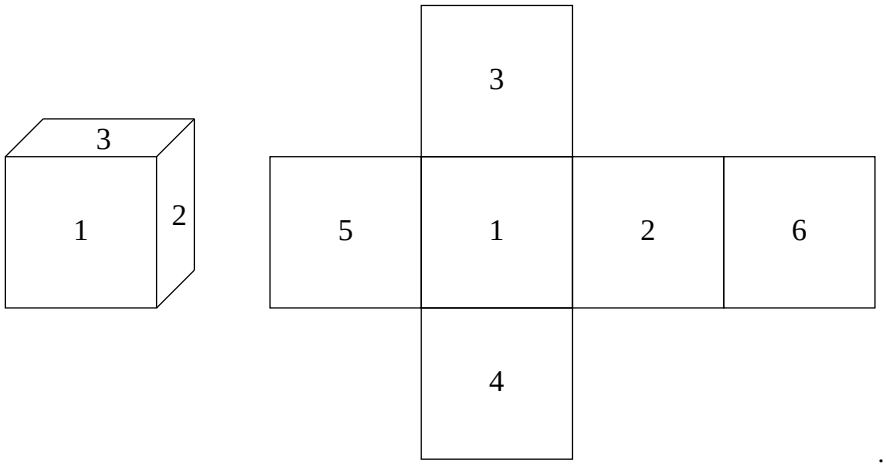
Since γ maps 0 onto 3, we look for an $h_0 \in \mathcal{U}_0$ that maps 0 onto 3. One is found: $h_0 = (0,3)(1,6)(2,9)(5,7)$. By multiplying γ with its inverse we obtain $h_0^{-1}\gamma = (0)(1,2)(3,7)(4,6)(5,8)(9)$. As here 1 maps onto 2, we look for an $h_1 \in \mathcal{U}_1$, that maps 1 onto 2. One is found: $h_1 = (1,2)(3,4)(6,7)$. Now $h_1^{-1}h_0^{-1}\gamma = (0)(1)(2)(3,6)(4,7)(5,8)(9)$. Here 2 maps onto 2; we look for an $h_2 \in \mathcal{U}_2$, that maps 2 onto 2. One is found: $h_2 = \mathbf{I}$. Now $h_2^{-1}h_1^{-1}h_0^{-1}\gamma = (0)(1)(2)(3,6)(4,7)(5,8)(9)$. Since here 3 maps onto 6, we look

for an $h_3 \in \mathcal{U}_3$ that maps 3 onto 6. One is found: $h_3 = (3,6)(4,7)(5,8)$. Now $h_3^{-1}h_2^{-1}h_1^{-1}h_0^{-1}\gamma = \mathbf{I}$. This maps all elements onto themselves, and kuvaa kaikki alkiot itselleen, ja $\mathbf{I} \in \mathcal{U}_n$ for all $n \geq 4$, so $\gamma \in G$.

The order $|G|$ of the group G is $|\mathcal{U}_0| \cdot |\mathcal{U}_1| \cdot |\mathcal{U}_2| \cdot |\mathcal{U}_3| \cdot \ldots \cdot |\mathcal{U}_9| = 10 \cdot 6 \cdot 2 = 120$.

Problem ** 35.

First label the six sides of the die in some way (for example, top, bottom, left, right, front, back); here we choose to use the integers 1, 2, 3, 4, 5, 6 as below:



.

Now any way of labeling the sides of the die with the integers 1, 2, 3, 4, 5, 6 can be expressed as a permutation π of $\{1,2,3,4,5,6\}$: the number of side i is $\pi(i)$ for all $i = 1,2,3,4,5,6$.

Due to the symmetry of the die essentially the same labeling appears more than once. For example

$$\pi(1) = 1, \quad \pi(2) = 2, \quad \pi(3) = 3, \quad \pi(4) = 4, \quad \pi(5) = 5, \quad \pi(6) = 6$$

is the same die as

$$\pi'(1) = 2, \quad \pi'(2) = 6, \quad \pi'(3) = 3, \quad \pi'(4) = 4, \quad \pi'(5) = 1, \quad \pi'(6) = 5,$$

as π' is obtained from π by rotating the cube 90 degrees around the axis passing through the midpoints of sides 3 and 4. This corresponds to permuting the names of the sides by $\gamma = (1,5,6,2)$. (Side i rotates to become side $\gamma(i)$ for all $i = 1,2,3,4,5,6$.)

The permutations $(1,5,6,2)$ and $(1,4,6,3)$ generate all the $6 \cdot 4 = 24$ rotational symmetries of the cube:

$$G = \left\{ \begin{array}{lll} \mathbf{I}, & (2,3,5,4), & (2,4,5,3), \\ (2,5)(3,4), & (1,2)(3,4)(5,6), & (1,2,3)(4,6,5), \\ (1,2,4)(3,6,5), & (1,2,6,5), & (1,3,2)(4,5,6), \\ (1,3,6,4), & (1,3)(2,5)(4,6), & (1,3,5)(2,6,4), \\ (1,4,2)(3,5,6), & (1,4,6,3), & (1,4)(2,5)(3,6), \\ (1,4,5)(2,6,3), & (1,5,6,2), & (1,5,4)(2,3,6), \\ (1,5,3)(2,4,6), & (1,5)(2,6)(3,4), & (1,6)(3,4), \\ (1,6)(2,3)(4,5), & (1,6)(2,4)(3,5), & (1,6)(2,5) \end{array} \right\}.$$

Two labelings π and π' are the same up to symmetry if and only if there exists a $\gamma \in G$ such that $\pi' = \pi\gamma^{-1}$. This equivalent with the condition $\pi^{-1}\pi' \in G$, so labelings π and π' are the same if and only if they belong to the same left coset of G in the symmetric group $\mathsf{Sym}(\{1,2,3,4,5,6\})$.

