Introduction: examples of algorithms and basics of complexity analysis

Raimondas Čiegis

Department of Mathematical Modelling, email: rc@vgtu.lt

September 4 d., 2023

1. Textbook (see references)

R. Čiegis. Duomenų struktūros, algoritmai ir jų analizė. Vilnius, "Technika", 2007.

2. Textbook (with pdf, see references)

T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduction to algorithms. The MIT Press, 2009.

3. Textbook (with pdf, see references)

T. Cormen. Algorithms unlocked. The MIT Press, 2013.

3. Video materials:

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/

It is clear that in different courses of computer science, mathematics and other subjects you've already studied many algorithms.

It is clear that in different courses of computer science, mathematics and other subjects you've already studied many algorithms.

We simply say that any algorithm is a set of actions and rules, such that after performing them all we achieve our goal (we solve the given task).

It is clear that in different courses of computer science, mathematics and other subjects you've already studied many algorithms.

We simply say that any algorithm is a set of actions and rules, such that after performing them all we achieve our goal (we solve the given task).

It is easy to find many interesting examples around us.

1. You want to prepare a special pizza for a lunch.

We take a book of culinary recipes, it describes all steps, that should be done and in a specific sequence. So we have an algorithm for cooking a tasty pizza.

2. You need to pass the exam on "Theory of Algorithms". .

The corresponding algorithm can be used:

a) perform all laboratory work tasks,

b) solve the homework tasks,

c) pass the intermediate exam,

d) get positive evaluation of the session exam.

Everything looks simple and clear, just execute this algorithm in given order.

In this course, you will systematize the knowledge you already have on various algorithms and will learn many new useful things:

a) how to create a new algorithm for the selected problem;

b) how to effectively implement algorithms (even a very good algorithm can be underestimated if not implemented properly);

c) you will become familiar with algorithms for solving important applied problems including search of information, data sorting, transport logistics and complicated design tasks.

Where are algorithms used?

Where are algorithms used?

We mention only few domains:

big data and analysis,
virtual reality,
artificial inteligence (AI),
artificial neural networks, machine learning,
internet of things,
digital audio, visual and video information: storage and transmission,
medicine,
robotics,
cryptocurrencies,
computer games ...

Algorithm is a well-defined calculation procedure, it takes the initial data, performes a finite number of operations and gives the result.

We can understand the numerical procedure as a computer program written in one of the programming languages.

Algorithm is a well-defined calculation procedure, it takes the initial data, performes a finite number of operations and gives the result.

We can understand the numerical procedure as a computer program written in one of the programming languages.

A more precise definition is obtained when the computing device is presented as the famous Turing machine. The latter theoretical machine was created, when Turing provided a constructive definition of the algorithm.

Next we define the algorithm complexity.

Next we define the algorithm complexity.

This information is very important when we want to predict how long it will take to solve the problem with the selected algorithm on a given computer (and there are many different types of computers).

Next we define the algorithm complexity.

This information is very important when we want to predict how long it will take to solve the problem with the selected algorithm on a given computer (and there are many different types of computers).

Information about tomorrow's weather will be completely irrelevant if the answer will be computed only after two days.

Next we define the algorithm complexity.

This information is very important when we want to predict how long it will take to solve the problem with the selected algorithm on a given computer (and there are many different types of computers).

Information about tomorrow's weather will be completely irrelevant if the answer will be computed only after two days.

How long will it take for a malicious program to "hack" your bank account password?

Is information about cryptocurrencies securely protected in the blockchain?

The amount of initial data is a very important characteristic of the problem, because the more data is given, the more computer memory is used to store this data, and the computations take longer.

The amount of initial data is a very important characteristic of the problem, because the more data is given, the more computer memory is used to store this data, and the computations take longer.

For example:

- The size of the vector X is equal to the number of its elements n,

The amount of initial data is a very important characteristic of the problem, because the more data is given, the more computer memory is used to store this data, and the computations take longer.

For example:

- The size of the vector X is equal to the number of its elements n,
- The size of the matrix A with m rows and n columns, is equal to mn,

The amount of initial data is a very important characteristic of the problem, because the more data is given, the more computer memory is used to store this data, and the computations take longer.

For example:

- The size of the vector X is equal to the number of its elements n,
- The size of the matrix A with m rows and n columns, is equal to mn,
- Let us take a graph $G = (V, E)$, for which the number of vertices V is n, and the size of the edge set E is m, then a total number of data is equal to $(m + n)$.

