ITMO UNIVERSITY

Saint Petersburg, Russia

Analysis and Development of Algorithms

Lecture 1

Introduction. Design of algorithms.

Complexity analysis of algorithms

About me

ITMO UNIVERSITY

Dr Petr Chunaev

NCCR site (in Russian)

- Candidate of Physical and Mathematical Sciences, PhD in Mathematics
- Senior Researcher at National Center for Cognitive Research of ITMO University

Possible topics of your scientific work:

My Google Scholar

- Predictability analysis of customer behavior (preference drift, etc.)
- Generation of synthetic data for modelling customer behavior
- Optimization theory on attributed and multiplex networks (community detection, link prediction, etc.)
- Dynamic pricing and stratification of customer behavior

ITMO more than a UNIVERSITY

Contact information

- chunaev@itmo.ru (NOT for sending reports, only questions)
- telegram @yamatico (ONLY urgent questions)
- 16 Birzhevaya Line, room 208 (send hand-writing letters)

How will we work?

ITMO UNIVERSITY

Telegram channel for lectures and tasks

Scores table

How your **Course Score** (at most 100) and your **Course Mark** (A, B, C, D, E or FX) correspond to each other:
if 95<=Score<=100, then Mark is A
if 85<=Score<95, then Mark is B
if 70<=Score<85, then Mark is C
if 60<=Score<70, then Mark is D
if 50<=Score<60, then Mark is E
if 0<=Score<50, then Mark is FX

For the reports, you get **Reports Score** (at most 100); for the exam, **Exam Score** (at most 100). Your **Course Score is the average of them**.
You can get your **Course Mark** (according to your **Reports Score only**) AUTOMATICALLY if all your reports are accepted (i.e. you have OK for all the tasks).

Course tutorial (in Russian)

To send reports **algorithms_itmo@mail.ru**

Instructions on practical tasks and reports will be given at the practical classes

Course content (8 lectures, 8 reports, 1 exam)

ITMO UNIVERSITY

Lecture 1. Introduction. Design of algorithms. Complexity analysis of algorithms. Significant operations. Experimental complexity analysis. Task 1. Experimental time complexity analysis.

Lecture 2. Unconstrained nonlinear optimization. Direct methods: one-dimensional (Exhaustive search, Dichotomy, Golden section search) and multidimensional (Exhaustive search, Gauss method, Nelder-Mead method). Task 2. Unconstraint nonlinear optimization. Direct algorithms.

Lecture 3. Unconstrained nonlinear optimization. First- and second-order methods: Gradient Descent, Conjugate Gradient, Newton method, Quasi-Newton method, Levenberg-Marquardt algorithm. Task 3. Unconstraint nonlinear optimization algorithms. First- and second-order methods.

Lecture 4. Unconstrained optimization algorithms. Stochastic and metaheuristic algorithms (Monte Carlo method, Simulated annealing, Evolutionary algorithms, Differential evolution, Particle swarm optimization). Task 4. Unconstraint optimization algorithms. Stochastic and metaheuristic algorithms.

Lecture 5. Introduction to graphs and basic algorithms on graphs. Graphs and their representations. Depth-first search and its applications. Breadth-first search and its applications. Task 5. Graphs and their representations. Depth-first search and its applications. Breadth-first search and its applications.

Lecture 6. Graph algorithms. Path search algorithms on weighted graphs. Dijkstra's algorithm. A* algorithm. Bellman-Ford algorithm Task 6. Path search algorithms on weighted graphs. Dijkstra's algorithm. A* algorithm. Bellman-Ford algorithm.

Lecture 7. Graph algorithms. Tools for network analysis. Task 7. Graph algorithms. Tools for network analysis.

Lecture 8. Analysis of modern algorithms. Task 8. Analysis of modern algorithms.

ITMOre than a UNIVERSITY

Exam: a test on the information from lectures and practical tasks

NB: The course is for refreshing some known information and making the knowledge level of students from different bachelor programs more or less equal.

