MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
State institution:
‘SOUTH UKRAINIAN NATIONAL PEDAGOGICAL UNIVERSITY NAMED AFTER K.D. USHINSKY’

Discrete mathematics
Lecture notes
(part 2)

Odesa 2025

UDC 519.101

Recommended for publication by the Academic Council of the State Institution ‘K.D. Ushynsky South Ukrainian National Pedagogical University’

Compiled by:

Doctor of Physical and Mathematical Sciences, Professor, Head of the Department of Higher Mathematics and Statistics Viacheslav Mykolaiovych Pyvovarchyk;

Doctor of Physical and Mathematical Sciences, Associate Professor Dmytro Semenovych Kaliuzhnyi-Verbovetskyi.

Phylosophy Doctor in Mathematics, Assistant Professor Anastasia Igorivna Dudko.

Lecture notes in English are intended for students of higher education institutions, in particular for students of the Educational and Research Institute of Natural and Mathematical Sciences, Informatics and Management of the Ushynsky University, who study the discipline ‘Discrete Mathematics’. The lecture notes (part 2) cover such a chapter as the theory of recurrence relations.

CONTENTS

CHAPTER 2. RECURRENCE RELATIONS

§ 2.1. Some examples

Let's consider sequences of numbers and the ways they can be defined. If we know how to calculate the element of the sequence a_n for each value of n, then such a sequence is called an explicit sequence. Often, a sequence is set explicitly using a formula for calculating a sequence member depending on its number. For example, a sequence of non-negative rational numbers given by the formula $a_n = \frac{1}{n}$, with $n = 4$ gives $a_4 = \frac{1}{4}$. However, explicitly specifying a sequence is not always convenient for solving problems. There is a way to recurrently specify the sequence: each element of the sequence is calculated by its number and one or more preceding elements of this sequence. For example, we can define a factorial in a recursive way as follows:

$n! = (n-1)! \cdot n$ for $n > 0$ with the initial condition 0!=1.

It is clear that to calculate any element of a sequence using a given recurrence relation, you need to calculate all its preceding elements.

Example 1: Hanoi Towers.

Let's start with the famous puzzle problem invented by the French mathematician Edouard Lucas in 1883. The problem was as follows: three rods are given, one of which has eight discs of different sizes strung on it, and always a smaller disc lies on top of a larger one. The task is to move the pyramid of eight discs in the least number of moves to another rod. Only one disc is allowed to be moved at a time, and you cannot place a larger disc on top of a smaller one.

The legend of Professor Lukas' puzzle reads that 'in the Great Temple of Benares, under the cathedral that marks the centre of the world, there is a bronze disc on which are fixed three diamond rods, one cubit high and as thick as a bee. A long time ago, at the very beginning of time, the monks of this monastery were in debt to the god Brahma. Angry, Brahma built three tall rods and placed 64 discs of pure gold on one of them. Each smaller disc lay on top of the larger one. As soon as

all 64 discs are transferred from the rod on which Brahma put them when he created the world to another rod, the tower and the temple will turn to dust and the world will perish with thunder.' It is usually suggested to estimate the complexity of the solution.

As we will show below, the number of movements is $a_n = 2^{64} - 1 = 18446744073709551615$ and, if we move the discs one time per second, the required movements would require $5{,}82 \cdot 10^{11}$ actions, i.e. the monks' work would last 584 billion years.

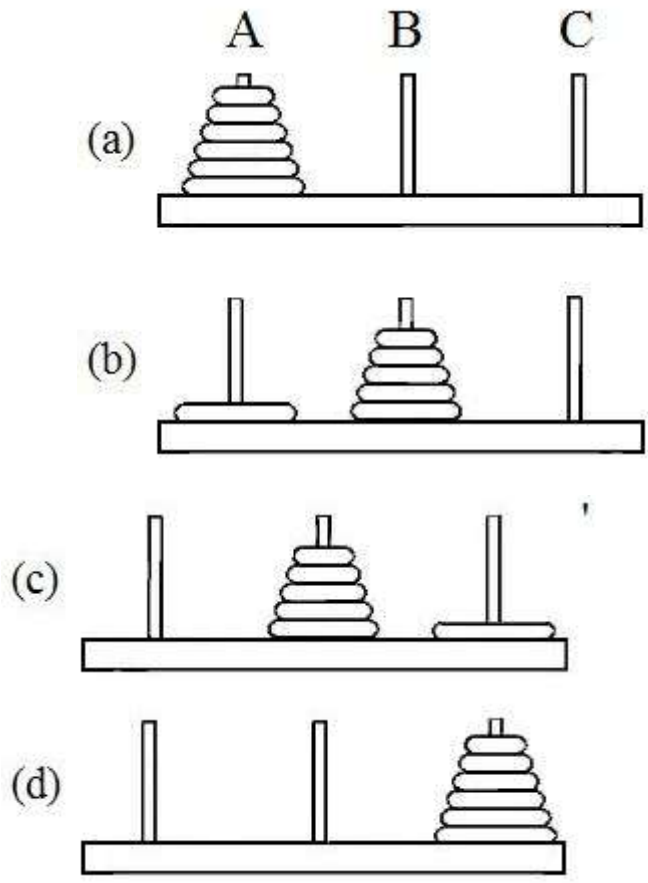


Figure 1. Hanoi towers (6 discs)

We will consider the solution to the problem in the general case when instead of eight discs, we will be moving n discs.

There are n discs of different diameters with holes in the centres and three vertical rods on which to place the discs. Initially, all the discs are on the same rod, ordered by size, with the largest at the bottom, forming a tower. The goal is to move the discs one by one to get the same tower on another rod. At no time should a disc

of a larger diameter lie on a disc of a smaller diameter. What is the minimum number of actions required?

Let's denote by a_n the required minimum number of actions to be performed. It is clear that $a_1 = 1$ and $a_2 = 3$. How do we find a_n? Obviously, in order to move the lowest disc, you need an empty rod on which to place the lowest disc. Therefore, the n-1st disc must be placed on the third rod. This requires a_{n-1} actions. Next, we move the lowest disc from the first rod to the second rod and then we need a_{n-1} more actions to move all the other discs from the third rod to the second rod. Thus, $a_n = 2a_{n-1} + 1$.

This recurrent relation $a_n = 2a_{n-1} + 1$ together with the initial condition $a_1 = 1$ makes it possible to find a_n.

We have, $a_2 = 2 \cdot 1 + 1 = 3, a_3 = 2 \cdot 3 + 1 = 7, a_4 = 2 \cdot 7 + 1 = 15$ ...

Note that $a_2 = 3 = 2^2 - 1$, $a_3 = 7 = 2^3 - 1$, etc. Using the method of mathematical induction, it is easy to prove that $a_n = 2^n - 1$.

Such a reasoning in the process of solving the Hanoi Towers problem, on the other hand, can be called recursive: the process ('moving n discs') contains itself ('moving n-1 disc') and contains a condition ('moving the lowest disc to the empty second rod in only one move') that can be used to determine when the task is complete. This condition is called a basic or nonrecursive condition.

Some functions also allow for a recursive definition, although they can be defined directly.

The possibility of a recursive function definition is not always obvious. For example, the function $f(n) = a^n, n \in N$, allows the following recursive definition

$$f(0) = 1, \quad f(k+1) = a \cdot f(k), \qquad k \in N.$$

The first condition is the basic one.

The *recursive algorithm* for solving problems is often and effectively used in many languages and environments of programming.

In programming, *recursion* means calling a function or procedure from itself, directly (simple recursion) or through other functions (complex or indirect

recursion). The number of nested calls to a function or procedure is called the depth of recursion. The use of recursion can be structurally simpler and more visual, especially if the programming language syntax does not contain cyclic structures for organising repetitive calculations. However, you should avoid using a large depth of recursion in recursive programs.

It should be noted that if the organisation of data processing is arranged in such a way that actions are repeated many times, but do not lead to challenges of their own, this is iteration (in the broad sense). In a narrow sense, an iteration is a single step in a cyclic process. In mathematics, iteration is the repeated application of any mathematical operation.

Example 2. There are 3^n *n*-digit sequences with each digit equal to 0, 1 or 2. How many of them have an odd number of zeros?

Let $b_n$denote the number of such sequences of length n that have an odd number of zeros. Each such sequence ends in 0, 1 or 2. A sequence ending in 1 has b_{n-1} possible sequences preceding the last 1. Similarly, there are b_{n-1} sequences ending in 2. If the sequence ends in 0, then this 0 must be preceded by a sequence of length $n-1$ with an even number of zeros, but the number of such sequences is 3^{n-1} (the total number of sequences of length $n-1$) minus b_{n-1} (the number of sequences of length $n-1$ with an odd number of zeros). Thus, there are $3^{n-1}-b_{n-1}$ sequences of length *n* with an odd number of zeros that have 0 at the end. By the principle of addition, we have

$$b_n = b_{n-1} + b_{n-1} + 3^{n-1} - b_{n-1}, \text{ i.e., } b_n = b_{n-1} + 3^{n-1}.$$

We can find b_n by iteration:

$$b_n = 3^{n-1} + b_{n-1} = 3^{n-1} + (3^{n-2} + b_{n-2}) = \cdots$$

$$\ldots = 3^{n-1} + 3^{n-2} + \cdots + 3 + b_1.$$

But $b_1 = 1$, so that $b_n = 1 + 3 + \ldots + 3^{n-1} = \frac{1}{2}(3^n - 1)$.

