# Data Structures and Algorithm Analysis

# 10

Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

# Root of the word Algorithm

- The word Algorithm comes from the name of the scientist al-Khowarizmi
  - He wrote a book about algebra and introduced some some techniques of mathematics …

# Algorithm -definition

- Informally, an *algorithm* is any well-defined *computational procedure* that takes some value, or set of values, as *input* and produces some value, or set of values, as *output*.
  - Input can be Numbers, Text, Image, Video, etc

- An algorithm is thus a sequence of computational steps that transform the input into the output.
- An algorithm is a step-by-step procedure for solving a problem in a finite amount of time.

# Algorithms are Every Where

- Operating Systems
  - Priorities, scheduling (queues, heaps)
- Networks:
  - Routing (Trees, graphs), Error detection/corrections
- Multimedia, Image Processing …
- Compilers
  - Storing data information, optimizations etc (different data structures, lists etc)
- Databases
  - Sorting, searching

# What is algorithm analysis ?

- Algorithm analysis has two aspects:
  - Running time:
    - *How much time is taken to complete the algorithm execution?*
  - Storage requirement
    - *How much memory is required to execute the program?*

- Mostly we'll deal with the Running times in this course

# Why do you need this course ?

A computer scientist must be prepared for tasks like:

*" … This is the problem.  Solve it ..."*

In such a situation it does not suffice to know how to code?

You must be able  to

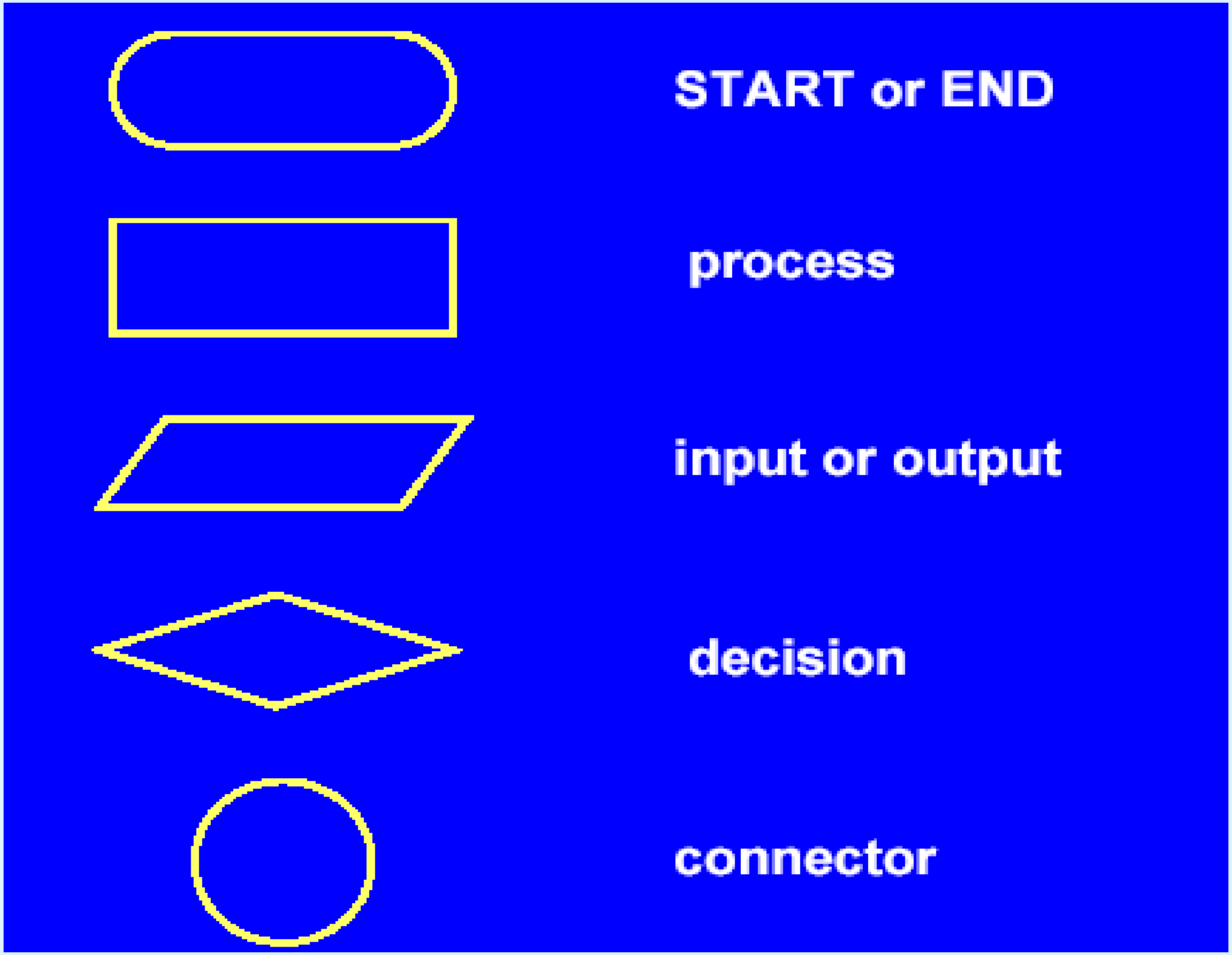
- find an adequate algorithm     or
- develop a new algorithm to solve the problem