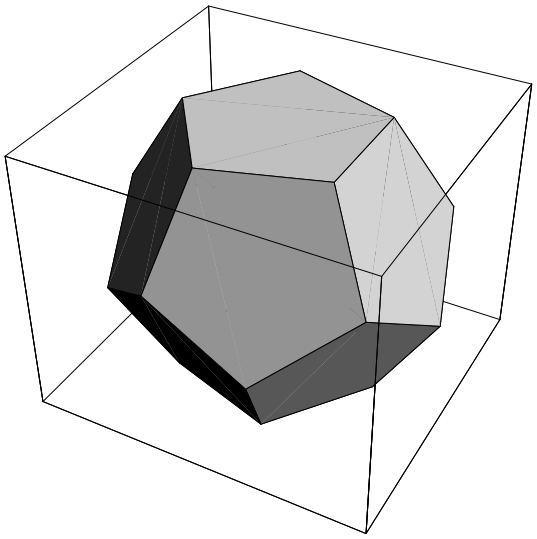
The number of cosets of G in $\mathsf{Sym}(\{1,2,3,4,5,6\})$ can be determined by Lagrange's Theorem (pp. 193, Theorem 6.2 in the book) to be

$$\frac{|\mathsf{Sym}(\{1,2,3,4,5,6\})|}{|G|} = \frac{6!}{24} = \frac{720}{24} = 30.$$

A left transversal of the cosets can be determined by for example Algorithm 6.17 on p. 224 of the book; it computes exactly the left transversal, although this is not explicitly mentioned in the book.

$$\begin{array}{llll} [1,2,3,4,5,6], & [1,2,3,4,6,5], & [1,2,3,5,4,6], & [1,2,3,6,4,5], \\ [1,2,3,5,6,4], & [1,2,3,6,5,4], & [1,2,4,3,5,6], & [1,2,4,3,6,5], \\ [1,2,5,3,4,6], & [1,2,6,3,4,5], & [1,2,5,3,6,4], & [1,2,6,3,5,4], \\ [1,2,4,5,3,6], & [1,2,4,6,3,5], & [1,2,5,4,3,6], & [1,2,6,4,3,5], \\ [1,2,5,6,3,4], & [1,2,6,5,3,4], & [1,2,4,5,6,3], & [1,2,4,6,5,3], \\ [1,2,5,4,6,3], & [1,2,6,4,5,3], & [1,2,5,6,4,3], & [1,2,6,5,4,3], \\ [1,6,2,3,4,5], & [1,6,2,3,5,4], & [1,6,2,4,3,5], & [1,6,2,5,3,4] \\ [1,6,2,4,5,3], & [1,6,2,5,4,3]. & & \end{array}$$

The dodecahedron consists of 12 pentagons (figure below) and the order of its rotational automorphism group is $12 \cdot 5 = 60$. (Choose a side and one of its neighbors. Any of the 12 sides can be rotated to the place of the first side chosen, and after fixing that, a neighboring side can be chosen from among the 5 neighbors. Two adjacent fixed sides uniquely determine the rotation.)



For the dodecahedron, the number of distinct labelings is thus

$$\frac{12!}{60} = \frac{479001600}{60} = 7983360.$$

Problem ** 36.

Let P be the partition of $X = \{1, 2, \ldots, 19\}$ given by

$$\{\{1,2,3,4,5\},\{6,7,8,9,10\},\{11,12,13,14,15\},\{16,17,18,19\}\}.$$

Clearly, all partitionings of X that satisfy the requirements can be formed from P by relabeling the elements. That is, the set of all partitionings that satisfy the requirements is the orbit of P under the group $G = \mathsf{Sym}(X)$, where permutation π acts on P by relabeling its points. For example

$$P' = \{\{1,3,4,5,7\},\{2,6,8,10,12\},\{9,11,13,14,15\},\{16,17,18,19\}\}$$

is obtained from P by swapping the points 2 and 7 and the points 9 and 12 with each other, that is, $P' = \pi(P)$, where $\pi = (2,7)(9,12)$.

By the orbit stabilizer theorem (p. 213, Lemma 6.9 in the book) we find the length of the orbit $G(P)$ to be

$$|G(P)| = \frac{|G|}{|G_P|},$$

where G_P is the subgroup of G that consists of all permutations that stabilize P, that is,

$$G_P = \{\pi \in G \mid \pi(P) = P\}.$$

Now $|G_P| = 3!(5!)^3 4!$, since the contents of the 4- and 5-subsets in P in P can be arbitrarily permuted without changing P; additionally there are 3! ways of ordering the 5-subsets. The number of distinct partitionings is thus

$$|G(P)| = \frac{19!}{3!(5!)^3 4!} = \frac{121645100408832000}{248832000} = 488864376.$$

Problem ** 37. (a)

The automorphism group of the given square is isomorphic to the dihedral group D_8. The group consists of the permutations

$$\begin{aligned}
g_0 &= (0)(1)(2)(3)(4)(5)(6)(7)(8)\\
g_1 &= (0,2,8,6)(1,5,7,3)(4)\\
g_2 &= (0,8)(1,7)(2,6)(5,3)(4)\\
g_3 &= (0,6,8,2)(1,3,7,5)(4)\\
g_4 &= (0,2)(3,5)(6,8)(1)(4)(7)\\
g_5 &= (1,5)(0,8)(3,7)(2)(4)(6)\\
g_6 &= (0,6)(1,7)(2,8)(3)(4)(5)\\
g_7 &= (1,3)(2,6)(5,7)(0)(4)(8).
\end{aligned}$$

The permutations are of types

$$\begin{aligned}
\text{type}(g_0) &= (9,0,0,0,0,0,0,0,0)\\
\text{type}(g_{1,3}) &= (1,0,0,2,0,0,0,0,0)\\
\text{type}(g_2) &= (1,4,0,0,0,0,0,0,0)\\
\text{type}(g_{4,5,6,7}) &= (3,3,0,0,0,0,0,0,0).
\end{aligned}$$

How many 5-subsets does each permutation map onto itself? If an element of a cycle in the permutation belongs to the subset, then all elements in the cycle must belong to the subset for the permutation to map the subset onto itself. So we compute the number of ways of choosing cycles from the permutations so that the sum of the lengths of the cycles is 5.

$$\begin{aligned}
\chi_5(g_0) &= \binom{9}{5} = 126 \\
\chi_5(g_{1,3}) &= \binom{2}{1} = 2 \\
\chi_5(g_2) &= \binom{4}{2} = 6 \\
\chi_5(g_{4,5,6,7}) &= \binom{3}{1}\binom{3}{2} + \binom{3}{3}\binom{3}{1} = 12
\end{aligned}$$

By Burnside's Lemma (p. 215, Theorem 6.10 in the book)

$$N_k = \frac{1}{|G|}\sum_{g\in G}\chi_k(g),$$

and we obtain $N_5 = \frac{1}{8}(126 + 2\cdot 2 + 6 + 4\cdot 12) = 23$.

Problem ** 37. (b)

We construct sets of representatives of k-subset orbits R_k for $0 \le k \le 2$. The set R_2 will be the answer to the problem.