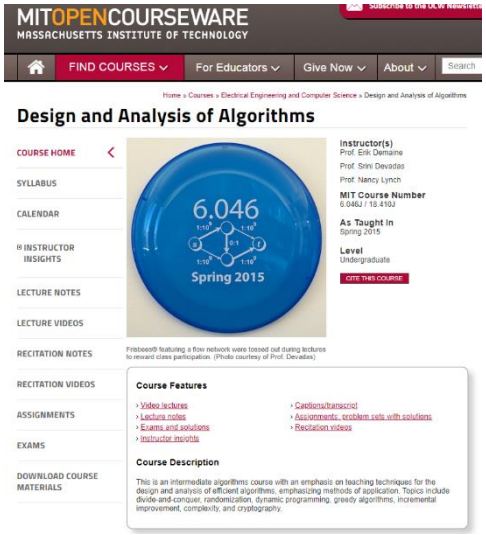
References

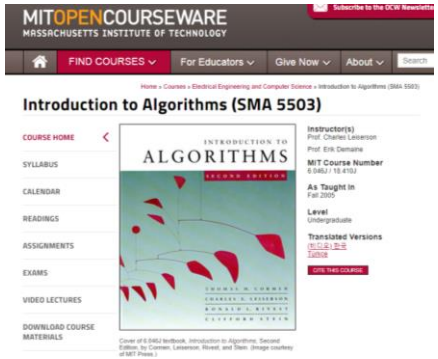
ITMO UNIVERSITY

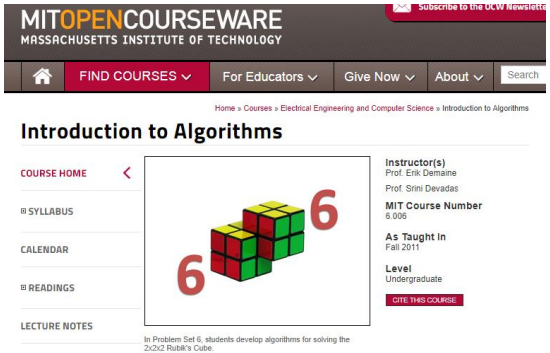
Algorithms and data structures

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest; Clifford Stein. Introduction to Algorithms. 3rd edition. MIT Press, 2009.
2. Ananiy Levitin. Introduction to the Design and Analysis of Algorithms. Addison Wesley, 2011.

Other textbooks, lecture notes and YouTube tutorials







ITMOre than a UNIVERSITY

Development of algorithms

Why do we study algorithms?

ITMO UNIVERSITY

Their impact is broad and far-reaching.

Internet. Web search, packet routing, distributed file sharing, ...

Biology. Human genome project, protein folding, ...

Computers. Circuit layout, file system, compilers, ...

Computer graphics. Movies, video games, virtual reality, ...

Security. Cell phones, e-commerce, voting machines, ...

Multimedia. MP3, JPG, DivX, HDTV, face recognition, ...

Social networks. Recommendations, news feeds, advertisements, ...

Physics. N-body simulation, particle collision simulation, ...

