**15B17CI371 – Data Structures Lab**

**ODD 2024**

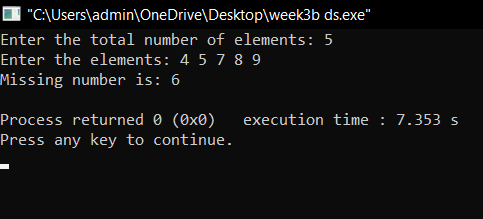
**Week 3-LAB B**

**Practice Lab**

**[CO: C270.2]**

**Q1. 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.**



int bsearch(int arr[],int n)

{

int low=0;

int high=n;

while(low<=high)

{

int mid=(low+high)/2;

if((arr[mid]<n)&&(arr[mid]!=mid))

{

high=mid-1;

}

else

{

low=mid+1;

}

}

return low;

}

int main()

{

int n;

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

cin>>n;

int \*arr=new int[n];

cout<<"Enter the elements: ";

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

{

cin>>arr[i];

}

cout<<"Missing number is: "<<bsearch(arr,n)<<endl;

}

**Q2. 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.**

**Example 1:**

**Input: numbers = [2,7,11,15], target = 9**

**Output: [1,2]**

**Hint: The sum of 2 and 7 is 9.**

**Example 2:**

**Input: numbers = [2,3,4], target = 6**

**Output: [1,3]**

**Hint: The sum of 2 and 4 is 6.**

void swap(int &a,int &b)

{

int temp=a;

a=b;

b=temp;

}

void insertionsort(int arr[],int n)

{

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

{

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

{

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

{

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

}

}

}

}

void sum(int arr[],int n)

{

int low=0;

int high=n-1;

int targ;

cout<<"Enter the sum target: ";

cin>>targ;

while(low<=high)

{

int mid=(low+high)/2;

int summ=arr[low]+arr[high];

if(summ<targ)

{

low=mid+1;

}

else if(summ>targ)

{

high=mid-1;

}

else

{

cout<<"Pair with sum equal to target is: "<<low+1<<" , "<<high+1;

break;

}

}

}

int main()

{

int n;

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

cin>>n;

int \*arr=new int[n];

cout<<"Enter the elements: ";

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

{

cin>>arr[i];

}

insertionsort(arr,n);

cout<<"Sorted array: ";

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

{

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

}

cout<<endl;

sum(arr,n);

}

Q3. 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.

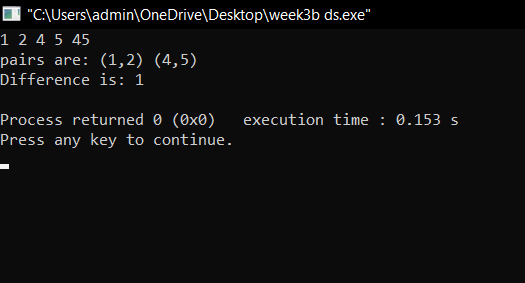
Example:

Arr = {2,5,4,89,1}

Sorted array= {1,2,4,5,89}

Pairs: {1,2}, {2,4}, {4,5}, {5,89}; Difference: 1,2,1,84

Output: Smallest: {1,2}, {4,5}



//3

#include <iostream>

#include <map>

#include <bits/stdc++.h>

using namespace std;

int part(int arr[],int low,int high)

{

int i=low;

int j=high;

int pivot=arr[low];

while(i<j)

{

while(arr[i]<=pivot && i<=high-1)

{

i++;

}

while(arr[j]>pivot && j>=low+1)

{

j--;

}

if(i<j)

{

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

}

}

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

return j;

}

void quicksort(int arr[],int low,int high)

{

if(low<high)

{

int p=part(arr,low,high);

quicksort(arr,low,p-1);

quicksort(arr,p+1,high);

}

}

int main()

{

int arr[5]={2,5,4,45,1};

int n=5;

quicksort(arr,0,n-1);

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

{

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

}

map<pair<int,int>,int> m;

int min=INT\_MAX;

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

{

int diff=(arr[i+1]-arr[i]);

if(diff<min)

{

min=diff;

}

m[{arr[i],arr[i+1]}]=diff;

}

map<pair<int, int>, int>::iterator it;

it=m.begin();

cout<<endl<<"pairs are: ";

while(it!=m.end())

{

if(it->second==min)

{

cout<<"("<<it->first.first<<","<<it->first.second<<")"<<" ";

}

it++;

}

cout<<endl<<"Difference is: "<<min<<endl;

}

Q4. Given a sorted array of size N and an integer K, find the position at which K is

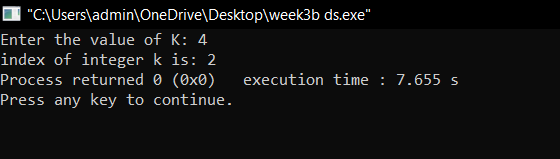
present in the array using interpolation search.

Example

Input: N = 5, arr[ ] = {1 2 3 4 5}, K = 4

Output: 3

Explanation: 4 appears at index 3



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

{

int low=0;

int high=n-1;

while(low<=high)

{

int probe=low+(k-arr[low])\*(high-low)/(arr[high]-arr[low]);

if(arr[probe]==k)

{

return probe;

}

else if(arr[probe]>k)

{

high=probe-1;

}

else

{

low=probe+1;

}

}

return -1;

}

int main()

{

int k;

int arr[5]={1,3,4,5,6};

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

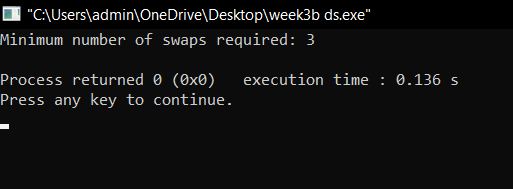
cin>>k;

cout<<"index of integer k is: "<<interpolation(arr,5,k);

}

Q5. 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.



int minSwapsToSort(vector<int>& arr) {

int n = arr.size();

vector<pair<int, int>> arrPos(n);

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

arrPos[i] = {arr[i], i};

}

// Sort the array by array values

sort(arrPos.begin(), arrPos.end());

vector<bool> visited(n, false);

int swaps = 0;

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

if (visited[i] || arrPos[i].second == i) {

continue;

}

int cycleSize = 0;

int j = i;

while (!visited[j]) {

visited[j] = true;

j = arrPos[j].second;

++cycleSize;

}

if (cycleSize > 1) {

swaps += (cycleSize - 1);

}

}

return swaps;

}

int main() {

vector<int> arr = {4, 3, 2, 1, 9, 6};

cout << "Minimum number of swaps required: " << minSwapsToSort(arr) << endl;

return 0;

}

Q6. Given an array of integers. Write a program to find the Inversion Count in

the array.

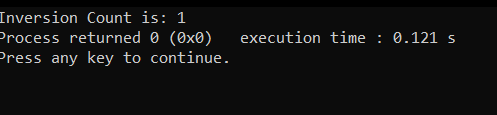
Inversion Count: For an array, inversion count indicates how far (or close) the

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 the

maximum.

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



int inversioncount(int arr[],int n)

{

int count=0;

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

{

int j=i+1;

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

{

count++;

}

}

return count;

}

int main()

{

int arr[5]={2,1,3,4,5};

int n=sizeof(arr)/sizeof(arr[0]);

cout<<"Inversion Count is: "<<inversioncount(arr,n);

}