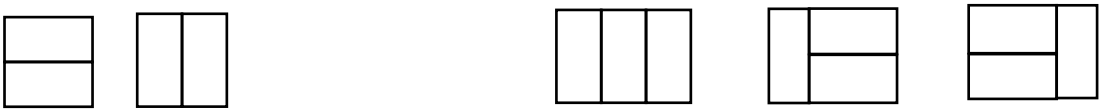
Example 3: Paving a garden path.

The garden path is 2 meter wide and *n* meter long. The stones for paving are 1 meter by 2 meter

. Find the number of ways to pave the path.

Let p_n denote the number of ways to pave a path of size 2 x *n*.

It is clear that $p_1 = 1$. It is also clear that $p_2 = 2$ (see Figure 1.a), and the value of $p_3 = 3$ (see Figure 1.b).



а) б)

Fig. 1. The number of ways to pave a path of two or three stones

It may seem that $p_n = n$, but this is not the case (see below). For a 2 x *n* path (n>2), paving should begin with one of the methods shown in Fig. 2.

Fig. 2. Possible ways to start paving a path 2 x *n*

In the first case, there are p_{n-1} ways of further paving, and in the second case, there are p_{n-2}. By the principle of addition

$$p_n = p_{n-1} + p_{n-2} \ (n \geq 3).$$

This is a 2nd order recurrence relation, since each number of ways is expressed in terms of the preceding two. We get

$p_4 = 3 + 2 = 5$, $p_5 = 5 + 3 = 8$, $p_6 = 5 + 8 = 13$, $p_7 = 13 + 8 = 21$, etc.

This is the well-known Fibonacci sequence (F_n):

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

Example 4. **Colouring the flag.**

The flag consists of n horizontal stripes, each stripe can be either red, blue or white, and adjacent stripes must not be of the same colour. Under these conditions, the topmost stripe can be any of three colours, the second can be two colours, the third can be two colours, etc. (the colour of a stripe must not be the same as the colour of the stripe above it). Thus, there are

$3 \cdot 2^{n-1}$ possible colouring options.

Now let's imagine that in order to avoid confusing the bottom and top, the bottom and top stripes should be different colours. Let a_n denote the number of such flags with n stripes. Then $a_1 = 0$, and $a_2 = 6$. Furthermore, since there is a one-to-one correspondence between flags with n stripes, where the top and bottom stripes are the same colour, and flags with $n-1$ stripes, where the bottom stripe is different from the top stripe, $a_n = 3 \cdot 2^{n-1} - c_n$, where c_n is the number of flags with n stripes with the same colour of the bottom and top stripes. Note that the number of flags of n stripes with the same colour of the bottom and top stripes is equal to the number of flags of $n-1$ stripes with different colours of the bottom and top stripes (the bottom strip of a flag of $n-1$ stripes must not be the same colour as the n-th strip of a flag of n stripes). Thus, $c_n = a_{n-1}$.

Therefore, $a_n = 3 \cdot 2^{n-1} - a_{n-1}$. Because from here $a_n + a_{n-1} = 3 \cdot 2^{n-1}$, we also have $a_{n-1} + a_{n-2} = 3 \cdot 2^{n-2}$. Thus,

$$2(a_{n-1} + a_{n-2}) = 3 \cdot 2^{n-1} = a_n + a_{n-1},$$

and $a_n = a_{n-1} + 2a_{n-2}$.

The result is a recurrence relation of the 2nd order. Now we will see how it can be solved.

§ 2.2. Solving recurrence relations using the characteristic equation

There are two main methods used to solve recurrence relations:

1) solving with the help of a characteristic equation;

2) solving with the help of a generating function.

In this section, we will consider solving recurrence relations using the characteristic equation. This method almost completely coincides with the method of solving linear differential equations with constant coefficients.

Definition. A recurrent relationship described by an equation of the form

$$a_n = \mathrm{B}_1 a_{n-1} + \mathrm{B}_2 a_{n-2} + \cdots + \mathrm{B}_k a_{n-k}, n \geq k, \qquad (2.1)$$

is called a homogeneous linear recurrence relation of the k-th order with constant coefficients.

The constants present in this equation, $B_1, B_2, \ldots, B_k$, are called the coefficients of the equation.

To get a more familiar form of the homogeneous equation, we move all the terms of equation (2.1) to the left side and redesignate the coefficients:

$$a_n + A_1 a_{n-1} + A_2 a_{n-2} + \cdots + A_k a_{n-k} = 0,\ n \geq k. \qquad (2.2)$$

For example, a geometric progression is defined by a linear homogeneous recurrence relation of the 1st order using the equation:

$$u_{n+1} = q u_n.$$

An arithmetic progression is defined by a linear homogeneous recurrence relationship of the 2nd order. Indeed, if we consider two relations written for two neighbouring values of *n*:

$$u_{n+2} = u_{n+1} + d \quad \text{and} \quad u_{n+1} = u_n + d,$$

then we will obtain from them by subtracting:

$$u_{n+2} - u_{n+1} = u_{n+1} - u_n,$$

or

$$u_{n+2} = 2u_{n+1} - u_n.$$

Definition. An algebraic equation of the form

$$x^k + A_1 x^{k-1} + A_2 x^{k-2} + \cdots + A_k = 0 \qquad (2.3)$$

is called the characteristic equation of the linear homogeneous recurrence relation (2.2).

Remark. It is obvious that for the linear homogeneous recurrence relation (3.1), the characteristic equation will be of the form

$$x^k = B_1 x^{k-1} + B_2 x^{k-2} + \cdots + B_k. \qquad (2.4)$$

According to the fundamental theorem of algebra, the characteristic equation (2.3) has k roots (real or complex). These roots play a decisive role in finding a sequence that corresponds to a given recurrence relation. In the following, we will focus on cases where the characteristic equation has only real roots. We will not consider the case of complex roots.

Taking the example of a linear homogeneous recurrence relation of the 2nd order

$$a_n = Aa_{n-1} + Ba_{n-2}, (n \geq 3), \quad (2.5)$$

where A and B are constant values, $B \neq 0$ and a_1 i a_2 are given, we will justify the method of solving the problem using the characteristic equation.

First, are there any real numbers $\alpha \neq 0$ such that the equality $a_n = \alpha^n$ satisfies (2.5)? Substituting $a_n = \alpha^n$ into the relation (2.5), we obtain

$$\alpha^n = A\alpha^{n-1} + B\alpha^{n-2},$$

i.e.

$$\alpha^{n-2}(\alpha^2 - A\alpha - B) = 0.$$

Hence, either $\alpha = 0$, but this is possible only if $a_1 = 0$ and $a_2 = 0$, (in this case, $a_n = 0$) or $\alpha^2 - A\alpha - B = 0$.

Thus, if $a_1 \neq 0$ or $a_2 \neq 0$, then $a_n = \alpha^n$ is a solution to equation (2.5) in those cases and only those cases when α is a solution to the auxiliary (characteristic) equation

$$x^2 = Ax + B\ . \quad (2.6)$$

As a consequence, if α and β – are different roots of equation (2.6), then $a_n = \alpha^n$ and $a_n = \beta^n$ are both solutions of equation (2.5).

If the auxiliary equation has a multiple root of α, then

$$x^2 - Ax - B = (x - \alpha)^2 = x^2 - 2\alpha x + \alpha^2\ , \quad A = 2\alpha, \qquad B = -\alpha^2.$$

In this case, $a_n = n\alpha^n$ also satisfies equation (2.5), since

$$Aa_{n-1} + Ba_{n-2} = A(n-1)\alpha^{n-1} + B(n-2)\alpha^{n-2} =$$
$$= 2(n-1)\alpha^n - (n-2)\alpha^n = n\alpha^n = a_n.$$

Let us consider two theorems that describe the structure of a general solution to a homogeneous linear recurrence relation of the *k*--th order, depending on the type of roots of the characteristic equation.

Theorem 2.1 If the characteristic equation of a homogeneous recurrence relation of the *k*-th order has *k* different roots $\lambda_1, \lambda_2, \ldots, \lambda_k$, then the general solution of the homogeneous recurrence relation has the form:

$$a_n = C_1\lambda_1^n + C_2\lambda_2^n + \cdots + C_k\lambda_k^n\ ,$$

where $C_1, C_2, \dots, C_k$ – are arbitrary constants.

Remark. To determine the constants $C_1, C_2, \dots, C_k$ *(finding a partial solution of the recurrence relation), it is necessary to specify k initial conditions, i.e., specify k elements of the sequence* $a_1, a_2, \dots, a_k$.

