Around Cayley's theorem

Bursian O., Kokhas D., Kokhas K.

This project is related with Cayley's theorem about tree counting. Tree is a connected graph without cycles. The theorem states that the number of labelled trees with n vertices is equal to n^{n-2}. There exist many proofs of this theorem, our goal is to introduce some of them and to investigate applications of different approaches to the theorem.

There are rather difficult problems in each project. By the rules of the Summer conference it is allowed to come together in teams for problems solving; you may join in different teams for different projects. We recommend you to find companions, and make cooperative investigations of this project.

First acquaintance with labelled trees

In olympiad problems usually graphs with "unnamed" vertices are used. For example, vertices are some towns and edges are roads between them, or vertices are persons (without names) and edges are acquaintances etc. Another point of view is taken in problems about graph counting: all graph vertices should be "individual". To exclude terminology issues we define a "labelled graph".

Let $[n] = \{1, 2, \ldots, n\}$. A tree (or an arbitrary graph as well) with n vertices which are enumerated by numbers from 1 to n, is called a *labelled tree* (accordingly, a *labelled graph*). To construct a labelled tree we can take a tree and number its vertices, or contrariwise: we can consider the set $[n]$ as a set of vertices and draw a tree by connecting these vertices by edges. The set of all labelled trees with n vertices is denoted by $\mathcal{T}_n$.

Two (unlabelled) graphs G_1 and G_2 with vertex sets V_1 and V_2 are called *isomorphic* (or, speaking plainly, are the same) if there exists a one-to-one mapping $f\colon V_1 \to V_2$ such that vertices A, $B \in V_1$ are connected by an edge in G_1 if and only if $f(A)$ and $f(B)$ are connected by an edge in G_2. For example, any tree with four vertices is isomorphic to either the tree "Chicken's feet" or the tree "Path of length three". In case when G_1 and G_2 both are labelled trees, $V_1 = V_2 = [n]$, the identity plays the role of f.

We assume that Cayley's theorem should not be used in solutions of problems from section "First acquaintance with labelled trees". The symbol T_n in problem statements denotes the number of labelled trees with n vertices, and the question of finding this value is not stated. Problem numbering is given according to the topics of the next sections.

1.1. A graph is called *unicyclic* if it is connected and contains exactly one cycle. Prove that the number of labelled trees with 100 vertices is greater than the number of labelled unicyclic graphs with 98 vertices.

1.2. Construct a bijection between the set of all mappings from $[n]$ to itself and the set of labelled trees with n vertices containing one vertex marked with red stamp and one vertex marked with blue stamp (it may happens that both stamps mark the same vertex).

1.3. There exist n^{n-1} different mappings from the set $[n-1]$ to $[n]$. Prove the identity

$$\sum_{j=1}^{n} \binom{n-1}{j-1} (n-j)^{n-j} T_j = n^{n-1},$$

by partitioning the set of these mappings into n parts in such a way that j-th term in the sum be equal to the number of mappings in j-th part.

3.1. A tree with n vertices and edges enumerated by numbers from 1 to $n-1$, is called an *edge labelled* tree. For example, there exist 4 different edge labelled trees with 4 vertices — see fig. 1. Prove that for $n \geq 3$ the number of different edge labelled trees with n vertices is equal to $\frac{1}{n} T_n$.

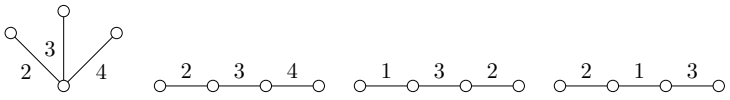

Figure 1. Edge labelled chicken's feet and paths of length 3

If we singled out one of the vertices in a tree (labelled or not) then we call such a tree *rooted* and the vertex that has been singled out is called a *root*. If we need to emphasize that none of the vertices is singled out then we call such a tree *free*. A *leaf* of a tree is a vertex of degree 1, except for the case when the tree is rooted and the root has degree 1, in this case the root is not considered as a leaf. A *forest* is a graph in which each connected component is a tree.

The set of forests on vertex set $[n]$ consisting of k rooted trees with roots 1, 2, ..., k, such that vertex n is in the tree with root 1 is denoted by $\mathcal{F}_n^k$ (fig. 2).

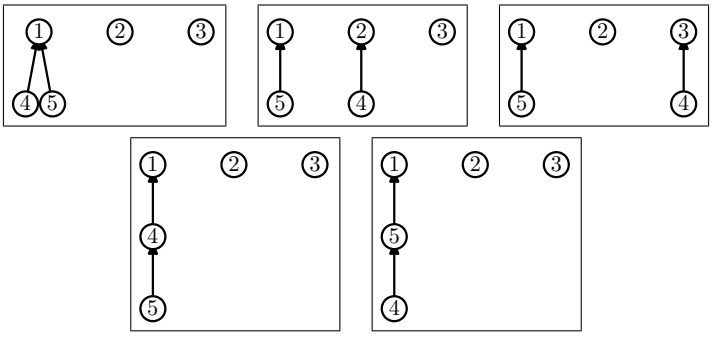

Figure 2. Set of forests $\mathcal{F}_5^3$. We orient each edge towards the root.

We denote sets by "handwritten" letters. Number of set elements is denoted by the same letter in italic font. For example, the number of elements of set $\mathcal{F}_n^k$ we denote by F_n^k.

1.4. Prove the recurrent relation for $2 \le k \le n-1$:

$$F_n^{k-1} = nF_n^k.$$

1.5. Let $V_1 = [r]$, $V_2 = \{r+1, \ldots, r+s\}$, $V = V_1 \cup V_2 = [r+s]$. Denote by $\mathcal{F}_{r,s}^k$ the set of forests consisting of k rooted trees on vertex set V with roots 1, 2, ..., k, such that vertex $r+1$ is in the tree with root 1, and each edge connects vertex from V_1 with vertex from V_2. Find the recurrence relation (by k) for the numbers $F_{r,s}^k$.

There are n parking spaces available along a one-way street and each of n drivers numbered from 1 to n has a preferred parking space. The drivers arrive consecutively in increasing order of their numbers. Each driver goes to his preferred parking place and parks on that place if it's not occupied; otherwise he goes farther to first free space and parks there, if all the spaces are occupied he goes away forever. By a preference sequence we call a list $a_1, a_2, \ldots, a_n$ of preferred parking places of the first, second, ..., n-th driver.

4.1. Prove that the number of preference sequences in which everyone will find a parking space is $(n+1)^{n-1}$.

4.2. Prove that success or unsuccess of the parking process does not depend on the order in which the cars arrive.

The next problem is out of the mainstream of our project. But it gives a possibility to understand what troubles arise in counting of unlabelled trees.

3.2. Prove that the number of different (that is nonisomorphic to each other) unlabelled trees with n vertices is less than 4^n.

1 Recursions, identities, bijections

1.6. Prove by combinatorial reasoning (interpreting numbers T_i as numbers of trees), that for $n > 1$

a) $T_n = \frac{n}{2} \sum_{k=0}^{n-2} \binom{n-2}{k} T_{k+1} T_{n-k-1}$;

b) Reduce this formula and also the formulas from problems 1.3 and 5.1 a) to each other algebraically.

1.7. Denote by $\mathcal{T}(n,k)$ the set of labelled rooted trees with n vertices in which the root is labelled by 1 and has degree k. Prove that

$$(n-1)(k-1)T(n,k) = (n-k)T(n,k-1).$$

A "triangle tree" is a graph defined by induction in the following way. The smallest triangle tree is a complete graph with two vertices (in contrast with its name). If some triangle tree is already given we can take its arbitrary edge AB, take new vertex C and add vertex C and edges AC, BC to the tree. By a labelled triangle tree we call a triangle tree with vertices numbered from 1 to n.

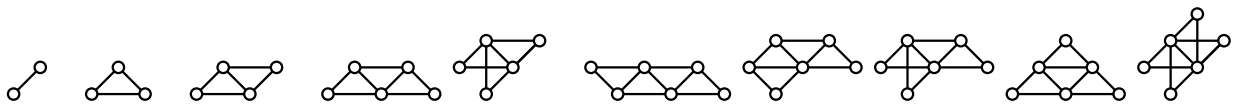

Figure 3. Unlabelled triangle trees on 2, 3, 4, 5 and 6 vertices

For example, for $n = 5$ there exist 2 unlabelled and 70 labelled triangle trees (fig. 3).
If some edge of a triangle tree is singled out then we call such tree "rooted".

1.8. Denote by $\Delta(n,k)$ the number of rooted labelled triangle trees with n vertices such that its rooted edge belongs to k triangles. Find recursion (by k) for the numbers $\Delta(n,k)$.

2 Prüfer code

Prüfer code corresponds a tree with numbered vertices to the sequence of its vertices in the following way. Prüfer code of a tree with two vertices is an empty word. If the number of vertices of a tree T is more than 2 then denote by v a leaf with smallest number, and by u the vertex adjacent to v. Then Prüfer code of tree T is obtained from Prüfer code of tree $T - v$ by appending the vertex u (to the left).

Check that you know how to solve problem 2.1 and come to jury to register your plus.

