**15B17CI371 – Data Structures Lab**

**ODD 2024**

**Week 5-LAB A**

**Practice Lab**

1.

#include<iostream>

using namespace std;

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

{ int c=0;

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

{

if(arr[i]==key)

{

cout<<key<<" belongs to the array.";

c++;

return c;

}

}

return c;

}

int main(){

int n,a;

cout<<"Enter the elements you want in the array:";

cin>>n;

int arr[n];

cout<<"Enter the elements:";

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

{

cin>>a;

arr[i]=a;

}

int key;

cout<<"Enter the key you want to search:";

cin>>key;

int x=linearsearch(arr,n,key);

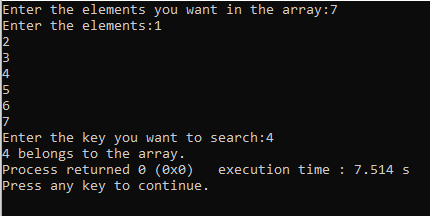
if(x==0)

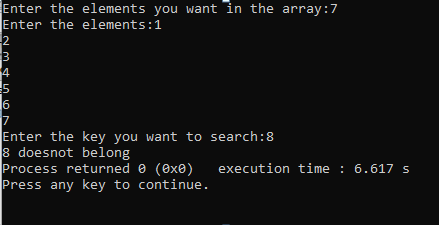
{

cout<<key<<" doesnot belong";

}

}





Time complexity:O(n);

Best case :first element O(1)

Worst case:last element O(n)

2.

#include<iostream>

using namespace std;

int binarysearch(int arr[],int r, int key,int l)

{

while(l<=r)

{int c=(l+r)/2;

if(arr[c]==key)

{

return c;

}

else if(arr[c]>key)

{

r=c-1;

}

else

{

l=c+1;

}

}

return -1;

}

int main(){

int n,a;

cout<<"Enter the elements you want in the array:";

cin>>n;

int arr[n];

cout<<"Enter the elements in sorted order:";

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

{

cin>>a;

arr[i]=a;

}

int key;

cout<<"Enter the key you want to search:";

cin>>key;

int x=binarysearch(arr,n-1,key,0);

if(x==-1)

{

cout<<key<<" doesnot belong";

}

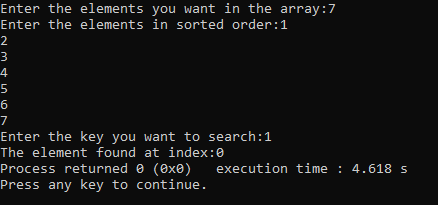
else

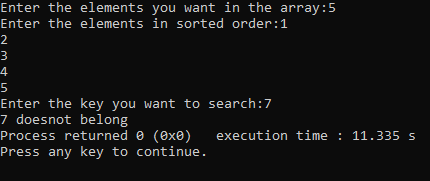
{

cout<<"The element found at index:"<<x;

}

}





Time complexity:O(logn)

3.

#include <iostream>

using namespace std;

void swap(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = low - 1;

for (int j = low; j < high; ++j) {

if (arr[j] < pivot) {

++i;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

return i + 1;

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void printarray( int arr[], int size) {

for (int i = 0; i < size; ++i) {

cout << arr[i] << " ";

}

cout << endl;

}

int smallest(int arr[],int k)

{

return arr[k-1];

}

int greatest(int arr[],int k,int n)

{

return arr[n-k];

}

int main() {

int size;

cout << "Enter the number of elements: ";

cin >> size;

int arr[size];

cout << "Enter the elements: ";

for (int i = 0; i < size; ++i) {

cin >> arr[i];

}

int key;

cout << "Enter which smallest and greatest element u want to search for: ";

cin >> key;

cout << "Original array: ";

printarray(arr, size);

quickSort(arr, 0, size - 1);

cout << "Sorted array: ";

printarray(arr, size);

int x=smallest(arr,key);

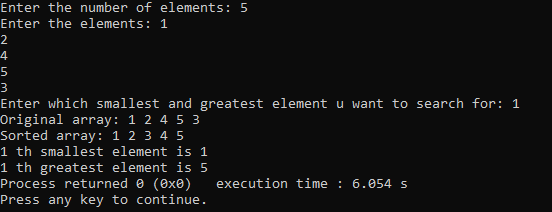
cout<<key<<" th smallest element is "<<x<<endl;

int y=greatest(arr,key,size);

cout<<key<<" th greatest element is "<<y;

return 0;

}



4.

#include<iostream>

using namespace std;

int interpolationsearch(int arr[],int r, int key,int l)

{ int c=l+(((key-arr[l])\*(r-l))/(arr[r]-arr[l]));

while(l<=r)

{

if(arr[c]==key)

{

return c;

}

else if(arr[c]>key)

{

r=c-1;

}

else

{

l=c+1;

}

}

return -1;

}

int main(){

int n,a;

cout<<"Enter the elements you want in the array:";

cin>>n;

int arr[n];

cout<<"Enter the elements in sorted order:";

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

{

cin>>a;

arr[i]=a;

}

int key;

cout<<"Enter the key you want to search:";

cin>>key;

int x=interpolationsearch(arr,n-1,key,0);

if(x==-1)

{

cout<<key<<" doesnot belong";

}

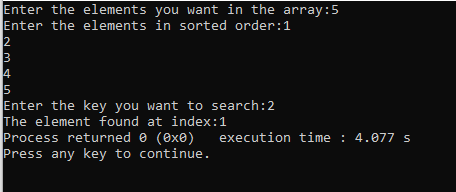
else

{

cout<<"The element found at index:"<<x;

}

}



5.

#include <iostream>

#include <string>

using namespace std;

int binarySearch(const string arr[], int size, const string& x) {

int left = 0;

int right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == x) {

return mid;

} else if (arr[mid] < x) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

int main() {

string arr1[] = {"Hi", "Folks", "ide", "for", "practice"};

int size1 = sizeof(arr1) / sizeof(arr1[0]);

string x1 = "ide";

int index1 = binarySearch(arr1, size1, x1);

cout << "Index of '" << x1 << "': " << index1 << endl;

string arr2[] = {"Hi", "Folks", "ide", "for", "practic"};

int size2 = sizeof(arr2) / sizeof(arr2[0]);

string x2 = "zz";

int index2 = binarySearch(arr2, size2, x2);

cout << "Index of '" << x2 << "': " << index2 << endl;

return 0;

}