Theorem 2.2. If the characteristic equation of a homogeneous recurrence relation of order *k* has a root λ of multiplicity *k*, the general solution of the homogeneous recurrence relation has the form:

$$a_n = (C_1 + C_2 n + \cdots + C_k n^{k-1})\lambda^n ,$$

where $C_1, C_2, \dots, C_k$ – arbitrary constants.

We will prove the theorems for the case k=2. For other values of *k*, the proofs are analogous.

Theorem 2.3. Let the sequence $\{a_n\}$ satisfy equation (2.1), the initial values a_1 та a_2 be defined, α and β be the roots of the characteristic equation of the sequence, then:

1) if $\alpha \neq \beta$ (the roots are real and distinct), then there exist constants K_1 і K_2 such that

   $a_n = K_1\alpha^n + K_2\beta^n$ for all $n \geq 1$;

2) if $\alpha = \beta$ (multiple root), then there exist constants K_3 і K_4 such that $a_n = (K_3 + nK_4)\alpha^n$ for all $n \geq 1$.

Proof

1) Let's choose K_1 і K_2 so that $a_1 = K_1\alpha + K_2\beta$, $a_2 = K_1\alpha^2 + K_2\beta^2$, that is, let's take $K_1 = \frac{a_1\beta - a_2}{\alpha(\beta-\alpha)}, K_2 = \frac{a_1\alpha - a_2}{\beta(\alpha-\beta)}$.

Then, obviously, the statement of the theorem is valid for $n = 1, 2$. Next, we apply induction.

Let the statement be true for every $n \leq k$. Then

$$a_{k+1} = Aa_k + Ba_{k-1} = A(K_1\alpha^k + K_2\beta^k) + B(K_1\alpha^{k-1} + K_2\beta^{k-1})$$
$$= K_1\alpha^{k-1}(A\alpha + B) + K_2\beta^{k-1}(A\beta + B) = K_1\alpha^{k+1} + K_2\beta^{k+1}.$$

Statement 1) is proven.

2) Let's choose K_3 i K_4 so that $a_1 = (K_3 + K_4)\alpha$ and $a_2 = (K_3 + 2K_4)\alpha^2$, that is, so that $K_3 = \frac{2a_1\alpha - a_2}{\alpha^2}, K_4 = \frac{a_2 - a_1\alpha}{\alpha^2}$.

Then the statement that $a_n = (K_3 + nK_4)\alpha^n$, is obviously valid for the values $n = 1,2$.

Let's assume it is true for every $n \leq k$. Then we have that

$$\begin{aligned} a_{k+1} &= Aa_k + Ba_{k-1} = A(K_3 + kK_4)\alpha^k + B(K_3 + (k-1)K_4)\alpha^{k-1} \\ &= K_3\alpha^{k-1}(A\alpha + B) + K_4\alpha^{k-1}(Ak\alpha + B(k-1)) \\ &= K_3\alpha^{k+1} + K_4\alpha^{k-1}\left(2k\alpha^2 - \alpha^2(k-1)\right) \\ &= K_3\alpha^{k+1} + K_4(k+1)\alpha^{k+1}. \end{aligned}$$

Statement 2) is proven.

A description of these cases can be found in [6]. The case of complex roots of equation (2.6) is considered in [2, 3].

Example 4 (continued). In the problem about flags, we obtained the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$, where $a_1 = 0, a_2 = 6$.

The corresponding characteristic equation has the form:

$$x^2 - x - 2 = 0.$$

This equation has two different roots $\alpha = -1, \beta = 2$. Therefore, we write the general solution of the recurrence relation as follows:

$$a_n = K_1(-1)^n + K_2 2^n,$$

where K_1 i K_2 can be determined from the initial conditions:

$$\begin{cases} 0 = -K_1 + 2K_2 \\ 6 = K_1 + 4K_2 \end{cases},$$

so that we obtain $K_1 = 2, K_2 = 1$.

Therefore,

$$a_n = 2(-1)^n + 2^n.$$

Example 5. The Fibonacci sequence is given as follows:

$$F_1 = 1, \qquad F_2 = 2, \qquad F_n = F_{n-1} + F_{n-2}, \ (n \geq 3).$$

The corresponding characteristic equation

$$x^2 - x - 1 = 0$$

has the roots $\frac{1}{2}(1 \pm \sqrt{5})$, so $F_n = K_1\alpha^n + K_2\beta^n$, where $\alpha = \frac{1}{2}(1 + \sqrt{5})$, $\beta = \frac{1}{2}(1 - \sqrt{5})$.

Substituting these into the initial conditions, we get

$$\begin{cases} 1 = K_1\left(\frac{1+\sqrt{5}}{2}\right) + K_2\left(\frac{1-\sqrt{5}}{2}\right), \\ 2 = K_1\left(\frac{1+\sqrt{5}}{2}\right)^2 + K_2\left(\frac{1-\sqrt{5}}{2}\right)^2. \end{cases}$$

Therefore, $K_1 = \frac{\alpha}{\sqrt{5}}$ and $K_2 = -\frac{\beta}{\sqrt{5}}$, so that

$$F_n = \frac{1}{\sqrt{5}}\alpha^{n+1} - \frac{1}{\sqrt{5}}\beta^{n+1} = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right). \quad (2.7)$$

This is the so-called *Binet formula*.

It may seem strange to have irrationalities in the formula for the n-th term of the sequence, since the Fibonacci numbers are integers, however, when the expression is transformed, irrationalities disappear.

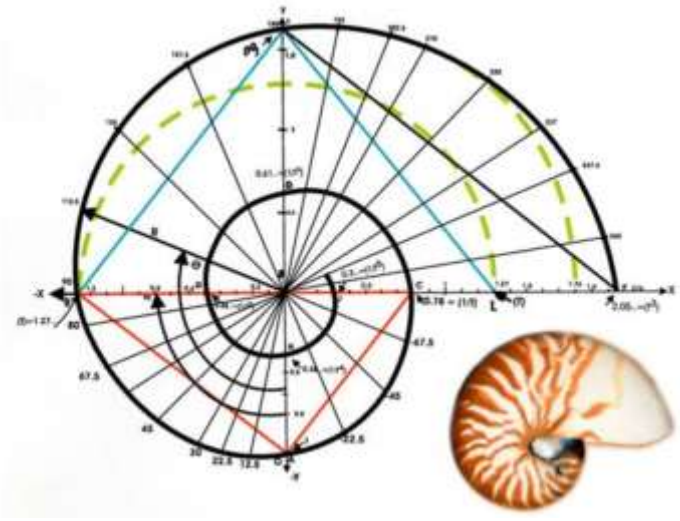
It is known that $\frac{F_{n+1}}{F_n} \to \frac{1+\sqrt{5}}{2}$. Number $\frac{1+\sqrt{5}}{2}$ called the *golden ratio*.

The golden ratio is a proportional division of a segment into unequal parts in which the length of the entire segment is related to the length of the larger part as the length of the larger part is related to the length of the smaller part.

$$\frac{a}{b} = \frac{a+b}{a} = \frac{1+\sqrt{5}}{2} = 1{,}618\ldots$$

The golden ratio can often be found in nature: in the structure of a sunflower, a mollusk shell, a spider web, a DNA molecule, etc.

It is generally accepted that the concept of the golden ratio was introduced into mathematics by Pythagoras (c. 570-490 BC), an ancient Greek philosopher and mathematician. It is believed that he borrowed the concept of the golden ratio from the Egyptians, who used the golden ratio in the construction of pyramids, temples, bas-reliefs, household items and jewelry.



Golden ratio in a mollusk shell

The ancient Greek philosopher Plato (427-347 BC) also knew about the division of a segment in the golden ratio.

The facade of the ancient Greek temple of the Parthenon contains the golden ratio. During its excavations, compasses used by architects and sculptors of the ancient world were discovered.

In ancient literature, the golden ratio is mathematically described in Euclid's "Principles" (about 325 - about 270 BC). The second book of "Principles" gives a geometric construction of the golden ratio. This number is found in a regular pentagon; in a "golden" equilateral triangle with an angle at the vertex of 36^0; in a regular decagon, etc.

After Euclid, the golden ratio was studied by Hypsicles (2nd century BC), Pappus (3rd century AD) and others. In medieval Europe, the first knowledge about the golden ratio was obtained from Arabic translations of Euclid's Elements.

Example 6. Solve the recurrence relation

$$a_n = 4a_{n-1} - 4a_{n-2}, \ \ (n \geq 3), \text{ where}$$

$$a_1 = 1, a_2 = 3.$$

The corresponding characteristic equation has the form

$$x^2 - 4x + 4 = 0,$$

that is, $(x-2)^2=0,\ x=2$ – a multiple root of the equation. Therefore, the general solution of the relation has the form: $a_n=(K_1+nK_2)2^n$.