$R_0 = \{\emptyset\}$, since the empty set is the only 0-element subset of the 9-element set. Now we add to each element in R_0 in turn each of the elements $\{0,\ldots,8\}$ in turn, and we append the result to R_1 unless R_1 already contains a 1-subset from the same orbit that precedes the result.

$$R_1 = \{\{0\},\{1\},\overbrace{\{2\}}^{\{0\}},\overbrace{\{3\}}^{\{1\}},\{4\},\overbrace{\{5\}}^{\{1\}},\overbrace{\{6\}}^{\{0\}},\overbrace{\{7\}}^{\{1\}},\overbrace{\{8\}}^{\{0\}}\} = \{\{0\},\{1\},\{4\}\}$$

Now we take in turn each element in R_1 and append each element in turn that is larger than all elements already in the set. We append the result to R_2 unless it already contains an earlier 2-subset from the same orbit.

$$\begin{aligned}
R_2 = \{\ & \{0,1\},\{0,2\},\overbrace{\{0,3\}}^{\{0,1\}},\{0,4\},\{0,5\},\overbrace{\{0,6\}}^{\{0,2\}},\overbrace{\{0,7\}}^{\{0,5\}},\{0,8\},\\
& \overbrace{\{1,2\}}^{\{0,1\}},\{1,3\},\{1,4\},\overbrace{\{1,5\}}^{\{1,3\}},\overbrace{\{1,6\}}^{\{0,5\}},\{1,7\},\overbrace{\{1,8\}}^{\{0,5\}},\\
& \overbrace{\{4,5\}}^{\{1,4\}},\overbrace{\{4,6\}}^{\{0,4\}},\overbrace{\{4,7\}}^{\{1,4\}},\overbrace{\{4,8\}}^{\{0,4\}}\ \}
\end{aligned}$$

so we have

$$R_2 = \{\{0,1\},\{0,2\},\{0,4\},\{0,5\},\{0,8\},\{1,3\},\{1,4\},\{1,7\}\}.$$

Note: Above in constructing R_{k+1} it suffices to only consider adding to each element of R_k elements that are larger than the elements already in there, since in this version of the algorithm an orbit is always represented by the lexicographical minimum element in it. If the $(k+1)$-subset $S \in R_{k+1}$ is the lexicographical minimum element in its orbit, it holds that also the set $S' = S \setminus \{\max(S)\}$ is the lexicographical minimum element of its orbit, so $S' \in R_k$. From this it follows that each element in R_{k+1} can be constructed from some element in R_k by adding an element that comes after the elements already in the set.

Problem ** 38.

The automorphism group $\mathsf{Aut}(\mathcal{G})$ of the graph $\mathcal{G}$ contains 48 elements:

I	$(0,2,3,1)(4,6,7,5)$	$(0,5,3)(2,4,7)$
$(2,4)(3,5)$	$(0,2,6,7,5,1)(3,4)$	$(0,5)(2,7)$
$(1,2)(5,6)$	$(0,3,6)(1,7,4)$	$(0,5,3,6)(1,7,2,4)$
$(1,2,4)(3,6,5)$	$(0,3,5,6)(1,7,4,2)$	$(0,5,6)(1,7,2)$
$(1,4,2)(3,5,6)$	$(0,3,6,5)(1,2,7,4)$	$(0,6,3)(1,4,7)$
$(1,4)(3,6)$	$(0,3)(1,2)(4,7)(5,6)$	$(0,6,3,5)(1,4,2,7)$
$(0,1,3,7,6,4)(2,5)$	$(0,3,5)(2,7,4)$	$(0,6,5,3)(1,2,4,7)$
$(0,1,3,2)(4,5,7,6)$	$(0,3)(4,7)$	$(0,6,5)(1,2,7)$
$(0,1,5,4)(2,3,7,6)$	$(0,4,6,7,3,1)(2,5)$	$(0,6)(1,7)(2,4)(3,5)$
$(0,1,5,7,6,2)(3,4)$	$(0,4,5,1)(2,6,7,3)$	$(0,6)(1,7)$
$(0,1)(2,3)(4,5)(6,7)$	$(0,4,6,2)(1,5,7,3)$	$(0,7)(1,6)(2,5)(3,4)$
$(0,1)(2,5)(3,4)(6,7)$	$(0,4)(1,5)(2,6)(3,7)$	$(0,7)(1,6)(2,3)(4,5)$
$(0,2,6,4)(1,3,7,5)$	$(0,4,5,7,3,2)(1,6)$	$(0,7)(1,5)(2,6)(3,4)$
$(0,2)(1,3)(4,6)(5,7)$	$(0,4)(1,6)(2,5)(3,7)$	$(0,7)(1,5,4,6,2,3)$
$(0,2,3,7,5,4)(1,6)$	$(0,5,6,3)(1,4,7,2)$	$(0,7)(1,3,2,6,4,5)$
$(0,2)(1,6)(3,4)(5,7)$	$(0,5)(1,4)(2,7)(3,6)$	$(0,7)(1,3)(2,5)(4,6)$

This group is generated by the permutations $(0,1,3,7,6,4)(2,5)$ and $(0,1,3,2)(4,5,7,6)$. The stabilizer subgroups can be determined either algorithmically by first generating the Schreier–Sims representation for the group (note that there is a typo in Algorithm 6.9 of the book; see the errata at `http://www.math.mtu.edu/~kreher/cages.html`) and then using Algorithm 6.6 to

test for each element whether they map $\{0,7\}$ and $\{0,1,2,3\}$ onto themselves. On the other hand, at least the stabilizers of $\{0,7\}$,

I	$(1,4,2)(3,5,6)$	$(0,7)(1,5)(2,6)(3,4)$
$(2,4)(3,5)$	$(1,4)(3,6)$	$(0,7)(1,5,4,6,2,3)$
$(1,2)(5,6)$	$(0,7)(1,6)(2,5)(3,4)$	$(0,7)(1,3,2,6,4,5)$
$(1,2,4)(3,6,5)$	$(0,7)(1,6)(2,3)(4,5)$	$(0,7)(1,3)(2,5)(4,6)$,

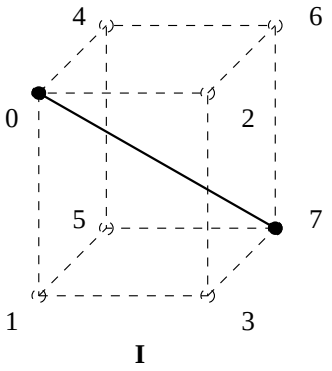
are easy to pick out from the list of all permutations. Here the stabilizer subgroup $\mathsf{Aut}(\mathcal{G})_{\{0,7\}}$ is of order 12, so the length of the orbit of $\{0,7\}$ and on the other hand the size of the left transversal of the stabilizer is

$$|\mathsf{Aut}(\mathcal{G})|/|\mathsf{Aut}(\mathcal{G})_{\{0,7\}}| = 48/12 = 4.$$

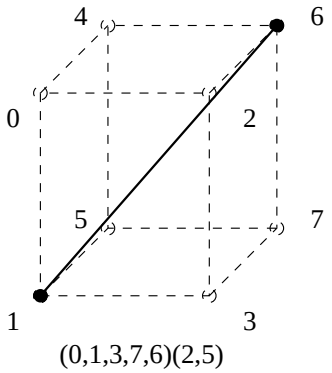
The left transversal could be computed by Algorithm 6.17 of the book, which computes the left transversal although this is not explicitly mentioned in the text. One left transversal is

$$g_1 = \mathbf{I} \qquad g_3 = (0,2,6,4)(1,3,7,5)$$
$$g_2 = (0,1,3,7,6,4)(2,5) \qquad g_4 = (0,3,6)(1,7,4) \quad .$$

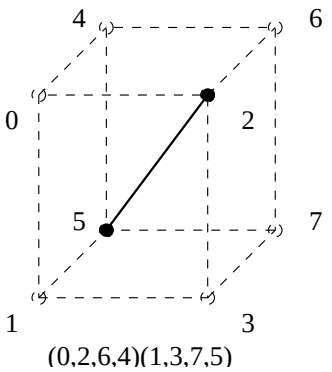
The orbit of $\{0,7\}$ and the corresponding transversal are given below.



