Design and Analysis of Algorithm

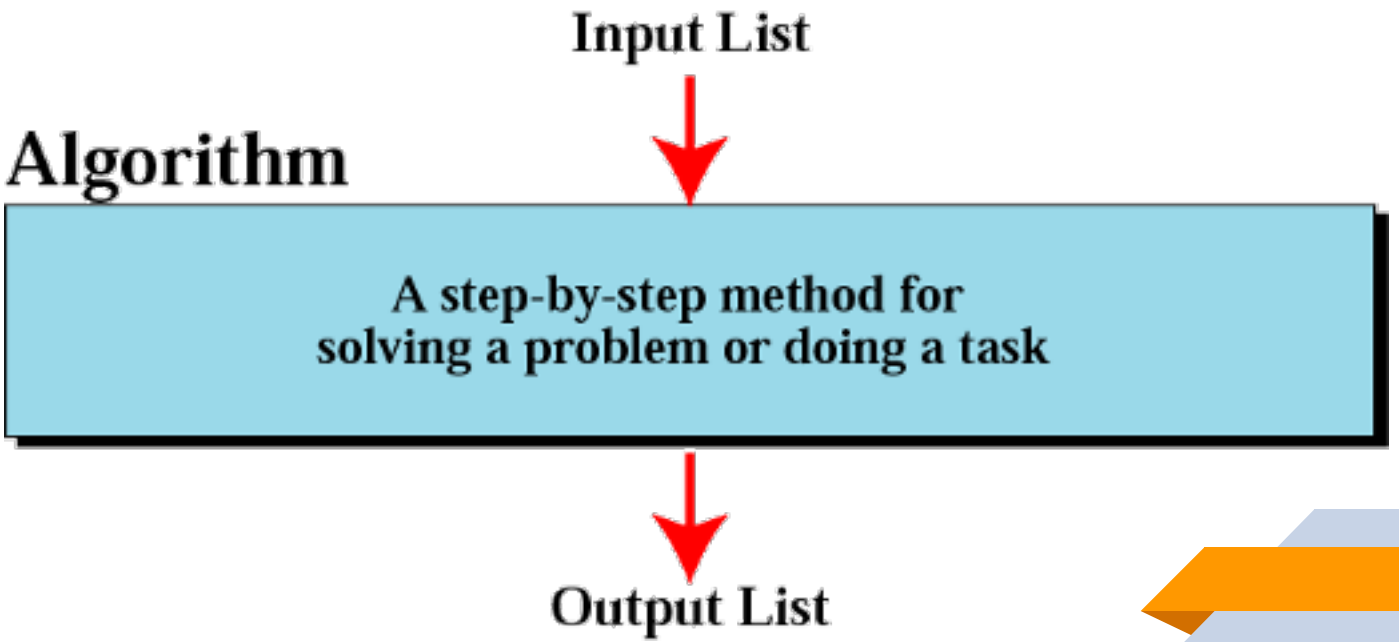
Lecture-1:
Introduction

Contents

Algorithm

Definition

An algorithm is "A finite set of precise instructions for performing a computation or for solving a problem".



Algorithm Specification

We can depict an algorithm in many ways.

- **Natural language:** implement a natural language like English
- **Flow charts:** Graphic representations denoted flowcharts
- **Pseudo code:** No particularity regarding syntax programming language.