2.1. a) Find Prüfer code of tree with vertices 1, 2, ..., 10 and edges (8,9), (8,4), (4,10), (10,3), (3,5), (10,6), (10,1), (1,7), (1,2).

b) Reconstruct tree by the Prüfer code 1, 1, 2, 5, 4, 2, 7.

c) Prove that Prüfer code defines one-to-one correspondence between the set of trees on the given set of n vertices and the set of words of length $n-2$ with "letters" from this set.

d) Prove that a vertex of degree d occurs $d-1$ times in the Prüfer code.

2.2. What is the number of rooted labelled trees with n vertices in which vertex n is a leaf (and hence it is not root)?

2.3. What is the number of free labelled trees with $n > 10$ vertices in which the degree of vertex 1 is equal to 10?

2.4. Find the number of labelled unicyclic graphs with n vertices having a cycle of length k.

2.5. Denote by $S(n,k)$ Stirling number of the second kind, by definition it is equal to the number of ways to partition set $[n]$ into k nonempty parts. Prove that the number of labelled trees on n vertices with exactly r leafs is equal to $\frac{n!}{r!} S(n-2, n-r)$.

2.6. Denote by $\tau(k_1, k_2, \ldots, k_n)$ the number of free labelled trees with n vertices such that the degree of i-th vertex is $k_i + 1$. Prove that $\tau(k_1, k_2, \ldots, k_n) = \dfrac{(n-2)!}{k_1!k_2!\ldots k_n!}$.

3 Results

You can submit several solutions of the next problems if they are quite different.

3.3. Deduce Cayley's theorem from problems: a) 1.4, b) 1.6, c) 1.7, d) 2.6.

3.4. Prove that the number of forests with vertices from set $[n]$ consisting of k rooted trees is equal to $\binom{n-1}{k-1}n^{n-k}$.

3.5. Prove that the number of forests with vertices from set $[n]$ consisting of two free trees is equal to $\frac{1}{2}n^{n-4}(n-1)(n-6)$.

3.6. Prove that the number of spanning trees of complete bipartite labelled graph $K_{r,s}$ with parts $V_1 = [r]$ and $V_2 = \{r+1, \ldots, r+s\}$ is equal to $r^{s-1}s^{r-1}$.

3.7. Let Δ_n be the number of labelled triangle trees with n vertices, Λ_n be the number of rooted labelled triangle trees with n vertices and rooted edge 1–2.

a) Prove that $\Lambda_n = (2n-3)^{n-3}$.

b) Find Δ_n.

Let n different objects are arranged in the row. A *cyclic* permutation is a permutation that moves all the objects along some cycle: it puts the first object on the place of the second one, puts the second one on the place of the third one and so on, the last object is put on the place of the first one. A *transposition* (ij) is a permutation that changes objects on the i-th и j-th place. If we consider $[n]$ as set of vertices of some graph then the transposition (ij) can be interpreted as an edge connecting vertex i with vertex j.

The result of consecutive applying of permutations s_1, s_2, …, s_{n-1} is a permutation that is called a *product* of these permutations and is denoted by $s_1 s_2 \ldots s_{n-1}$. We consider the products that differ by the order of factors as different. For example, if permutation s is a transposition changing objects on the first and second places, permutation t is a transposition changing objects on the third and fourth places, then the products st and ts determine the same permutations but we consider them as different products.

3.8. a) n objects are arranged in the row. Prove that the result of consecutive applying of transpositions s_1, s_2, …, s_{n-1} is a cyclic permutation if and only if the graph with vertex set $[n]$ and edge set s_1, s_2, …, s_{n-1} is a tree.

b) Prove that the number of ways in which a cyclic permutation of set $[n]$ can be represented as a product of $n-1$ transpositions is T_n.

4 *Parking functions*

There are n parking spaces available along a one-way street and each of n drivers numbered from 1 to n has a preferred parking space. The drivers arrive consecutively in increasing order of their numbers. Each driver goes to his preferred parking place and parks on that place if it's not occupied; otherwise he goes farther to first free space and parks there, if all the spaces are occupied he goes away forever. A preference sequence for which everyone finds a parking space is called a *parking function*. The set of parking functions is denoted by $\mathcal{P}_n$.

4.3. Find the number of parking functions $(a_1, a_2, \ldots, a_n)$ such that any two consecutive drivers have different preferences that is $a_k \neq a_{k+1}$ for $k = 1, \ldots, n-1$?

4.4. Let only $m < n$ cars enter the street with n parking spaces. How many preference sequences exist such that all the drivers find a parking space?

4.5. Prove that the number of parking functions such that exactly k drivers $(1 \leq k \leq n)$ prefer to park in the first place is equal to $\binom{n-1}{k-1} n^{n-k}$.

4.6. For each parking function $a = (a_1, a_2, \ldots, a_n)$ define the sequence of differences $c(a) = (c_1, c_2, \ldots, c_{n-1})$ by the rule

$$c_i = a_{i+1} - a_i \pmod{n+1}.$$

Let the parking function a correspond to the labelled tree $t(a)$ with $n+1$ vertices that is given by Prüfer code $c(a)$.

Prove that this correspondence is a bijection between $\mathcal{P}_n$ and $\mathcal{T}_{n+1}$.

4.7. Prove that $(n+1)^{n-1} = \sum_{\substack{0 \leq k_n \leq 1 \\ 0 \leq k_{n-1}+k_n \leq 2 \\ 0 \leq k_{n-2}+k_{n-1}+k_n \leq 3 \\ \ldots \\ 0 \leq k_2+k_3+\ldots+k_{n-1}+k_n \leq n-1}} \frac{n!}{(n-k_2-k_3-\ldots-k_n)!k_2!k_3!\ldots k_n!}$.

4.8. We call a parking function $a = (a_1, \ldots, a_n)$ *sure* if at least $j+1$ drivers prefer park in the first j places for all j, $1 \leq j \leq n-1$. Prove that the number of sure parking functions in the street with n parking spaces is equal to $(n-1)^{n-1}$.

4.9. Prove by combinatorial arguments the recurrent relation: $P_{n+1} = \sum_{k=0}^{n} \binom{n+1}{k} P_k (n-k)^{n-k}$.

5 *Inversions on trees and deficiencies of parking functions*

5.1. Prove the recurrent relation by combinatorial arguments:

a) $T_n = \sum_{k=0}^{n-2} \binom{n-2}{k} (k+1) T_{k+1} T_{n-k-1}$; b) $P_{n+1} = \sum_{k=0}^{n} \binom{n}{k} (k+1) P_k P_{n-k}$;

Consider a labelled tree T with $n+1$ vertices. Assign vertex $n+1$ to be a root and define directions on the tree edges towards the root. We "accept as correct" that the vertex labels increase when we move towards the root. We say that vertices i and j form *inversion*, $1 \le i < j \le n$, if vertex i lies on the path from vertex j to the root. Check all pairs of vertices and denote by $\mathrm{inv}(T)$ the total number of inversions in tree T. A maximum value of $\mathrm{inv}(T)$ is equal to $\frac{n(n-1)}{2}$, it is achieved when T is a path of n edges whose vertices are labelled as $n+1$, $1, 2, \ldots, n$.

Let $a = (a_1, a_2, \ldots, a_n)$ be a parking function. Let first driver parked in p_1-th place, the second driver parked in p_2-th place etc. We call the value

$$D(a) = \sum_{i=1}^{n} (p_i - a_i) = \frac{n(n-1)}{2} - \sum_{i=1}^{n} a_i.$$

the *deficiency* of parking function a. Function $(1, 1, \ldots, 1)$ has the largest possible deficiency, it is equal to $\frac{n(n-1)}{2}$.

Theorem. Let n be arbitrary natural number. Then for all k, $0 \le k \le \frac{n(n-1)}{2}$, the number of rooted labelled trees on $n+1$ vertices with root $n+1$ having k inversions is equal to the number of parking functions with deficiency k in the street with n parking spaces.

Introduce polynomials $F_n(x)$ and $H_n(x)$ that "number" inversions and deficiencies:

$$F_0(x) = 1, \quad F_n(x) = \sum_{T \in \mathcal{T}_{n+1}} x^{\mathrm{inv}(T)}; \qquad H_0(x) = 1, \quad H_n(x) = \sum_{a \in \mathcal{P}_n} x^{D(a)},$$

We call them *inversion enumerator* и *deficiency enumerator*.

5.2. Let $\mathcal{T}^*$ be the set of labelled rooted trees with $n+3$ vertices such that its root has degree 2 and is labelled by $n+3$, and two sons of the root are labelled by $n+1$ and $n+2$. Express the inversion enumerator of the set $\mathcal{T}^*$

$$F_n^*(x) = \sum_{T \in \mathcal{T}^*} x^{\mathrm{inv}(T)}$$

in terms of the polynomials $F_i(x)$.

5.3. Prove that polynomials $F_n(x)$ and $H_n(x)$ satisfy the same recurrent relations for $n \ge 0$

$$F_{n+1}(x) = \sum_{k=0}^{n} \binom{n}{k} (x^k + x^{k-1} + \ldots + 1) F_k(x) F_{n-k}(x). \tag{a}$$

$$H_{n+1}(x) = \sum_{k=0}^{n} \binom{n}{k} (x^k + x^{k-1} + \ldots + 1) H_k(x) H_{n-k}(x). \tag{b}$$

The eheorem about inversions and deficiencies immediately follows from two previous problems. Though you might suggest a bijective proof.