Although the number of data characterizes the size of the task, it does not yet describe the complexity of the algorithm completely. Let's examine two important operations: addition of two matrices $A + B$ and multiplication of two matrices AB.

Let us assume that we have square matrices of size $n \times n$.

Although the number of data characterizes the size of the task, it does not yet describe the complexity of the algorithm completely. Let's examine two important operations: addition of two matrices $A + B$ and multiplication of two matrices AB.

Let us assume that we have square matrices of size $n \times n$.

Let us consider the matrix addition $A + B$

$$C = A + B, \quad C = (c_{ij}), \;\; 1 \leqslant i, j \leqslant n\,,$$
$$c_{ij} = a_{ij} + b_{ij}\,.$$

We perform n^2 summation operations.

Although the number of data characterizes the size of the task, it does not yet describe the complexity of the algorithm completely. Let's examine two important operations: addition of two matrices $A + B$ and multiplication of two matrices AB.

Let us assume that we have square matrices of size $n \times n$.

Let us consider the matrix addition $A + B$

$$C = A + B, \quad C = (c_{ij}), \;\; 1 \leqslant i, j \leqslant n\,,$$
$$c_{ij} = a_{ij} + b_{ij}\,.$$

We perform n^2 summation operations.
This number of operations is of the same order as the number of initial data, since the number of coefficients for both matrices is $2n^2$.

Next let us consider the multiplication of two matrices

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad 1 \leqslant i, j \leqslant n \,.$$

In total we compute n^3 multiplication and $n^2(n-1)$ addition operations, or $(2n^3 - n^2)$ arithmetical operations.

Next let us consider the multiplication of two matrices

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad 1 \leqslant i, j \leqslant n \, .$$

In total we compute n^3 multiplication and $n^2(n-1)$ addition operations, or $(2n^3 - n^2)$ arithmetical operations.

We can make an important conclusion that the complexity of multiplication of matrices is one order larger than for matrix addition.

Next let us consider the multiplication of two matrices

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad 1 \leqslant i, j \leqslant n \,.$$

In total we compute n^3 multiplication and $n^2(n-1)$ addition operations, or $(2n^3 - n^2)$ arithmetical operations.

We can make an important conclusion that the complexity of multiplication of matrices is one order larger than for matrix addition.

The complexity of both operations is estimated by using the same metric – the number of arithmetical operations.

However, not all algorithms are suitable for this measure.
For example, in data sorting and search algorithms the most important actions are comparing data and swapping two elements of the set.

However, not all algorithms are suitable for this measure.
For example, in data sorting and search algorithms the most important actions are comparing data and swapping two elements of the set.

Therefore we consider a more general definition:

The complexity of any algorithm is equal to the number of basic operations of this algorithm.

However, not all algorithms are suitable for this measure.
For example, in data sorting and search algorithms the most important actions are comparing data and swapping two elements of the set.

Therefore we consider a more general definition:

The complexity of any algorithm is equal to the number of basic operations of this algorithm.

Then the size of the problem is equal to the complexity of the known best algorithm used to solve it.

Let us recall some simple mathematical results, which greatly facilitate the analysis of algorithms.

Let us recall some simple mathematical results, which greatly facilitate the analysis of algorithms.

Very often the same algorithm is used to solve a given problem with various initial data.

For example, a company re-sorts information (data records) about their customers every day. As we will see data retrieval is in particular efficient when the data is already sorted.

1. The best case complexity

A given algorithm can be particularly efficient for some types of data. It is important to know how quickly we can solve such tasks and be able to recognize such data in advance.

This estimate is also called optimistic.

1. The best case complexity

A given algorithm can be particularly efficient for some types of data. It is important to know how quickly we can solve such tasks and be able to recognize such data in advance.

This estimate is also called optimistic.

Let D_n be a set of all cases of initial data.
There are n elements in each collection d_m:

$$D_n = \{d_m : \ d_m = (a_1, a_2, \ldots, a_n), \quad a_j \in A, \ \ j = 1, 2, \ldots, n\}.$$

Denote by $T(d_m)$ a complexity of the algorithm for a specific collection of data d_m.

1. The best case complexity

A given algorithm can be particularly efficient for some types of data. It is important to know how quickly we can solve such tasks and be able to recognize such data in advance.

This estimate is also called optimistic.