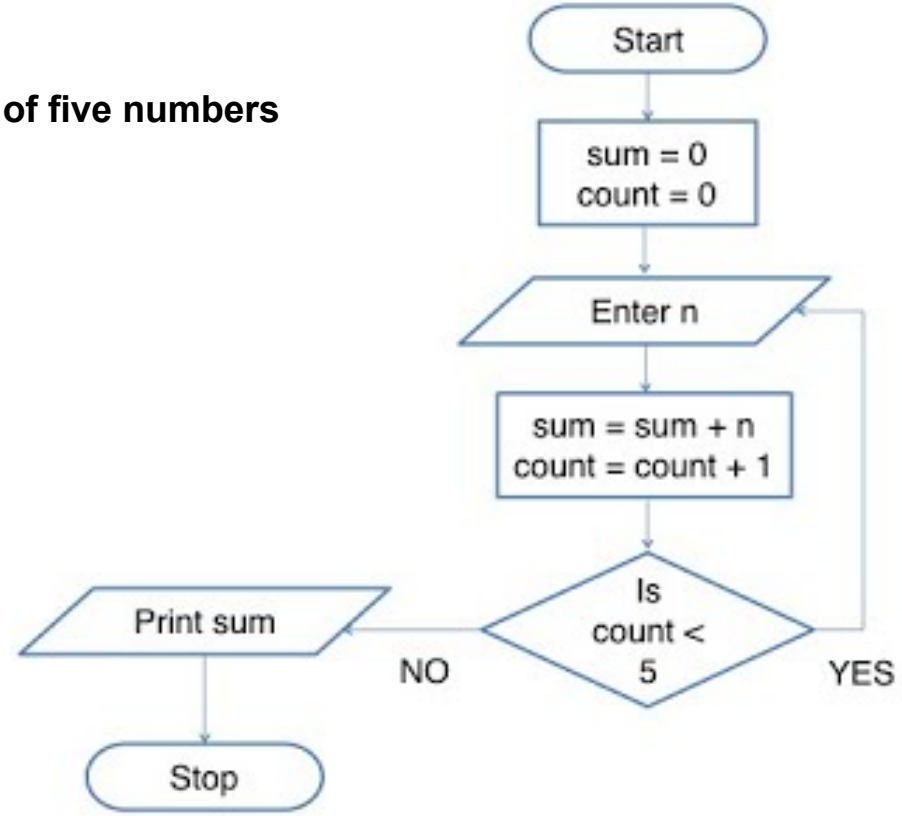
Algorithm Specification

Example: Algorithm for calculating factorial value of a number

```
Step 1: a number n is inputted
Step 2: variable final is set as 1
Step 3: final= final * n
Step 4: decrease n by 1
Step 5: verify if n is equal to 0
Step 6: if n is equal to zero, goto step 8 (break out of loop)
Step 7: else goto step 3
Step 8: the result final is printed
```

Flow charts

Find the sum of five numbers



Algorithm Specification

Pseudocode

```
Algorithm Max(A, n)
        // A is an array of size n.
      {
        Result:=A[1];
        for i :=2 to n do
              if A[i] >Result then Result:=A[i];
        return Result;
      }
```

Algorithm Characteristics

All algorithms must satisfy the following criteria:

Input:
There are zero or more quantities that are externally supplied.

Output:
At least one quantity is produced.

Definiteness:
Each instruction is clear and unambiguous.

Finiteness:
If we trace out the instructions of an algorithm, then for all cases, the algorithm terminates after a finite number of steps.

Effectiveness:
Every instruction must be basic enough to be carried out.

Steps to design an algorithm

- Problem definition
- Constraint to be satisfied
- Approach to design algorithm
  - Divide and conquer
  - Greedy
  - Dynamic
  - Backtracking
- Draw flow chart
- Verification
- Implementation
- Analysis

Analysing algorithms

Analysing an algorithm mean predicting the resources that the algorithm requires.

Resources such as memory, communication bandwidth, or computer hardware are of primary concern.

Most often it is computational time (time complexity) that we want to measure.

While analyzing the time complexity it is assumed that a constant amount of time is required to execute each line of the pseudocode and we ignore not only the actual cost of each statement but also the abstract cost say c_i

e.g. If running time is $an^2 + bn + c$

Analysis of algorithm

How many times a particular statement is executed while running.

```
main()
{
   a=b+c
   for (i=1;i≤n;i+1)
   {
      x=y+z;
    }
    for (i=1;i≤n;i+1)
     {
       for (j=1;j≤n;j+1)
           {
              p=q+r;
} } }
```

Find the time complexity

```
main()
{
   int n;
   while(n>1)
   {
    n=n/2
    }
}
```

J ⇒ C

F ⇒ D

D ⇒ E

C ⇒¥¥

16 ⇒ 1

8 ⇒ 2

4 ⇒ 3

2 ⇒ 4

1 ⇒??

Complexity=$O(\log_2 n)$

Find the time complexity

Problem 1

```
main()
{
        for (i = 1; i
      ≤ 133; i++)
   {
          a = a + b
    }
}
```

Problem 2

```
main()
{
 for  (i=1; i≤n; i++)
   {
       for  (j=1; j≤n; j++)
       {
          for
(k=1; k≤133; k++)
          {
          x=y+a
          }
      }
   }
}
```

Problem 3

```
main()
{
 for  (i=1; i≤n; i++)
   {

for(j=1; j≤i^2; j++)
      {
for(k=1; k≤133; k++)
         {
         x=y+a
         }
     }
   }
}
```