# How can algorithms be described?

There are two basic instruments to describe algorithms:

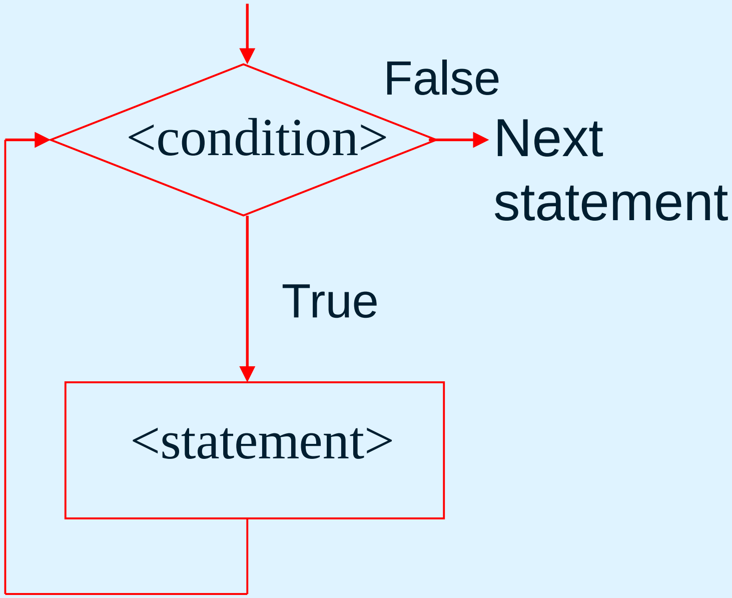
- Flowcharts:
  - graphical description of the flow of processing steps
  - at present it is only of historical importance
  - however, sometimes are used to describe the overall structure of an application
- Pseudocode:
  - artificial language based on
    - vocabulary (set of keywords)
    - syntax (set of rules used to construct the language "phrases")
  - *"A not so restrictive"* as a programming language

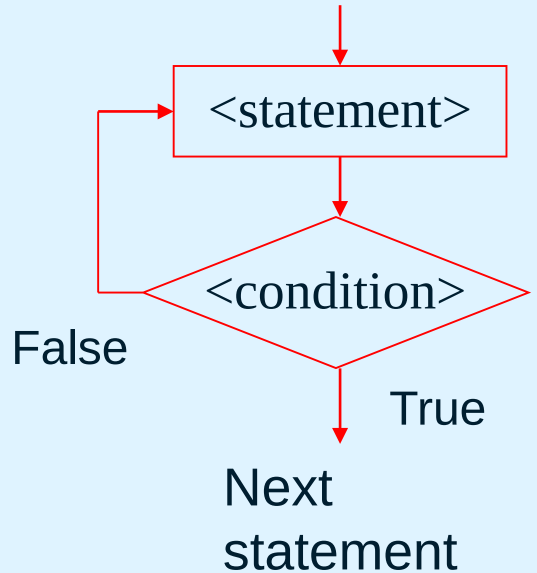
# Flowchart Symbols



# Flow Charts - Example

For Loop




Loop

# Pseudocode: Rules

## Assignment (operator)

$$v := \text{<expression>}$$

or $$v \leftarrow \text{<expression>}$$

Expression consists of Operators and Operands

- Operands: variables, constant values
- Operators: arithmetical, relational, logical