We use the initial conditions and get:

$$\begin{cases}1=2(K_1+K_2)\\3=4(K_1+2K_2)\end{cases}\quad\text{or}\quad\begin{cases}2K_1+2K_2=1\\4K_1+8K_2=3.\end{cases}$$

Where $K_1=K_2=\frac{1}{4}$. |Hence, $a_n=(n+1)2^{n-2}$.

Example 7. Let $a_1=3, a_2=6, a_3=14$ and for any $n\geq 4$, $a_n=6a_{n-1}-11a_{n-2}+6a_{n-3}$. Find a_n.

The characteristic equation has the form

$$x^3-6x^2+11x-6=0, \text{ or } (x-1)(x-2)(x-3)=0.$$

The roots of the equation are the numbers 1, 2, and 3. Therefore, the general solution of the relation is written in the form $a_n=K_1+K_2 2^n+K_3 3^n$.

Using initial conditions and solving the corresponding system of linear equations

$$\begin{cases}K_1+2K_2+3K_3=3\\K_1+4K_2+9K_3=6\\K_1+8K_2+27K_3=14\end{cases},$$

we get that $K_1=1, K_2=\frac{1}{2}, K_3=\frac{1}{3}$, so then $a_n=1+2^{n-1}+3^{n-1}$.

Example 8. Find a_n if $a_0=1, a_1=2, a_2=12$, and for any $n\geq 3$, $a_n=6a_{n-1}-12a_{n-2}+8a_{n-3}$.

The characteristic equation has the form:

$$x^3-6x^2+12x-8=0, \text{ or } (x-2)^3=0.$$

This is the case of a multiple root of the characteristic equation. Therefore, the general solution has the form $a_n=(\mathrm{C}_1+nC_2+n^2C_3)2^n$.

Using the initial conditions, solving the corresponding system of linear equations

$$\begin{cases}\mathrm{C}_1=1\\2\mathrm{C}_1+2\mathrm{C}_2+2\mathrm{C}_3=2\\4\mathrm{C}_1+8\mathrm{C}_2+16\mathrm{C}_3=12\end{cases},$$

we get that $\mathrm{C}_1=1,\ \mathrm{C}_2=-1,\ \mathrm{C}_3=1$, and $a_n=(1-n+n^2)2^n$.

The case when the characteristic equation has roots of different multiplicities is considered in [2, 3].

§ 2.3. Inhomogeneous linear recurrence relations with constant coefficients

In this section, we briefly consider linear inhomogeneous recurrence relations with constant coefficients and one of the methods for their solving.

Definition. A recurrence relation defined by an equation of the form

$$a_n = \mathrm{B}_1 a_{n-1} + \mathrm{B}_2 a_{n-2} + \cdots + \mathrm{B}_k a_{n-k} + f_n, \ \ n \geq k,$$

23.8)

is called an inhomogeneous linear recurrence relation of order *k* with constant coefficients.

Here, the constants $\mathrm{B}_1, \mathrm{B}_2, \ldots, \mathrm{B}_k$ are given, f_n is some given function of n that is not identically zero.

Let us write equation (2.8) in a more familiar form, renaming the coefficients:

$$a_n + A_1 a_{n-1} + A_2 a_{n-2} + \cdots + A_k a_{n-k} = f_n, \ n \geq k. \qquad (2.9)$$

In §3.2, we considered the method of solving homogeneous linear recurrence relations with constant coefficients using the characteristic equation. It turns out that there is a connection between a linear inhomogeneous recurrence relation with constant coefficients and its corresponding homogeneous recurrence relation.

Theorem 2.4. The general solution of a linear inhomogeneous recurrence relation with constant coefficients can be written as the sum of the general solution of the corresponding homogeneous recurrence relation and any partial solution of this linear inhomogeneous recurrence relation.

How to find a partial solution of a linear inhomogeneous recurrence relation with constant coefficients? In the general case, it is not possible to propose a universal method. But if the right-hand side of the inhomogeneous recurrent relation has a special form (for example, a polynomial), then there are universal algorithms for finding a partial solution.

Example 9. Solve a recurrence relation

$$a_n = -a_{n-1} + 3 \cdot 2^{n-1}, \quad a_1 = 0.$$

It is necessary to first solve the corresponding homogeneous relation $a_n = -a_{n-1}$. Since its characteristic equation is $x + 1 = 0$ has a single root $x = -1$, there exists a general solution of the homogeneous recurrence relation of the form

$$a_n = C \cdot (-1)^n, \;\; C - const.$$

We will seek a partial solution of an inhomogeneous relation $a_n = -a_{n-1} + 3 \cdot 2^{n-1}$ in the form $a_n = A \cdot 2^n$, the constant *A* needs to be determined. Substituting into the original equation gives:

$$A \cdot 2^n = -A \cdot 2^{n-1} + 3 \cdot 2^{n-1},$$

where $2A = -A + 3$, that is, *A=1*.

According to Theorem 2.4, the general solution of the inhomogeneous recurrence relation has the form $a_n = C \cdot (-1)^n + 2^n, \; also \;\; C = const.$

Because $a_1 = 0$, we get from $0 = C \cdot (-1)^0 + 2^0,$ that C=2 and, finally, $a_n = 2 \cdot (-1)^n + 2^n$.

Note that we apply the initial condition at the very end of the procedure.

Example 10. Find the general solution of the relation

$$a_{n+2} - 6a_{n+1} + 9a_n = 2n.$$

Let us first find the solutions of the corresponding homogeneous relation

$$a_{n+2} - 6a_{n+1} + 9a_n = 0.$$

Its characteristic equation $x^2 - 6x + 9 = 0$ has a root x=3 of multiplicity 2, i.e., the general solution of the homogeneous recurrence relation has the form

$$a_n = C_1 \cdot 3^n + C_2 \cdot 3^n \cdot n = (C_1 + C_2 n) \cdot 3^n, \qquad C_1, C_2 - const.$$

We will seek a partial solution to the inhomogeneous relation in the form of the right-hand side, i.e. a polynomial of the 1st degree $a_n = Ax + B$, the constants *A* and *B* need to be determined. Substituting this into the original equation, we obtain:

$$A(n+2) + B - 6(A(n+1) + B) + 9(An + B) = 2n.$$

Using the method of undetermined coefficients, we obtain

$$(\mathrm{A} - 6\mathrm{A} + 9\mathrm{A})n + 2A + B - 6A - 6B + 9B = 2n,$$

or

$$4\mathrm{A}n = 2n$$

and

$$-4A + 4B = 0,$$

and therefore

$$A = B = 0{,}5.$$

According to Theorem 2.4, the general solution of the inhomogeneous recurrence relation has the form

$$a_n = (C_1 + C_2 n) \cdot 3^n + 0{,}5n + 0{,}5, \qquad C_1, C_2 \ - const.$$

More details on the theory of inhomogeneous linear recurrent relations with constant coefficients can be found in [8].

To solve many computational problems, *iterative methods* are used, which, in essence, are procedures for solving recurrence relations.

Thus, the simple iteration method for solving a nonlinear algebraic equation $f(x) = 0$ is described by a recurrent equation of the form

$$x(n+1) = x(n) + f\big(x(n)\big).$$

Newton's method, or the tangent method, is another classical algorithm for solving an algebraic equation $f(x) = 0$. This is an algorithm for numerically solving a recurrence equation $x(n+1) = x(n) - \frac{f(x(n))}{f'(x(n))}$.

Recurrent equations are used for numerical solving and analysis of solutions of ordinary differential equations, partial differential equations, integral equations, functional equations. You may read more about this in [8].

§ 2.4. Generating function of a recurrent sequence

Consider a homogeneous recurrent sequence $\{a_n\}$ of order k $\{a_n\}$which satisfies the relation

$$a_{n+k} + A_1 a_{n+k-1} + A_2 a_{n+k-2} + \cdots + A_k a_n = 0, \ \ n = 0, 1, 2 \ldots \qquad (2.10)$$

The first k terms of the sequence $(a_0, a_1, \ldots, a_{k-1})$ are given. The numbers included in the sequence can have different natures.

Definition. Formal power series

$$f(x) = \ a_0 + a_1 x + \cdots + a_n x^n + \cdots = \ \sum_{i=0}^{\infty} a_i x^i \qquad (2.11)$$

is called *the generating function* or *generatrix* of a recurrent sequence (2.10).

If all the terms of the sequence (2.10), starting from some, are zero, then the generating function is a *generating polynomial*.

An example of a generating polynomial is Newton's binomial:

$$(x+1)^n = C_n^0 x^n + C_n^1 x^{n-1} + C_n^2 x^{n-2} + \cdots + C_n^n,\ n \in \mathrm{N}.$$

It determines the sequence $C_n^0, C_n^1, C_n^2, \ldots, C_n^n, 0, 0, \ldots$

In fact, in the context of this section, we are only interested in the coefficients of the formal power series. For example, for the Fibonacci sequence $\{F_n\} = \{1, 1, 2, 3, 5, 8, \ldots\}$, the generating function has the form:

$$f(x) = 1 + x + 2x^2 + 3x^3 + \cdots + F_n x^n + \cdots$$

Generating function for a sequence $\{2^n\}$ looks like:

$$f(x) = 1 + 2x + 2^2x^2 + 2^3x^3 + \cdots + 2^n x^n + \cdots$$

The explicit form of the generating function (via a formula) is called the closed form in the theory of *recurrent sequences*.

