Xtreme track

IEEE CS GECB

Complexity Analysis

The **time complexity** is the computational complexity that describes the amount of **time** it takes to run an algorithm.

Space Complexity of an algorithm is total space taken by the algorithm with respect to the input size. Space complexity includes both Auxiliary space (temporary space) and space used by input.

Asymptotic Notation

Asymptotic Notations are languages that allow us to analyze an algorithm’s running time by identifying its behavior as the input size for the algorithm increases. This is also known as an algorithm’s growth rate.

Asymptotic Notation

1. Big O	- Worst Case or Upper bound
2. Big Omega	- Best Case or Lower bound
3. Big Theta	- Average case or Tight bound

```
a[] ={ 1,2,3,4,5,6,7,8,9,10}
int k;
cin>>k;                          //Between 1 and 10, both inclusive
for( int i=0; i<10; i++)
{
        if( a[i] == k)
        {
                cout<<" Found ! "
                break;
        }
}
```

```
a[n][n]                                    //matrix of size n x n
int k;
cin>>k;
for( int i=0; i<n; i++)
{
        for( int j=0; j<n; j++)
        {
                if( a[ i ][ j ] == k)
                {
                        cout<< ” Found ! ”
                        break;
                }
        }
}
```

Algorithms

You can think of a **programming algorithm** as a recipe that describes the exact steps needed for the computer to solve a problem or reach a goal.

```
Step 1: START
Step 2: Set n as 5
Step 3: If n greater than or equal to 0 goto Step 5
Step 4: Else goto step 8
Step 5: Print Hello World !
Step 6: n=n-1
Step 7: goto step 3
Step 8: STOP
```