DATA STRUCTURES AND ALGORITHMS

Algorithms: types, representation, evaluation

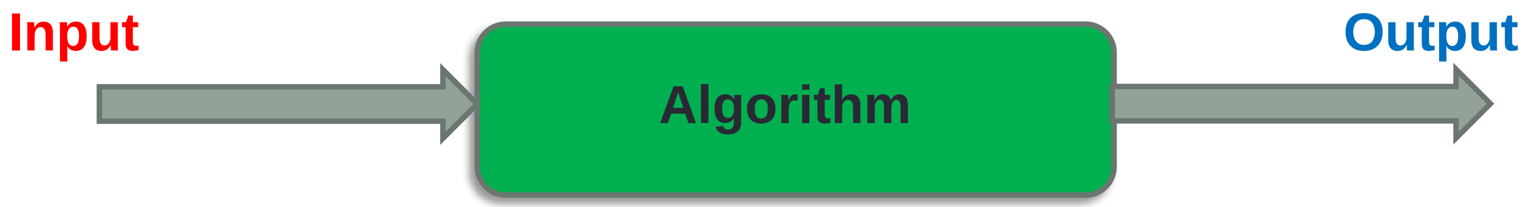
Summary of the previous lecture

- AVL tree
  - Rotation to the left and to the right
- Bayer tree (B-tree)
- Binary heap

Algorithms

An algorithm **is an effective method expressed as a finite list of well-defined instructions for calculating a function.**

Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that will proceed through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state.



Algorithms

Algorithms are essential to the way computers process data.

Computer programs contain algorithms that detail the specific instructions a computer should perform (in a specific order) to carry out a specified task, such as calculating employees' paychecks or printing students' report cards.

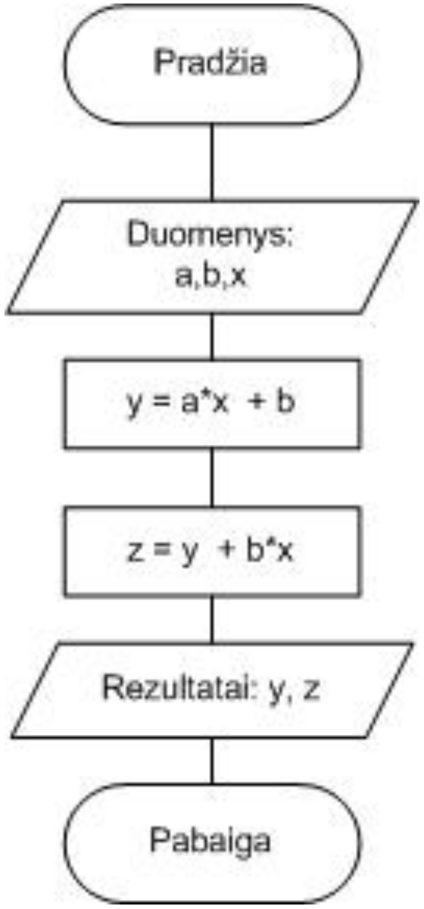
Algorithm can be considered to be any sequence of operations that can be simulated by a computer system.

