15B17CI371 – Data Structures Lab

ODD 2024

Week 3-LAB B

Practice Lab

**1. Given an array containing n distinct numbers in the range [0,n]. Write a program to return the only number in the range that is missing from the array.**

#include <iostream>

using namespace std;

int missingElement(int \*arr)

{

for(int i=0;arr!=NULL;i++)

if(\*(arr+i)!=i)

return i;

}

int main()

{

int n;

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

cin>>n;

cout<<"Input the elements : ";

int \*arr=new int[n];

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

cin>>arr[i];

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

for(int j=i+1;j<n;j++)

if(arr[j]<arr[i])

{

int temp=arr[j];

arr[j]=arr[i];

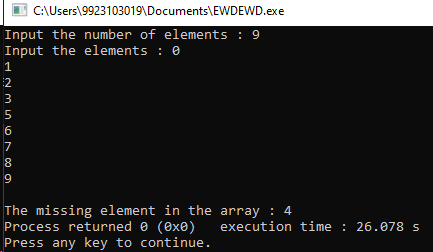
arr[i]=temp;

}

int miss=missingElement(arr);

cout<<"\nThe missing element in the array : "<<miss;

}



**2. Given a 1D array of integers,first sort the array in non-decreasing order,and then find two numbers such that the sum of two numbers add up to a specific value. If such a pair of numbers can be found in the array,return the indices,else return a suitable message.**

#include <iostream>

using namespace std;

void sort(int \*arr,int n)

{

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

for(int j=i+1;j<n;j++)

if(arr[j]<arr[i])

{

int temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

int main()

{

int n,target,i,j;

bool found=false;

cout<<"Input the target : ";

cin>>target;

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

cin>>n;

cout<<"Input the elements : ";

int \*arr=new int[n];

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

cin>>arr[i];

sort(arr,n);

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

{

for(j=i+1;j<n;j++)

{

if(arr[i]+arr[j]==target)

{

found=true;

break;

}

if(arr[i]+arr[j]>target)

{

found=true;

break;

}

}

if(found)

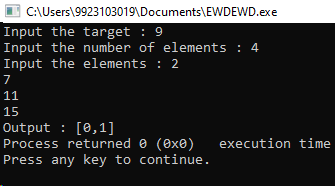
break;

}

cout<<"Output : ["<<i<<","<<j<<"]";

}

**Output :**

****

**3. You are given a list of numbers. They are unsorted. Sort this list. Assume that consecutive elements form pairs of numbers. Determine which pair or pairs of elements have the smallest absolute difference between them.**

#include <iostream>

using namespace std;

void sort(int \*arr,int n)

{

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

for(int j=i+1;j<n;j++)

if(arr[j]<arr[i])

{

int temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

int main()

{

int n,i,j,min\_diff;

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

cin>>n;

cout<<"Input the elements : ";

int \*arr=new int[n];

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

cin>>arr[i];

sort(arr,n);

cout<<"Sorted array : ";

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

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

cout<<"\nPairs : ";

min\_diff=arr[1]-arr[0];

for(i=0;i<n-1;i++)

{

if(arr[i+1]-arr[i]<min\_diff)

min\_diff=arr[i+1]-arr[i];

cout<<"{"<<arr[i]<<","<<arr[i+1]<<"} ,";

}

cout<<"\nSmallest Difference Pairs : ";

for(i=0;i<n-1;i++)

{

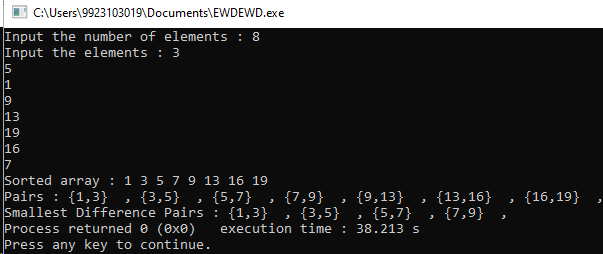
if(arr[i+1]-arr[i]==min\_diff)

cout<<"{"<<arr[i]<<","<<arr[i+1]<<"} ,";

}

}

**Output :**

****

**4. Given a sorted array of size N and an integer K,find the position at which K is present in the array using interpolation search.**

#include <iostream>

using namespace std;

int interpolationSearch(int arr[],int n,int K)

{

int low=0,high=n-1;

while (low<=high&&K>=arr[low]&&K<=arr[high])

{

if (low==high)