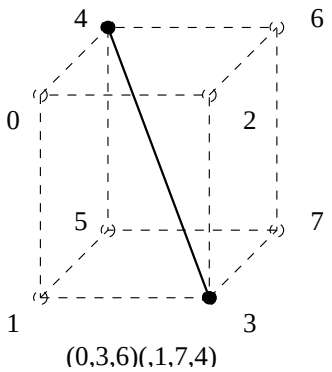
I



(0,1,3,7,6)(2,5)



(0,2,6,4)(1,3,7,5)



(0,3,6)(,1,7,4)

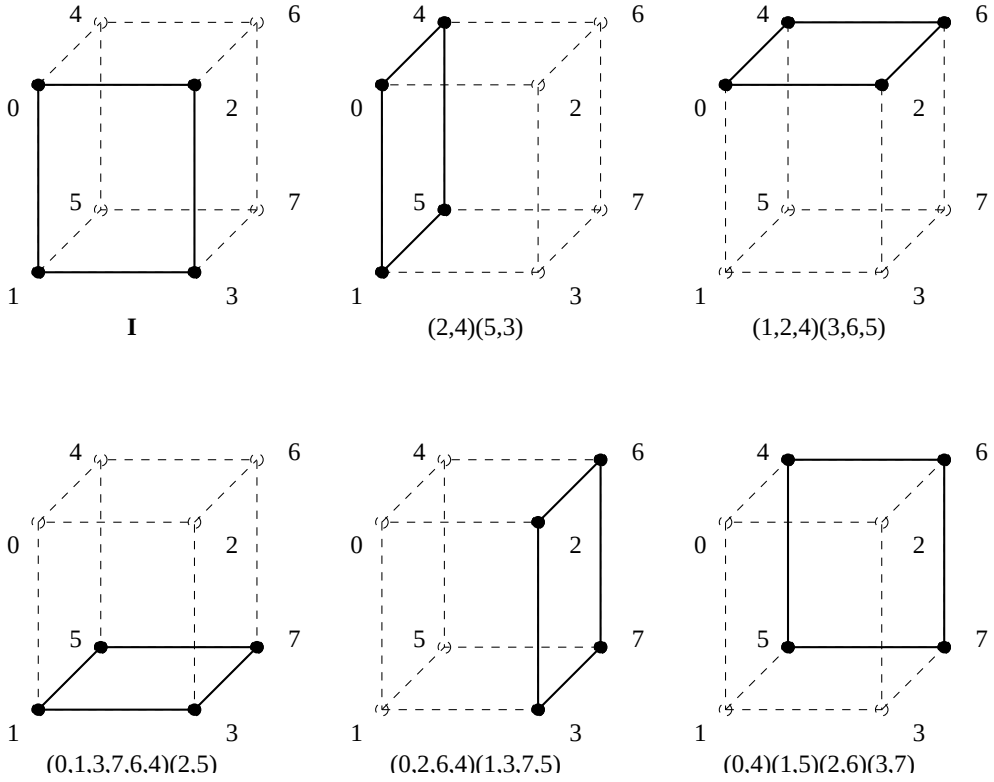
Similarly the stabilizer subgroup $\mathsf{Aut}(\mathcal{G})_{\{0,1,2,3\}}$ of $\{0,1,2,3\}$ consists of the permutations

$$\begin{array}{ll} \mathbf{I} & (0,2)(1,3)(4,6)(5,7) \\ (1,2)(5,6) & (0,2,3,1)(4,6,7,5) \\ (0,1,3,2)(4,5,7,6) & (0,3)(1,2)(4,7)(5,6) \\ (0,1)(2,3)(4,5)(6,7) & (0,3)(4,7) \end{array} ,$$

so the orbit length of the subset $\{0,1,2,3\}$ and also the size of the left transversal of $\mathsf{Aut}(\mathcal{G})_{\{0,1,2,3\}}$ is $48/8 = 6$. One left transversal consists of the permutations

$$\begin{array}{lll} g_1 = \mathbf{I} & g_3 = (1,2,4)(3,6,5) & g_5 = (0,2,6,4)(1,3,7,5) \\ g_2 = (2,4)(5,3) & g_4 = (0,1,3,7,6,4)(2,5) & g_6 = (0,4)(1,5)(2,6)(3,7) \end{array} .$$

The orbit of the subset and the corresponding transversal elements are shown below.



Problem * 39.

The graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ are isomorphic (denoted by $\mathcal{G}_1 \cong \mathcal{G}_2$) if there is a bijection $\pi : \mathcal{V}_1 \to \mathcal{V}_2$ such that for all $u, v \in \mathcal{V}_1$ it holds that $\{u, v\} \in \mathcal{E}_1$ if and only if $\{\pi(u), \pi(v)\} \in \mathcal{E}_2$. The bijection π is an isomorphism.

Let $\mathcal{F}$ be a family of graphs and X be some set. The mapping $\Phi : \mathcal{F} \to X$ is *invariant* for the family $\mathcal{F}$ if for all $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{F}$ it holds that $\Phi(\mathcal{G}_1) = \Phi(\mathcal{G}_2)$ always when $\mathcal{G}_1 \cong \mathcal{G}_2$. Thus, if $\Phi(\mathcal{G}_1) \neq \Phi(\mathcal{G}_2)$ then we must have $\mathcal{G}_1 \ncong \mathcal{G}_2$.

To solve the problem it therefore suffices to find an invariant that can distinguish the given graphs.