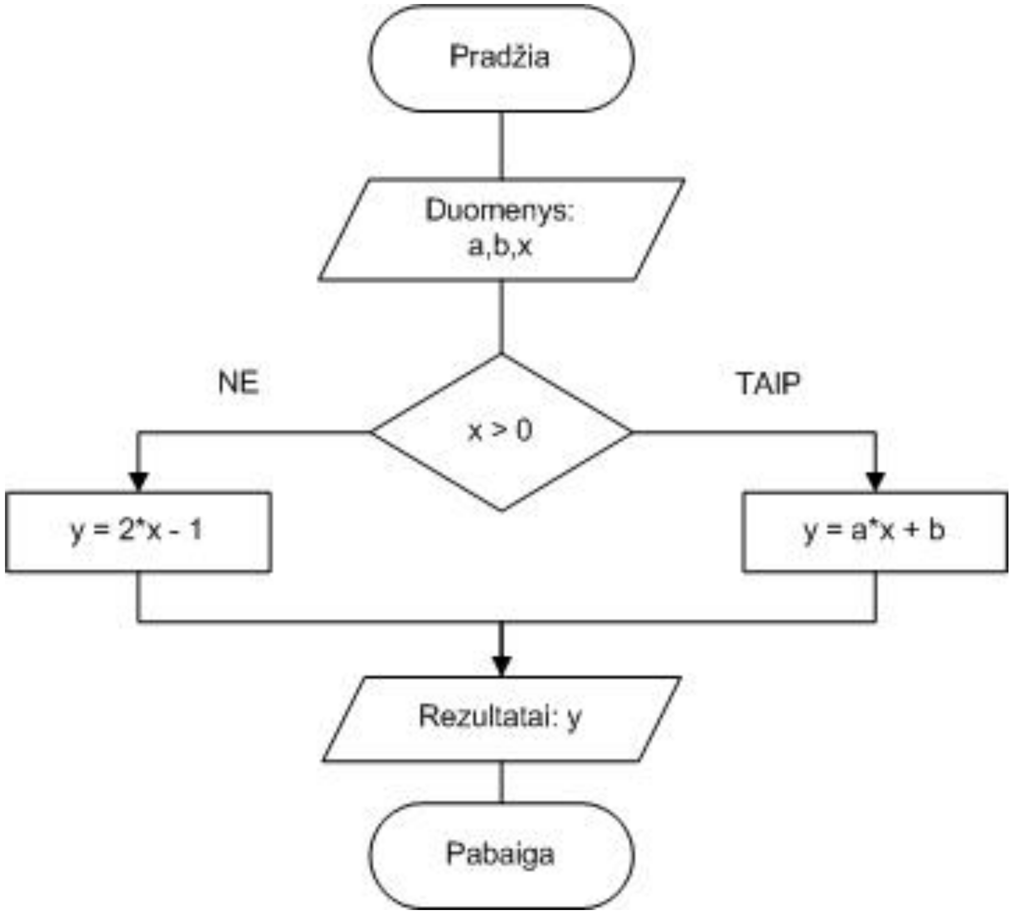
Types of algoritms

Types of the algorithms:

- Linear
  - $y = ax + b$
- Branching
  - $y = ax + b$ (if $x < 0$) or $y = 2x – 1$ (if $x > 0$)
- Complex
  - e-mail, use authorization

Types of algorithms





Algorithms

Features of the algorithms:

- Completeness
- Explicitness
- Input and output data
- Efficiency

Efficiency of the algorithms can be evaluated based on:

- Number of basic operations
- Amount of memory required for calculations

Example 1

Let's analyze multiplication of two matrices A and B. Assume that each matrix has **n** rows and **n** columns.

Elements of the matrix C = A*B can be calculated as follows:

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

Number of multiplication operations is $\mathbf{n^3}$

Number of adding operations is $\mathbf{n^2(n-1)}$

Total number of operations is $\mathbf{2n^3 - n^2}$.

Example 2

Assume that you have three one dimensional arrays **A**, **B**, **C**. Size of all arrays is the same and is equal to **N.** You must check if there is any combination of three numbers (each number from different array A, B, C) that sum of these three numbers is equal to 0.

Input:

A = {5; 44; 23; 12}   B = {-45; 32; -25; 11}   C = {7; -8; -19; 31}

Output:

44 – 25 – 19 = 0

Example 2

Algorithm No. 1

Worst algorithm: N^3

- Calculate all possible combination of numbers from A, B, C

Algorithm No. 2

Middle algorithm: $N^2/2 + 2*N*\log_2 N + N*(N+1)$

- Sort arrays A (descending), B (ascending)
- Calculate sums of all combinations of elements from A and B
- Searching opposite value of the sums obtained from step No.2

Algorithm No. 3

Best algorithm: $N^2/2 + N*\log_2 N + N*\log_2 N$

- Calculate sums of all combinations of elements from A and B
- Sort arrays C and searching using binary search

Classification

Classification of the algorithms based on implementation.

- **Recursion**
  - A recursive algorithm is one that invokes (makes reference to) itself repeatedly until a certain condition matches
- **Iteration**
  - Iterative algorithms use repetitive constructs like loops and stacks to solve the given problems.
- **Logical**
  - An algorithm may be viewed as controlled logical deduction. This notion may be expressed as: Algorithm = logic + control

Classification

- **Deterministic or non-deterministic**
  - Deterministic algorithms solve the problem with exact decision at every step of the algorithm whereas non-deterministic algorithms solve problems via guessing although typical guesses are made more accurate through the use of heuristics.
- **Serial, parallel or distributed**
  - If computers execute one instruction of an algorithm at a time such algorithms is called serial. Parallel algorithms take advantage of computer architectures where several processors can work on a problem at the same time, whereas distributed algorithms utilize multiple machines connected with a network.