For the given example, the closed form of the generating function is as follows:

$$f(x) = \frac{1}{1-2x}, \quad |x| < \frac{1}{2}.$$

Limitation $|x| < \frac{1}{2}$ is due to the fact that, in the interval $\left(-\frac{1}{2}; \frac{1}{2}\right)$, the series $1 + 2x + 2^2x^2 + 2^3x^3 + \cdots + 2^n x^n + \cdots$ converges, and outside it diverges.

From the course of mathematical analysis, it is known that the power series on the right-hand side of (3.11) has a convergence region, which is an interval (open, closed or semi-closed, it can even be a point). The question of finding the convergence interval can be found, for example, in [4]. We use these series as an auxiliary tool, and the exact description of the convergence interval is not important to us. The method of the generating function is applied for those values of x at which the series converges. Note that the derivative and integral of the generating function are also defined formally.

The following functions and series are most often considered:

Power series	Function	Sequence
$1 + x + \cdots + x^n + \cdots$	$\frac{1}{1-x}$	1, 1, 1, 1, …
$1 - x + \cdots + (-1)^n x^n + \cdots$	$\frac{1}{1+x}$	1, -1, 1, -1,...

$1 + 2x + 3x^2 \ldots + (n+1)x^n + \cdots$	$\frac{1}{(1-x)^2}$	1, 2, 3, 4, …
$1 + 2x + 4x^2 \ldots + 2^n x^n + \cdots$	$\frac{1}{1-2x}$	1, 2, 4, 8, 16, …
$1 + \frac{m}{1!}x + \frac{m(m-1)}{2!}x^2 + \frac{m(m-1)(m-2)}{3!}x^3 + \cdots$	$(1+x)^m$	$1, m, \frac{m(m-1)}{2!}, \ldots$

Let us consider the method for finding the generating function in closed form and the formula for the *n*-th term of a sequence using examples, if the sequence is given by a recurrence relation. You can read more about the theoretical foundations of the method in [1].

Example 11. Consider the stationary sequence 1, 1, 1, …, which can be given by the recurrence relation $a_{n+1} = a_n$. It is defined by the function

$$f(x) = 1 + x + \cdots + x^n + \cdots.$$

Let's find the expression for this function in closed form. To do this, multiply both sides of the equality by x. Then we get

$$xf(x) = x + x^2 + \cdots$$

or $xf(x) = f(x) - 1,$ whence $f(x) = \frac{1}{1-x}$.

Example 12. Consider the recurrence relation of Example 9,

$$a_n = 3 \cdot 2^{n-1} - a_{n-1},\ a_1 = 0,\ n \geq 2.$$

Let $f(x) = a_1 x + a_2 x^2 \ldots + a_n x^n + \cdots$, then

$$f(x) = a_1 x + (3 \cdot 2^1 - a_1)x^2 + (3 \cdot 2^2 - a_2)x^3 + \cdots =$$

$$= a_1 x + 3(2x^2 + 2^2 x^3 + \cdots) - (a_1 x^2 + a_2 x^3 + \cdots).$$

This equality can be rewritten as follows:

$$f(x) = 6x^2(1 + 2x + 2^2 x^2 + \cdots) - xf(x),$$

where

$$f(x) = \frac{6x^2}{(1-2x)(1+x)}, \qquad |x| < \frac{1}{2}.$$

We have obtained the form of the generating function in closed form.

Now we will find the formula for the *n*-th term of the sequence. To do this, we will decompose the fraction into elementary fractions and apply the method of undetermined coefficients:

$$f(x) = \frac{6x^2}{(1-2x)(1+x)} = 6x^2\left(\frac{\mathrm{A}}{1-2x} + \frac{B}{1+x}\right),$$

where $A = \frac{2}{3},\ B = \frac{1}{3}$.

Then $f(x) = 2x^2\left(\frac{2}{1-2x} + \frac{1}{1+x}\right)$.

Expanding each term in parentheses into a Maclaurin series, we obtain

$$f(x) = 4x^2(1 + 2x + 2^2x^2 + \cdots) + 2x^2(1 - x + x^2 - \cdots).$$

Let's find the coefficient at x^n:

$$a_n = 4 \cdot 2^{n-2} + 2 \cdot (-1)^{n-2} = 2^n + 2 \cdot (-1)^n.$$

Example 13. $a_n = 4a_{n-1} - 4a_{n-2}, a_1 = 1, a_2 = 3, n \geq 3$.

To find the generating function, we perform the steps similar to those in Example 12.

$$\begin{aligned} f(x) = {} & a_1x + 3x^2 + (4a_2 - 4a_1)x^3 + (4a_3 - 4a_2)x^4 + \cdots \\ & = x + 3x^2 + 4(a_2x^3 + a_3x^4 + \cdots) - 4(a_1x^3 + a_2x^4 + \cdots). \end{aligned}$$

This equality can be rewritten in the following form:

$$f(x) = x + 3x^2 + 4x(f(x) - a_1x) - 4x^2f(x),$$

where

$$f(x) = \frac{x - x^2}{(1-2x)^2}.$$

Using Maclaurin's series again, we have

$$\frac{1}{(1-2x)^2} = 1 + 2 \cdot 2x + 3 \cdot 2^2x^2 + \cdots f(x)$$

$$= (x - x^2)(1 + 2 \cdot 2x + 3 \cdot 2^2x^2 + \cdots).$$

Now let's find the formula for the *n*-th term of the sequence.

$$a_n = 4a_{n-1} - 4a_{n-2} = n2^{n-1} - (n-1)2^{n-2} = (n+1)2^{n-2}.$$

Example 14. Find the formula for the *n*-th term of the Fibonacci sequence.

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, \qquad n \geq 2.$$

In this example, we will do it differently than in the previous two. We will multiply all equalities by $x^0, x^1, \ldots, x^n$ respectively and sum the obtained equalities. We get:

$$x^1F_1 + \sum_{n=2}^{\infty} x^n F_n = x + \sum_{n=2}^{\infty} x^n F_{n-1} + \sum_{n=2}^{\infty} x^n F_{n-2}.$$

Let us rewrite this equation in the form

$$x + \sum_{n=2}^{\infty} x^n F_n = x + x\sum_{n=2}^{\infty} x^{n-1} F_{n-1} + x^2\sum_{n=2}^{\infty} x^{n-2} F_{n-2.}$$

Denote $f(x) = x + \sum_{n=2}^{\infty} x^n F_n$, then $f(x) = x + xf(x) + x^2f(x)$, where $f(x) = \frac{x}{1-x-x^2}$ is the generating function of the Fibonacci sequence.

Let's represent the generating function as a power series. To do this, we decompose the fraction into elementary fractions and apply the method of undetermined coefficients. The roots of the equation $1 - x - x^2 = 0$:

$$x_1 = \frac{-1+\sqrt{5}}{2},\ x_2 = \frac{-1-\sqrt{5}}{2}.$$

Let's use the method of undetermined coefficients. We decompose the fraction into elementary fractions:

$$\frac{x}{1-x-x^2} = \frac{A}{x-x_1} + \frac{B}{x-x_2}.$$

Reducing the right-hand side to a common denominator and solving the corresponding linear system, we obtain $A = \frac{-1+\sqrt{5}}{2\sqrt{5}}, B = \frac{1+\sqrt{5}}{2\sqrt{5}}$. Then

$$\frac{x}{1-x-x^2} = \frac{-1+\sqrt{5}}{2\sqrt{5}(x-x_1)} + \frac{1+\sqrt{5}}{2\sqrt{5}(x-x_2)} = \frac{-1+\sqrt{5}}{2\sqrt{5}x_1} \cdot \frac{1}{\left(1-\frac{x}{x_1}\right)} + \frac{-1+\sqrt{5}}{2\sqrt{5}x_2} \cdot \frac{1}{\left(1-\frac{x}{x_2}\right)}.$$

Using the formula $\frac{1}{1-\alpha x} = 1 + \alpha x + (\alpha x)^2 + \cdots + (\alpha x)^n + \cdots$ and applying some algebraic transformations of expressions under the summation sign, we obtain

$$f(x) = \sum_{n=0}^{\infty} x^n F_n = \frac{1}{\sqrt{5}} \sum_{n=1}^{\infty} x^n \left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \right),$$

hence $F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \right)$ (Binet's formula again).

The foundations of the theory of linear recurrent sequences were laid in the first third of the 18th century by the English mathematician Abraham de Moivre (1667-1754) and the Dutch mathematician Daniel Bernoulli (1700-1782). Leonard Euler set out this theory in the thirteenth chapter of his "Introduction to the Analysis of Infinitesimals" (1748).