Let D_n be a set of all cases of initial data.
There are n elements in each collection d_m:

$$D_n = \{ d_m : \ d_m = (a_1, \, a_2, \, \dots, \, a_n), \quad a_j \in A, \ \ j = 1, 2, \dots, n \} \, .$$

Denote by $T(d_m)$ a complexity of the algorithm for a specific collection of data d_m.

The best case complexity is given by

$$T_G(n) = \min_{d_m \in D_n} T(d_m) \, .$$

2. The worst case complexity

The given algorithm may be inefficient for some types for data. It is important to know how long we may have to wait for the result and be able to identify such data.

This estimate is also called pessimistic.

2. The worst case complexity

The given algorithm may be inefficient for some types for data. It is important to know how long we may have to wait for the result and be able to identify such data.

This estimate is also called pessimistic.

The complexity of the worst case is defined as

$$T_B(n) = \max_{d_m \in D_n} T(d_m) \, .$$

3. The average complexity

The given algorithm may be inefficient for some types for data, but often such sets are very few. Therefore, the most important information for most users is to know how long it takes on average to wait for the result. Such information allows a rational usage of human and hardware resources.

3. The average complexity

The given algorithm may be inefficient for some types for data, but often such sets are very few. Therefore, the most important information for most users is to know how long it takes on average to wait for the result. Such information allows a rational usage of human and hardware resources.

The avarage case complexity of the algorithm is given by

$$T_V(n) = \frac{1}{|D_n|} \sum_{d_m \in D_n} T(d_m) \,,$$

3. The average complexity

The given algorithm may be inefficient for some types for data, but often such sets are very few. Therefore, the most important information for most users is to know how long it takes on average to wait for the result. Such information allows a rational usage of human and hardware resources.

The avarage case complexity of the algorithm is given by

$$T_V(n) = \frac{1}{|D_n|} \sum_{d_m \in D_n} T(d_m)\,,$$

Examples of such estimates will be given for most algorithms presented in our lectures.

Basic methods for development of algorithms

Each day new problems are formulated in our virtual reality.

In many cases they are nontrivial and it is not sufficient to use existing algorithms to solve them.

Basic methods for development of algorithms

Each day new problems are formulated in our virtual reality.

In many cases they are nontrivial and it is not sufficient to use existing algorithms to solve them.

Thus we should be ready to develop new algorithms suited for the given problem.

It is important to know how to attack such challenges, what training is needed and how long the studies will take.

Basic methods for development of algorithms

Each day new problems are formulated in our virtual reality.

In many cases they are nontrivial and it is not sufficient to use existing algorithms to solve them.

Thus we should be ready to develop new algorithms suited for the given problem.

It is important to know how to attack such challenges, what training is needed and how long the studies will take.

The good news is that only a few basic methods exist for development of efficient algorithms. Thus we always have a short list of starting points.

No clouds are in the blue sky?

No clouds are in the blue sky?

Unfortunately, this does not mean that familiarization with the basic methods will guarantee that without efforts you will be able to construct efficient algorithms for any new problem.

No clouds are in the blue sky?

Unfortunately, this does not mean that familiarization with the basic methods will guarantee that without efforts you will be able to construct efficient algorithms for any new problem.

I don't promise that you won't have to read textbooks and solve practical programming tasks (and more is better).

No clouds are in the blue sky?

Unfortunately, this does not mean that familiarization with the basic methods will guarantee that without efforts you will be able to construct efficient algorithms for any new problem.

I don't promise that you won't have to read textbooks and solve practical programming tasks (and more is better).

I don't promise that in one month you will be able to rediscover by yourself the existing most famous algorithms.

No clouds are in the blue sky?

Unfortunately, this does not mean that familiarization with the basic methods will guarantee that without efforts you will be able to construct efficient algorithms for any new problem.

I don't promise that you won't have to read textbooks and solve practical programming tasks (and more is better).

I don't promise that in one month you will be able to rediscover by yourself the existing most famous algorithms.

But You will see that parctically all algorithms in Top 10 list are developed by using these methods.

I hope, that this note is already a strongly motivating argument.

Splitting of the problem (Divide-and-Conquer method)

Most fast algorithms are developed by using the following simple idea – divide a task into smaller tasks, solve them and construct a solution of the initial full problem.

Splitting of the problem (Divide-and-Conquer method)

Most fast algorithms are developed by using the following simple idea – divide a task into smaller tasks, solve them and construct a solution of the initial full problem.

