

# **Algorithms Design and Analysis**

Ch.1: Introduction to Algorithms

*Prepared By  
Dr. Ibrahim Attiya*

# Outline

---

- ☐ Introduction
- ☐ Importance of Algorithms
- ☐ Algorithm: Definition
- ☐ Characteristics of an Algorithm
- ☐ Ways of Writing an Algorithm
- ☐ Design and Analysis vs Analysis and Design
- ☐ Present and Future
- ☐ Points to Remember
- ☐ Key Terms

# The Goals of this Course

- In this course we will learn:
  - To *think algorithmically* and get the spirit of how algorithms are designed.
  - To get to know a *toolbox* of *classical* algorithms.
  - The various ways of writing an algorithm
  - The concept of designing an algorithm
  - A number of algorithm design *techniques* (such as divide-and-conquer).
  - To reason (in a precise and formal way) about the *efficiency* and the *correctness* of algorithms.

# Algorithm

- **Algorithms** refer to the steps to be carried out in order to accomplish a particular task. A good algorithm should use the resources like the CPU time and memory judiciously. It should also be unambiguous and understandable.
- The **five most essential things** are to be considered while writing an algorithm are as follows:
  - Time taken
  - Memory usage
  - Input
  - Process
  - Output.

# Characteristics of an Algorithm

---

- An algorithm is a sequence of steps in order to carry out a particular task.
  - ❖ It can have zero or more inputs.
  - ❖ It must have at least one output.
  - ❖ It should be efficient both in terms of memory and time.
  - ❖ It should be finite.
  - ❖ Every statement should be unambiguous.

# An example to demonstrate the importance of algorithms

- Nowadays, algorithms have become important even for biological endeavours.
- Governments across the world want to make a global database of DNAs to combat the menace of terrorism. *This would be possible only if sorting and searching algorithms are developed*, which can extract information from billions of DNAs.
- In order to accomplish this task, nature-based algorithms are being developed.

# Why Study Algorithms?

## **Reasons why one should study algorithms :**

- ✓ The study of algorithms helps in enhancing the thinking process. They are like brain stimulants that will give a boost to your thinking process.
- ✓ The study of algorithms helps in solving many problems in Computer Science, Computational Biology and Economics.
- ✓ Without the knowledge of algorithms you can become a coder but not a programmer.



# Why Study Algorithms?

---

## **Reasons why one should study algorithms :**

- ✓ A good understanding of algorithms will help you to get a job. There is an immense need in the software Industry of good programmers who can analyze the problem as well.
- ✓ It helps in creating an interest in the subject.
- ✓ The problems faced in the past pave way for deducing the possible solutions in the future.



# Ways of Writing an Algorithm

- **English like Algorithm:** It can be written in simple English but this methodology comes with its package. i.e. this methodology also has some demerits.
- Natural languages can be ambiguous and therefore
- lack the characteristic of being definite.
- Since, each step of an algorithm should be clear and should not have more than one meaning, English language like algorithms are not considered good for most of the tasks.

# Ways of Writing an Algorithm

## **Algorithm 1.1** English-like algorithm of linear search

**Step 1.** Compare 'item' with the first element of the array, A.

**Step 2.** If the two are same, then print the position of the element and exit.

**Step 3.** Else repeat the above process with the rest of the elements.

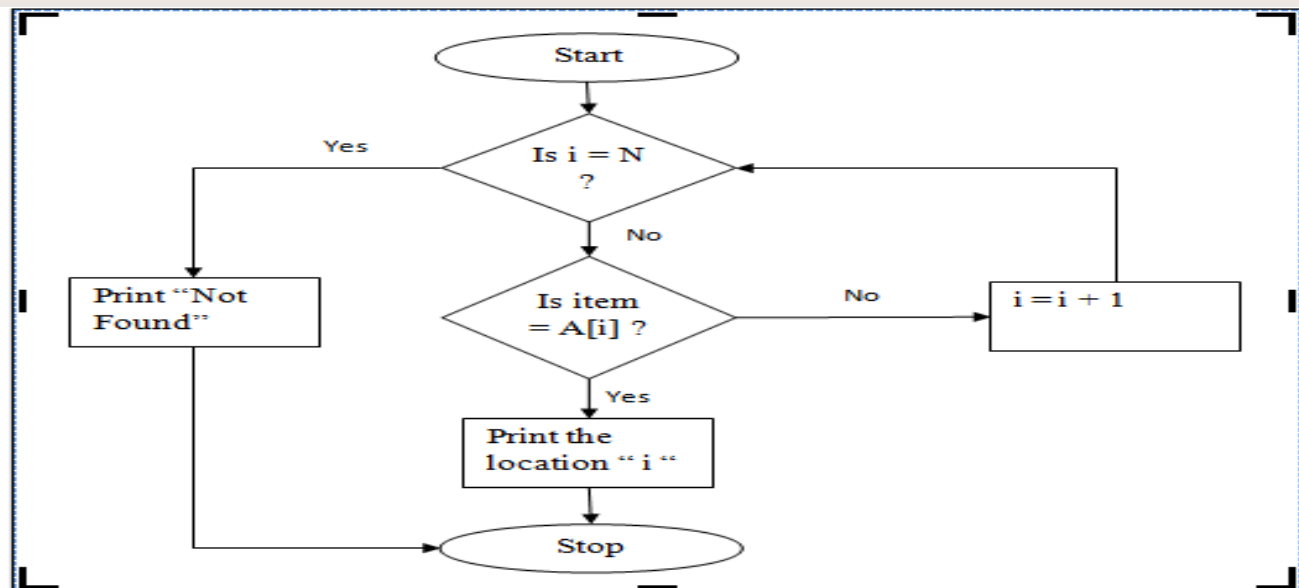
**Step 4.** If the item is not found at any position, then print 'not found' and exit.