(a) We can choose, for example, the list of the degrees of the vertices (in ascending order). The isomorphism π must map the degree list $[\deg(v_1), \ldots, \deg(v_n)]$, when $\deg(v_1) \leq \deg(v_2) \leq \cdots \leq \deg(v_n)$ onto the corresponding list $[\deg(\pi(v_1)), \ldots, \deg(\pi(v_n))]$ so that $\deg(\pi(v_1)) \leq \deg(\pi(v_2)) \leq \cdots \leq \deg(\pi(v_n))$. (Obviously two graphs must have the same number of vertices to be isomorphic.) The degree list of the graph on the left is $[2, 3, 3, 3, 3]$ and the degree list of the graph on the right is $[2, 2, 3, 3, 4]$. Therefore the graphs are nonisomorphic.

(b) We can consider the number of triangles in the graph: an isomorphism π must map an arbitrary triangle $\{\{u, v\}, \{u, w\}, \{v, w\}\} \subseteq \mathcal{E}_1$ onto the corresponding triangle $\{\{\pi(u), \pi(v)\}, \{\pi(u), \pi(w)\}, \{\pi(v), \pi(w)\}\} \subseteq \mathcal{E}_2$, so two isomorphic graphs $\mathcal{G}_1, \mathcal{G}_2$ must contain the same number of triangles. Now we observe that the graph on the left contains no triangles while there are two in the graph on the right. Thus the graphs cannot be isomorphic.

Problem ** 40.

We take as the symmetry group of K_8 the dihedral group D_8, which is generated by the permutations $(0, 1, 2, 3, 4, 5, 6, 7)$ and $(0, 7)(1, 6)(2, 5)(3, 4)$. (The cyclic group C_8 is a subgroup of D_8 so a D_8-symmetric coloring is also C_8-symmetric.) Symmetricity of a coloring means that edges of the same color map onto each other when an element of the symmetry group acts on the vectices, that is, the edges in each D_8-orbit must be of the same color.

The D_8-orbits of the edge set of K_8 are (see also Figure 6):

$$\Delta_1 = \{\{0, 1\}, \{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, [5, 6\}, \{6, 7\}, \{0, 7\}\},$$
$$\Delta_2 = \{\{0, 2\}, \{2, 4\}, \{4, 6\}, \{0, 6\}, \{1, 3\}, \{3, 5\}, \{5, 7\}, \{1, 7\}\},$$
$$\Delta_3 = \{\{0, 3\}, \{0, 5\}, \{1, 4\}, \{1, 6\}, \{2, 5\}, \{2, 7\}, \{3, 6\}, \{4, 7\}\},$$
$$\Delta_4 = \{\{0, 4\}, \{1, 5\}, \{2, 6\}, \{3, 7\}\}.$$

There are 5 D_8-orbits $\Gamma_1, \ldots, \Gamma_5$ of 3-subsets, and their lexicographically mini-

mum representatives are

$$\gamma_1 = \{0,1,2\},\ \gamma_2 = \{0,1,3\},\ \gamma_3 = \{0,1,4\},\ \gamma_4 = \{0,2,4\},\ \gamma_5 = \{0,2,5\}.$$

Similarly there are 8 D_8-orbits $\Sigma_1, \ldots, \Sigma_8$ of 4-subsets, and their lexicographically minimum representatives are

$$\sigma_1 = \{0,1,2,3\},\ \sigma_2 = \{0,1,2,4\},\ \sigma_3 = \{0,1,2,5\},$$
$$\sigma_4 = \{0,1,3,4\},\ \sigma_5 = \{0,1,3,5\},\ \sigma_6 = \{0,1,3,6\},$$
$$\sigma_7 = \{0,1,4,5\},\ \sigma_8 = \{0,2,4,6\}.$$

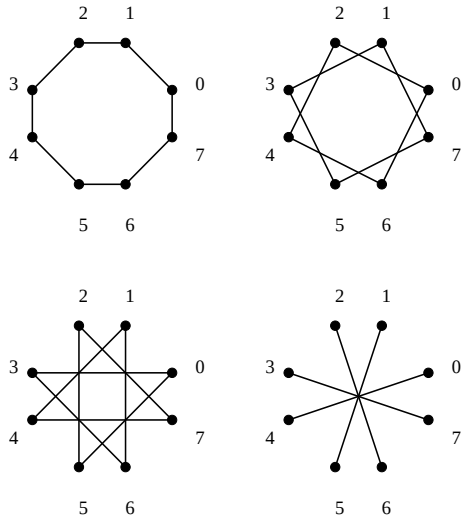


Figure 6: The D_8-orbits of the edges of K_8

The orbit indicence matrix of the 2-subset and 3-subset orbits is

	Γ_1	Γ_2	Γ_3	Γ_4	Γ_5
Δ_1	2	2	2	0	0
Δ_2	1	2	0	2	1
Δ_3	0	2	2	0	2
Δ_4	0	0	4	2	0

.

Similarly the orbit incidence matrix of 2-subset and 4-subset orbits is

	Σ_1	Σ_2	Σ_3	Σ_4	Σ_5	Σ_6	Σ_7	Σ_8
Δ_1	3	4	2	2	2	1	1	0
Δ_2	2	4	1	1	4	2	0	1
Δ_3	1	2	2	2	4	3	1	0
Δ_4	0	4	2	2	4	0	2	1

.

In the orbit incidence matrix the element on row i in column j tells how many elements of the orbit Γ_j (similarly Σ_j) contain a chosen element of the orbit Δ_i. For example the edge $\{0,1\}$ from orbit Δ_1 is contained in exactly two 3-subsets of orbit Γ_1, that is, $\{0,1,2\}$ and $\{0,1,6\}$.

The desired edge coloring can now be found by using orbit incidence matrices. Clearly every 3-subset represents a triangle of three edges, and each 4-subset represents a 4-clique with $\binom{4}{2} = 6$ edges. If some triangle appears in the edge coloring with color 1, by symmetry of the coloring all triangles in the same orbit Γ_j are colored with color 1. Thus then all edges incident with triangles (3-subsets) in Γ_j must be colored with color 1. By the structure of the orbit incidence matrix this is clearly possible exactly when the orbits colored with color 1 include all orbits Δ_i with a nonzero value in column j of the orbit incidence matrix.

Thus we can color the orbit sets $\{\Delta_1\}$, $\{\Delta_2\}$, $\{\Delta_3\}$, $\{\Delta_4\}$,
$\{\Delta_1, \Delta_3\}$, $\{\Delta_1, \Delta_4\}$ and $\{\Delta_3, \Delta_4\}$ with color 1. Since we must be able to color the remaining orbits with color 2 so that no 4-clique in color 2 appears, by using the second orbit incidence matrix we find two colorings that satisfy the conditions: in solution 1 we color the orbits $\{\Delta_1, \Delta_4\}$ with color 1 and orbits $\{\Delta_2, \Delta_3\}$ with color 2; in solution 2 we color the orbits $\{\Delta_3, \Delta_4\}$ with color 1, and orbits $\{\Delta_1, \Delta_2\}$ with color 2.

Problem * 41.