5.4. Prove the theorem about inversions and dificiencies by constructing one-to-one correspondence between the sets $\mathcal{T}_{n+1}$ and $\mathcal{P}_n$ such that a parking function with dificiency j corresponds to the tree with j inversions.

6 *Additional tasks*

By a *labelled plane tree* we call a rooted labelled tree such that the sons of any of its vertex are lineary ordered. Denote by $\mathcal{P}_n^k$ the set of forests consisting of k labelled plane trees on the set $[n]$ with roots 1, 2, ..., k such that vertex number n is in the tree with root 1.

1.9. Prove that the recurrent relation holds for $2 \le k \le n-1$

$$P_n^{k-1} = (2n-k)P_n^k.$$

3.9. Prove that the number of spanning trees in the complete bipartite labelled graph $K_{r,s}$ containing $k+\ell$ trees rooted in vertices 1, 2, ..., k and $r+1$, $r+2$, ..., $r+\ell$ is equal to

$$(r\ell + sk - k\ell)r^{s-\ell-1}s^{r-k-1}.$$

3.10. Prove that the number of spanning trees in complete tripartite labelled graph $K_{r,s,t}$ with parts $V_1 = [r]$, $V_2 = \{r+1, \ldots, r+s\}$ and $V_3 = \{r+s+1, \ldots, r+s+t\}$ is equal to

$$(r+s+t)(r+s)^{t-1}(s+t)^{r-1}(t+r)^{s-1}.$$

3.11. What is the number of plane labelled rooted trees with $n+1$ vertices?

A tetrahedral tree is defined by induction similarly as a triangle one. The simplest tetrahedral tree (degenerate) is a triangle (complete graph with three vertices), the next simplest is a tetrahedron (complete graph with four vertices). If some tetrahedral tree is already given then we may add a new vertex by taking any triangle face in any of tetrahedrons and constructing a new tetrahedron with the new vertex using the chosen face as a base. Formally it means that we add to the graph one new vertex and three new edges (and three new triangle faces, if you wish). Note that the tetrahedrons may intersect in the three-dimensional space as well as triangles of a triangle tree may intersect in the plane.

3.12. How many labelled tetrahedral trees with n vertices exist?

4.10. Prove that $(n-1)^{n-1} = \sum_{\substack{0\le k_{n-1}\le 1\\ 0\le k_{n-2}+k_{n-1}\le 2\\ 0\le k_{n-3}+k_{n-2}+k_{n-1}\le 3\\ \ldots\\ 0\le k_2+k_3+\ldots+k_{n-2}+k_{n-1}\le n-2}} \frac{n!}{(n-k_2-k_3-\ldots-k_{n-1})!k_2!k_3!\ldots k_{n-1}!}.$

4.11. Prove that $n^n = \sum_{\substack{0\le k_1\le 1\\ 0\le k_1+k_2\le 2\\ 0\le k_1+k_2+k_3\le 3\\ \ldots\\ 0\le k_1+k_2+\ldots+k_{n-1}\le n-1}} \frac{n!}{k_1!k_2!\ldots k_{n-1}!}.$

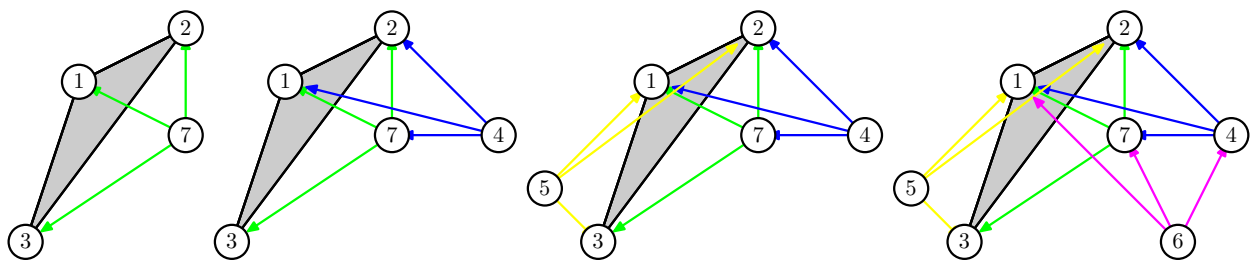

Figure 4. Construction of a tetrahedral tree. We add cosequently vertices 7, 4, 5, 6 to the initital triangle (1, 2, 3)

Solutions

1 Recursions, identities, bijections

1.1. We took this problem from [3]. Let A be a vertex with the largest label on the cycle, and B be one of the two adjacent vertices that has larger label. Remove edge AB from the tree, hang a leaf with label 99 to vertex A and a leaf with label 100 to vertex B. We constructed injective mapping from the set of unicyclic graphs to the set of trees.

1.2. See [6, problem 8.5] or the first proof in [1, chapter 26]. Consider the path from the blue vertex to the red one. Let $A \subset [n]$ be the set of labels of this path. Our tree consists of this path and several trees that grow on the vertices of this path. The sequence of labels along the path can be interpreted as a permutation of elements of set A. Draw this permutation as a set of cycles (with vertices on the elements of set A). We obtain a set of cycles, and several trees that grow on its vertices.

From the other hand, mapping from set $[n]$ to itself is given by the directed graph: for each i draw an arrow from vertex i to vertex $f(i)$ (loops are allowed). Since all the vertices of this graph have outgoing degree 1, the graph is a union of several cycles and several trees that grow on the vertices of these cycles (speaking more carefully the trees "pour in" the vertices of the cycles).

This is the required bijection.

1.3. We took this statement from [17, p. 3.9]. The rigth hand side of the equality counts all possible mappings from $[n-1]$ to $[n]$. Any such mapping f can be drawn as a directed graph with n vertices: each vertex i is a starting point of an arrow that goes to $f(i)$. Vertex n can have ingoing edges only, and the connected component of vertex n is a tree in which all the arrows directed "towards n". The left hand side of the equality counts such graphs classifying them by the trees containing vertex n.

1.4. [14, theorem 2.1]. The root of a tree determines a direction from the root to the periphery. Construct a mapping from set $\mathcal{F}_n^{k-1}$ to $\mathcal{F}_n^k$. For this we take an arbitrary forest with $k-1$ trees, find vertex k in it, cut off this vertex together with the branch that grows on it, and put it as a separate rooted tree. Ususally the result is a tree from $\mathcal{F}_n^k$ except the case when we cut off the branch from first tree and this branch contains vertex n. In this case we do a correction: exchange labels 1 and k.

Now count how many preimages for this mapping has an arbitrary forest from $\mathcal{F}_n^k$. In order to construct a preimage of a forest, we have to take its k-th tree and attach it as a branch to an arbitrary vertex of the first, second, ..., $(k-1)$-th tree of the forest. This construction imply that there was no correction. And for preimages with correction, we have to attach the first tree as a branch to an arbitrary vertex of k-th tree and then exchange labels 1 and k. Thus, each of n vertices of the graph can be used for construction of a unique preimage. Hence, $F_n^{k-1} = nF_n^k$.

1.5. Answer: $F_{r,s}^{k-1} = sF_{r,s}^k$ for $2 \le k \le r$. [14, Corollary 3.1]. Recursion can be constructed as in problem 1.4. Since the vertex k must belong to set V_1, in inverese mapping the subtree with root k can be attached to any of s vertices of set V_2.

1.6. a) This result is from [18]. Denote by E_n the number of labelled trees with vertices on $[n]$ containing one specified edge, for example, edge 1–2. Counting all the edges in all the trees we obtain an identity

$$\frac{n(n-1)}{2} E_n = (n-1)T_n.$$

Thus, $T_n = \frac{1}{2} n E_n$. Now to find E_n we have to choose how many vertices do the tree hanging on vertex 1 and the tree hanging on vertex 2 contain, and choose trees with these numbers of vertices. We obtain formula $T_n = \frac{n}{2} \sum\limits_{k=0}^{n-2} \binom{n-2}{k} T_{k+1} T_{n-k-1}$.

b) The equality in this problem can be derived from the equality of problem 5.1 a) by a "Gauss method": sum up two copies of the sum in the problem 5.1 a), change index of the summation k with $n-2-k$ and sum up these sums term by term.

Now derive the identity of problem 1.3 from the identity 5.1 a). For this write separately the summand for $j=n$, and change the summation index in the remaining sum by the rule $j=k+1$:

$$n^{n-2}+\sum_{k=0}^{n-2}\binom{n-1}{k}(k+1)^{k+1}(n-k-1)^{n-k-1}=n^{n-1}.$$

Subtract n^{n-2} from both sides and apply the formula $\binom{n-1}{k}(n-k-1)=(n-1)\binom{n-2}{k}$ in the left hand side:

$$\sum_{k=0}^{n-2}(n-1)\binom{n-2}{k}(n-k-1)(k+1)^{k+1}(n-k-1)^{n-k-3}=n^{n-2}(n-1).$$

By cancelling $(n-1)$ we obtain the formula from the problem 5.1 a) up to a change of the summation index.

1.7. [4] Let $A\in\mathcal{T}(n,k-1)$ be one of the trees, v be any of its $n-k$ vertices adjacent to vertex 1. Replace the edge connecting vertex v to its ancestor with the edge connecting this vertex and the root. Denote the obtained tree by B (obviously $B\in\mathcal{T}(n,k)$), we will call a pair of trees (A,B) a *bundle*.