Classification

Classification of algorithms is by their design methodology or paradigm:

- Brute-force
- Divide and conquer
- Dynamic programming
- Linear programming

Algorithms

Algorithms can be presented as:

- Pseudo code
- Flowchart
- Natural language
- Programing language
- Drakon chart
- Control tables

Pseudo code $\Rightarrow$

```
begin int Factorial (n)
   s = 1;
  for ( i = 2; i <= n ; i++ )
           s = s * i;
  return ( s );
 end Factorial
```

Flowchart

Flowcharts are used in designing and documenting complex processes or programs.

Like other types of diagrams, they help visualize what is going on and thereby help the viewer to understand a process, and perhaps also find flaws, bottlenecks, and other less-obvious features within it.

There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions.

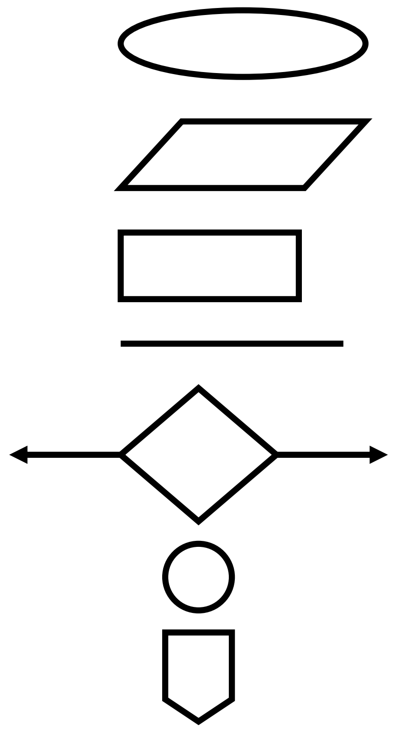
The two most common types of boxes in a flowchart are:

- **a processing step**, usually called *activity*, and denoted as a rectangular box
- **a decision**, usually denoted as a diamond.

Flowchart

Symbols of flowchart:

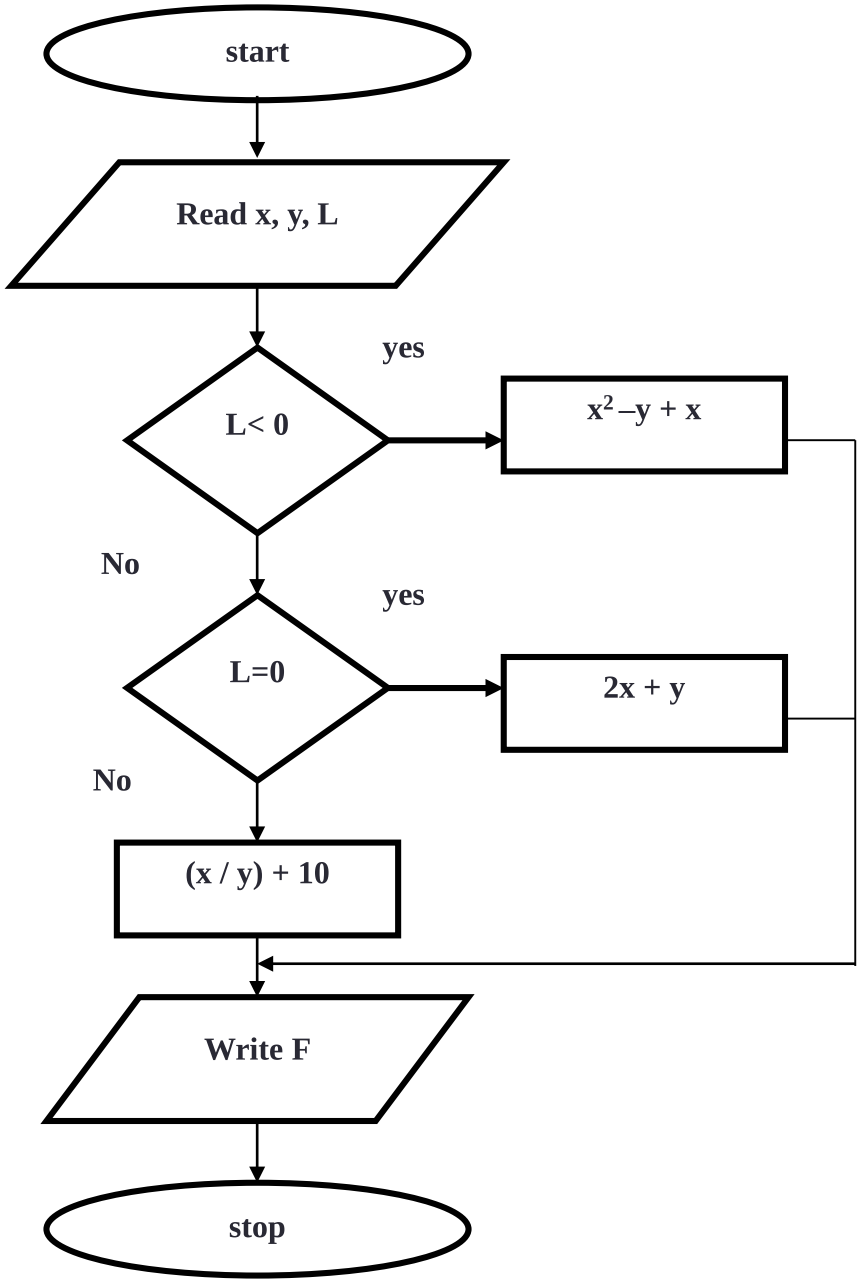
1. Terminal (start, stop)
2. Input / output
3. Processing
4. Flow: concerned with direction
5. Decision: for logic comparison
6. Connector
7. Off-page connector