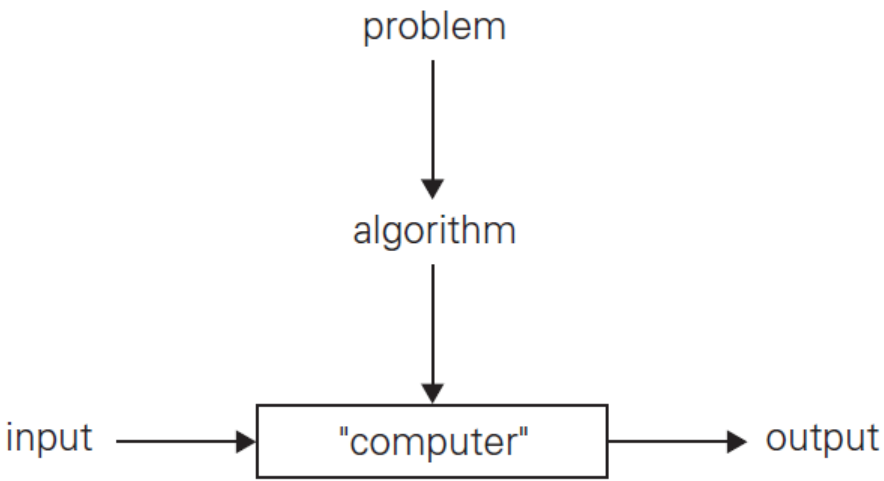
⋮

Definition of an algorithm

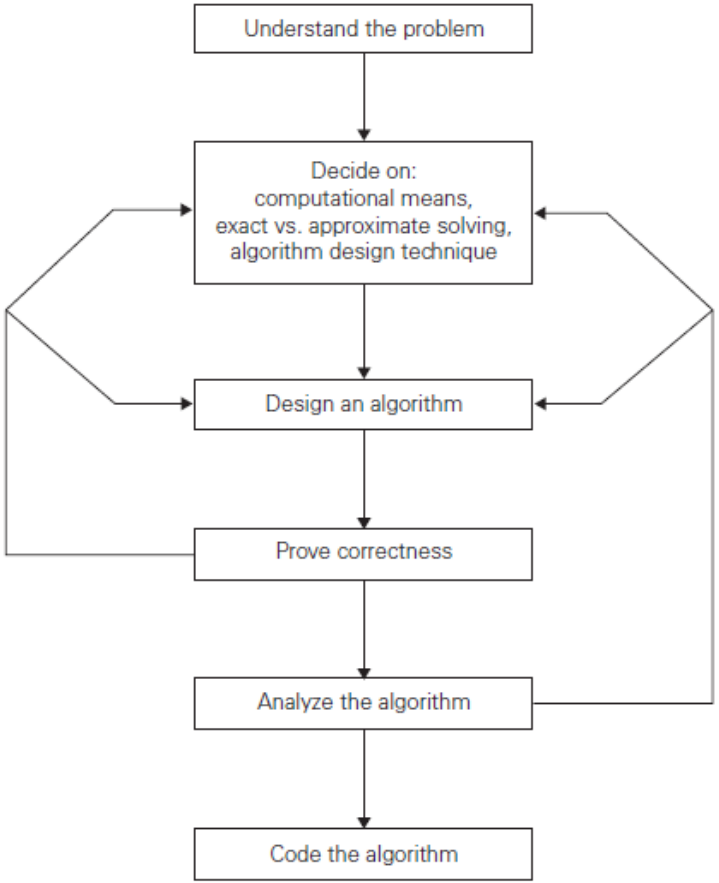
ITMO UNIVERSITY

An ***algorithm*** is a sequence of clear instructions for solving a problem, i.e., for obtaining a required output for any acceptable input in a finite amount of time.

- The class of inputs for which an algorithm works has to be specified carefully.
- The same algorithm can be represented in several different ways.
- There may exist several algorithms for solving the same problem.
- Algorithms for the same problem can be based on very different ideas and can solve the problem with dramatically different speed.


problem
algorithm
input
"computer"
output

Algorithm development process



Algorithm development process

ITMO UNIVERSITY

Understanding the problem

Determining the capabilities of a computing device

Sequential algorithms
Parallel algorithms
Learning algorithms without being tied to the parameters of a specific computer system

Choosing between exact or approximate method for solving the problem
Exact method
Approximation method

Choosing the right data structures
Many algorithms require very specific data structures

Algorithm development process

ITMO UNIVERSITY

Algorithm Design Techniques

A design technique is a versatile approach applied to algorithmic solutions to a wide range of problems

Algorithm Representation Methods

- Natural language
- Pseudocode: *a mixture of one of the natural languages (input / output, comments) and constructs specific to the programming language (for, if and while)*
- Flowcharts: *drawings consisting of a sequence of geometric shapes connected by arrows, with the help of which each step of the algorithm execution is described (inconvenient for large algorithms)*

Algorithm development process

ITMO UNIVERSITY

Algorithm Correctness Evaluation

Prove that the chosen algorithm produces the required result for any correct values of the input data in a limited period of time (mathematical induction is a universal method)

Algorithm Analysis

- ✓ Time efficiency, or time complexity: *running time function*
- ✓ Space efficiency, or space complexity: *the amount of RAM required for the algorithm to work*
- ✓ Simplicity
- ✓ Generality, or universality
- ✓ Optimality

Algorithm Encoding

Debugging and testing

Analysis of algorithms. Design techniques. Data structures

Important Problem Types

ITMO UNIVERSITY

- **Sorting:** to rearrange the items of a given list in some order (e.g. using a *key*)
- **Searching:** to find a given value, called a *search key*, in a given (multi)set
- **String processing**: to process a *string* that is a sequence of characters from an alphabet
- **Graph problems:** to analyse a graph, i.e. a collection of points called vertices, some of which are connected by line segments called edges, informally
- **Combinatorial problems:** to find a combinatorial object – such as a permutation, a combination, or a subset – that satisfies certain constraints
- **Geometric problems:** to explore object geometry
- **Numerical problems:** to solve equations and systems of equations, compute definite integrals, evaluate functions, etc.