Count the number of bundles by two different ways. From the one hand, the number of bundles is equal to $(n-k)T(n,k-1)$, because the bundle is uniquely determined by the tree A with vertex v. The tree can be chosen in $T(n,k-1)$ ways, then the vertex v can be choosen in $(n-k)$ ways. From the other hand, the bundle may be obtained in the following way: choose a tree B in $T(n,k)$ ways, remove one edge outgoing from the root, and connect the "broken branch" obtained with any of non root vertices of the remaining tree. In total we obtain

$$(n-1-n_1)+(n-1-n_2)+\ldots+(n-1-n_k)=(n-1)(k-1)$$

ways where n_i is the number of vertices in the "broken branch" formed after removal of the i-th edge outgoing from the root. So we have $(n-1)(k-1)T(n,k)$ cases. Thus, $(n-1)(k-1)T(n,k)=(n-k)T(n,k-1)$.

1.8. Answer: $(n-k-2)\Delta(n,k)=k(2n-4)\Delta(n,k+1)$. This statement is from [9].

Note that the construction in the definition of a triangle tree can start from any edge: take any edge, add the triangles that contain this edge, then add triangles that contain any of the already constructed edges and so on. Furthermore, let the first edge be fixed, and during the appending to the tree the next vertex C, we draw arrows on the edges CA, CB outgoing from vertex C. Then the directed graph that represents the triangle tree does not depend on the order in which the vertices had been added!

Consider a triangle tree G in which the rooted edge uv belongs to k triangles, denote them by uvw_1, uvw_2, …, uvw_k, and construct a tree with $k+1$ triangles by the following construction (fig. 5). Take an arbitrary vertex w, coinsiding with no one of the vertices w_i, and consider a minimal triangle subtree containing vertices u, v and w. Vertex w has two outgoing edges in this subtree since it is minimal one. Remove these edges and draw edges wu and wv instead of them. As a result of this operation we obtain a rooted tree that has one more triangle. The parts of tree G that were hunging on the removed edges, hang now on the new edges wu and wv. (It may happens that during these actions we remove the edge connecting w with u or v, and immediately restroe it.) Vertex w may be chosen in $n-k-2$ ways, so we have $(n-k-2)\Delta(n,k)$ ways of implementing this construction.

Now describe the inverse operation: given a tree in which the rooted edge uv belongs to $k+1$ triangles, construct a tree in which the rooted edge uv belongs to k triangles. Take one of the $k+1$ triangle vertices connected to the root, let it be vertex w, and "displant" (move) triangle

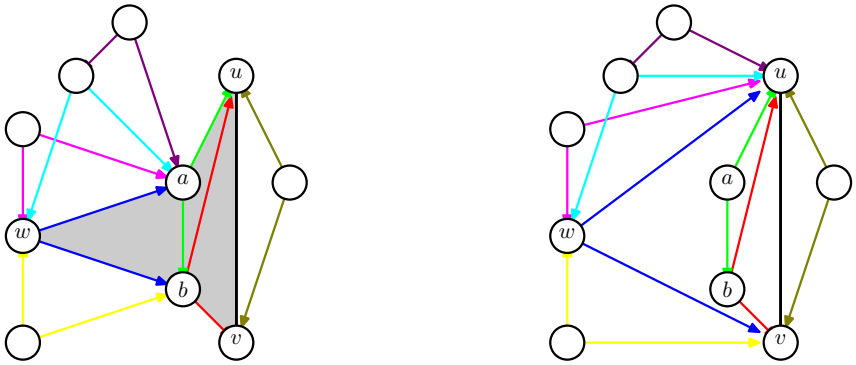

Figure 5. A tree with k triangles on the rooted edge uv (left figure) and a tree with $k+1$ triangles ($k=1$) (right figure). The minimal tree containing vertices w, u and v is colored gray. Two arrows go from each vertex except u and v to the endpoints of the edge on which the triangle corresponding to the vertex had been hunged

uvw from the edge uv to another edge, say, ab. It can be done in the following way: let G_1 and G_2 be triangle trees growing on the edges wu and wv. Choose an arbitrary edge ab, not belonging to $G_1 \cup G_2$ (and not coinsiding with uv), and replace edges wu and wv with edges wa and wb. The subtrees G_1 and G_2, hanging on edges wu and wv, we displant on the edges wa and wb correspondingly.

Count in how many ways this construction can be implemented. Any of $2n-4$ non root edges can be assigned as ab. If edge ab belongs to the triangle subtree, hanging on the rooted triangle uvw', then we can take any of k rooted triangles uvw, where $w \neq w'$, as triangle uvw, which we hang on this edge. In total, $k(2n-4)\Delta(n,k+1)$ ways.

1.9. [14, corollary 4.1]. Try to construct a recursion similar to that of in problem 1.4. In this problem we have more places where the subtree with root k may be hanged. For each vertex the number of places is equal to the number of its descendants plus one. In total for all vertices we obtain the number of all edges plus the number of all vertices. And the number of edges in forest with k trees is equal to $n-k$.

2 *Prüfer code*

2.1. This is problem 2.2.4 from [3], it is just an exercise for understanding what is a Prüfer code.

2.2. Answer: $(n-1)^{n-1}$.

Prüfer code of unrooted tree in which vertex n is a leaf does not contain symbol "n". Hence there are $(n-1)^{n-2}$ such codes. The choise of the root increases the number of cases in $(n-1)$ times.

2.3. Answer: $\binom{n-2}{9}(n-1)^{n-11}$. Since this number is equal to the number of fillings of n cells by $n-2$ different objects for which the first cell contains 9 objects.

2.4. Answer: $\frac{1}{2}(n-1)(n-2)\dots(n-k+1)n^{n-k}$.

The number of ways to choose a cycle is equal to $\frac{n(n-1)(n-2)\dots(n-k+1)}{2k}$: we choose the first vertex of cycle, the second one and so on consequently, after that we take into account that it is not important which vertex of the cycle is the first and what is the cycle direction. It remains to count the number of unicyclic graphs containing cycle $(n, n-1, \dots, n-k+1)$. Remove the edge connecting n and $n-k+1$. The number of possible Prüfer codes for obtained tree is equal to kn^{n-1-k}. Indeed, after $n-k$ steps of Prüfer encoding the remaining part of the tree is exactly the path n, $n-1$, …, $n-k+1$. Then the first $n-k-1$ elements of Prüfer code can be assigned arbitrarily, $(n-k)$-th element is the number from $n-k+1$ to n, and all other elements are determined.

We took this problem from [3].

2.5. We took this problem from [5]. Attentive reader can also find it in [7]. Due to Prüfer code, it seems to be almost a tavtology.

2.6. [7]. Prüfer's algorithm maps labeled trees to monomials $x_{i_1}x_{i_2}\dots x_{i_{n-2}}$, which can be "reduced" to the form $x_1^{k_1}x_2^{k_2}\dots x_n^{k_n}$ (where $k_1+k_2+\dots+k_n=n-2$). Thus the number of different codes is equal to the polynomial coefficient $\frac{(n-2)!}{k_1!k_2!\dots k_n!}$.

3 Results

3.1. We took this statement in [17]. For each labeled tree assign vertex n to be a root, then the direction to the root is defined on each edge and the relation "ancestor–descendant' ' is given. Now to each labeled tree with n vertices we put into correspondence an edge-labeled tree with n vertices. For this we put the label i on each edge ij , where i is a descendant of j.

The obtained edge labeled tree allows us to recover the initial vertex labelling if we indicate which vertex was a root and label it by n. Hence each edge labeled tree could be obtained in our mapping from n different labeled trees.

3.2. We took this problem in [3]. Let us count the number of rooted trees. Draw a tree on the plane and draw an arrow on each edge towards the periphery. Now put a code into correspondence to the tree: starting from the root move along the tree as if it is a system of walls on the plane. If we move along the arrow write 1, if against write 0. We obtain a sequence of $2n-2$ zeroes and ones. It is clear that each sequence determines at most one tree.

3.3. a) Cayleys theorem follows from problem 1.4 and the equality $F_n^{n-1} = 1$.

c) Cayley's theorem follows from problem 1.7 and the equality $T(n,k) = \binom{n-2}{k-1}(n-1)^{n-k-1}$. Indeed, the number of labeled trees is equal to the sum

$$\sum_{k=1}^{n-1} T(n,k) = \sum_{k=1}^{n-1} \binom{n-2}{k-1}(n-1)^{n-k-1} = \sum_{k=0}^{n-2} \binom{n-2}{k}(n-1)^{n-2-k} = ((n-1)+1)^{n-2}.$$

b) In [18] Cayley's theorem is drived from problem 1.6 by means of generation functions and Lagrange inverse formula, but may be elementary method exist.

d) Cayley's theorem is drived from problem 2.6 by the summation like in the polynomial theorem

$$\sum_{\substack{k_1+k_2+\ldots+k_n=n-2 \\ k_i \geq 0}} \frac{(n-2)!}{k_1!k_2!\ldots k_n!} = n^{n-2}.$$

3.4. See [14, corollary 2.3].

3.5. We took this problem in [17, п. 4.3].

3.6. See [14, corollary 3.1]. We can prove the statement by means of Prüfer code. At the end of the Prüfer encoding we obtain two vertices from different parts of the graph. Therefore the code contains $r-1$ labels from the first part and $s-1$ labels from the second part and the total numbers of codes equals $r^{s-1}s^{r-1}$.