We should answer the following two questions:

Splitting of the problem (Divide-and-Conquer method)

Most fast algorithms are developed by using the following simple idea – divide a task into smaller tasks, solve them and construct a solution of the initial full problem.

We should answer the following two questions:

1. How to divide the task into a finite number of smaller tasks?

Splitting of the problem (Divide-and-Conquer method)

Most fast algorithms are developed by using the following simple idea – divide a task into smaller tasks, solve them and construct a solution of the initial full problem.

We should answer the following two questions:

1. How to divide the task into a finite number of smaller tasks?
2. How to reduce the number of smaller tasks, since the direct analysis of all smaller tasks may fail even with the fastest supercomputers.

Testing of all cases

Let us consider a simple but powerfull method, which is based on complete checking of all cases of tasks. This method became popular with the advent of computers.

Testing of all cases

Let us consider a simple but powerfull method, which is based on complete checking of all cases of tasks. This method became popular with the advent of computers.

An important advantage of this strategy is that, after considering all options, no additional calculations are required.

Testing of all cases

Let us consider a simple but powerfull method, which is based on complete checking of all cases of tasks. This method became popular with the advent of computers.

An important advantage of this strategy is that, after considering all options, no additional calculations are required.

How to determine the age of three children?

John and Peter are good friends and they met today. OK, the last their meeting was quite a long time ago...

Testing of all cases

Let us consider a simple but powerfull method, which is based on complete checking of all cases of tasks. This method became popular with the advent of computers.

An important advantage of this strategy is that, after considering all options, no additional calculations are required.

How to determine the age of three children?

John and Peter are good friends and they met today. OK, the last their meeting was quite a long time ago...

During conversation Peter explained that this day is very special for him, because all three his children – Inga, Julia and Justin – celebrate their birthdays. Peter suggested to John, who is a professional mathematician, to guess the age of each child.

To make the task easier, Peter pointed out that the sisters are no older than Justin, and Inga does not have a younger sister.

To make the task easier, Peter pointed out that the sisters are no older than Justin, and Inga does not have a younger sister.

In addition he also informed that after multiplying the ages of all three children, we get the number 36.

To make the task easier, Peter pointed out that the sisters are no older than Justin, and Inga does not have a younger sister.

In addition he also informed that after multiplying the ages of all three children, we get the number 36.

After some thought, John stated, that he still does not have enough information.

Then Peter said that the sum of the years of all three children's ages coincides with the number of windows of the house, by which they stand.

To make the task easier, Peter pointed out that the sisters are no older than Justin, and Inga does not have a younger sister.

In addition he also informed that after multiplying the ages of all three children, we get the number 36.

After some thought, John stated, that he still does not have enough information.

Then Peter said that the sum of the years of all three children's ages coincides with the number of windows of the house, by which they stand.

John thought again and stated that the new information is really very important, yet it is not sufficient to tell the answer. Therefore, he needs a little help.

To make the task easier, Peter pointed out that the sisters are no older than Justin, and Inga does not have a younger sister.

In addition he also informed that after multiplying the ages of all three children, we get the number 36.

After some thought, John stated, that he still does not have enough information.

Then Peter said that the sum of the years of all three children's ages coincides with the number of windows of the house, by which they stand.

John thought again and stated that the new information is really very important, yet it is not sufficient to tell the answer. Therefore, he needs a little help.

New Peter's remark was short – the eyes of the eldest child are blue. Upon learning this, John immediately told the age of each child!

From the first condition we learn, that the product of the ages of three children equals 36. It is easy to verify that there are only eight different options when this condition is met. They are given in the table.

TABLE: Eight options, when the product equals 36

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
Inga	1	1	1	1	1	2	2	3
Julia	1	2	3	4	6	2	3	3
Justin	36	18	12	9	6	9	6	4

We assume that John knows how many windows has a house where both friends met.
So from the second condition he founds out what the sum of the children's years is equal to.

We assume that John knows how many windows has a house where both friends met.
So from the second condition he founds out what the sum of the children's years is equal to.

But he still didn't have enough information to tell the answer.

We compute the sums of ages for each option:

$$
\begin{aligned}
1+1+36 &= 38, & 1+2+18 &= 21, \\
1+3+12 &= 16, & 1+4+\ \ 9 &= 14, \\
1+6+\ \ 6 &= 13, & 2+2+\ \ 9 &= 13, \\
2+3+\ \ 6 &= 11, & 3+3+\ \ 4 &= 10.
\end{aligned}
$$

We assume that John knows how many windows has a house where both friends met.
So from the second condition he founds out what the sum of the children's years is equal to.