{

if (arr[low]==K) return low;

return -1;

}

int pos=low+(((double)(high-low)/(arr[high]-arr[low]))\*(K-arr[low]));

if (arr[pos]==K)

return pos;

if (arr[pos]<K)

low=pos+1;

else

high=pos-1;

}

return -1;

}

int main()

{

int n,i,j,k;

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

cin>>n;

cout<<"Input the elements : ";

int \*arr=new int[n];

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

cin>>arr[i];

cout<<"Input the value of K : ";

cin>>k;

int index=interpolationSearch(arr,n,k);

if (index!=-1)

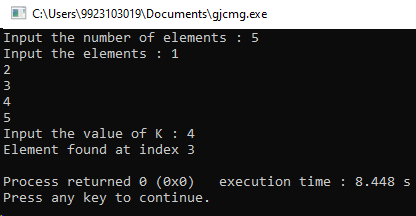
cout<<"Element found at index "<<index<<endl;

else

cout<<"Element not found."<<endl;

}

**Output :**

****

**5. Given an array of n distinct elements. Write a program to find the minimum**

**number of swaps required to sort the array in strictly increasing order.**

#include <iostream>

#include <vector>

using namespace std;

int minSwaps(int \*arr,int n) {

int swaps=0;

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

{

int min=arr[i];

int index=i;

for(int j=i+1;j<n;j++)

if(min>arr[j])

{

min=arr[j];

index=j;

}

int temp=arr[i];

arr[i]=arr[index];

arr[index]=temp;

swaps++;

}

return swaps;

}

int main()

{

int n;

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

cin>>n;

cout<<"Input the elements : ";

int \*arr=new int[n];

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

cin>>arr[i];

int a=minSwaps(arr,n);

cout<<"Sorted array : ";

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

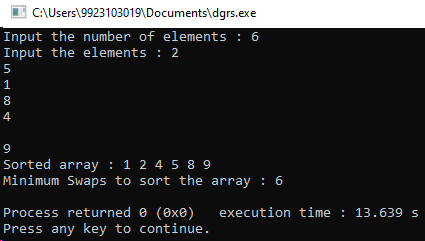
cout<<arr[i]<<" ";

cout<<"\nMinimum Swaps to sort the array : "<<a<<endl;

return 0;

}

**Output :**

****

**6.** **Given an array of integers. Write a program to find the Inversion Count in**

**the array.Inversion Count: For an array,inversion count indicates how far (or close) the**

**array is from being sorted. If the array is already sorted then the inversion count**

**is 0. If an array is sorted in the reverse order then the inversion count is themaximum.**

**Formally,two elements a[i] and a[j] form an inversion if a[i] > a[j] and i<j.**

#include <iostream>

#include <vector>

using namespace std;

int mergeAndCount(int \*arr,int left,int mid,int right)

{

int n1=mid-left+1;

int n2=right-mid;

vector<int> leftArr(n1);

vector<int> rightArr(n2);

for (int i=0; i<n1; i++)

leftArr[i]=arr[left+i];

for (int i=0; i<n2; i++)

rightArr[i]=arr[mid+1+i];

int i=0,j=0,k=left,swaps=0;

while (i<n1 && j<n2)

{

if (leftArr[i]<=rightArr[j])

arr[k++]=leftArr[i++];

else

{

arr[k++]=rightArr[j++];

swaps+=(n1-i);

}

}

while (i<n1) arr[k++]=leftArr[i++];

while (j<n2) arr[k++]=rightArr[j++];

return swaps;

}

int mergeSortAndCount(int\* arr,int left,int right)

{

int count=0;

if (left<right)

{

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

count+=mergeSortAndCount(arr,left,mid);

count+=mergeSortAndCount(arr,mid+1,right);

count+=mergeAndCount(arr,left,mid,right);

}

return count;

}

int main()

{

int n;

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

cin>>n;

cout<<"Input the elements : ";

int \*arr=new int[n];

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

cin>>arr[i];

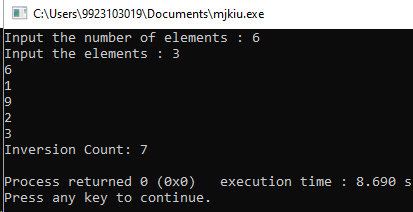
int result=mergeSortAndCount(arr,0,n-1);

