Dept. of CSE
Varendra University

Introduction to Algorithms

Course Instructor:

Sumaiya Tasnim

Lecturer, Department of CSE

Varendra University

Acknowledgement

Dr. Md Nakib Hayat Chowdhury

Head, Department of CSE

Bangladesh Army University of Science and Technology (BAUST)

Mosiur Rahman Sweet

Former Lecturer, Department of CSE

Varendra University

Md. Muktar Hossain

Lecturer, Department of CSE

Varendra University

Why study Algorithms?

Professor Donald E. Knuth stated "Computer Science is the study of algorithms"

Programs will not exist without algorithms...

What is an Algorithm?

“An Algorithm is a Set of Well-defined Instructions in Sequence to Solve a Problem.”

An algorithm is a tool for solving a **well-specified computational problem**. It is a sequence of computational steps that transform the input into the output.

- An algorithm is said to be correct if, for every input instance, it halts with the correct output.
- An incorrect algorithm might not halt at all on some input instances, or it might halt with other than the desired output.

Algorithms did not start with Computers,
they have been with us since ancient time;
nor they are limited to Computer Science…

Characteristics of Algorithm:

Unambiguous

Algorithm should be clear and unambiguous. It must lead to only one meaning.

Input

Algorithm should have 0 or more well-defined inputs.

Output

Algorithm should have 1 or more well-defined outputs and should match the desired output.

Characteristics of Algorithm:

Finite

An algorithm must terminate after a finite number of steps.

Feasible

An algorithm should be feasible with the available resources.

Independent

Algorithm should have step-by-step directions which should be independent of any system.

Algorithm VS Program:

Algorithm	Program
An algorithm is a set of well-defined instructions in sequence to solve a problem.	A program is an implementation of an algorithm to be run on a specific system.
It does not deal with machine specific details.	It deals with machine specific details.
Algorithm is written using natural language.	Programs are written using a specific programming language.
Think of it as a class.	Think of it as a object of the class.

Algorithm Design:

Design: Focuses on creating algorithms to solve a specific problem.

- Understanding the Problem
- Choosing a Strategy
- Developing the Algorithm
- Verification

Algorithm Design Techniques /Methods/Strategies/Approaches

Algorithm Design Techniques:

- Exhaustive Search or Brute-Force Search
- Backtracking, Branch and Bound
- Greedy Method
- Divide and Conquer
- Dynamic Programming
- Machine Learning
- Randomized Algorithm
- Online algorithm and offline algorithm

How to Design an Algorithm?

- The **problem** that is to be solved by this algorithm i.e. clear problem definition.
- The **constraints** of the problem must be considered while solving the problem.
- The **input** to be taken to solve the problem.
- The **output** is to be expected when the problem is solved.
- The **solution** to this problem is within the given constraints.

How to Analyze an Algorithm?

The analysis focuses on determining the correctness of an algorithms and evaluating the performance of an algorithm.

- Correctness
- Complexity Analysis
- Best/Worst/Average Case Analysis
- Comparative Analysis

Analysis of an Algorithm: Correctness

A correct algorithm yields the correct result for all legal input values.

?

Proving the correctness of an algorithm is not at all an easy task.
It consists of two steps:

- Proving that the algorithm will always terminate.
- Proving that it will always produce correct result.

Analysis of an Algorithm: Performance

- Two main concern: Time and Space
- We actually do not count the time!
- We count the **number of steps!!**
- The lesser the number of steps, the faster the algorithm.
- Space?

Complexity Analysis:

Two factors (time and space) define the efficiency of an algorithm.
Time Complexity: Refers to the amount of time required by the algorithm to execute and get the result.

$$T(P) = C + V$$

Constant Time Part (C): Any instruction that is executed just once comes in this part.
Variable Time Part (V): Any instruction that is executed more than once, say ***n*** times, comes in this part.

Calculating exact complexity is difficult to deal with precisely because the details are very complicated. It is easier to talk about upper and lower bounds of the function.

Complexity Analysis:

Two factors (time and space) define the efficiency of an algorithm.
Space Complexity: Refers to the amount of memory required by the algorithm to store the variables and get the result.

$$S(P) = C + V$$

Fixed Part (C): The space that is required by the algorithm.
Variable Part (V): The space that can be different based on the implementation of the algorithm. For example, temporary variables, dynamic memory allocation, recursion stack space, etc.

Representing an Algorithm

Pseudocode

- Pseudo is from the Greek word “psevdes” meaning **false.**
- It is a high-level way of representing an algorithm by annotations and informative text written in plain English.
- It has no syntax like any of the programming language and thus can’t be compiled or interpreted by the computer.