Algorithm Performance

ITMO UNIVERSITY

- The term “analysis of algorithms” usually means an investigation of algorithm’s efficiency with respect to two resources: running time and memory space (aka **computational complexity**).

- Time
  - Instructions take time.
  - How fast does the algorithm perform?
  - What affects its runtime?
- Space
  - Data structures take space
  - What kind of data structures can be used?
  - How does choice of data structure affect the runtime?

➢ **We will focus on time:**

- How to estimate the time required for an algorithm?

Naive Approach: run an algorithm for different size of input data and analyze the results (actually, your Task 1).

Analysis of Algorithms

ITMO UNIVERSITY

- Analyze algorithms independently of
  - *specific implementations, computers,* or *data.*

- To analyze algorithms:
  - First, we start to count the number of significant operations in a particular solution to assess its efficiency.
  - Then, we will express the efficiency of algorithms using growth functions.

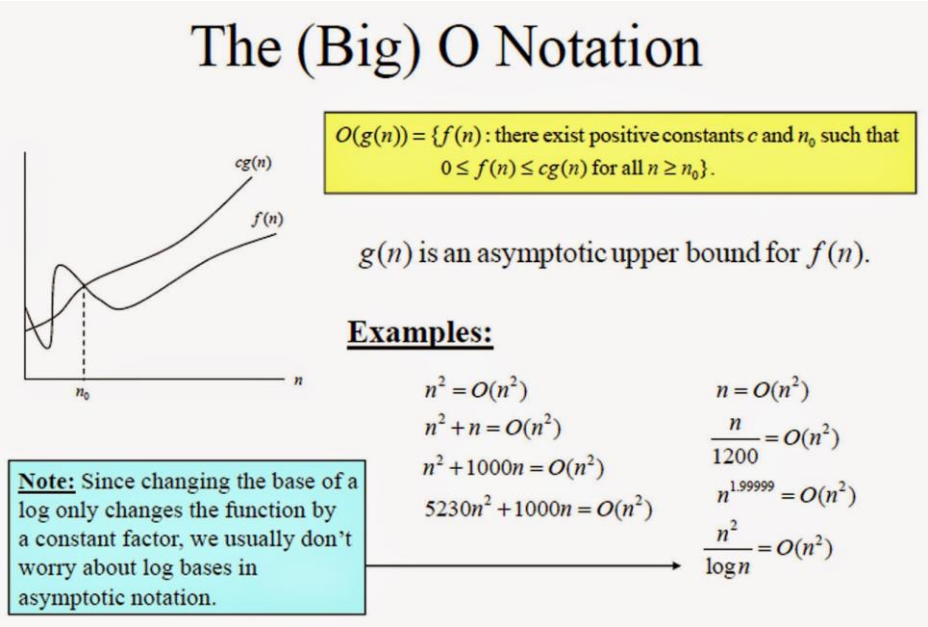
What is Important?

- If the problem size is always very small, we can probably ignore the algorithm's efficiency.
- We have to find the trade-off between the algorithm's time requirement and its memory requirements.

ITMO more than a UNIVERSITY

Analysis of Algorithms (Worst Case)

- A running time function, T(n), yields the time required to execute the algorithm of a problem of size n.
  - T(n) may contain unspecified constants.
  - We cannot determine this function exactly.
  - Example: $T(n) = an^2 + bn + c$, where a, b and c are unspecified constants.
- We can compare the efficiency of two algorithms by comparing their growth rates. The lower the growth rate, the faster the algorithm, at least for large values of n.
- *For example,* $T(n) = an^2 + bn + c$ has growth rate $O(n^2)$
- The *goal* of the *algorithm designer:* an algorithm with as low a growth rate of the running function, T(n), of that algorithm as possible.



