Design and Analysis of Algorithm (KCS503)

Getting Started

Lecture - 1

Overview

- Start using frameworks for describing and analysing algorithms.
- Examine algorithms :
    - simple algorithms
    - Sorting algorithm insertion sort
- See how to describe algorithms in pseudo code.
- Begin using asymptotic notation to express running-time analysis.

What is an Algorithm ?

- 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.
- We can also view an algorithm as a tool for solving a well specified computational problem.

Characteristics of an Algorithm

- Input: provided by the user.
- Output: produced by algorithm.
- Definiteness: clearly define.
- Finiteness: It has finite number of steps.
- Effectiveness: An algorithm must be effective so that it’s output can be carried out with the help of paper and pen.

Analysis of an Algorithm

- Done in three steps:
  - Initialization
  - Maintenance
  - Termination
- It deals with predicting the resources that an algorithm requires to its completion such as memory and CPU time.
- To main measure for the efficiency of an algorithm are Time and space.

Complexity of an Algorithm

- Complexity of an Algorithm is a function, f(n) which gives the running time and storage space requirement of the algorithm in terms of size n of the input data.
- Space Complexity: Amount of memory needed by an algorithm to run its completion.
- Time complexity: Amount of time that it needs to complete itself.

Cases in Complexity Theory

- Best Case: Minimum time
- Worst Case: Maximum amount of time
- Average Case: Expected / Average value of the function f(n).

Example 1

```
 A()
{
  int i;
    for(i=1;i<=n;i++)
       {
                   printf(“ABCD”);
        }
}
```

Complexity of Example 1

- $T = O(n)$

Example 2

```
A()
{
    int i=1 ,s=1;
    scanf("%d", &n);
    while(s<=n)
       {
           i++;
           s=s+i;
           printf("abcd");
          }
}
```

Complexity of Example 2

- $T(n) = O(\sqrt{n})$

Example 3

```
A()
{
    int i=1;
       for(i=1; i pow 2<=n; i++)
       {
            printf(“abcd”);
          }
}
```

Complexity of Example 3

- $T(n) = 0(\sqrt{n})$

Example 4

```
A()
{
   int i = 1;
     for(i = 1; i ≤ n;  i + +)
   {
      for(j = 1; j ≤ i^2;  j + +)
      {
          for(k = 1; k ≤ n/2; k + +)
          {
             printf(“abcd”);
          }
      }
   }
}
```

Complexity of Example 4

$T(n) = O(n^4)$

$i = 1\ 2\ 3\ 4\ 5\ \dots\dots\dots..$

$J = 1\ 4\ 9\ 16\ \dots\dots\dots\dots\dots..$

$K = \frac{n}{2}, \frac{4n}{2}, \frac{9n}{2}, \dots\dots\dots\dots..$

$sum\ of\ squares\ of\ natural\ numbers..$

$[n(n+1)(2n+1)]/6$