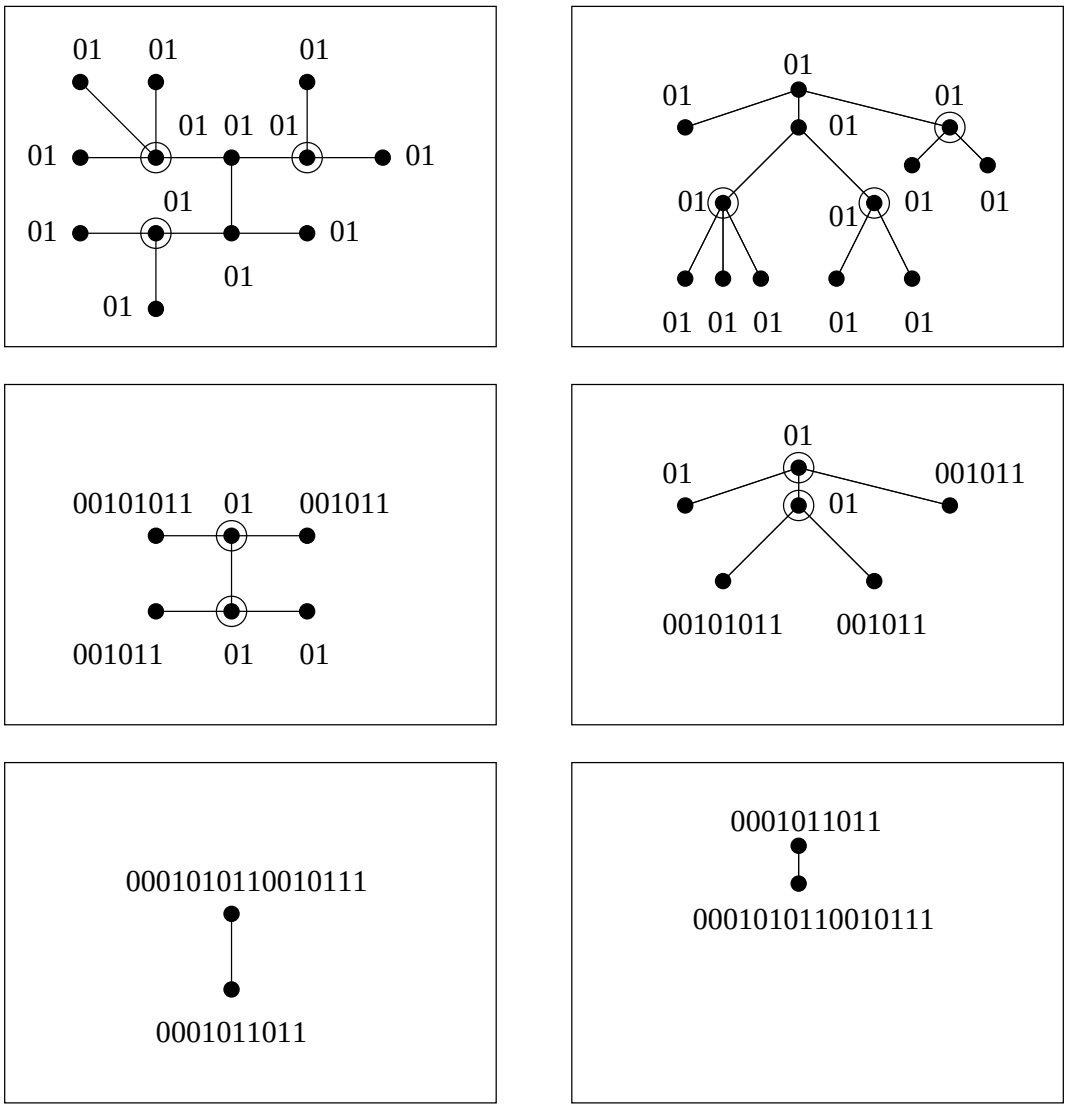
We use the algorithm in Section 7.3.1 of the book after adding a bit of precision:

1. Label all vertices in the tree by 01.
2. Repeat until there are no more than two vertices:
   (a) Let the set T consist of all nonleaf vertices, that have at most one nonleaf neighbor. (In a tree, a vertex is a leaf if it has at most one neighbor.)
   (b) For each $x \in T$ repeat:
      i. Let the set Y be the set of labels of nonleaf neighbors of x, and x too, with the beginning 0 and ending 1 removed.
      ii. Replace the name of x with the name obtained by concatenating the elements of Y in lexicographical order, prepending a 0 and appending a 1.
      iii. Remove from the tree leaf neighbors of x.

3. If only one vertex remains, its label is the certificate.

4. If two vertices remain, the certificate is their labels concatenated in lexicographical order.

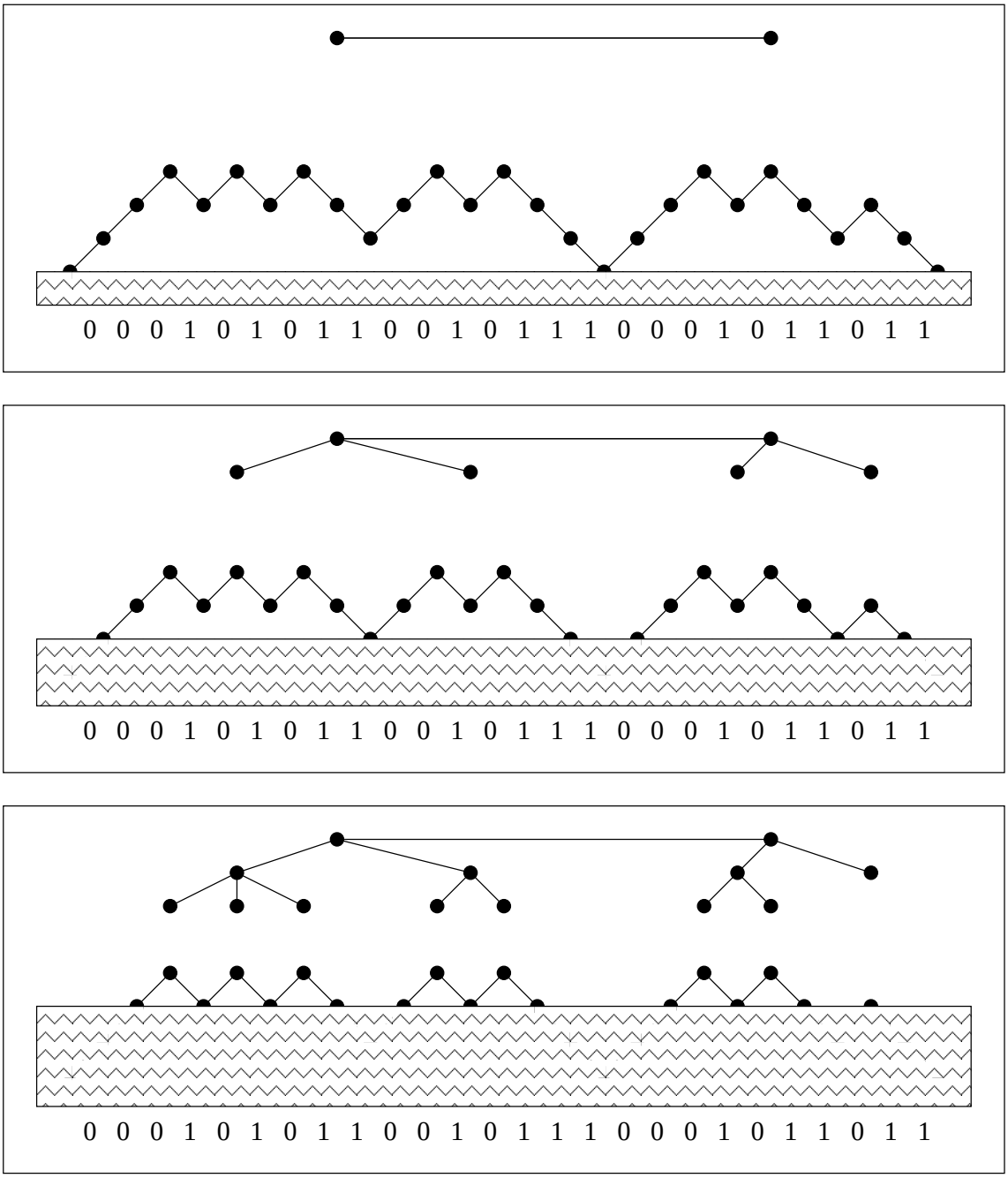
The computation of the algorithm for the trees given is presented in the figure below. The circled vertices belong to T.



