Understanding Time Complexity

Complexity of Algorithms

- **The complexity of an algorithm is the function which gives the running time and/or space in terms of the input size.**
- **In order to compare algorithms, we must have some criteria to measure the efficiency of a algorithm.**
- **Suppose *M* is an algorithm, and suppose *n* is the size of the input data.**
- **The *TIME* and *SPACE* used by the algorithm M are the two main measures for the efficiency of *M.***
- **The TIME is measured by counting the number of key operation-in sorting and searching algorithms. (the # of comparison)**
- **The SPACE is measured by counting the maximum of memory needed by the algorithm**

“IN Computer Science, time complexity describes the amount of time that takes to run an algorithm or Code.”

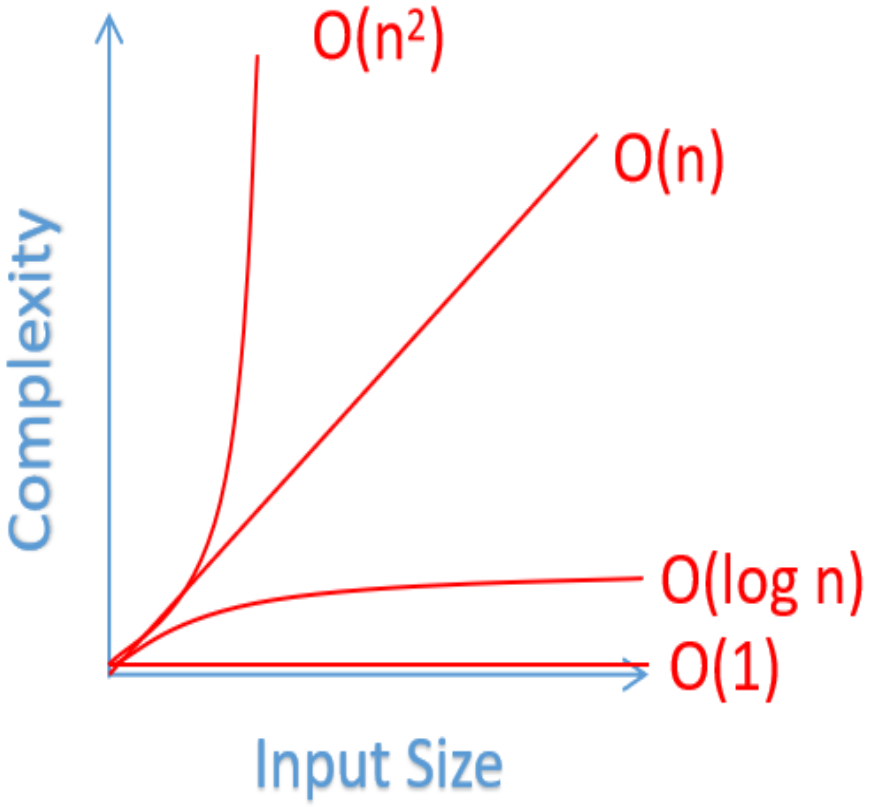
How to calculate Time Complexity ?

“Time complexity is calculated by counting the Number of Operation , which takes a fixed amount of time to perform.”

Big O Notation

“Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.”

- O(1) Constant Time complexity
- O(N) Linear Time Complexity
- O(N^2) Quadratic Time Complexity
- O(N log N) Logarithmic Time Complexity



- Input (in single operation)
- Output (in single operation)

```
#include <iostream>
using namespace std;


int main() {

int a,b;
cin >> a >> b;
 //O(1+1)=O(1) constant operation.
int sum = a+b;
 cout<< sum <<endl;
 return 0;
}
```

Linear Time Complexity - O(N)

- **While doing operation with *Single* for or while loop.**

- **Total Complexity(O(N))**
- **In 1 Sec 1e7 Or maximum 1e8 Iteration can be done.**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    cin >> t;
        int n;
       cin >> n;
       int arr[n];
       for(int i=0;i<n;i++){
        cin >> arr[i];
       }      // O(N) *here N is Number of
element
       sort(arr,arr+n);
       for(int i=0;i<n;i++){
        cout << i << " ";
       }     // O(N) *here N is Number of
element
       cout << endl;
}
```

Quadratic Time Complexity – O(N^2)

- **“If Two nested loop used and iterate through all the elements. Then complexity is O(n^2).”**
- **Explanation : iteration for the inner loop : (n-1)*(n-2)……*2*1.**
- **So the result is n*(n-1) which is O(n^2).**

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int n;
   cin >> n;
   int arr[n];
   for(int i=0;i<n;i++) cin >> arr[i];
   //Bubble sort
   for( int i = 0; i < n; i++ )
  {
       for(int j = 0; j < (n-i-1); j++) {
          if(arr[j] > arr[j+1]){
             swap( arr[j] , arr[j+1] );
          }
       }   }
   for(auto u : arr){
      cout << u << " ";
   }
   cout << endl;
}
```

Logarithmic Time Complexity – O(log N)

Per Operation mid divides into half Value. That means

n/2 + n/4 + n/8 + n/16 + ……..