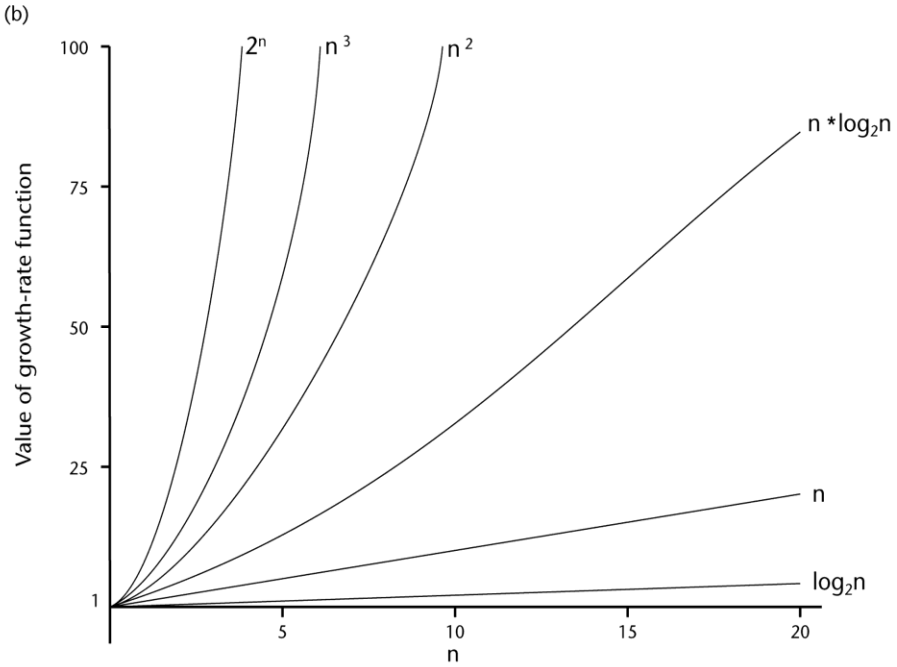
General Rules for Estimation

- **Loops**
  - The running time of a loop is at most the running time of the statements inside of that loop times the number of iterations.
- **Nested Loops**
  - Running time of a nested loop containing a statement in the inner most loop is the running time of statement multiplied by the product of the size of all loops.
- **Consecutive Statements**
  - Just add the running times of those consecutive statements.
- **If/Else**
  - Never more than the running time of the test plus the larger of running times of the options.

HW: why?

order of growth	name	typical code framework	description	example
1	constant	`a = b + c;`	statement	add two numbers
log N	logarithmic	`while (N > 1) { N = N / 2; ... }`	divide in half	binary search
N	linear	`for (int i = 0; i < N; i++) { ... }`	loop	find the maximum
N^2	quadratic	`for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) { ... }`	double loop	check all pairs
N^3	cubic	`for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) for (int k = 0; k < N; k++) { ... }`	triple loop	check all triples
2^N	exponential	Combinatorial algorithms	exhaustive search	check all subsets

A Comparison of Growth-Rate Functions



https://www.youtube.com/watch?v=YoZPTyGL2IQ&ab_channel=CS50

The Execution Time of Algorithms

Example: Simple If-Statement

The input is n

	Cost	Times
`if (n < 0)`	c1	1
`  av = -n;`	c2	1
`else`		
`  av = n;`	c3	1

Total Cost = c1 + max(c2,c3) = O(1)

The Execution Time of Algorithms

ITMO UNIVERSITY

Example: Simple Loop

The input is n

	Cost	Times
`i = 1;`	c1	1
`sum = 0;`	c2	1
`while (i <= n) {`	c3	n+1
`  i = i + 1;`	c4	n
`  sum = sum + i;`	c5	n
`}`		

Total Cost = c1 + c2 + (n+1)c3 + n(c4 + c5)=O(n)

The Execution Time of Algorithms

ITMO UNIVERSITY

Example: Nested Loop

The input is n

```
                                Cost        Times
i=1;                            c1            1
sum = 0;                        c2            1
while (i <= n) {                c3            n+1
 j=1;                           c4            n
 while (j <= n) {               c5            n*(n+1)
     sum = sum + i;             c6            n*n
     j = j + 1;                 c7            n*n
   }
   i = i +1;                    c8            n
}
```

Total Cost = $c1 + c2 + (n+1)*c3 + n*c4 + n*(n+1)*c5+n*n*c6+n*n*c7+n*c8=O(n^2)$

A ***data structure*** is a particular scheme of organizing related data items.

The nature of the data items is dictated by the problem at hand; they can range from elementary data types (e.g., integers or characters) to data structures (e.g., a one-dimensional array of one-dimensional arrays is often used for implementing matrices).