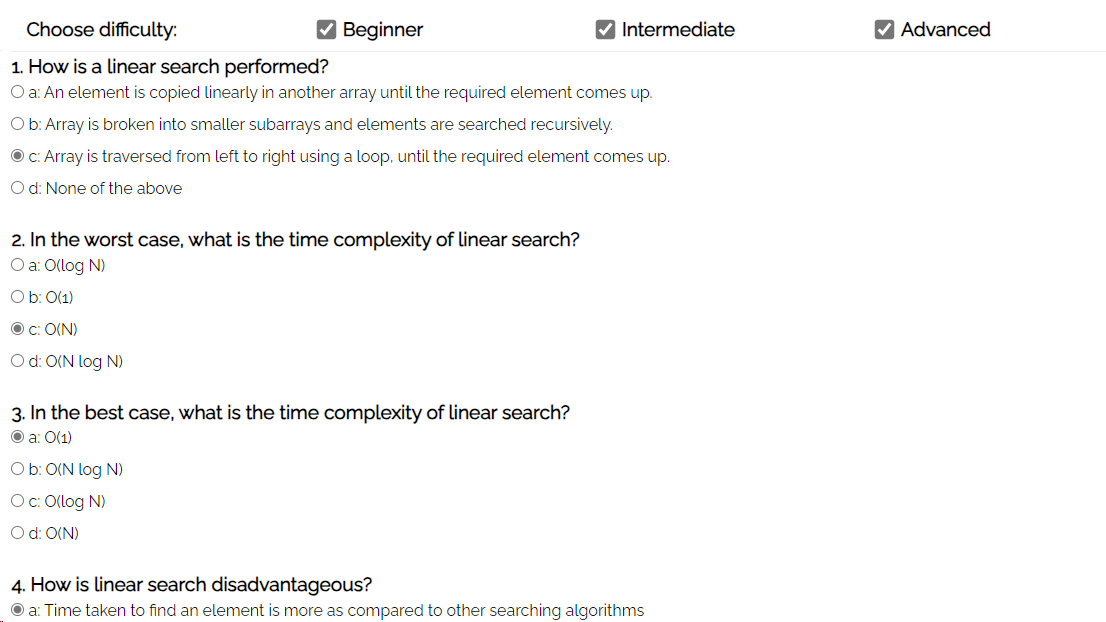
cout<<"Inversion Count: "<<result<<endl;

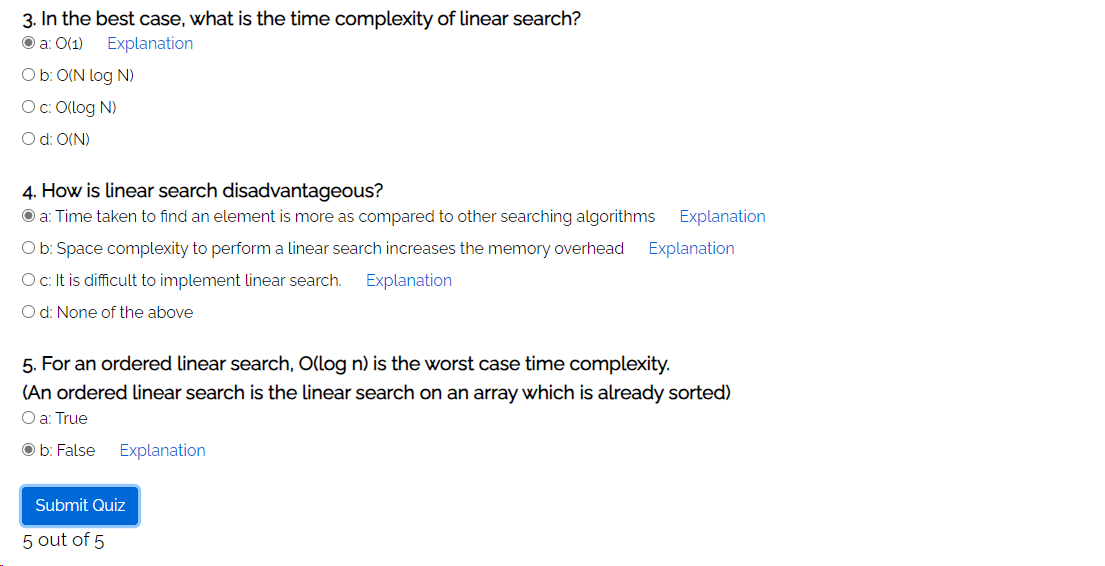
return 0;

}

**Output :**

****

**Virtual Labs : **

****