The beginning of the generating function method was laid by Abraham de Moivre, and the further development and continuation of this method was facilitated by the work of Leonard Euler. A. Moivre was engaged in obtaining a formula for the general term of the Fibonacci sequence. To do this, he developed the generating function method, which is also used in modern mathematics in problems of combinatorics, probability theory and number theory.

The algorithm for forming the Fibonacci sequence is very simple, but mathematicians could not obtain a general formula for the *n*-th term of the sequence for several centuries. Abraham Moivre in 1730, five hundred years after the description of the sequence in Leonardo Fibonacci's work (1202), derived the formula for the *n*-th term of the sequence using the generating function method,

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \right).$$

Now the formula is called Binet's formula (Jacques Philippe Marie Binet, 1786-1856, a French mathematician, mechanic and astronomer, rediscovered this formula a hundred years after Moivre).

The summation of recurrent sequences is a problem of finite difference calculus. Such problems were addressed by A. Moivre, D. Bernoulli, L. Euler, I. Newton, B. Taylor, J. Stirling, etc.

The calculus of finite differences acquired the status of an independent mathematical discipline only at the beginning of the 18th century in the works of Isaac Newton ("Method of Differences", 1811). The theory of recurrent sequences is a part of the calculus of finite differences, which, in turn, is a section of the mathematical analysis of functions of integer arguments (functions with a discretely variable argument).

§ 2.5. Derangements

Suppose n people leave their coats in the cloakroom of a theatre. After the performance they pick their coats up at random. What is the probability that none of them gets their coat?

Definition. The derangement of the order of the numbers $1,2,\dots,n$ is a renumbering $\pi(1),\pi(2),\dots,\pi(n)$ in such a way that $\pi(i) \neq i$, for all $i = 1,2,\dots,n$.

For example, there are nine permutations of numbers 1 2 3 4:

2 4 1 3, 2 1 4 3, 2 3 4 1, 3 1 4 2, 3 4 2 1, 3 4 1 2, 4 1 2 3, 4 3 1 2, 4 3 2 1.

In all these combinations, 1 is not in the first place, 2 is not in the second place, and so on.

Denote by d_n the number of derangements of the numbers $1,2,\dots,n$. It is easy enough to calculate that $d_1 = 0$, $d_2 = 1$, d_3=2, $d_4 = 9$. Our goal is to find recurrence relations for d_i, and then the formula for calculating d_n.

Let us first note that d_n is the number of ways to place n objects in n boxes, when for each object one box is forbidden and when each box is forbidden for one

of the objects. Above, the boxes and objects were numbered $1,2,\dots,n$, where the i-th box is forbidden for the i-th object. In general, the numbering can be arbitrary.

We further note that among the nine derangements of the numbers 1, 2, 3, 4, there are three where the number 4 is interchanged with another number, these permutations: 2 1 4 3, 3 4 1 2 and 4 3 2 1.

In the other six permutations, the number 4 is not interchanged with another number.

Given this, we have

$$d_n = e_n + f_n,$$

where $e_n$is the number of disorderings in which the number n is exchanged for some other number, and f_n – the number of rearrangements in which the number n does not swap places with another number. If the number n swaps places with the number i (there are $(n-1)$ ways to choose i in total), then the remaining $(n-2)$ numbers can be reordered in d_{n-2} in ways.

So, $e_n = (n-1)d_{n-2}$.

If n leaves its place and a number r comes to this place (the number of ways to choose this r is $(n-1)$), then there are $n-1$ places and $n-1$ numbers left, and again one place is forbidden for each number, and one number is forbidden for each place (all numbers have their places forbidden, and for n the r-th place is forbidden). Therefore, $f_n = (n-1)d_{n-1}$. So,

$$d_n = (n-1)(d_{n-1} + d_{n-2}) \qquad (2.12)$$

Using this recurrence relation, we obtain that

$$d_5 = 4(9+2) == 44,$$
$$d_6 = 5(44+9) = 265$$

and so on.

The recurrence relation (2.12) cannot be solved by the auxiliary equation method, since the coefficients at d_{n-1} and d_{n-2} (both levels $(n-1)$) are not constant. However, we can reduce (2.6) to a more convenient form.

We rewrite equality (2.12) in the form

$$d_n - nd_{n-1} = -(d_{n-1} - (n-1)d_{n-2}).$$

The expression on the right-hand side is obtained by changing the sign on the left-hand side and replacing n with $n-1$. Therefore, iterating, we obtain

$$d_n - nd_{n-1} = -(d_{n-1} - (n-1)d_{n-2}) = (-1)^2(d_{n-2} - (n-2)d_{n-3}) =$$
$$\cdots = = (-1)^{n-2}(d_2 - 2d_1) = (-1)^n(1-0) = (-1)^n,$$

that is,

$$d_n - nd_{n-1} = (-1)^n \,. \tag{2.13}$$

So,

$$\frac{d_n}{n!} - \frac{d_{n-1}}{(n-1)!} = \frac{(-1)^n}{n!}.$$

Adding equalities $\frac{d_m}{m!} - \frac{d_{m-1}}{(m-1)!} = \frac{(-1)^m}{m!}$, where $m = 2,3,\dots,n$, we get

$$\frac{d_n}{n!} - \frac{d_1}{1!} = \frac{(-1)^2}{2!} + \frac{(-1)^3}{3!} + \cdots + \frac{(-1)^n}{n!} = \sum_{m=2}^{n} \frac{(-1)^m}{m!} = \sum_{m=0}^{n} \frac{(-1)^m}{m!}.$$

or $d_1 = 0$, therefore we have

$$d_n = n!\sum_{m=0}^{n} \frac{(-1)^m}{m!} = n!\left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + \frac{1}{n!}\right). \tag{2.14}$$

One interesting consequence of this result is that

$$\frac{d_n}{n!} \xrightarrow[n\to\infty]{} \frac{1}{e}.$$

Thus, the probability that no one gets their coat is tending as $n \to \infty$ to $\frac{1}{e} =$ 0,36788 ....

For example, for $n = 6$ we have $\frac{d_6}{6!} = \frac{265}{720} \approx 0{,}36806$, which coincides to the third digit with $\frac{1}{e}$.

Example 15. Find the number of permutations of the numbers $1,2,\dots,n$, in which k numbers are at their places.

There are C_n^k ways to choose k numbers which we fix. The remaining $n-k$ must be located in different places. The number of ways of such locations is equal to d_{n-k}. Therefore, by the multiplication principle, we get the answer $C_n^k d_{n-k}$.

§ 2.6. Sorting algorithms

Often in problems it is necessary to be able to estimate the number of elementary operations or the time required by the computer to execute the entire algorithm. This is important because some operations take longer to execute than others.

Sometimes it is also necessary to take into account the amount of computer memory, the accuracy of calculations, etc. for the execution of the algorithm. Obviously, this is important only for programs that require significant time to execute, which may depend on such a factor as the amount of data. That is why the concept of time complexity of an algorithm is often considered. Despite the fact that the time complexity function of an algorithm can be defined exactly in some cases, in most cases it is pointless to look for its exact value.

Consideration of the input data size and estimating the order of increasing the algorithm's running time lead to the concept of the asymptotic complexity of the algorithm. In this case, an algorithm with a lower asymptotic complexity is more efficient for all input data, with the possible exception of small data. Asymptotic notation is used to record the asymptotic complexity of algorithms. In particular, the phrase "the complexity of an algorithm is «O big of $f(n)$» means that as the amount of input information to the algorithm increases, the running time of the algorithm will grow no faster than some constant multiplied by $f(n)$ and is denoted as $O(f(n))$. The letter " n " here is the size of the input data, and the function, for example, «$f\ (n) = n^2$» inside "O ( )" gives us an idea of how complex the algorithm is in relation to the amount of input data.

If some algorithm needs to execute $17n^3 + 3n$ conditional operations to process n elements of input data, then as n increases, the final running time will be significantly more affected by raising n to the cube than by multiplying n by 17 or adding 3n. The time complexity of the algorithm is $O(n^3)$, i.e. depends on the size of the input data cubically.

The use of the capital letter O (or the so-called O-notation) came from mathematical analysis, where it is used to compare the asymptotic behavior of

functions. The running time of an algorithm can be classified depending on which function is under the O-notation. For example, an algorithm with complexity $O\ (n)$ is called an algorithm with linear running time, an algorithm with $O(\ n^{\alpha}\ )$ for some $\alpha > 1$ is called polynomial.

For example, if we consider the algorithm for adding two matrices of size $m \times n$, then obviously mn addition operations are performed. Then the number of all arithmetic operations performed has a complexity index $O(N^2)$, were $N = \max\{m, n\}$.

Similarly, the number of arithmetic operations performed when multiplying a matrix of size $m \times p$ by a matrix of size $p \times n$ is of the order $O(N^3)$, where $N = \max\{m, n, p\}$ (mnp operations of addition and mnp multiplication operations).

