GROUP-2

L2- Samuel, L3- Satyajit, L4- Vinay, L5- Sunil,

L6- Farheen, L7- Rehman, L8- Alisha, L9- Suhail.

AIM: To write a program and implement the following

a)Linear Search b)Binary Search c)Fibonacci Search

LINEAR SEARCH

Linear search or sequential search is a method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched.

AIGORITHM:

Step 1 - Read the search element from the user.

Step 2 - Compare the search element with the first element in the list.

Step 3 - If both are matched, then display "Given element is found at position: " and terminate the function

Step 4 - If both are not matched, then compare search element with the next element in the list.

Step 5 - Repeat steps 3 and 4 until search element is compared with last element in the list.

Step 6 - If last element in the list also doesn't match, then display "Element is not found" and terminate the function.

PROGRAM:

#include<iostream>

using namespace std;

int main()

{

int arr\_search[50],i,n,element;

cout<<"\n enter size of array:";

cin>>n;

cout<<"enter the elements:";

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

cin>>arr\_search[i];

cout<<"\n your data:";

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

{

cout<<"\t"<<arr\_search[i];

}

cout<<"\nenter element to search:";

cin>>element;

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

{

if(arr\_search[i]==element)

{

cout<<"element found at : position "<<i+1;

break;

}

}

if(i==n)

cout<<"element"<<element<<"not found";

}

Sample Input & output:

enter the size of array:6

enter the elements:

1

6

8

9

2

7

Your data:

1. 6 8 9 2 7

Enter element to search:6

Element found at: position 2

BINARY SEARCH

Binary search is used for finding an item from a sorted list of items.It works by repeatedly dividing in half the portion of list that could contain the item, upto the location one.

ALGORITHM:

Step 1 - Read the search element from the user.

Step 2 - Find the middle element in the sorted list.

Step 3 - Compare the search element with the middle element in the sorted list.

Step 4 - If both are matched, then display "Given element is found at position" and terminate the function.

Step 5 - If both are not matched, then check whether the search element is smaller or larger than the middle element.

Step 6 - If the search element is smaller than middle element, repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.

Step 7 - If the search element is larger than middle element, repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.

Step 8 - Repeat the same process until we find the search element in the list or until sublist contains only one element.

Step 9 -If the element is not found then display “element not found” and terminate the function.

PROGRAM:

#include <iostream>

#include<conio.h>

#include<stdlib.h>

#define MAX\_SIZE 5

using namespace std;

int main() {

int arr\_search[MAX\_SIZE], i, element;

int low = 0, high = MAX\_SIZE, mid;

cout << "\nEnter " << MAX\_SIZE << " Elements for Searching : " << endl;

for (i = 0; i < MAX\_SIZE; i++)

cin >> arr\_search[i];

cout << "\nYour Data :";

for (i = 0; i < MAX\_SIZE; i++) {

cout << "\t" << arr\_search[i];

}

cout << "\ nElement to Search : ";

cin>>element;

while (low <= high) {

mid = (low + high) / 2;

if (arr\_search[mid] == element) {

cout << "\nSearch Element : " << element << " : Found : Position : " << mid + 1 << ".\n";

break;

} else if (arr\_search[mid] < element)

low = mid + 1;

else

high = mid - 1;

}

if (low > high)

cout << "\nSearch Element : " << element << " : Not Found \n";

getch();

}

Sample input & output:

Enter 5 elements for searching:

1

4

5

8

9

Your data:

1. 4 5 8 9

Enter element to search:8

Search element: 8 : Found : Position: 4

FIBONACCI SEARCH

Fibonacci search uses Fibonacci numbers to search an element in a sorted array.

FIBONACCI NUMBERS:

The first two numbers are 0 and 1 and each subsequent numbers in the series is equal to the sum of previous two numbers.

Fibonacci numbers are 0,1,1,2,3,5,8…………

ALGORITHM:

STEP 1:Find the smallest number>=n.let the number be fibm, let the two Fibonacci numbers preceeding it be m1,m2

STEP 2:While the array has elements

Compare x with last element of the range covered by m2

1.Else if x is less than the element move the 3 fibonacci variables two fibonacci down, indicating the elimination of 2/3rd of remaining array.

2. Else x is greater than the element move the 3 fibonacci variables one fibonacci down, indicating the elimination of 1/3rd of remaining array. Reset offset to index.

STEP 3:Since there might be single element remaining for comparision. check if m1 is 1.If yes, compare x with that remaining element. If match return index

PROGRAM:

#include<iostream>

using namespace std;

int main(){

int arr[100],n,fibm,m1,m2,i,l;

cout<<"enter size of array";

cin>>n;

cout<<"enter elements";

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

cin>>arr[i];

int x;

cout<<"enter search element";

cin>>x;

m1=1;

m2=0;

fibm=m1+m2;

while(fibm<n){

m2=m1;

m1=fibm;

fibm=m1+m2;

}

int offset=0;

int flag=0;

int co=0;

while(co<=n+1){

l=offset+m2;

if(n>l)

i=l;

else

{

i=n;

}

if(arr[i]==x)

{

cout<<"element found at "<<i;

flag=1;

break;

co+=1;

}

else if(arr[i]>x)

{

fibm=m2;

m1=m1-fibm;

m2=fibm-m1;

co+=1;

}

else

{

fibm=m1;

m1=m2;

m2=fibm-m1;

offset=i;

co+=1;

}

}

if(arr[0]==x)

{

cout<<"found at position 0";

flag=1;

}

if(flag==0)

{

cout<<"element not found";

}

}

Sample input & output:

Enter size of array: 5

Enter elements:

1

4

6

8

9

Enter search element:4

Element found at position 2