As a result we obtain the same certificate for each tree, so the trees are isomorphic.

Problem * 42.

We use the rising water algorithm presented in the book (pp. 248–252). The computation is represented in the following figure.



Problem ** 43.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph considered. Denote the set of neighbors of a vertex $v \in \mathcal{V}$ by $N_{\mathcal{G}}(v)$. Define for subsets $T \subseteq \mathcal{V}$:

$$D_T(v) = |N_{\mathcal{G}}(v) \cap T|,$$

that is, $D_T(v)$ is the number of neighbors of v in the set T. The ordered partitioning

$$B = [B[0], B[1], \ldots, B[|B| - 1]]$$

of the vertex set $\mathcal{V}$ is *equitable*) if for all $i, j \in \{0, 1, \ldots, |B| - 1\}$ it holds that $D_{B[j]}(u) = D_{B[j]}(v)$ for all $u, v \in B[i]$. The ordered partition B is a *refinement* of the ordered partition A if

- for each subset $B[i]$ there is some subset $A[j]$ such that $B[i] \subseteq A[j]$; and
- if $u \in A[i_1]$ and $v \in A[j_1]$, where $i_1 < j_1$, then $u \in B[i_2]$, $v \in B[j_2]$, where $i_2 < j_2$.

Conversely we can say that A is a *coarser* partition than B. B is a strict refinement (strictly coarser than) A if B is a refinement (coarser than) A and additionally $A \neq B$.

Let A be an ordered partition of the vertex set of the graph. For the partition A there exists a unique (up to the order of the components) coarsest equitable partition B, which is a refinement of A. The algorithm given below finds an ordered coarsest equitable partition B, which is a refinement of A.

1. Let $B = A$.
2. Push the subsets of B onto the stack $\mathcal{S}$.
3. Repeat until $\mathcal{S} = \emptyset$:
   (a) Pop the subset T from the top of the stack $\mathcal{S}$.
   (b) Repeat for all subsets $B[i]$ in B:
      i. Let $L[h] = \{v \in B[i] \ : \ D_T(v) = h\}$ for all $h = 0, 1, \ldots, |\mathcal{V}| - 1$.
      ii. If there are more than one nonempty sets in L, replace $B[i]$ by the nonempty sets $L[0], L[1], L[2], \ldots$, and push the nonempty sets onto the stack $\mathcal{S}$.

The degrees of the vertices of the graph in question are

Solmu	0	1	2	3	4	5	6
Asteluku	2	5	4	4	4	1	2

.

Partition the vertices into increasing order of their degree

$$A = [\{5\}, \{0, 6\}, \{2, 3, 4\}, \{1\}].$$

Execute the given algorithm step by step. Initially

$$B = [\{\overset{0}{5}\}, \{\overset{1}{0}, \overset{1}{6}\}, \{\overset{1}{2}, \overset{1}{3}, \overset{1}{4}\}, \{\overset{0}{1}\}]$$
$$\mathcal{S} = [\{5\}, \{0, 6\}, \{2, 3, 4\}, \underbrace{\{1\}}_{=T}]$$

(above each vertex v in B is the corresponding value $D_T(v)$). Clearly no component can be split, as $D_T(v)$ is constant in all components. Things are the same in the next step:

$$B = [\{\overset{1}{5}\}, \{\overset{1}{0}, \overset{1}{6}\}, \{\overset{2}{2}, \overset{2}{3}, \overset{2}{4}\}, \{\overset{3}{1}\}]$$
$$\mathcal{S} = [\{5\}, \{0, 6\}, \underbrace{\{2, 3, 4\}}_{=T}].$$

At the third step $\{2, 3, 4\}$ is split into parts $\{3\}$ and $\{2, 4\}$:

$$B = [\{\overset{0}{5}\}, \{\overset{0}{0}, \overset{0}{6}\}, \{\overset{1}{2}, \overset{0}{3}, \overset{1}{4}\}, \{\overset{2}{1}\}]$$
$$\mathcal{S} = [\{5\}, \underbrace{\{0, 6\}}_{=T}].$$

Now the partition remains unchanged until the stack $\mathcal{S}$ is emptied:

$$B = [\{\overset{0}{5}\}, \{\overset{1}{0}, \overset{1}{6}\}, \{\overset{2}{3}\}, \{\overset{1}{2}, \overset{1}{4}\}, \{\overset{2}{1}\}]$$
$$\mathcal{S} = [\{5\}, \{3\}, \underbrace{\{2, 4\}}_{=T}].$$

$$B = [\{\overset{1}{5}\}, \{\overset{0}{0}, \overset{0}{6}\}, \{\overset{0}{3}\}, \{\overset{1}{2}, \overset{1}{4}\}, \{\overset{1}{1}\}]$$
$$\mathcal{S} = [\{5\}, \underbrace{\{3\}}_{=T}].$$

$$B = [\{\overset{0}{5}\}, \{\overset{0}{0}, \overset{0}{6}\}, \{\overset{1}{3}\}, \{\overset{0}{2}, \overset{0}{4}\}, \{\overset{0}{1}\}]$$
$$\mathcal{S} = [\underbrace{\{5\}}_{=T}].$$

The desired partition is thus

$$B = [\{5\}, \{0, 6\}, \{3\}, \{2, 4\}, \{1\}].$$

Problem *** 44.