- However, Algorithm 1.1, in spite of being simple, is not commonly used.

# Ways of Writing an Algorithm

## Flowchart:







- Flowcharts pictorially depict a process.
- They are easy to understand and are commonly used in case of simple problems.



Flowchart of linear search

# Ways of Writing an Algorithm

**Table 1.1** Flowchart conventions

S. No.	Name	Element Representation	Meaning
1.	Start/End		An oval is used to indicate the beginning and end of an algorithm.
2.	Arrows		An arrow indicates the direction of flow of the algorithm.
3.	Connectors		Circles with arrows connect the disconnected flowchart.
4.	Input/Output		A parallelogram indicates the input or output.
5.	Process		A rectangle indicates a computation.
6.	Decision		A diamond indicates a point where a decision is made.

# Ways of Writing an Algorithm

## Pseudo Code :

- The pseudo code has an advantage of being easily converted into any programming language.
- This way of writing algorithm is most acceptable and most widely used.
- In order to be able to write a pseudo code, one must be familiar with the conventions of writing it.

```
Algorithm LinearSearch(A, n, item)
{
    for i := 1 to n step 1 do
    {
        if(A[i] = item) then
        {
            write i;
            exit;
        }
    }
    write "Not Found"
}
```

# Ways of Writing an Algorithm

S. No.	Construct	Meaning
1.	// Comment	Single line comments start with //
2.	/* Comment Line 1 Comment Line 2 ○ ○ Comment Line n */	Multi-line comments occur between /* and */
3.	{ statements }	Blocks are represented using { and }. Blocks can be used to represent compound statements (collection of simple statements) or the procedures.
4.	;	Statements are delimited by ;

5.  $a > b$

6.  $a < b$

7.  $a \geq b$

8.  $a \leq b$



# Ways of Writing an Algorithm

9.  $a \neq b$

10.  $a == b$

11. if<condition>then<statement>

12. if<condition>then<statement1> else<statement2>

13. Case

{

:<condition 1>: <statement 1>



:<condition n>: <statement n>

:default: <statement n+1>

}



# Ways of Writing an Algorithm

S. No.	Construct	Meaning
18.	<pre>while&lt;condition&gt;do {     statements }</pre>	The statement depicts a <b>while</b> loop
19.	<pre>repeat     statements until&lt;condition&gt;</pre>	The statement depicts a <b>do-while</b> loop
20.	<pre>for variable = value1 to value2 {     statements }</pre>	The statement depicts a <b>for</b> loop
21.	Read	Input instruction
22.	Print	Output instruction
23.	Algorithm<name> (<parameter list>)	The name of the algorithm is <name> and the arguments are stored in the <parameter list>

# Design and Analysis

- In order to accomplish a task, a solution needs to be developed. This is called designing of an algorithm.
- The next step would be to analyze the time complexity of the algorithm.

## **Algorithm 1.3** Finding maximum element from an array

```
Algorithm Max(A, n)
{
    Max = A[1];
    for i = 2 to n step 1 do
    {
        if(A[i]>Max) then
        {
            Max=A[i];
        }
    }
    Print "The maximum element is A[i]"
}
```

# Analysis and Design

- First of all you have to analyze the maximum time complexity, CPU usage, memory complexity etc. and then you will decide on the algorithms you may be using in order to accomplish the tasks.
- It may be stated at this point that a general approach is being used in Design and Analysis, however Analysis and Design is far more practical and hence implementable.



**Figure 1.2** Design and analysis



**Figure 1.3** Analysis and design

# Present and Future

- **To summarize, in developing an algorithm, the following things are taken care of:**
  1. Make sure that the solution is correct.
  2. Try to make sure that the time consumption is least.
  3. Try to make sure that the memory consumption is also least.
  4. Try to keep each statement unambiguous and definite.
- However, doing so can be a problem in the following cases:
  1. When the search space is so large that it is not possible to obtain an exact solution.
  2. The algorithm implements a non-deterministic machine.

# Present and Future

---

➤ **The future algorithms will depend on**

1. Artificial intelligence techniques, which would help in optimization.
2. Ability to utilize hardware capabilities and process power as well.
3. Non-determinism which will have to be integrated so as to solve real-time problem and parallel processes.

# Points to Remember

- An algorithm is different from a program. An algorithm is finite; a program can be infinite.
- An algorithm can be pictorially depicted by a flow chart.
- The analysis of an algorithm is essential in order to judge whether it can be implemented in the given conditions.
- The analysis of an algorithm may consider time or space or both.
- The design of an algorithm can be followed by the analysis of the requirements. This approach is referred to as analysis followed by design.
- The algorithm can be designed in order to accomplish a task, and then can be analyzed. This approach is referred to as analysis and design.

# Key Terms

---

- ❑ **Algorithm** It is a sequence of steps to accomplish a particular task efficiently and effectively.
- ❑ **Constraint** the conditions that control the selection of elements in backtracking.
- ❑ **Explicit Constraint** the conditions that determine how should various  $x_i$ 's are related to each other.
- ❑ **Implicit Constraint** An element  $x_i$  can take its values only from a legal set of values called domain.



A silver-colored metal spiral binding is visible along the left edge of the page, consisting of a series of loops.

*Any Questions?*