There are a few data structures that have proved to be particularly important for computer algorithms.

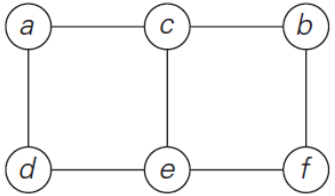
An abstract collection of objects with several operations that can be performed on them is called an **abstract data type**.

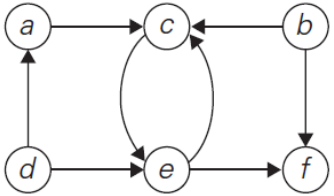
List, *stack*, *queue*, *priority queue,* and *dictionary* are important examples of abstract data types.

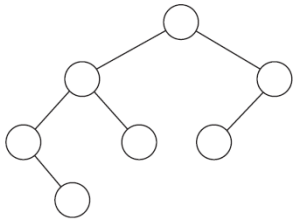
Basic Data Structures

ITMO UNIVERSITY

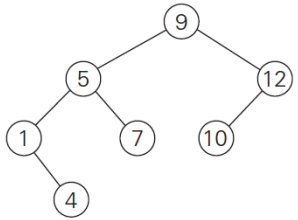
- Graph
- Tree
- Forest



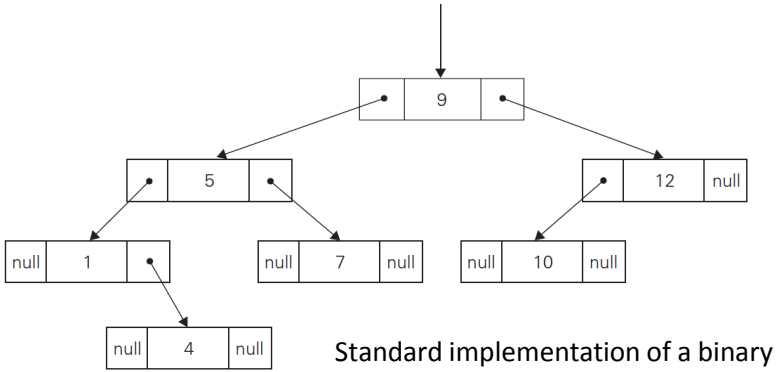




Binary tree



Binary search tree



Standard implementation of a binary tree

A binary tree is usually implemented for computing purposes by a collection of nodes corresponding to vertices of the tree. Each node contains some information associated with the vertex (its name or some value assigned to it) and two pointers to the nodes representing the left child and right child of the vertex, respectively.

Algorithm design techniques

ITMO UNIVERSITY

- Algorithm *design techniques* (or “strategies” or “paradigms”) are general approaches to solving problems algorithmically, applicable to a variety of problems from different areas of computing.

Basic techniques:

- Brute-force or exhaustive search
- Divide and Conquer
- Greedy Algorithms
- Dynamic Programming
- Backtracking
- etc.

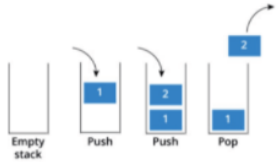
Active learning (20 min)

ITMO UNIVERSITY

ITMO More than a UNIVERSITY

What is the data structure shown in the picture? Name and describe it. Give examples of algorithms where it can be used.

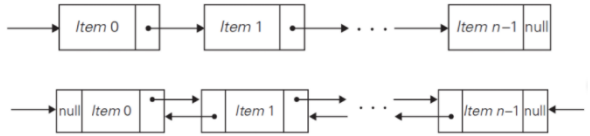
What is the data structure shown in the picture? Name and describe it. Give examples of algorithms where it can be used.



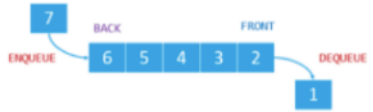
Describe the data structure called a hash table. Give examples of algorithms where it can be used.

Describe the data structures called a set and a dictionary. Give examples of algorithms where it can be used.

Which data structures are shown in the picture? Name and describe them. Give examples of algorithms where they can be used.



What is the data structure shown in the picture? Name and describe it. Give examples of algorithms where it can be used.



- Brute-force or exhaustive search
- Divide and Conquer
- Greedy Algorithms
- Dynamic Programming
- Backtracking

Thank you for your attention!

ITMOre than a UNIVERSITY