3.7. a) Solution 1. The statement is taken from [9] where it is proven by five different ways. We present the third proof in [9].

Applying the recursion from solution 1.8 and the "initial condition" $\Delta(n,k) = 1$ for $k = n-2$, we obtain

$$\Delta(n,k) = (2n-4)\frac{k}{n-2-k}\Delta(n,k+1) = (2n-4)^2\frac{k(k+1)}{(n-2-k)(n-2-(k+1))}\Delta(n,k+2) =$$
$$= \ldots = (2n-4)^{n-2-k}\frac{k(k+1)\ldots(n-3)}{(n-2-k)(n-2-(k+1))\cdots 1}\Delta(n,n-2) =$$
$$= (2n-4)^{n-2-k}\frac{(n-3)!}{(k-1)!(n-3-(k-1))!}\Delta(n,n-2) = (2n-4)^{n-2-k}\binom{n-3}{k-1}.$$

By summing over all possible k and applying the binomial expansion we compute the number of rooted triangle trees: $\sum\limits_{k=1}^{n-2}\binom{n-3}{k-1}(2n-4)^{n-k-2} = (2n-3)^{n-3}$.

Solution 2 (Prüfer code). We encode a labelled rooted triangle tree by a code similar to Prüfer code.

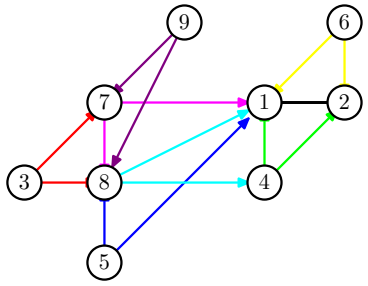

Figure 6. Example of the construction of code for the triangle tree (7b)(12)(8a)(7b)(8a)(4a).

Construction of the code by a tree. Let a triangle labelled rooted tree with rooted edge 1–2 be given. At first, given labelling of the vertices, define a special labelling of edges. Draw arrows on the edges by the same way as in solution of problem 3.7. Then each vertex has two outgoing arrows (except the vertices of the root edge). If a vertex has label x, then label its outgoing edges as xa and xb, where the endpoint of the edge xa has smaller label than the endpoint of the edge xb. Denote the root edge by 1–2. Thus, our tree contains $2n-3$ edges with "unified" names:

$$12,\ 3a,\ 3b,\ 4a,\ 4b,\ \ldots,\ na,\ nb.$$

We write the code using new names of edges. Vertices of degree 2 we call leaves. In each step we choose the leaf of the current triangle tree with maximal label, remove the chosen vertex and write the name of its ancestor edge. In the last step the last vertex is connected with endpoints of root edge 1–2, and we skip writing.

Recovering the tree by a code.

All the vertices x for which both edges xa and xb do not belong to the code are leaves. (The vertices 1 and 2 are not the leaves by definition.) So looking at the code we can write the List of leaves. In this list the vertex z with the largest label has been removed the first. Therefore the first symbol in the code is the name of the edge-ancestor of z. Write the vertex z and its edge-ancestor to the second list, call it the "List of triangles", remove the first symbol from the code, remove z from the List of leaves. Look at the (shortened) code once again, seek a new vertices that become leaves and update the List of leaves. Now take the vertex with the largest label and the first symbol in the code and so on.

At the end of this algorithm we obtain the List of triangles containing the sequence of vertices with their edge-ancestors. Unlike the usual trees it is not easy to recover the triangle tree by reading List of triangles from the beginning because the unified name of an edge (xa or xb) "hides" the labels of its endpoints. But if we start to read the List of triangles from the tail, it can be done easily. Draw the root edge 1–2. And start the construction of the tree like in its defintion. Vertex which is not contained in List of triangles should be appended to the tree the first. In the next step the vertex y has the edge-ancestor with the name of the form xa or xb, and since this edge has been already drawn, we know the labels of both its endpoints.

Why does the list always contain at least one leaf? The current code has $n-3-k$ positions where k is the number of already considered positions. And the number of vertices that could be a leaves is equal to $n-2-k$ (we subtract 2 because the vertices 1 and 2 are not leaves), that is greater by 1.

The code contains $n-3$ positions, the tree contains $2n-3$ edges. Each edge can be written in any position, so the number of codes is equal to $(2n-3)^{n-3}$. Since eaxctly one tree corresponds to each code, and any tree determines a code, the number of rooted triangle trees coinsides with the number of codes.

Example. Let $n=8$ and the code is (7b, 1–2, 8a, 7b, 8a, 4a).

Number of step	Current code	List of leaves	List of triangles
1	(7b, 1–2, 8a, 7b, 8a, 4a)	3, 5, 6, 9	9, 7a
2	(1–2, 8a, 7b, 8a, 4a)	3, 5, 6	6, 1–2
3	(8a, 7b, 8a, 4a)	3, 5	5, 8a
4	(7b, 8a, 4a)	3	3, 7b
5	(8a, 4a)	7	7, 8a
6	(4a)	8	8, 4a

List of triangles does not contain vertex 4, it implies that this vertex has been hung to the edge 1–2 at the first step. So we draw the triangle with vertices 1, 2, 4. Then vertices 8, 7, 3, 5, 6, 9 have been hung to the edges 4a, 8a, 7b, 8a, 1–2, 7b correspondingly. Hang vertex 8 to the edge 4a (i. e. to the edge 2–4) and so on. The obtained triangle tree is depicted in fig. 6.

b) Since the root edge of a triangle tree may be labelled by any pair of labels, not only 1–2, and any of $2n-3$ edges in triangle tree may be chosen as the root, the relation holds

$$\binom{n}{2}\Lambda_n = (2n-3)\Delta_n.$$

3.8. This statement of J. Dénes is proven in [11] (lemma 3.15 and theorem 3.16).

3.9. Answer: $kn^{n-k-1}(n-2)(n-3)\dots(n-k)$.

Similarly to problem 2.4. We can choose a path from the first vertex to the second in $(n-2)(n-3)\dots(n-k)$ ways. The number of Prfer codes for trees that contain this path equals kn^{n-k-1}. We took this problem in [5, problem 5.93].

3.10. We took this problem in [17, p. 3.5].

3.11. [14, Corollary 3.3]. Let $V_1=[r]$, $V_2=\{r+1,\dots,r+s\}$, $V_3=\{r+s+1,\dots,r+s+t\}$ and $V=V_1\cup V_2\cup V_3$. Denote by $F^k_{r,s,t}$ the number of rooted tripartite forests on the vertex set V consisting of k trees with roots 1, 2, …, k in which the vertex number $r+1$ is a descendant of vertex 1, and by $\bar F^k_{r,s,t}$ the number of rooted tripartite forests on the vertex set V with roots 2, 3, …, $k+1$ in whcih the vertex number 1 is a descendant of vertex $r+1$. We are interested in the number of trees, i. e. $F^1_{r,s,t}$.

$$F^1_{r,s,t} \underset{(2)}{=} (s+t)^{r-1}F^r_{r,s,t} \underset{(3)}{=} (s+t)^{r-1}\bar F^r_{r,s,t} \underset{(4)}{=}$$
$$\underset{(4)}{=} (s+t)^{r-1}(r+t)^{s-1}\bar F^{r+s-1}_{r,s,t} \underset{(5)}{=} (s+t)^{r-1}(r+s+t)(r+s)^{t-1}.$$

(2) similarly to problem 1.5 for $2\le k\le r$ we obtain the recurrent formula

$$F^{k-1}_{r,s,t} = (s+t)F^k_{r,s,t}$$

and applying this formula several times we turn to the forest with r trees;

(3) apply the equality $F^r_{r,s,t}=\bar F^r_{r,s,t}$ (forests in these sets differ only by the choise of the root for the tree containing vertex 1);

(4) for $r+1\le k\le r+s-1$ we obtain the recurrent formula

$$\bar F^{k-1}_{r,s,t} = (r+t)\bar F^k_{r,s,t}$$

and again similarly to problem 1.5 turn to the forest with $r+s-1$ trees; The transition from forests with roots 1, 2, ..., r to forests with roots 2, 3, ..., $r+1$ is caused by the condition that in this recurrent equality the vertex number k has to belong to second part.

(5) the number of forests with large number of trees can be easily computed:

$$\bar{F}^{r+s-1}_{r,s,t} = (r+s)^t + t(r+s)^{t-1},$$

in the first summand the vertex 1 is a son of vertex $r+1$, and the remaining t vertices of the third part can be connected with any of $r+s$ vertices of the two other parts; in the second summand the vertex 1 is a son of some vertex from the third part, and the remaining $t-1$ vertices similarly connected with any of $r+s$ vertices.

3.12. Answer: $(2n)!/n!$. Thus the number of plane labelled unrooted trees with $n+1$ vertices is equal to $\frac{1}{n+1}\binom{2n}{n}$. [14, Corollary 4.2].

3.13. Answer: $\binom{n}{3}(3n-8)^{n-5}$ из [17].

a) The number of rooted tetrahedral trees may be counted by encoding similarly to the number of rooted triangle trees.