We can also consider the *constant complexity of the algorithm*, O(1). This means that the selected operation requires constant time. Such operations do not depend on the amount of data. For example, to determine the value of the fifth element of an array, you do not need to memorize the elements or go through the elements several times. You always just need to wait for the fifth element in the input stream and this will be the result, the calculation of which takes approximately the same time for any amount of data.

There are also algorithms with *logarithmic complexity*, denoted as $O(log\ n)$. The last notation is the standard notation for logarithmic complexity algorithms regardless of the base of the logarithm. This notation is used because computers use the binary number system, so we need to use 2 as the base of the logarithm (i.e. we get the notation $log_2 n$). However, when changing the base of the logarithm $log_2 n$ and $\log_b n$ differ only by a constant factor $log_b\ 2$, which is discarded in O-notation. The simplest algorithm with logarithmic complexity is binary search. Indeed, if the data array is sorted, then we can check whether there is any specific value in it by dividing it in half. Next, we check the middle element, if it is greater than the one we are looking for, then we will discard the second half of the array, since it definitely does not have the desired element. If it is less, then on the contrary - we will discard the first half of the array. Similarly, we will further divide the resulting

"half-array" in half, each such operation reduces the amount of input data by half, as a result we will check $log\ n$ array elements.

For example, the algorithm for finding the largest element in an unsorted array has *linear complexity* $O(n)$. We need to go through all n elements of the array to figure out which one is the largest.

Quadratic complexity $O(n^2)$ has, for example, an insertion sort algorithm. In its canonical implementation, it is two nested loops: one to go through all the elements of the array, and the second loop to find a place for the next element in the already sorted part. Thus, the number of operations will depend on the size of the array, $n * n$, i.e. n^2. Sometimes algorithms with such complexity are unavoidable, but quadratic complexity is rather a reason to reconsider the algorithms or data structures used. An example of an algorithm with quadratic complexity is the bubble sort algorithm for an array.

There are also algorithms with linear-logarithmic $O(n \cdot logn)$, *polylogarithmic* $O(log^k n)$, *quasilinear* $O(n \cdot log^k n)$ complexities for some k, etc. An example of a linear-logarithmic algorithm is a binary tree sorting algorithm, an example of a quasi-linear algorithm is quick sort or heap sort algorithms, etc.

Algorithms with *polynomial complexity* $O(n^k)$ for some k form a class of algorithms P, which is central to the theory of computational complexity. All basic arithmetic operations are implemented by the algorithm in polynomial time.

Problems which require exponential execution time are called *exponential*. Their complexity is bounded by the function $e^{P(n)}$, were $P(n)$ – some polynomial that depends on the size of the input data n.

Note that exponents have a higher complexity than polynomials. So the algorithm complexity $O(2^n)$ is more complicated than $O(n^{99})$.

Note that factorials are more complex than powers. A brief proof of this fact is to understand that factorials and powers contain the same number of factors, but the numbers we multiply increase for factorials and remain the same for powers.

The study of algorithms of varying complexity is the subject of algorithm theory.

Let us consider some of the most common sorting algorithms: the bubble sort algorithm and the merge sort algorithm.

Let us consider a pile of papers with written answers from students. We want to sort them, that is, arrange them in ascending or descending order of grades. Is there an efficient way to do this? Let us start with a simple but not very efficient procedure.

Bubble Sort

First, let's note that sorting always requires repeated actions, for which either loops or recursion are used. The efficiency of the algorithm is assessed by the number of necessary actions. Most often, for this, we calculate the number of pairwise comparisons of sorted data elements (arrays) and the number of exchanges of adjacent elements in pairs.

The bubble sort algorithm is a fairly simple to implement algorithm for sorting arrays. In the literature, you can find other names for this algorithm, e.g., sorting by simple exchanges. The algorithm is called bubble because the larger (or smaller) value "pops" (shifts) to the edge of the array after each iteration. The bubble sort algorithm consists of repeated passes through the array being sorted. At each iteration, adjacent elements are compared sequentially, and if the order in the pair is incorrect, the elements are swapped. For each pass through the array, at least one element is replaced, so it is necessary to make no more than $n - 1$ passes to sort the array, where n is the size of the array.

Let's illustrate the algorithm with an example.

Let's take n sheets of students' work in random order, they form an array. We will sort the sheets in ascending order of students' grades.

Compare the first two and swap them if they are not in ascending order of grades. Then compare the second with the third and swap them again if necessary. Doing this until the end of the sequence, we will get the highest grade in the last

place. Then we repeat the procedure for the first $n-1$ numbers. Then the second highest number will be in the penultimate place. Repeat the procedure for the first $n-2$ numbers and so on.

The total number of sheet movements (comparisons) in this procedure is

$$(n-1)+(n-2)+\cdots+2+1=\frac{1}{2}n(n-1)=\frac{1}{2}n^2-\frac{1}{2}n.$$

In this case, we say that the algorithm has a complexity index $O(n^2)$.

Example 16. Let there be sheets with grades 7, 10, 4, 6, 3. Sort them in ascending order of grades using a bubble sort.

After the first 4 moves of the sheets, we have 7, 4, 6, 3, 10. After the next 3 moves: 4, 6, 3, 7, 10. After the next two: 4, 3, 6, 7, 10. After the last comparison of grades: 3, 4, 6, 7, 10.

The total number of moves is 10.

Merge sort

The idea is to divide the given set into two (approximately equal) parts. Then we sort each part, and then merge (combine) them. The process of merging two sorted sets of lengths l and m into one set requires $l+m-1$ comparisons. This is explained as follows. Let there be two sorted sets in ascending order. We compare the first (smallest) numbers of the sets and choose the smaller one as the first number of the new set. In doing so, we cross out this number from the set from which it was taken. We repeat the procedure and find the second number of the new set, and so on. Obviously, the number of comparisons does not exceed $l+m-1$, since no comparison is required to choose the last one.

Before comparing two sorted parts, they need to be ordered. Let t_n – the number of comparisons required to sort n numbers by the chosen sorting method. If we divide the set of n elements into two with l and k elements, we get

$$t_n=t_l+t_k+l+k-1=t_l+t_k+n-1.$$

Thus, if we consider the particular case $n = 2^m$ so that the set can be divided in half at each stage, then we get

$$t_{2^m} = 2\, t_{2^{m-1}} + (2^m - 1)$$

Let's substitute $a_m = t_{2^m}$. Then the recurrence relation has the form

$$a_m = 2\, a_{m-1} + (2^m - 1). \qquad (2.15)$$

Let us first solve the homogeneous recurrence relation

$$a_m = 2\, a_{m-1}.$$

Obviously, the solution has the form $a_n = A2^n$, were A – is an arbitrary constant. Let us now find a partial solution to relation (2.15). Let us use the method of undetermined coefficients.

The method of finding a solution in the form $a_n = B2^n + C$ will not lead to a result, because 2^n is a solution of the homogeneous recurrence relation, so we are looking for a partial solution in the form

$$a_n = Bn2^n + C.$$

After substitution we get

$$Bn2^n + C = 2B(n-1)2^{n-1} + 2C + 2^n - 1,$$

that is,

$$0 = -B2^n + 2^n - 1 + C,$$

from which we get $B = C = 1$ and, accordingly,

$$a_n = A2^n + n2^n + 1$$

We have $a_1 = 1$, and accordingly, $A = -1$. Finally, we have that

$$a_n = 2^n(n-1) + 1.$$

So, $t_{2^m} = 1 + 2^m(m-1)$.

Let's substitute $n = 2^m$, then we get $t_n = 1 + n(log_2 n - 1)$.

Therefore, this sorting method has a complexity index of $O(nlog_2 n)$. This is better than using the previous method, the complexity of which is $O(n^2)$, because n^2 increases with increasing n faster than $nlog_2 n$.

Below, is a table of common array sorting algorithms and their time complexity.

Algorithm	Data structure	Time complexity
Merge sort	Massive	$O(n \log n)$
Pyramid sorting	Massive	$O(n\ log\ n)$
Bubble sort	Massive	$O(n^2)$
Insertion sort	Massive	$O(n^2)$
Sort by selection	Massive	$O(n^2)$

§ 2.7. Catalan numbers

In this section we will introduce a well-known sequence of numbers called Catalan numbers, which occurs in many problems of various types. Catalan numbers are named after the Belgian mathematician Catalan (E. S. Catalan 1814-1894), who considered them in his publications. But they were considered even earlier by other mathematicians, including L. Euler in his work on the division of a polygon into triangles (we will discuss this briefly later).

We will describe one case in full where Catalan numbers occur and start with the following simple problem.

Example 17. How many “up-to-right” paths are there from point A to point B (Fig. 1)?

