**PART A**

**Linear search**

**Implementation of linear search in C++ given bellow:**

#include<iostream>

using namespace std;

int linearsearch(int arr[],int n,int x)

{

int i,index,flag=0;

for(i=0; i<n; i++)

{

if(arr[i]==x)

{

index = i;

flag = 1;

break;

}

}

if(flag==1){

return index;

}

else{

return -1;

}

}

int main()

{

int n,res,x,i;

cin >> n;

int arr[n];

for(i=0; i<n; i++)

{

cin >> arr[i];

}

cin >> x; //element x that want to search

res = linearsearch(arr,n,x);

if(res==-1)

{

cout<<"Value not found in the list"<<endl;

}

else

{

cout << "Value found....\nThe position of the value is "<<res <<endl;

}

}

**Time Complexity Analysis of linear search:**

Best Case:

In the best possible case,

The element being searched may be found at the first position.

In this case, the search terminates in success with just one comparison.

Thus in best case, linear search algorithm takes O(1) operations.

Worst Case:

In the worst possible case,

The element being searched may be present at the last position or not present in the array at all.

In the former case, the search terminates in success with n comparisons.

In the next case, the search terminates in failure with n comparisons.

Thus in worst case, linear search algorithm takes O(n) operations.

Thus, we have-

Time Complexity of Linear Search Algorithm is O(n).

Here, n is the number of elements in the linear array.

**Binary search**

**Implementation of Binary search in C++ given bellow:**

#include<iostream>

using namespace std;

int binarysearch(int arr[],int n,int x)

{

int i,left,right,mid;

left = 0;

right = n-1;

while(left<right)

{

mid = (left+right)/2;

if(x==arr[mid])

{

return mid;

}

else if(x<arr[mid])

{

right = mid-1;

}

else

{

left = mid+1;

}

}

return -1;

}

int main()

{

int n,i,res,x;

cin>> n; //how many values in the list?

int arr[n];

for(i=0; i<n; i++)

{

cin >> arr[i];

}

cin >> x; //value that want to search

res = binarysearch(arr,n,x);

if(res==-1)

{

cout<<"Value not found in the list"<<endl;

}

else

{

cout<<"Value found\nThe position of the value is "<<res<<endl;

}

}

**Time Complexity Analysis of binary search:**

Binary Search works on a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, shift the interval to the lower half. Otherwise, shift it to the upper half. Repeatedly checking until the value is found or the interval is empty.

Binary Search Algorithm searches an element by comparing it with the middle most element of the array. Then, following three cases are possible-

**Case 01:** If the element being searched is found to be the middle most element, its index is returned.

**Case 02:** If the element being searched is found to be greater than the middle most element, then its search is further continued in the right sub array of the middle most element.

**Case 03:** If the element being searched is found to be smaller than the middle most element, then its search is further continued in the left sub array of the middle most element.

This procedure keeps on repeating on the sub arrays until the desired element is found or size of the sub array reduces to zero.

In each iteration or in each recursive call, the search gets reduced to half of the array. So for n elements in the array, there are iterations or recursive calls.

Let consider an array of {10,20,30,40} 4 elements.

Using log base 2 of 8, (= 2 ) means that dividing 2 times the list.

Thus, we have-

Time Complexity of Binary Search Algorithm is O().

Here, n is the number of elements in the sorted linear array.

This time complexity of binary search remains unchanged irrespective of the element position even if it is not present in the array.

In base case situation, if we get the value in the middle then the complexity becomes **O (1)** and in worst case it becomes **O (log n).**

**It is efficient than linear search while time complexity of linear search is O(n) but time complexity of Binary search is O (log n).**