|  |  |
| --- | --- |
| Assignment No. : | 10 |
| Title: | Write a modular program using object oriented programming features to implement different searching methods. (Binary Search, Sequential  Search |
| Subject: | Data Structures Laboratory |
| Class: | S.Y. (C.S.E.) |
| Roll No.: |  |
| Assessment (Marks): |  |
| Signature and Date of Assessment: |  |

Experiment No.: 10

|  |  |
| --- | --- |
| Title: | Write a modular program using object oriented programming features to implement different searching methods. (Binary Search, Sequential Search). |
| Objectives: | 1. To understand the concept of binary search. 2. To understand the concept of sequentially sort. |
| Problem | Write a modular program using object oriented programming features to implement different |
| Statement : | searching methods. (Binary Search, Sequential Search) |
| Outcomes: | Understanding the concept of binary search.  Understanding the concept sequential search. |

 Understanding the concept and use of bubble sort.

Binary search:

Binary search is the search technique which works efficiently on the sorted lists. Hence, in orderto search an element into some list by using binary search technique, we must ensure that the list is sorted.Binary searchfollows divide and conquer approach in which, the list is divkled into twohalves and the item is compared with the middle element of the list. If the match is found then, the location of middle element is returned otherwise, we search into either of the halves depending upon the result produced through the match.

Binary search algorithm is given below.

lower\_bound, upper\_bound, VAL) step 1 : [INITIALIZE] SET BEG = lower\_bound

END = upper\_bound, POS = - I step 2: Repeat steps 3 and 4 while BEG CEND step 3: SET MID = (BEG + END)t2 step 4: IF AIMIDI = VAL

SET POS = MID

PRINT POS

Go to Step 6

ELSE IF A[MIDI > VAL

SET END = MID - 1

ELSE

SET BEG = MID + 1

[END OF IF]

[END OF LOOPI

step 5: IF POS = -1

PRINT "VALUE IS NOT PRESENT IN THE ARRAY"

[END OF IF] step 6: EXIT

Example

Let us consider an array arr = { l, 5, 7, 8, 13, 19, 20, 23, 29). Find the location ofthe item 23 in the array.

In 1st step :

BEG = O

END = 8ron

MID = 4 a[mid] = a141 = 13 < 23, therefore in Second step:

Beg = mid +1 = 5

End = 8 mid = 13/2 = 6 a[midl = a161 = 20 < 23, therefore; in third step:

beg= mid + I = 7 End = 8



a[mid] = a[71 al7] = 23 = item;

252

therefore, set location = mid;

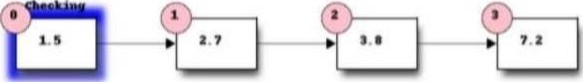
The location of the item will be 7.

Binary Search Program using Recursion

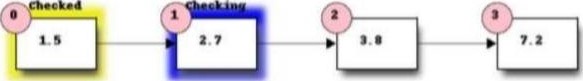
Linear search :

Linear search is the most basic kind of search methcxi. It involves checking each element of the list in turn, until the desired element is found.

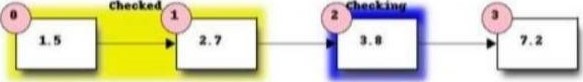
For example, suppose that we want to find the number 3.8 in the following list:



We start with the first element, and perform a comparison to see if its value is the value that we want. In this case, 1.5 is not equal to 3.8, so we move onto the next element:



We perform another comparison, and see that 2.7 is also not equal to 3.8, so we move onto the next element:



We perform another comparison and determine that we have found the correct element Now we can end the search and return the position of the element (index 2). We had to use a total of 3 comparisons when searching through this list of 4 elements. How many comparisons we need to perform depends on the total length of the list, but also whether the element we are looking for is near the beginning or near the end of the list. In the worst-case scenario, if our element is the last element of the list, we will have to search through the entire list to find it. If we search the same list many times, assuming that all elements are equally likely to be searched for, we will on average have to search through half of the list each time. The cost (in comparisons) of performing linear search thus scales linearly with the length of the list

Consider-

* There is a linear array 'a' of size
* Linear search algorithm is being used to search an element 'item' in this linear array.
* If search ends in success, it sets loc to the index of the element otherwise it sets loc to -l .
* Algorithm for Linear Search.

Begin fori = O to (n- l)by I do if (alil = item) then set loc = i Exit endif

endfor set loc = -l

Questions: l) What is searching?

* 1. Write algorithm for linear search
  2. Write algorithm for Binary Search

Conclusion: Thus, we have learned the logic of linear search

Both linear and binary search algorithms can be useful depending on the application. While a linear search is repetitive or iterative in nature as well as uses the sequential approach, the binary search goes about its task by using the divide and conquer strategy.

Code:-Linear Search

#include<iostream>

using namespace std;

class Search{

    int i,Arr[10],n,flag=0,key;

public: void Linearsearch(){

    cout<<"ENTER THE NUMBER OF ELEMENTS: "<<endl;

    cin>>n;

    cout<<"ENTER YOUR ELEMENTS: "<<endl;

    for(i=0;i<n;i++){

        cin>>Arr[i];

    }cout<<"ENTER THE ELEMENT TO BE SEARCHED: "<<endl;

    cin>>key;

    for(i=0;i<n;i++){

        if(Arr[i]==key){

            cout<<"THE ELEMENT IS PRESENT AT LOCATION: "<<i+1<<endl;

            flag=1;

            break;

        }

    }if(flag==0){

        cout<<"ELEMENT NOT PRESENT. "<<endl;

    }

}

};

int main(){

Search s;

s.Linearsearch();

}

Output:-

ENTER THE NUMBER OF ELEMENTS:

3

ENTER YOUR ELEMENTS:

12

34

56

ENTER THE ELEMENT TO BE SEARCHED:

56

THE ELEMENT IS PRESENT AT LOCATION: 3

Code:-Binary Search

#include<iostream>

using namespace std;

class Search{

    int a[100],low,mid,high,n,i,key;

public: void BinarySearch(){

    cout<<"ENTER THE NUMBER OF ELEMENTS: "<<endl;

    cin>>n;

    cout<<"ENTER THE ELEMENTS: "<<endl;

    for(i=0;i<n;i++){

        cin>>a[i];

    }

   low=0;

   high=n-1;

   cout<<"ENTER THE ELEMENT YOU WANT TO SEARCH: "<<endl;

   cin>>key;

   while(low<=high){

    mid=(low+high)/2;

    if(a[mid]==key){

        cout<<"THE ELEMENT IS FOUND AT LOCATION: "<<mid+1<<endl;

        exit(0);

    }else if(a[mid]<key){

        low=mid+1;

    }else if(a[mid]>key){

        high=mid-1;

    }

   }cout<<"ELEMENT NOT FOUND "<<endl;

}

};

int main(){

Search s;

s.BinarySearch();

}

Output:-

ENTER THE NUMBER OF ELEMENTS:

3

ENTER YOUR ELEMENTS:

12

34

56

ENTER THE ELEMENT TO BE SEARCHED:

56