Pseudocode Conventions

- Give a valid name for the pseudocode procedure.
- Use line numbers for each line of code.
- Use proper indentation for every statement in a block structure
- For flow control statements use if-else. Always end an statement with an end-if. Both if, else and end-if should be aligned vertically in same line
- Use “=“ or “←” operator for assignment statements

Pseudocode Conventions

- Array elements can be represented by specifying the array name followed by the index in square brackets.
  Example: A[i] indicates the ith element of array A

- For looping or iteration use for or while statements. Always end a for loop with end-for and a while loop with end-while.

- The conditional expression of for or while must maintain same indentation.

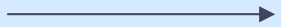
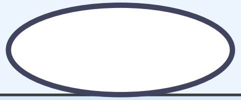
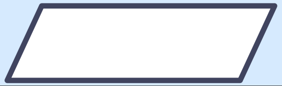
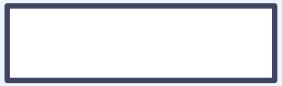
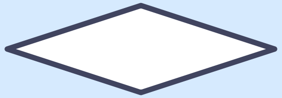
Pseudocode of Insertion Sort

```
INSERTION SORT:
1.   for i ← 2 to length[A]
2.        do key ← A[i]
3.        # insert A[i] into sorted sequence list
          a[1 .. i-1]
4.        j ← i-1
5.        while j>=0 and A[j] > key
6.             do A[j+1] ← A[j]
7.             j ← j-1
8.        end-while
9.        A[j+1] ← key
10.  end-for
```

Flowchart

- Flowchart refers to diagrammatic representation that illustrates a solution model to a given problem.
- It is a visual representation of an algorithm which describes graphically in detail the logical operations and steps of the algorithm.

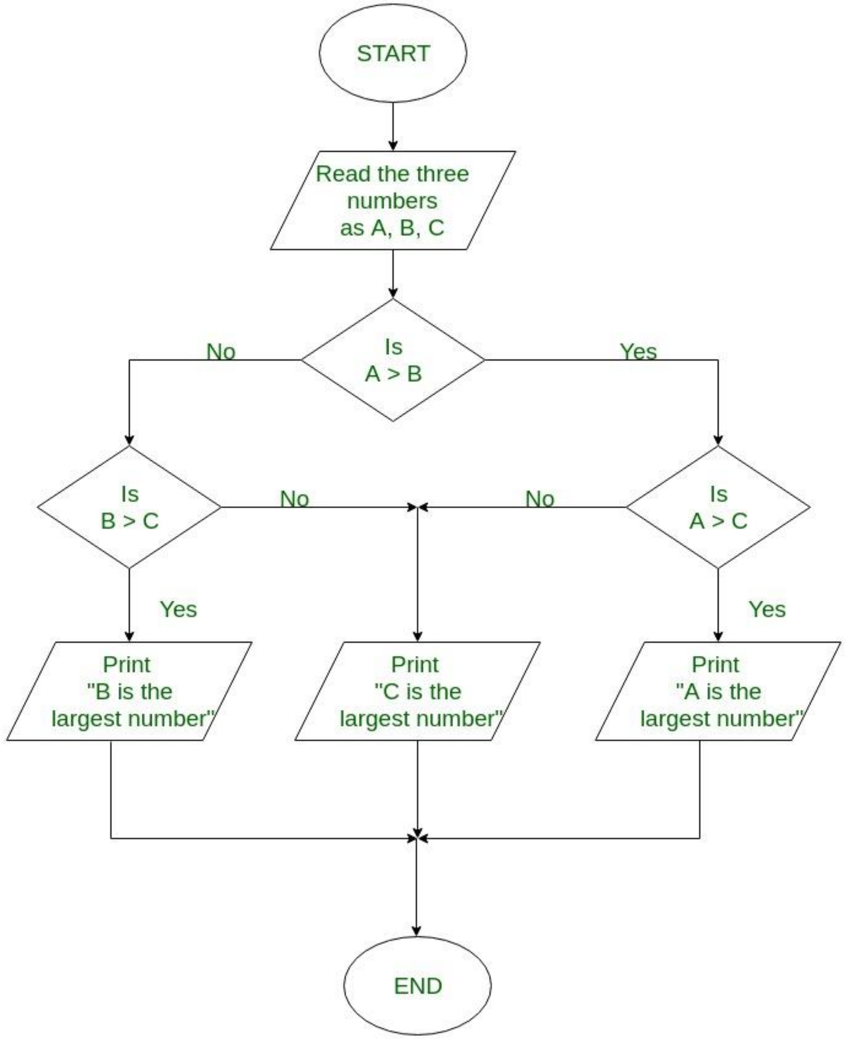
Flowchart Convention

Symbol	Purpose	Description
	Flow Line	Indicates the flow of logic by connecting symbols.
	Terminal	Represents the start and the end of a flowchart.
	I/0	Used for input and output operation.
	Processing	Used for arithmetic operations and data-manipulations.
	Decision	Used for decision making between two or more alternatives.