Construction of the code by a tree. Choose the rooted triangle of the tree, for example, triangle $(1,2,3)$. The elements of code sequence are triangles (faces of tetrahedrons). Let vertex x be hanged to the face with vertex labels a, b and c, where $a<b<c$, we call this face ancestor of vertex x. Denote the face (x,a,b) by $x\alpha$, (x,a,c) by $x\beta$, (x,b,c) by $x\gamma$. Therefore, the faces of all tetrahedrons are denoted $x\alpha$, $x\beta$, $x\gamma$, where $x=4, \ldots, n$, and the root face we denote by $(1,2,3)$. The tree has $3n-8$ faces. Any vertex of degree 3 we call a leaf of a tetrahedral tree. In each step we remove the leaf with the largest labelr and write its face-ancestor. When the last tetrahedron remains (4 vertices), we stop.

Construction of the tree by a code. Leaves are vertices of the tetrahedral tree with those labels $x \in \{4, \ldots, n\}$, for which the current code does not contain no one of the faces $x\alpha$, $x\beta$, $x\gamma$.

Example. Let $n=7$ and the code is $(4\beta,\ (1,2,3),\ 7\alpha)$.

Number of step	Current code	List of leaves	Hanged vertex
1	$(4\beta,\ (1,2,3),\ 7\alpha)$	6, 5	6
2	$((1,2,3),\ 7\alpha)$	5, 4	5
3	(7α)	4	4
4	$()$	7	7

We read the list of hanged vertices in the reverse order and obtain that vertex 7 had been hung to the face $(1,2,3)$, vertex 4 had been hung to the face $7\alpha=(7,1,2)$, vertex 5 to the face $(1,2,3)$, vertex 6 to the face $4\beta=(4,1,7)$ consequently.

The obtained tetrahedral tree is depicted in fig. 4.

Since the code contains $n-4$ faces, we have in total $(3n-8)^{n-4}$ possible codes.

b) Since the root tringle of a tetrahedral tree can be labelled by any triple of labels, not only 1, 2, 3, and each of $3n-8$ triangles in a tetrahedral tree can be assigned as the root, the relation holds:

$$\binom{n}{3}\Lambda_n = (3n-8)\Delta_n.$$

4 *Parking functions*

4.1. This famous problem has been posed in [15].

Enlarge the parking lot by appending the $(n+1)$-th parking place, and enclose the street in a cycle so that it leads from the $(n+1)$-th place to the first one. For the enlarged street we have exactly $(n+1)^n$ preference sequences (each of the n drivers independently prefers one of the $n+1$ places). Now, for any preferences all the n drivers park successfully, and one parking place remains free. The preference sequence satisfies the requirements of the initial problem if and only if the $(n+1)$-th place is remained free. Indeed, the fact that the $(n+1)$-th place remains free means that no one prefers it and no one drives by it seeking the free place for parking (in the opposite case he had to park there and not to drive by). Therefore, in this case we can remove the $(n+1)$-th parking place with the adjacent part of the road (i. e. return to initial problem), without breaking the process of parking.

Partition all preference sequences for the cyclic parking lot into $(n+1)^{n-1}$ groups consisting of $n+1$ sequences each: along with sequence $(a_1, \ldots, a_n)$ we group all its cyclic shifts, i. e. $(a_1+1, \ldots, a_n+1)$, $(a_1+2, \ldots, a_n+2)$, $\ldots$, $(a_1+n, \ldots, a_n+n)$ (sums are taken modulo $n+1$). Observe that each group contains exactly one preference sequence satisfying the requirement of the initial problem, namely the sequence for which the $(n+1)$-th parking lot remains free.

Thus, the number of preferennce sequences is equal to the number of groups, i. e. $(n+1)^{n-1}$.

4.2. It is more or less evident what happens when two consequitive cars change their order.

4.3. Answer: n^{n-1}.

Similarly to the problem 4.1, we obtain that for the extended parking street the number of all possible preference sequences is equal to $(n+1)n^{n-1}$, because the first driver can have $n+1$ favourite places, and each next driver can have n favourite places. As in problem 4.1, each group of $n+1$ sequences that differ by a cyclic shift, contains one parking function only. We took this problem from book [12].

4.4. Answer: $(n+1-m)(n+1)^{m-1}$ sequences.

Solution from book [2, problem 96.54].

For every integer m, $0 \le m \le n$, denote by $N(n,m)$ the number of preference sequences for m drivers which lead to successful parking (in particular, $N(n,0)=1$). We prove by induction on n that $N(n,m) = (n+1-m)(n+1)^{m-1}$ for all m. For $n=1$ the statement is trivial.

Induction step $(n-1) \to n$. Let k drivers like parking places greater than 1 and $m-k$ drivers like the first place $(0 \le k \le m)$. Those k drivers may be chosen in C_m^k ways. It is easily seen that, given the preferences, success or unsuccess of parking does not depend on the order the drivers arrive. So we can assume that the $m-k$ drivers that like the first place arrive the latest. Those $m-k$ drivers can park for sure — they occupy the first $m-k$ free places remainig after parking of the first k drivers. Thus, success or unsuccess of the parking process depends only on the preferences of the first k drivers (on $n-1$ places). Therefore, after we have chosen those k drivers there exist exactly $N(n-1,k)$ successful preference sequences. Thus,

$$N(n,k) = \sum_{k=0}^{m} \binom{m}{k} \cdot N(n-1,k)\,. \tag{1}$$

Apply the hypothesis of induction and transform:

$$N(n,k) = \sum_{k=0}^{m} \binom{m}{k}(n-k)n^{k-1} = \sum_{k=0}^{m} \binom{m}{k} n^k - \sum_{k=1}^{m} k\binom{m}{k} n^{k-1} = \sum_{k=0}^{m} \binom{m}{k} n^k - \sum_{k=1}^{m} m\binom{m-1}{k-1} n^{k-1} =$$

$$= \sum_{k=0}^{m} \binom{m}{k} n^k - m \cdot \sum_{k=0}^{m-1} \binom{m-1}{k} n^k = (n+1)^m - m \cdot (n+1)^{m-1} = (n+1-m) \cdot (n+1)^{m-1}\,,$$

as desired. We use here the trivial formula $k \cdot \binom{m}{k} = m \cdot \binom{m-1}{k-1}$ and the binomial formulan for expansions of $(n+1)^m$ and $(n+1)^{m-1}$ in powers of n.

4.5. Let b_1, b_2, ..., b_n be the preference sequence written in nondecreasing order. Then $b_1 = b_2 = \ldots = b_k = 1$, $2 \le b_{k+1} \le b_{k+2} \le \ldots$. Note that preference sequence b_{k+1}, b_{k+2}, ..., b_n allows $n-k$ drivers park in the street with $n-1$ parking places (from 2-nd to n-th). The number of such sequences is computed in the previous problem and is equal to $k \cdot n^{n-k-1}$. The number of ways to choose k drivers that prefer to park at the first place is equal to $\binom{n}{k}$. Therefore, the total number of parking functions for which k drivers want to park in the first place is equal to $\binom{n}{k} \cdot k \cdot n^{n-k-1} = \binom{n-1}{k-1} n^{n-k}$.

"In order to check" this result, sum up the obtained expressions over k:

$$\sum_{k=1}^{n} \binom{n-1}{k-1} n^{n-k} = (n+1)^{n-1}$$

(binomial expansion). We obtain the number of parking functions. That's good.

We see that the number of parking functions in this problem is the same as the number of forests on set $[n]$ consisting of k rooted trees (problem 3.4).

We present the bijection from [13] that gives the combinatorial proof of this fact. Let a tree with n+1 vertices be given. Construct a queue of the breadth first search by this tree. At first we add the root to the queue. Then in each step we remove the first vertex in the queue and add its sons to the queue in increasing order of their numbers. And so on while queue is not empty. Let A_i be the set of vertices added in i-th step (all these vertices are sons of some vertex), a_i be the number of elements of A_i. If in step i none of vertices had been added then A_i is empty. Since the tree is a connected graph with $n+1$ vertices, the queue of the breadth first search for it contains at least one element, while vertex $n+1$ is not removed, and the queue will be empty when we will remove all $n+1$ vertices, that means that a_i comply with restrictions:

$$\begin{cases} a_1 \ge 1 \\ a_1 + a_2 - 1 \ge 1 \\ \ldots \\ a_1 + a_2 + \ldots + a_k - k \ge 1 \\ a_1 + a_2 + \ldots + a_{n+1} = n \end{cases}$$

We define a parking function in the following way: elements of A_i be the numbers of cars that prefer to park at i-th place. It can be proved that the described mapping is a bijection between parking functions for n cars and trees with $n+1$ vertices.

4.6. It is sufficient to describe the inverse mapping, i. e. the rule that allows, given an arbitrary sequence $c = (c_1, c_2, \ldots, c_{n-1})$ where all c_i are remainders modulo $n+1$, to construct a parking function $A = (a_1, \ldots, a_n)$ for which c is a sequence of differences. It can be easily obtained from the solution of problem 4.1.

Ideed, if all the differences $c_i = a_{i+1} - a_i \pmod{n+1}$ are fixed, then there exist exactly $n+1$ preference sequences with those differences for a cyclic parking street. In solution 4.1 those $(n+1)$ sequences are united in one group and it is proved that exactly one parking function belongs to this group. Obviously, that is what we wanted.