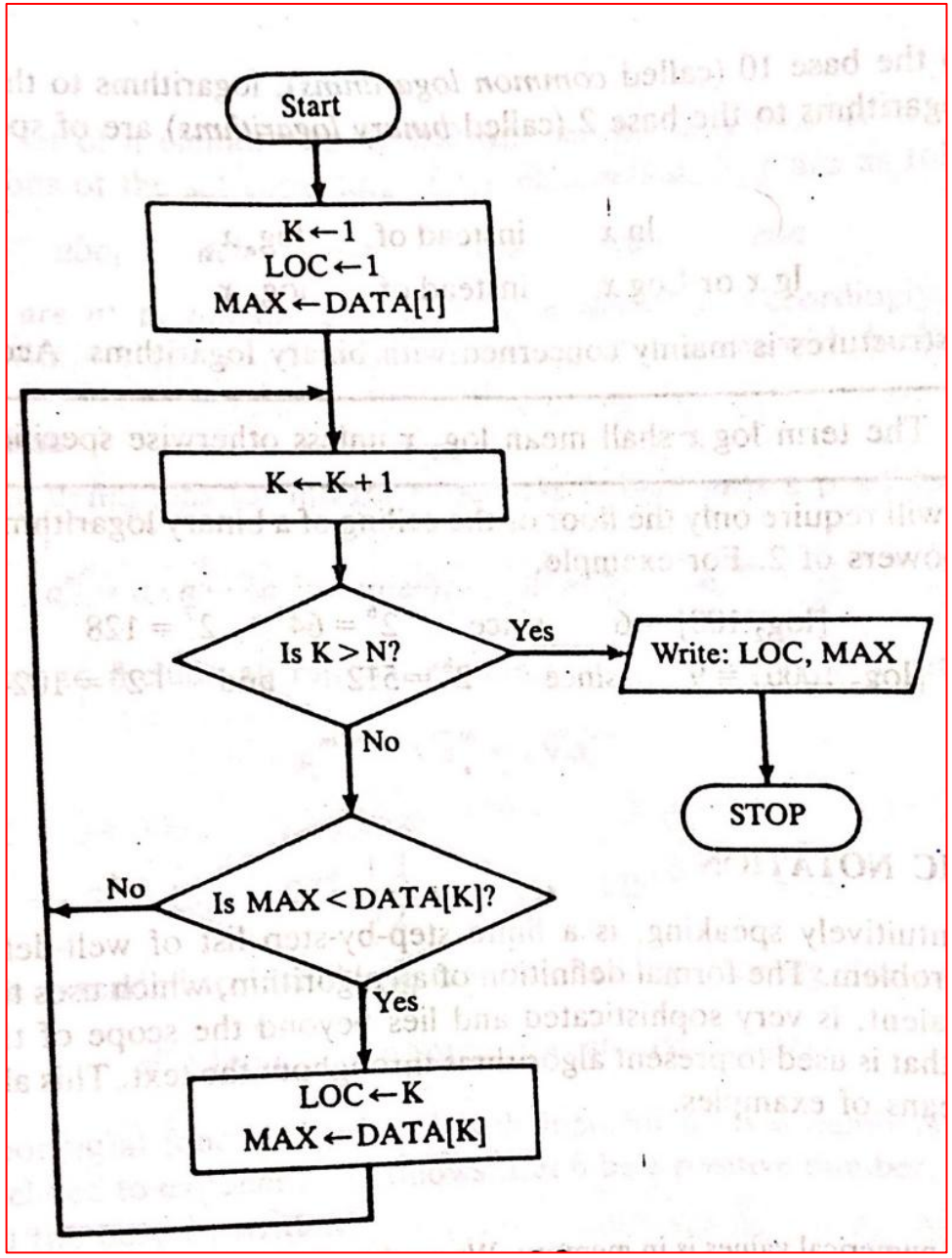
= n (1/2 + 1/4 + 1/8 + ……..)

= log(n)) // $log_2(y) = x$, $2^x = y$;

So, Time Complexity of Binary search is O(log(n))

```
int main()
{  // Binary Search
   int n , v[n]; cin >> n;
   for(int i=0; i<n; i++){
   cin >> v[i];
   }
   int lo=0,hi=n-1,mid,fnd;
   cin >> fnd;
   while(hi>=lo){
      mid=(hi+lo)/2;
      if(v[mid] == fnd) {
         cout << mid << endl;
         break;
      }
      if(v[mid] <= fnd){
      lo=mid+1;       }
      else
      {
         hi = mid-1;
      }   }}
```

Flowcharts


Start
K←1
LOC←1
MAX←DATA[1]
K←K+1
Is K>N?
Yes
Write: LOC, MAX
No
STOP
No
Is MAX<DATA[K]?
Yes
LOC←K
MAX←DATA[K]