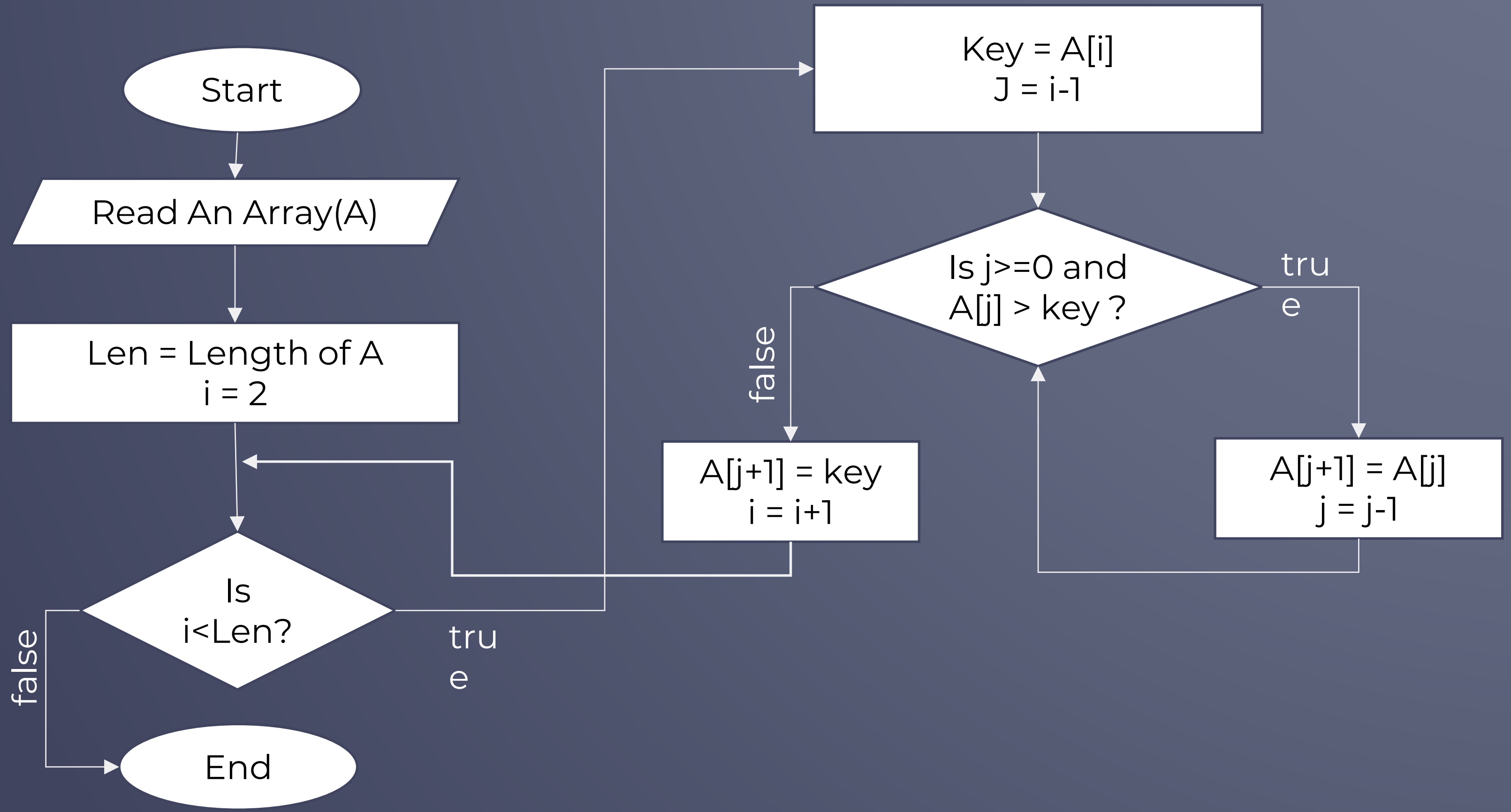
https://www.gliffy.com/blog/how-to-flowchart-basic-symbols

Flowchart Example

flowchart to find largest of 3 numbers


START
Read the three numbers as A, B, C
Is A > B
No
Yes
Is B > C
No
No
Is A > C
Yes
Yes
Print "B is the largest number"
Print "C is the largest number"
Print "A is the largest number"
END

Flowchart of Insertion Sort



"Students shouldn't go out into life without the ability to communicate. Your success in life will be determined largely by:

-your ability to speak,
-your ability to write &
-the quality of your ideas,

in that order.“

— Late MIT Prof. Patrick Winston