The Petersen graph is an example of an extremely regular graph, whose vertex set no invariant inducing function can partition into more than one parts. Thus for example Algorithm 7.1 in the book will execute a brute force backtrack search to find isomorphisms regardless of which invariants are used.

A slightly better result can be achieved by using the ideas developed in the context of computing certificates. An isomorphism could be found for example by computing a certificate for both graphs, and storing some vertex permutation π, that produces the least certificate matrix. (Assume that the vertices of the graph on the right are numbered so that $a \mapsto 0$, $b \mapsto 1$, jne.)

If the permutation $\pi_1 \in \mathrm{Sym}(\{0,\ldots,9\})$ produces the certificate from the adjacency matrix A of the left-hand graph, and $\pi_2 \in \mathrm{Sym}(\{0,\ldots,9\})$ similarly from the adjacency matrix B of the right hand graph, we find $A_{\pi_1}[i,j] = B_{\pi_2}[i,j]$ for all i,j, since the graphs are isomorphic and the certificates must be equal. Now

$$\begin{aligned} B[i,j] = B[\pi_2(\pi_2^{-1}(i)), \pi_2(\pi_2^{-1}(j))] = B_{\pi_2}[\pi_2^{-1}(i), \pi_2^{-1}(j)] = \\ = A_{\pi_1}[\pi_2^{-1}(i), \pi_2^{-1}(j)] = (A_{\pi_1})_{\pi_2^{-1}}[i,j] = A_{\pi_1\pi_2^{-1}}[i,j], \end{aligned}$$

so the permutation $\pi_1\pi_2^{-1}$ is the desired isomorphism that maps vertices of the left-hand graph onto vertices of the right-hand graph such that edges map onto edges.

With pen and paper an isomorphism can be found for example by examining the neighborhood of a vertex (this works especially well in the case of a transitive automorphism group, as one can pick any vertex to start with). In the left-hand graph of Figure 7 the neighborhood of vertex 0 is considered: 0 has the neighbors 1, 4, and 5. Furthermore, in addition to 0, the vertex 1 has the neighbors 2 and 6, the vertex 4 has 3 and 9, and vertex 5 has 7 and 8. If this 2-neighborhood of 0 is drawn radially, we can get the graph on the left. If we repeat this for the graph on the right starting from vertex j we obtain (in one case) the bottom graph in Figure 7, and the isomorphism can be directly read out.

The order of the automorphism group can be determined either by the algorithms in the book (say, algorithm 7.2 or 7.9), or by examining the symmetries. From the structure of the graphs in Figure 7 we can see that after choosing the initial vertex, its neighbors can be permuted at will, and the vertices still remaining can be ordered so that we still get the same graph. Once the order of the neighbors of the initial vertex is fixed, we can see that we can still choose the order of the vertices in one "branch" (say, vertices 2 and 6, both neighbors of vertex 1, in the graph on the left) and still fix the remaining vertices so that we obtain the same

graph. After fixing the vertices in one "branch" the remaining vertices are clearly fixed.

Since the original graph on the left clearly has cyclic rotational symmetry, and on the other hand in Figure 7 identical neighborhood graphs have been presented for vertices 0 ja 5, it is clear that the automorphism group must be transitive, that is, the same neighborhood graph can be drawn starting from any vertex. Now there are 10 ways of choosing the initial vertex, its neighbors can be permuted in $3! = 6$ ways, and the 2 vertices in one "branch" in 2 ways. Thus there are exactly $10 \cdot 6 \cdot 2 = 120$ ways of mapping the neighborhood graph onto itself.

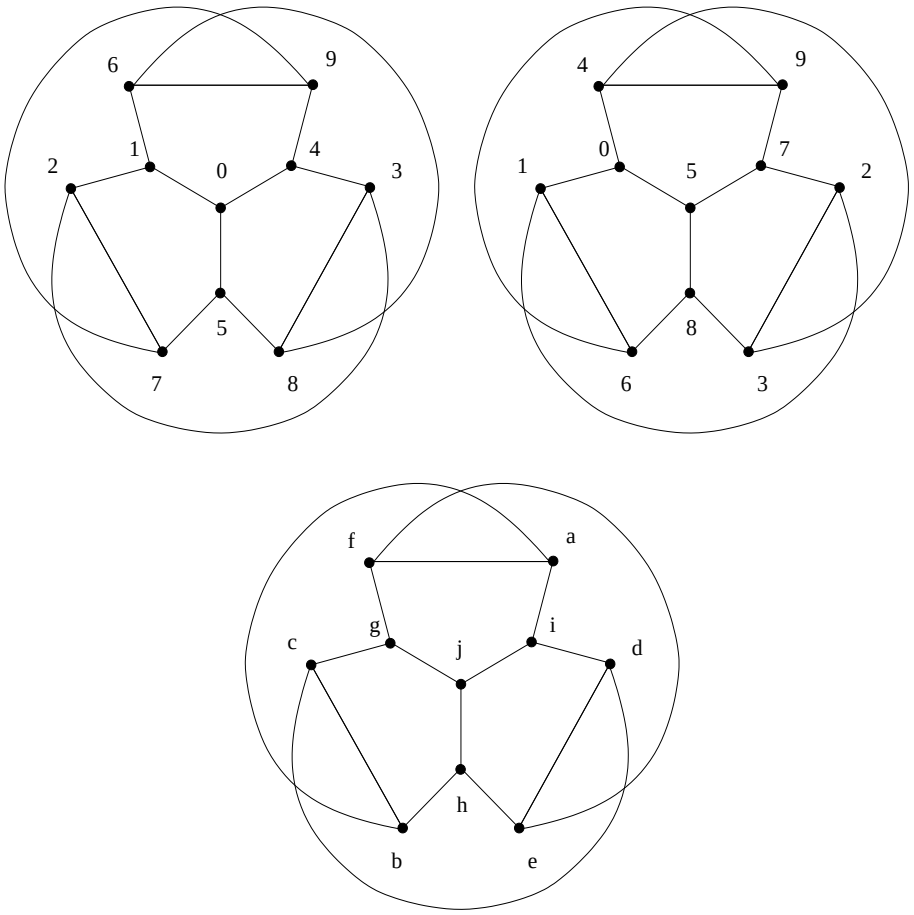


Figure 7: Isomorphisms of Petersen graphs.