Example: $v \leftarrow a+b*c$

# Writing Operators in Algorithms

- Arithmetical:

  + (addition), - (subtraction), *(multiplication),
  / (division),  ^ (power),
  DIV (integer quotient),
  MOD (remainder)

- Relational:

  = (equal), <> (different),
  < (less than), <= (less than or equal),
  >(greater than) >= (greater than or equal)

- Logical:

  OR (disjunction), AND (conjunction), NOT (negation)

## Representing Input, Output and Conditional Statements in algorithms

- READ v1,v2 // Inputs of variables
- WRITE e1,e2 // Output of variables

- IF THEN condition in algorithms

  **IF <condition>**

  **THEN**

  **statements**

- IF THEN ELSE in algorithms

  **IF <condition>**

  **THEN Statement**

  **ELSE**

  **Statements**

# Loops in *Algorithms*

**While Loop**

WHILE <condition>
DO
<statements>

**Repeat Loop**

REPEAT
*<statements>*
UNTIL <condition>

# Properties an Algorithm

- Generality
- Finiteness
- Non-ambiguity
- Efficiency

# Correctness or Generality

- An algorithm is said to be ***correct*** if, for every input instance, it gives a correct output.
  - It means that an algorithm applies to all instances of input data not only for few particular instances

- *Example:*

  Let's consider the problem of increasingly ordering a sequence of values.

  (2,1,4,3,5) → (1,2,3,4,5)

  input data → output

# Generality: *Example*

Method:

Step 1: 2 ⟷ 1 4 3 5

Step 2: 1 2 4 3 5

Step 3: 1 2 4 ⟷ 3 5

Step 4: 1 2 3 4 5

Algorithm:

- compare the first two elements: if they are not in the desired order then swap them
- compares the second and the third element and do the same

.....

- continue until the last two elements were compared

The sequence has been ordered

# Generality:*Example*

- Is this algorithm sufficiently general ?
- Does it assure the ordering of ANY sequence of values ? NO

Example:

3 2 1 4 5

2 3 1 4 5

2 1 3 4 5

2 1 3 4 5

In this case the algorithm doesn't work

# Finiteness

- An algorithm has to terminate, i.e. it should always stop after a finite number of steps. Algorithm should have terminate condition or state.

Example: Generate all odd numbers less than 10

Step1: Assign 1 to x;

Step2: Increase x by 2;

Step3: If x=10 then

STOP;

else

GO TO Step 2

How does this algorithm work ?

# Finiteness:*Example*

How does this algorithm work and what does it produce?

- Step1: Assign 1 to x;
- Step2: Increase x by 2;
- Step3: If x=10

  then STOP;

- else Print x; GO TO Step 2;

x=1

x=3 x=5 x=7 x=9 x=11

The algorithm generates odd numbers but it does not stop.
The above algorithm does not have Finiteness property.

# Finiteness:*Example*

The following algorithm has now Finiteness property:

Step1: Assign 1 to x;

Step2: Increase x by 2;

Step3: If x >= 10

then STOP;

else Print x; GO TO Step 2

# Non-ambiguity

- The operations in an algorithm must be EXPLICITLY specified. At each step of execution, the next step has to be very clear.

The following is an example of ambiguous algorithm

Step 1: Set x to value 0

Step 2: Either increment x with 1 or decrement x with 1

Step 3: If x € [-2,2]

GO TO Step 2;

else Stop.

Step 2 is ambiguous and not clear.

# Non-ambiguity *(Example)*

Let's modify the previous algorithm as follows:

Step 1: Set x to value 0
Step 2: Flip a coin
Step 3: IF one obtains tail
THEN increment x with 1
ELSE decrement x with 1
Step 3: If x € [-2,2]
GO TO Step 2,
else
Stop.

- The algorithm can be executed but … different executions can be different

# Efficiency

- An algorithm should need a reasonable amount of computing resources:
  - memory and time

- We will study Efficiency in terms of time in detail.
- Assessing efficiency needs knowledge of Analysis of Algorithm