4.7. The sum counts all possible parking functions. In fact, to determine a parking function one may start with choosing, for each parking place i, the number k_i which shows how many drivers like that place. Non negative integer numbers k_i should comply with the following restrictions:

$k_n \le 1$ since in the opposite case there exist two drivers who wish park in the last place, so the parking will fail;

$k_{n-1} + k_n \le 2$ since in the opposite case there exist three drivers who wish to park in two last places, so the parking will fail;

and so on till the inequality $k_2 + \ldots + k_{n-1} + k_n \le n-1$.

After numbers k_2, k_3, ..., k_n had been chosen, let $k_1 = n - k_2 - k_3 - \ldots - k_n$ and assign the last k_1 drivers to prefer the first place.

It is easy to understand that the numbers k_1, k_2, ..., k_n satisfying the above restrictions produce a parking function. Indeed, by problem 4.2 the success of the parking process does not depend on the order in which cars enter. Let those who want to park in n-th place go first, then those who want to park in $(n-1)$-th place go after them, and so on. They will perfectly park all cars.

It remains to observe that the number of ways to realize the set k_1, k_2, ..., k_n as a parking functions is equal to the term under the summation sign.

4.8. Solution 1 (from book [8, problem 5.49.f]). Let k_i be the number of elements of sequence a that are equal to i, in particular, $k_n = 0$. The condition on a preference sequence to be a sure parking function is equivalent to the following restrictions:

1) all partial sums of the sequence $k_1 - 1$, $k_2 - 1$, ..., $k_{n-1} - 1$ are positive;
2) $\sum\limits_{i=1}^{n-1}(k_i - 1) = 1$.

Lemma. For any finite integer sequence with sum 1 there exists a unique cyclic permutation such that all its partial sums are positive.

The proof of the lemma is left to the reader.

Obviously, in the sure preference sequence all a_i are numbers from 1 to $n-1$. If we take an arbitrary sequence of n numbers from $[n-1]$, then then numbers k_i defined by this sequence satisfy the condition 2), but, generally speaking, do not satisfy the condition 1). But by the lemma this sequence has a cyclic permutation (exactly one) satisfying the first condition!

Therefore, the number of sure parking functions is $n-1$ times less then the number of elements of $[n-1]^n$.

Solution 2 (direct computation). Consider an arbitrary sure parking function, let k drivers want to park at the first place for this function ($2 \le k \le n$). We give a red hat to one of them and remove him. The preferences of the remaining drivers form a usual (not necessarily sure) parking function in the street with $n-1$ parking places and $k-1$ drivers that prefer to park in the first place. So, to obtain a parking function for which k drivers want to park in first place, we have to choose one driver in hat (n ways) and assign prefernces to others (by problem 4.5 it can be done by $(n-1)^k\binom{n-2}{k-2}$ ways). As a result we obtain not just a parking function, but a parking function such that one of k drivers pretending on the first parking place wears a red hat. Thus, the total number of parking functions is expressed by the sum $\sum\limits_{k=2}^{n}\frac{n}{k}\cdot(n-1)^{n-k}\binom{n-2}{k-2}$. It remains to check the identity

$$\sum_{k=2}^{n}\frac{n}{k}\cdot(n-1)^{n-k}\binom{n-2}{k-2} = (n-1)^{n-1}.$$

Transform the left hand side of the expression: replace the summation index k by $i = n-k$, apply the equality $\frac{n}{n-i}\cdot\binom{n-2}{i}(n-1) = (n-1-i)\binom{n}{i}$ and after that split each summand as a difference of two terms:

$$\sum_{k=2}^{n}\frac{n}{k}\cdot(n-1)^{n-k}\binom{n-2}{k-2} = \sum_{i=0}^{n-2}\frac{n}{n-i}\cdot\binom{n-2}{i}(n-1)^i = \sum_{i=0}^{n-2}(n-1-i)\cdot\binom{n}{i}(n-1)^{i-1} =$$
$$= \sum_{i=0}^{n-2}\binom{n}{i}(n-1)^i - \sum_{i=0}^{n-2}\binom{n}{i}\cdot i(n-1)^{i-1}.$$

By the binomial formula the first sum is equal to

$$n^n - \binom{n}{n}(n-1)^n - \binom{n}{n-1}(n-1)^{n-1}. \qquad (2)$$

For the second sum observe that the 0-th summand vanishes, apply formula $\binom{n}{i}\cdot i = n\binom{n-1}{i-1}$, and after that apply the binomial formula:

$$\sum_{i=0}^{n-2}\binom{n}{i}\cdot i(n-1)^{i-1} = n\sum_{i=1}^{n-2}\binom{n-1}{i-1}(n-1)^{i-1} = n\Big(n^{n-1}-\binom{n-1}{n-1}(n-1)^{n-1}-\binom{n-1}{n-2}(n-1)^{n-2}\Big). \tag{3}$$

It remains to note that the difference of expressions (2) and (3) is equal to $(n-1)^{n-1}$.

4.9. Take an arbitrary parking function. Let b_1, b_2, ..., b_{n+1} be a sequence of numbers of preferred places, written in increasing order. The success of the parking process means that the following inequalities hold: $b_1 \le 1$ (and hence $b_1 = 1$), $b_2 \le 2$, ..., $b_{n+1} \le n+1$.

Choose the largest k, $0 \le k \le n$, such that $b_{k+1} = k+1$. Then the preference sequence b_1, ..., b_k is a parking function for the street with k parking places, and the sequence $b_{k+1}-k$, $b_{k+2}-k$, ..., $b_{n+1}-k$ is a sure parking function for the street with $n+1-k$ parking places. (It is a sure function due to maximality of k.)

Thus, in order to construct an arbitrary parking function in the street with $n+1$ parking places, one may choose an arbitrary k, choose arbitrary k drivers (in $\binom{n+1}{k}$ ways), assign one of the P_k parking functions for their preferences, and assign one of the $(n-k)^{n-k}$ sure parking functions to determine the preferences of the other $n+1-k$ drivers. Thus, we obtain a combinatorial interpretation of the formula under consideration.

4.10. Similarly to problem 4.7, the formula counts the number of sure parking functions for the parking lot of length n.

4.11. In [10] this formula is proved by a complicated bijection with the set of labelled trees. We prove it similarly to the previous problem.

Say that an *enchanced* parking function is a parking sequence $(a_1, \dots, a_n)$ in which all the elements which equal 1 are numbered. For example, $(\underset{1}{1}, 2, \underset{2}{1}, 3, 3)$ and $(\underset{2}{1}, 2, \underset{1}{1}, 3, 3)$ are two different enchanced parking functions (for $n=5$). We denote by $N^*(n)$ the number of enchanced parking functions for the parking lot of length n.

By arguments as in problem 4.7 we obtain that the sum counts the number of enchanced parking functions. It remains to check that $N^*(n) = n^n$.

Consider an enchanced parking sequences in which k drivers like NOT the first place, and $n-k$ drivers prefer to park in the first place. Similarly to problem 4.5we conclude that the number of such sequences equals $\frac{n!}{k!}N(n-1,k)$. Then the total number of enchanced parking functions is given by the formula similar to formula (1):

$$N^*(n) = \sum_{k=0}^{n}\frac{n!}{k!}\cdot N(n-1,k) = \sum_{k=0}^{n}\frac{n!}{k!}\cdot(n-k)n^{k-1}.$$

That is a telescoping series, it is equal to n^n.

5 Inversions on trees and deficiencies of parking functions

5.1. Both statements are inspired by [16].

a) Consider an arbitrary labelled tree with n vertices. Let n–u be the first edge on the path from vertex n to vertex 1 (the case $u=1$ is possible). Remove this edge, the tree breaks up into the two trees A_n and A_1. Vertex u is singled out in tree A_1 in a natural way, i.e. we consider this tree as rooted. Let tree A_1 have $k+1$ vertex, $0 \le k \le n-2$. Then tree A_n has $n-k-1$ vertices. Obviously, the labelled forest (A_n, A_1) is uniquely defined by the initial labelled tree with n vertices.

Now we demonstrate how the inverse mapping is constructed. For all k, $0 \le k \le n-2$, choose vertices for tree A_1. Since vertex 1 has to be in tree A_1, and vertex n has to be in tree

A_n, we need to choose k vertices from the elements of set $[n]$, except 1 and n. This can be done in $\binom{n-2}{k}$ ways; the other $n-k-1$ vertices belong to tree A_n. Then construct trees A_1 and A_n themselves, it can be done in $(k+1)T_{k+1}$ and T_{n-k-1} ways, respectively. Finally, we need "to fasten" the trees, so we need to draw an edge connecting vertex n with the root of tree A_1. We obtain a combinatorial interpretation of the k-th summand on the right hand side of the identity: it is equal to the number of spanning trees on set $[n]$ consisting of exactly two trees such that one tree contains $k+1$ vertices (including vertex n), and the other tree is rooted and contains vertex 1.

b) Consider an arbitrary parking function $a = (a_1, a_2, \ldots, a_{n+1})$ for a street with $n+1$ parking places. Let drivers enter the street in increasing order of their numbers, and assume that the last of them parks on place k. Fix the preferences of the first n drivers and try to increase the preference of the last driver as much as posiible, i. e. find a maximum value of a_{n+1} for which the sequence $(a_1, a_2, \ldots, a_{n+1})$ is a parking function. It is clear that:

— this maximum value is $a_{n+1} = k$,

— none of the other drivers prefers the k-th place, and

— no one of those who found his favourite place occupied drives by the k-th place.

