Disha College

Data Structure

BCA III

Subject Teacher

Seema Pathak

Unit 1

Objective

- Algorithm
- Properties of algorithm
- Analysis of algorithm
- Complexity of algorithm
- Types of complexity

Algorithm

- A well-defined computational procedure that accepts input and produces output.
- It can also be defined as sequence of computational steps to solve a particular problem.

An algorithm can be expressed in three ways:-

a) in any natural language such as English, called pseudo code.

b) in a programming language

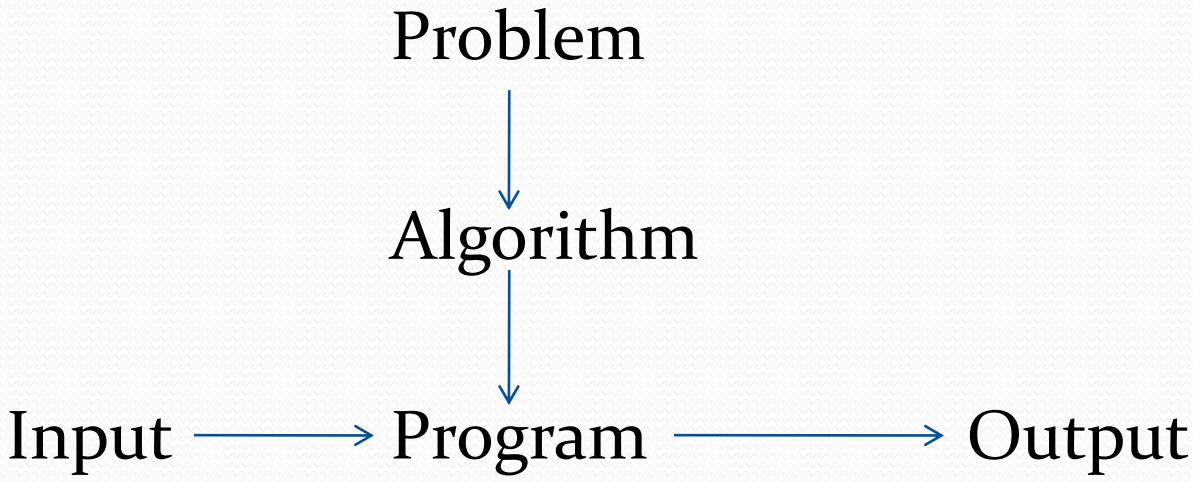
c) in the form of a flowchart.

Properties of an Algorithm

- Finiteness:- An algorithm must terminate after finite number of steps.
- Definiteness:-The steps of the algorithm must be precisely defined.
- Generality:- An algorithm must be generic enough to solve all problems of a particular class.
- Effectiveness:- The operations of the algorithm must be basic enough to be put down on pencil and paper.
- Correctness:- An algorithm must work correctly for all possible input

Algorithm

Once the algorithm is designed, checked for its correctness, and proved that it is efficient, the next step is to write a computer program.



Analysis of algorithm

Efficiency of the algorithm is the property which relate to the amount of resources (space and time) used by the algorithm.

The analysis of algorithm is a major task in computer science. To compare algorithms we must have some criteria to measure the efficiency of our algorithm.

Analysis of algorithm

There are several issues that may influence the efficiency:

- Input size
- Type of Computer used
- Method used to solve the problem

Analysis of algorithm

- Two main measures for the efficiency of an algorithm are the time and space used by the algorithm.
- Time and space both are directly dependent on the input size n i.e. the number of data items used by the algorithm.
- So calculation of efficiency of algorithm is performed by using input sixe n as key factor.

Complexity of algorithm

The complexity of an algorithm M is the function f(n) which gives the running and/or storage space requirement of the algorithm in terms of the size n of the input data.

The storage space required by an algorithm is simply a multiple of the data size n. Accordingly , unless otherwise stated or implied, the term complexity shall refer to the running time of an algorithm.

Types of complexity

There are two types of complexity

- **Time Complexity**
- **Space Complexity**

Time Complexity

The Time Complexity of an algorithm is a function of the running time of the algorithm. It gives the information about how much time is needed by the algorithm to complete its execution.

The time is measured by counting the number of key operations because the key operations are so defined that the time for other operations is much less than or at most proportional to the time for the key operations. In case of sorting and searching algorithms comparison is the key operation.

Space complexity

- The Space Complexity of an algorithm is a function of the space needed by the algorithm to run to completion.
- The space required by an algorithm is simply a multiple of the data size n, so time complexity is taken into consideration all most of the times.
- Exact memory requirement for a program is to be known in advance as without sufficient memory either the program work slowly or may not work totally.
- When we design a program we must ensure that it should take minimum memory.

Different cases for analysis

In complexity theory there are different cases for analysis of algorithm

- worst case
- Average case
- Best case

Worst Case

- The worse case occurs when the value of f(n) is maximum for any possible input.
- It provides an upper bound on running time.
- It gives an absolute guarantee that the algorithm would not run longer, no matter what the input is.

Average Case

The average case gives the expected value of f(n).

It provides a prediction about the running time.

The analysis of average case assumes that all possible cases are equally likely to occur.

Average case analysis uses concept of probability theory.

Suppose the number n_1, n_2, ……n_k occur with respective probabilities p1, p2,…..pk then average value (E) is

$E = n_1p_1 + n_2p_2 + \ldots\ldots\ldots\ldots\ldots n_kp_k$

Best Case

- Best case gives minimum possible value of f(n)
- In best case algorithm shows ideal performance, ie. It takes minimum time for execution.

Algorithm for linear search

```
#include<stdio.h>

int main()
{
   int a[20],i,x,n;
   printf("How many elements?");
   scanf("%d",&n);

   printf("Enter array elements:n");
   for(i=0;i<n;++i)
      scanf("%d",&a[i]);

   printf("nEnter element to search:");
   scanf("%d",&x);

   for(i=0;i<n;++i)
      if(a[i]==x)
         break;

   if(i<n)
      printf("Element found at index %d",i);
   else
      printf("Element not found");

   return 0;
}
```

Complexity of linear search

Worst case:- occurs when element not found in the list.

No of comparisons = C(n) = n

where n is the input size ie. No of elements in the list

Complexity of linear search

- Average case:- occurs when element may appear at any location with equal probability which is 1/n
- Total no of comparisons =
- $C(n) = 1 * 1/n + 2 * 1/n + 3 * 1/n + ... n * 1/n$

$= 1/n(1 + 2 + 3 + 4 + ...n)$

$= 1/n (n * (n+1))/2$

$= (n + 1)/2$

$= n/2$ when n is a large

It means on average half of the elements is to be compared to search a single element.

Complexity of linear search

Best Case:- occurs when element found at first location.

No of comparisons

$C(n) = 1$

So best case complexity of linear search is 1.