But he still didn't have enough information to tell the answer.

We compute the sums of ages for each option:

$$
\begin{aligned}
1+1+36&=38, & 1+2+18&=21,\\
1+3+12&=16, & 1+4+\ \ 9&=14,\\
1+6+\ \ 6&=13, & 2+2+\ \ 9&=13,\\
2+3+\ \ 6&=11, & 3+3+\ \ 4&=10.
\end{aligned}
$$

Now it is clear that this sum of years is equal to 13, because in all other cases John would already know the age of each child.

Two options remain $(1, 6, 6)$ and $(2, 2, 9)$.

Two options remain (1, 6, 6) and (2, 2, 9).

Since Justin is the eldest child only in the second case (that his eyes are blue, of course, does not matter), we conclude that Peter is raising twins Inga and Julia, who have turned two years old, and nine-year-old son Justin.

Security of information, passwords

In most cases we are looking for new algorithms with the minimal complexity. But now we give an example when the large complexity of the full search (so called brute force search algorithms) is a very desirable and useful feature.

Security of information, passwords

In most cases we are looking for new algorithms with the minimal complexity. But now we give an example when the large complexity of the full search (so called brute force search algorithms) is a very desirable and useful feature.
The secrecy and security of communications is currently a very important task and its importance will only increase in the future. Unauthorized reading of information and hybrid attacks on data centers grows also fastly.

Security of information, passwords

In most cases we are looking for new algorithms with the minimal complexity. But now we give an example when the large complexity of the full search (so called brute force search algorithms) is a very desirable and useful feature.
The secrecy and security of communications is currently a very important task and its importance will only increase in the future. Unauthorized reading of information and hybrid attacks on data centers grows also fastly.

Security of modern public key cryptographic algorithms is based on mathematical results for some very classical number theory problems. The state of the art in this field still is such, that the solution of these problems is possible only after a complete re-selection of all cases, and the total number of different options is so high that it is not possible to "crack" the password while the information is still relevant.

The Recursion Method

The object (task, data structure) is defined by using recurrences if the definition is based on the smaller objects of the same structure.

The Recursion Method

The object (task, data structure) is defined by using recurrences if the definition is based on the smaller objects of the same structure.

Factorials. In mathematics, the factorial of a non-negative integer $n \in \mathbb{N}$, denoted by $n!$, is the product of all positive integers less than or equal to n.

The Recursion Method

The object (task, data structure) is defined by using recurrences if the definition is based on the smaller objects of the same structure.

Factorials. In mathematics, the factorial of a non-negative integer $n \in \mathbb{N}$, denoted by $n!$, is the product of all positive integers less than or equal to n.

The recurrence definition can be given as:

$$n! = \begin{cases} n \cdot (n-1)!, & \text{if } \; n > 0, \\ 1, & \text{if } \; n = 0. \end{cases}$$

The object (task, data structure) is defined by using recurrences if the definition is based on the smaller objects of the same structure.

Factorials. In mathematics, the factorial of a non-negative integer $n \in \mathbb{N}$, denoted by $n!$, is the product of all positive integers less than or equal to n.
The recurrence definition can be given as:

$$n! = \begin{cases} n \cdot (n-1)!, & \text{if } \ n > 0, \\ 1, & \text{if } \ n = 0. \end{cases}$$

It follows from this definition that

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1 \, .$$

Fibonacci numbers. Fibonacci numbers are a sequence of numbers where every number is the sum of the preceding two numbers.
They are defined by:

$$f_n = \begin{cases} f_{n-1} + f_{n-2}, & \text{if} \;\; n > 1, \\ f_0 = 1, \;\; f_1 = 1\,. \end{cases}$$

Fibonacci numbers. Fibonacci numbers are a sequence of numbers where every number is the sum of the preceding two numbers.
They are defined by:

$$f_n = \begin{cases} f_{n-1} + f_{n-2}, & \text{if} \;\; n > 1, \\ f_0 = 1, \;\; f_1 = 1\,. \end{cases}$$

Task 1. Compute f_4 value.

Fibonacci numbers. Fibonacci numbers are a sequence of numbers where every number is the sum of the preceding two numbers.
They are defined by:

$$f_n = \begin{cases} f_{n-1} + f_{n-2}, & \text{if } \ n > 1, \\ f_0 = 1, \ \ f_1 = 1 \, . \end{cases}$$

Task 1. Compute f_4 value.