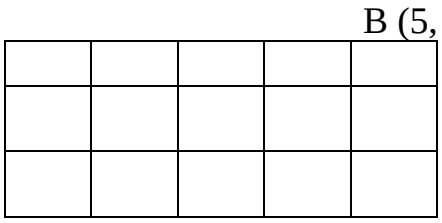


Fig. 1

By the "up-right" path we mean moving along the sides of the small squares always up or to the right. Each path consists of 5 moves to the right and 3 moves up. Thus, the total number of possible paths is C_8^3. In the general case, if a rectangle has length m and height n, then the number of such paths is C_{m+n}^n.

Let's say we have now an $n \times n$ square and we need to find the number p_n of paths "right-up" from the lower left corner to the upper right corner, which pass under the diagonal AB (possibly touching it). In the case of n = 3, shown in Fig. 2, there are 5 such paths, which can be described as: RURURU, RURRUU, RRUURU, PPVPVV, RRRUUU, where R means right, U means up. So $p_3 = 5$. And what is p_n equal to?

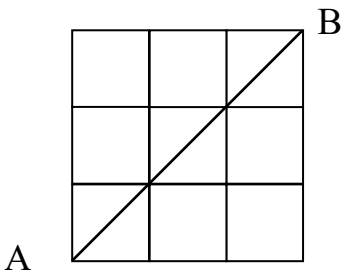


Fig. 2

We will call any path that satisfies our conditions “correct”. Such a path touches the diagonal at least once before reaching point B (at least at point A). Consider any correct path. Let its last touch of the diagonal before B be at point C(*m*,*m*), where $1 \leq m < n$. Then p_m – is the number of correct paths from A to C. The path from point C must continue to point D(*m+1*, *m*), and from there to the point E(*n*, *n-1*) (see. fig. 3), but it should not go over the line DE anywhere, otherwise C would not be the last point of tangency of the diagonal. But the points D and E are opposite vertices of a square with side $n - m - 1$, and the number of correct paths from D to E is p_{n-m-1}.

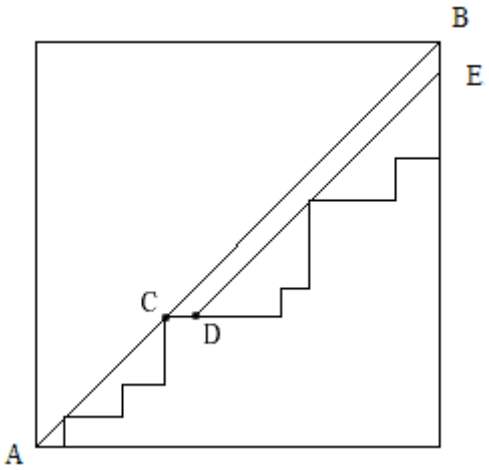


Fig. 3

By the product rule, the number of correct paths from A to B with the last diagonal tangent to point B (*m*,*m*) is equal to $p_m p_{n-m-1}$.

Since *m* can take all values from 0 to *n-1*, by the sum rule we get

$$p_n = \sum_{m=0}^{n-1} p_m p_{n-m-1}. \qquad (2.16)$$

The initial condition $p_1 = 1$ is obvious, then by formula (2.16) we get $p_0^2 = 1$.

So, let's choose $p_0 = 1$.

This recurrence relation differs from the previously considered ones in that it is not linear. To solve the relation, we will apply the method of the generating function.

Let $f(x) = p_0 + p_1 x + p_2 x^2 + \ldots$

Then, using (3.16), we obtain

$$f^2(x) = (p_0 + p_1 x + p_2 x^2 + \cdots)(p_0 + p_1 x + p_2 x^2 + \ldots) =$$

$$= \sum_{n=0}^{\infty} x^n (p_0 p_n + p_1 p_{n-1} + \cdots + p_n p_0) = \sum_{n=0}^{\infty} p_{n+1} x^n$$

So, $xf^2(x) = f(x) - p_0 = f(x) - 1$, from which we get

$$xf^2(x) - f(x) + 1 = 0.$$

Solving this quadratic equation, we get:

$$f(x) = \frac{1 - \sqrt{1-4x}}{2x} = \frac{1}{2x}\left(1 - (1-4x)^{\frac{1}{2}}\right).$$

We choose a minus sign before the root (any sign can be chosen) to avoid having a term of the form $\frac{1}{x}$ in $f(x)$.

Developing $(1-4x)^{\frac{1}{2}}$ into the Maclaurin series, we have:

$$f(x) = \frac{1}{2x}\left\{1 - \left(1 - \frac{1}{2}\cdot 4x - \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4^2x^2}{2!} - \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{3}{2}\cdot\frac{4^3x^3}{3!} - \cdots\right)\right\} =$$

$$= \frac{1}{2x}\left(\frac{1}{2}\cdot 4x + \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{4^2x^2}{2!} + \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{3}{2}\cdot\frac{4^3x^3}{3!} + \cdots\right) =$$

$$= 1 + \frac{1}{2}\cdot\frac{4x}{2!} + \frac{1}{2}\cdot\frac{3}{2}\cdot\frac{4^2x^2}{3!} + \frac{1}{2}\cdot\frac{3}{2}\cdot\frac{5}{2}\cdot\frac{4^3x^3}{4!} + \cdots$$

Thus, for $n \geq 1$,

$$p_n = \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1)}{2^n (n+1)!} \cdot 4^n = \frac{2^n \cdot 1 \cdot 3 \cdot 5 \cdot (2n-1)}{(n+1)!} = \frac{2^n (2n)!}{(n+1)!\, 2^n n!}$$

$$= \frac{1}{n+1} C_{2n}^n$$

For example, $p_3 = \frac{1}{4} C_6^3 = 5$, and $p_4 = \frac{1}{5} C_8^4 = 14$. Note that $p_0 = 1$ satisfies the condition $C_0^0 = 1$.

Definition. Numbers p_n are called *Catalan numbers*, they are usually designated as K_n.

Therefore,

$$K_n = \frac{1}{n+1} C_{2n}^n. \qquad (2.17)$$

The sequence $\{K_n\}_{n \geq 0}$ starts like this: 1, 1, 2, 5, 14, 42, 139, 429, …

From (2.17) we obtain

$$K_m = K_0 K_{m-1} + K_1 K_{m-2} + \cdots + K_{m-1} K_0. \qquad (2.18)$$

As mentioned earlier, Catalan numbers arise in various situations. Thus, replacing R and U with 0 and 1 respectively, we obtain that Catalan numbers K_n are equal to the number of binary sequences of length $2n$ containing n zeros and n ones, provided that in each section starting from the left end, the number of ones does not exceed the number of zeros.

L. Euler first encountered this sequence when solving the following problem: in how many ways can a convex n-gon be cut into triangles by disjoint diagonals? It is obvious that for n = 3 this problem has no solutions, for n = 4 there are two ways, for n = 5 there are 5 ways, for n = 6 there are 14 ways.

In the general case, the Catalan number K_{n-2} is equal to the number of ways to divide a convex n-gon into triangles by $(n-3)$ diagonals. For example, $K_3 = 5$ The method of dividing a pentagon into triangles is shown in Fig. 4.

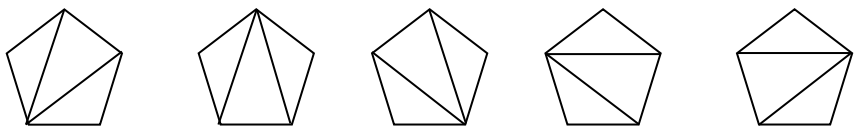


Fig. 4

In 1838, Catalan solved the following problem: "Let there be a chain of n letters arranged in a given order. It is necessary to arrange $(n-1)$ pairs of brackets so that inside each pair there are exactly two "expressions". These paired expressions can be either two adjacent letters or two adjacent expressions. In how many ways can the brackets be arranged?" The solution of the problem led to the mathematics of the numerical sequence, which later began to bear his name.

We would like to note that Catalan numbers often arise when solving combinatorial problems. You can get acquainted with this sequence in more detail, for example, in [5, 6, 7, 8].

List of references

1. Anderson J. A. Discrete mathematics and combinatorics. Translated from English. Moscow: Williams, 2004. 960 p.

2. Bilous N. V., Dudar Z. V., Lisna N. S., Shubin I. Yu. Fundamentals of combinatorial analysis. Kharkiv: XTYPE , 2000. 110 p.

3. Bondarenko M. F., Bilous N. V., Rutkas A. G. Computer discrete mathematics: textbook. Kharkiv: SMIT Company, 2004. 480 p.

4. Dorogovtsev A. Ya. Mathematical analysis: textbook: in two parts. Kyiv: Lybid, 1994.

5. Sloan N. J. A. Directory of numerical sequences: website. URL: https://bit.ly/3jBR5Uw (date of application: 03/27/2022)

6. Anderson I. A first course in discrete mathematics. Berlin: Springer, 2001. 200 p.

7. Dossey J. A., Otto A. D., Spence L. E., Vanden Eynden C. Discrete mathematics. New York: Addison Wesley, 2001. 600 p.

8. Grimaldi R.P. Discrete and combinatorial mathematics. NY: Addison Wesley, 2003. 1005 p.