It follows that exactly $k-1$ drivers prefer the places from 1-st to the $(k-1)$-th, and exactly $n+1-k$ drivers prefer the places from the $(k+1)$-th to the $(n+1)$-th. Let $\widetilde{a} = (a_1, a_2, \ldots, a_n, k)$, call this sequence the *enlargement* of the sequence a.

Obviously, for any $\ell \le k$ the sequence $(a_1, a_2, \ldots, a_n, \ell)$ is a parking function, and all k such parking functions (and only them) have the same enlargement function $\widetilde{a}$.

How to construct a sequence $\widetilde{a}$? The last driver is singled out from the beginning. We need to choose $k-1$ drivers who will park in the first $k-1$ places ($\binom{n}{k-1}$ ways) and assign them a parking function (P_{k-1} ways). Then we need to assign a parking function for those who park in the places after the k-th ($P_{n-(k-1)}$ ways). So, there are $\binom{n}{k-1}P_{k-1}P_{n-(k-1)}$ ways to choose a sequence $\widetilde{a}$. Remembering that $\widetilde{a}$ is the enlargement sequence for k different parking functions, we conclude that the number of parking functions P_{n+1} is computed by the formula

$$P_{n+1} = \sum_{k=1}^{n+1} \binom{n}{k-1} k P_{k-1} P_{n-(k-1)}.$$

5.2. Answer: $F_n^*(x) = \sum\limits_{k=0}^{n} \binom{n}{k} F_k(x) F_{n-k}(x)$.

5.3. Both statements are from [16]. These recursions are detailed versions of the recursions from problem 5.1.

a) Note that in the definition of $\mathrm{inv}(T)$ the labels of vertices can be considered as arbitrary real numbers. Besides that, let us demand that the root has no contribution to the number $\mathrm{inv}(T)$, therefore the number $\mathrm{inv}(T)$ does not depend on the root label.

We give a useful definition. Consider a labelled rooted tree T on set $[\ell]$ and fix a set of numbers $S = \{s_1, s_2, \ldots, s_\ell\}$. Define a collection consisting of ℓ rooted trees T_1, T_2, …, T_ℓ, such that the vertices of each of them are labelled by the elements of set S. Tree T_i is obtained from tree T by replacing its labels by labels from set S in the following way. The root of tree T has some label, replace it by label s_i. Arrange the other elements of the set S in increasing order and place them in unrooted vertices of tree T in the same way "as it has been done in tree T": the smallest label is put at vertex 1, the next label is put at vertex 2, etc. Is is evident that the numbers of inversions in all trees in this collection coincide. Call the set of trees of this collection *equally-inversional* to each other.

Now return to the solution. Append the inversion counting to the reasonings of solution 5.1 a). In order not to change notations, we will prove the required relation with n replaced with $n-2$ (recall that polynomial F_n is generated by the set of forests with $n+1$ vertices):

$$F_{n-1}(x) = \sum_{k=0}^{n-2} \binom{n-2}{k} (x^k + x^{k-1} + \ldots + 1) F_k(x) F_{n-k-2}(x).$$

So in solution 5.1 a) all possible labelled trees are in one to one correspondence with forests (A_n, A_1). To define forest (A_n, A_1), we need to choose k vertices for tree A_1 from all elements of set $[n]$ except 1 and n, (we have $\binom{n-2}{k}$ ways, the other $n-k-1$ vertices automatically belong to tree A_n) and construct trees A_1 and A_n themselves (the tree A_1 is rooted). Then we "fasten" trees by drawing an edge that connects vertex n with the root of tree A_1.

To count the inversions, fix trees A_n and A_1 and group in one cluster the $k+1$ forests of the form (A_n, A), where A ranges over the set of rooted trees which are equally-inversional to tree A_1. The number of inversions in each tree $T \in \mathcal{T}_n$ obtained in this way is equal to $\mathrm{inv}(A_n)+\mathrm{inv}(A)+\delta_A$, where the correction term δ_A equals the number of inversions of the root vertex of A with other vertices of A (being the root of tree A, it does not participate in the counting of the value $\mathrm{inv}(A)$, but after fastening the trees it participates). As we noted, $\mathrm{inv}(A) = \mathrm{inv}(A_1)$ for all trees A that are equally-inversional to tree A_1. As for the term δ_A, it is clear that it takes each of the values 0, 1, 2, ..., k exactly once, as A runs over the set of all trees that are equally-inversional to tree A_1.

Each forest (A_n, A) of our cluster determines summands $x^{\mathrm{inv}(A_n)}$, $x^{\mathrm{inv}(A)}$ in the enumerators $F_{n-k-2}(x)$, $F_k(x)$. The exponent of their product $x^{\mathrm{inv}(A_n)+\mathrm{inv}(A)}$ equals the number of inversions of the obtained graph without taking into account the correction term. Summing the contributions of the trees that are equally-inversional to tree A_1 we obtain the expression

$$(x^k + x^{k-1} + \ldots + 1)x^{\mathrm{inv}(A_n)+\mathrm{inv}(A)},$$

which equals the sum of monomials of the enumerator $F_{n-1}(x)$ for the trees generated by our cluster. Summing over all clusters we obtain the required formula.

b) We append the deficiency counting to the reasonings of solution 5.1 b). In that solution the set of parking functions is partitioned into clusters. A cluster generated by a parking function $a = (a_1, a_2, \ldots, a_{n+1})$ consists of all parking functions $(a_1, a_2, \ldots, a_n, i)$ where $1 \le i \le k$, and k is the maximal possible preference of $(n+1)$-th driver for which the sequence $(a_1, a_2, \ldots, a_n, k)$ is a parking function.

For counting all possible parking functions with given parameter k we consider all possible functions a' for those drivers who park in the first $k-1$ places, and all possible functions a'' for those who park in the last $n-(k-1)$ places. Roughly speaking, the deficiency of these parking cases is equal to $D(a')$ and $D(a'')$, and in the language of enumerators it equals $x^{D(a')}$ and $x^{D(a'')}$. As for the $(n+1)$-th driver who parks in the k-th place for all functions of our cluster, his contribution to the whole deficiency for functions of our cluster takes each of the values 0, 1, 2, ..., k exactly once. Thus, the total deficiency of parking functions from one cluster is expressed by $(x^{k-1}+x^{k-2}+\ldots+1)x^{D(a')}x^{D(a'')}$. Summing over all clusters we obtain the required formula.

5.4. See [19].

References

[1] *Айгнер М., Циглер Г.* Доказательства из Книги. М.:Мир, 2006.

[2] *Берлов С. Л., Иванов С. В., Кохась К. П.* Петербургские математические олимпиады. СПб.: "Лань", 2000.

[3] *Глибичук А. А., Дайняк А. Б., Ильинский Д. Г., Купавский А. Б., Райгородский А. М., Скопенков А. Б., Чернов А. А.* Элементы дискретной математики в задачах. М.:МЦНМО, 2016.

[4] *Иванов О. А.* Элементарная математика для школьников, студентов и преподавателей. М.:МЦНМО, 2009.

[5] Комбинаторный анализ. Задачи и упражнения. / Под. ред. К. А. Рыбникова. М.: Наука, 1982.

[6] *Ландо С. К.* Введение в дискретную математику. М.: МЦНМО, 2012.

[7] *Сачков В. Н.* Введение в комбинаторные методы дискретной математики. М.: МЦНМО, 2004.

[8] *Стэнли Р.* Перечислительная комбинаторика. Т. 2. М.: Мир, 2005.

[9] *Beineke L. W., Moon J. W.* Several proofs of the number of labelled 2-dimensional trees. In "Proof Techniques in Graph Theory" (F. Harary, editor). New York: Academic Press, 1969. P. 11-20.

[10] *Benjamin A. T., Juhnke F.* Another way of counting N^N. // SIAM J. Disc. Math. 1992. V. 5. No. 3. P. 377–379.

[11] *Bóna M.* Combinatorics of permutations. Chapman & Hall/CRC, 2004.

[12] *Bóna M.* A walk through combinatorics. 3rd ed. World Scientific Publ. Co., 2011.

[13] *Chassaing P. , Marckert J.-F.* Parking functions, empirical processes, and the width of rooted labelled trees // Electron. J. Combin. 2001. V. 8. № 1. Research Paper 14.

[14] *Guo S., Guo V.* A recursive algorithm for trees and forests // `http://arxiv.org/pdf/math/1702.01744v1.pdf`

[15] *Konheim A. G., Weiss B.* An occupancy discipline and applications. // SIAM J. Appl. Math. 1966. Vol. 14. № 6. P. 1266–1274.

[16] *Kreweras G.* Une famille de polynômes ayant plusieurs propriétés énumeratives // Period. Math. Hungar. V. 11. 1980. № 4. P. 309–320.

[17] *Moon J. W.* Counting labelled trees. William Clowes and Sons, 1970.

[18] *Shukla A.* A short proof of Cayley's tree formula // `https://arxiv.org/pdf/0908.2324.pdf`

[19] *Yan C. H.* Parking functions// in Handbook of Enumerative Combinatorics. M. Bóna, ed.