Task 2. This task is more complicated: compute f_{30} and f_{31} values. What conclusion You can make?

We will show how recursion helps to solve logical games and puzzles.

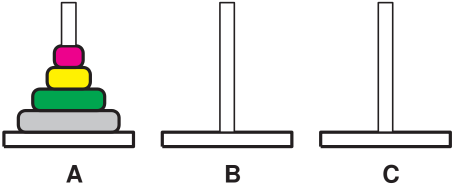
We will show how recursion helps to solve logical games and puzzles.

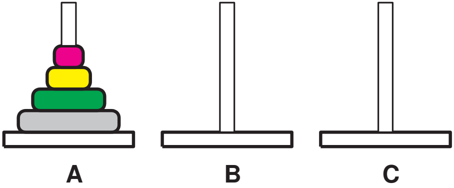
The Towers of Hanoi is a mathematical game or puzzle consisting of three rods and a number of disks of various diameters, which can slide onto any rod.
The puzzle begins with the disks stacked on one rod in order of decreasing size, the smallest at the top, thus approximating a conical shape. The objective of the puzzle is to move the entire stack to the last rod, obeying the following rules:

1. Only one disk may be moved at a time.

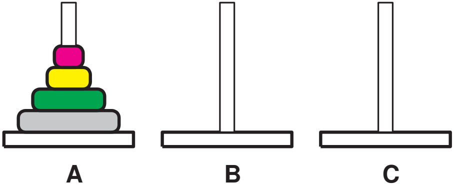
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.

3. No disk may be placed on top of a disk that is smaller than it.


A
B
C

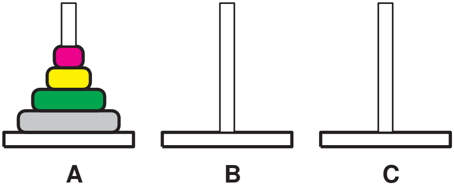


We can solve this problem by applying the following strategy:



We can solve this problem by applying the following strategy:

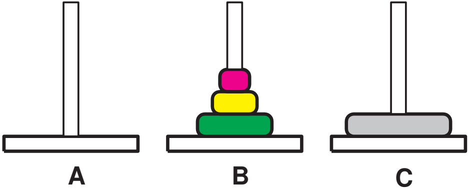
First, we move the $(n-1)$ top discs of A on the B rod, and we use the empty rod C as an auxiliary one.



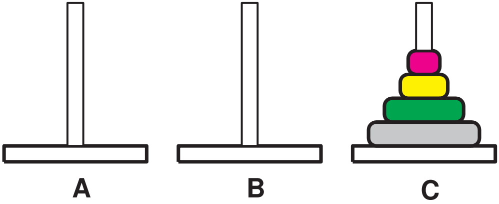
We can solve this problem by applying the following strategy:

First, we move the $(n-1)$ top discs of A on the B rod, and we use the empty rod C as an auxiliary one.

Then we move the largest disc of A on C rod.



In the last step, by applying the same algorithm we move the discs from B onto C and use A as an auxiliary rod.



Recursive algorithm to solve the problem of Hanoi towers

```
HanoiTowers (n, A, B, C)
begin
  (1)  if  (  n > 0  )
  (2)      HanoiTowers (n-1, A, C, B);
  (3)      move (A, C);
  (4)      HanoiTowers (n-1, B, A, C);
  (5)  end if
end  HanoiTowers
```

Divide-and-conquere method

Divide-and-conquere method

1. This method recursively breaks down a problem into two or more smaller subproblems.

Divide-and-conquere method

1. This method recursively breaks down a problem into two or more smaller subproblems.
2. We find solutions of all smaller subproblems.

Divide-and-conquere method

1. This method recursively breaks down a problem into two or more smaller subproblems.
2. We find solutions of all smaller subproblems.
3. From them we make the solution of the whole problem.

Divide-and-conquere method

1. This method recursively breaks down a problem into two or more smaller subproblems.
2. We find solutions of all smaller subproblems.
3. From them we make the solution of the whole problem.

Divide-and-conquere method

1. This method recursively breaks down a problem into two or more smaller subproblems.
2. We find solutions of all smaller subproblems.
3. From them we make the solution of the whole problem.

Smaller subproblems again can be solved by applying the divide-and-conquere algorithm.

The recursive division is done until the received tasks are easily solved.