Example

Draw a flowchart to compute:

$$F = \begin{cases} x^2 - y + x & \text{if} \quad L < 0 \\ 2x + y & \text{if} \quad L = 0 \\ (x / y) + 10 & \text{if} \quad L > 0 \end{cases}$$

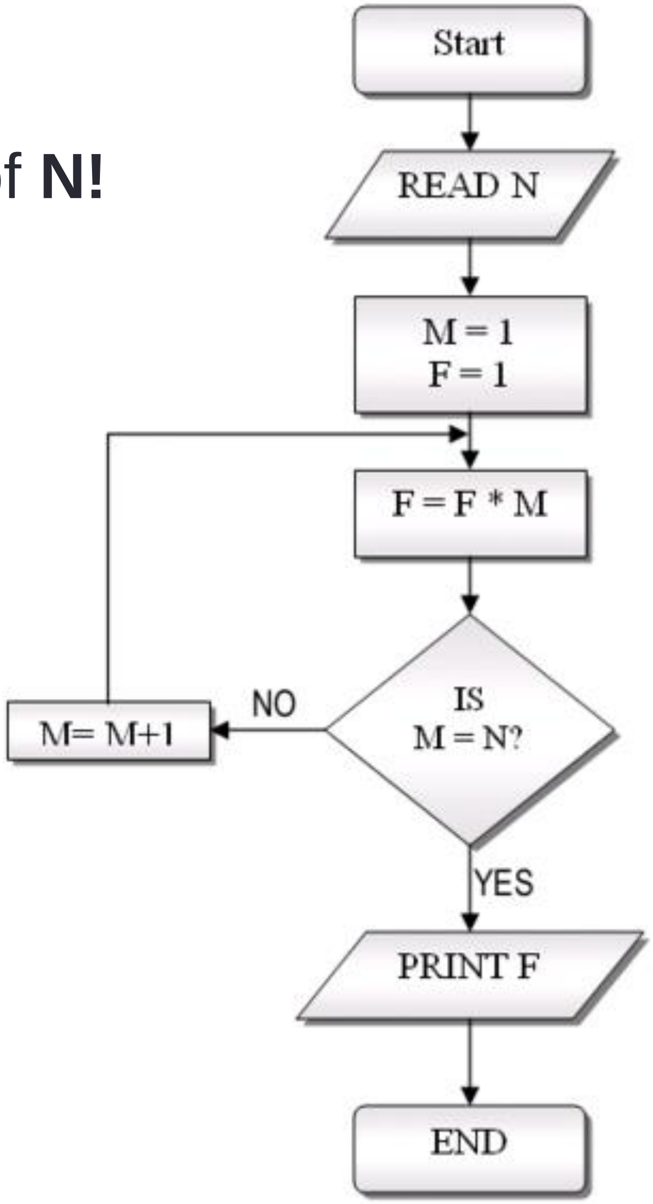


Flowchart example

Example for computing the factorial of **N!**

```
int Factorial (n)
{   s = 1;
  for ( i = 2; i <= n ; i++ )
            s = s * i;

 return  s;
```



Software and Algorithms

The following steps are recommended to solve the problem:

- Problem formulation (*what data is given and what to find*)
- Model selection
- Algorithm development
- Algorithm evaluation (*known data and results*)
- Algorithm realization (*variable, data structures, etc*.)
- Algorithm analysis (runing time, efficiency)
- Software testing

Software testing

Software testing includes:

- Does software calculates things that were required
- Application limits of the algorithm
- Testing of typical use cases
- Efficiency evaluation with small and large data sets
- Evaluation of the documentation

Practice

No.1

Built flowchart for the following algorithm.

Find max and min numbers in one-dimensional array and swap these numbers.