1.last and First Occurance

#include<iostream>

using namespace std;

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

{

int s=0;

int e=n-1;

int ans=-1;

while(s<=e)

{

int mid=(s+e)/2;

if(arr[mid]==key)

{

ans=mid;

e=mid-1;

}

else if(arr[mid]<key)

{

s=mid+1;

}

else

{

e=mid-1;

}

}

return ans;

}

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

{

int s=0;

int e=n-1;

int ans=-1;

while(s<=e)

{

int mid=(s+e)/2;

if(arr[mid]==key)

{

ans=mid;

s=mid+1;

}

else if(arr[mid]<key)

{

s=mid+1;

}

else

{

e=mid-1;

}

}

return ans;

}

int main()

{

int arr[]={1,2,5,8,8,8,8,8,10,12,15,20};

int n=sizeof(arr)/sizeof(int);

cout<<firstOccurance(arr,n,8)<<endl;

cout<<lastOccurance(arr,n,8);

}

Binary Search STL::::

#include<iostream>

#include<algorithm>

using namespace std;

int main(){

int arr[]={1,2,5,8,8,8,10,12,140};

int n=sizeof(arr)/sizeof(int);

bool present=binary\_search(arr,arr+n,150);

if(present){

cout<<"Present"<<endl;

}

else{

cout<<"Absent"<<endl;

}

}

//Search lower and Upper Bound

#include<iostream>

#include<algorithm>

using namespace std;

int main(){

int arr[]={1,2,5,8,8,8,10,12,140};

int n=sizeof(arr)/sizeof(int);

bool present=binary\_search(arr,arr+n,150);

if(present){

cout<<"Present"<<endl;

}

else{

cout<<"Absent"<<endl;

}

//Two more thing

//Get the Index of the element

//lower\_bound(s,e,key) and upper\_bound(s,e,key)

//lower bound

auto lb=lower\_bound(arr,arr+n,8);

cout<<"lower bound "<<(lb-arr)<<endl;

//upper bound

auto up=upper\_bound(arr,arr+n,8);

cout<<"upper bound "<<(up-arr)<<endl;

cout<<"Occurance frequency of 8 is "<<(up-lb)<<endl;

}

Helps Rahul

#include<iostream>

using namespace std;

int find\_key(int a[],int n,int key)

{

int s=0;

int e=n-1;

while(s<=e)

{

int mid=(s+e)/2;

if(a[mid]==key)

{

return mid;

}

else if(a[s]<=a[mid])

{

//2 cases -> element lies in the left or the right of the mid

if(key>=a[s] and key<=a[mid])

{

e=mid-1;

}

else{

s=mid+1;

}

}

else{

if(key>=a[mid] and key<=a[e])

{

s=mid+1;

}

else

{

e=mid-1;

}

}

}

return -1;

}

int main()

{

int n;

cin>>n;

int a[1000];

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

{

cin>>a[i];

}

int key;

cin>>key;

cout<<find\_key(a,n,key);

}

Square Root a number

#include<iostream>

using namespace std;

float square\_Root(int n,int p)

{

int s=0;

int e=n;

float ans=-1;

while(s<=e)

{

int mid=(s+e)/2;

if(mid\*mid==n)

{

return mid;

}

if(mid\*mid<n)

{

ans=mid;

s=mid+1;

}

else {

e=mid-1;

}

}

//Floating Part

//Brute Force

float inc=0.1;

for(int times=1;times<=p;times++)

{

while(ans\*ans<=n)

{

ans=ans+inc;

}

ans=ans-inc;

inc=inc/10;

}

return ans;

}

int main()

{

int n;

cin>>n;

cout<<square\_Root(n,5)<<endl;

}