15B17CI371 – Data Structures Lab

ODD 2024

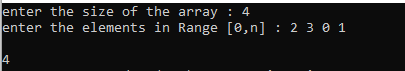
Week 3-LAB B

9923103023 – F1

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.

Ans :



#include<iostream>

#include<cmath>

using namespace std;

int main()

{

int n;

cout<<"enter the size of the array : ";

cin>>n;

cout<<"enter the elements in Range [0,n] : ";

int \*arr = new int[n];

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

{

cin>>arr[i];

}

int missing;

int sum=0;

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

{

sum+= arr[i];

}

int req\_sum = ((n)\*(n+1))/2;

missing = req\_sum - sum;

cout<<"\n"<<missing;

}

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.

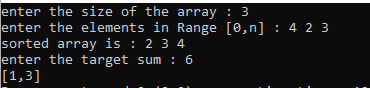
Example 2:

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

Output: [1,3]

Hint: The sum of 2 and 4 is 6.

Ans :



#include<iostream>

#include<cmath>

using namespace std;

void sort\_arr(int \*arr,int n)

{

int tp;

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

{

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

{

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

{

tp = arr[i];

arr[i] = arr[j];

arr[j] = tp;

}

}

}

}

void fun(int \* arr,int n)

{

int indi1,indi2,target;

cout<<"enter the target sum : ";

cin>>target;

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

{

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

{

indi1=i+1;

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

{

indi2 = j+1;

cout<<"["<<indi1<<","<<indi2<<"]";

return;

}

}

}

}

void printar(int \* ra,int n)

{

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

{

cout<<ra[i]<<" ";

}

}

int main()

{

int n;

cout<<"enter the size of the array : ";

cin>>n;

cout<<"enter the elements in Range [0,n] : ";

int \*arr = new int[n];

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

{

cin>>arr[i];

}

cout <<"sorted array is : ";

sort\_arr(arr,n);

printar(arr,n);

cout <<"\n";

fun(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:

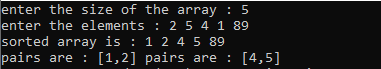
Arr = {2,5,4,89,1}

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}

Ans :



#include<iostream>

#include<cmath>

using namespace std;

void sort\_arr(int \*arr,int n)

{

int tp;

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

{

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

{

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

{

tp = arr[i];

arr[i] = arr[j];

arr[j] = tp;

}

}

}

}

void abs\_pairs(int \* arr,int n)

{

int \* par = new int[n-1];

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

{

par[i] = abs((arr[i+1])-(arr[i]));

}

int indi1=0;

int mini = par[0];

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

{

if(par[i]<mini)

{

mini = par[i];

}

}

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

{

if(par[i]==mini)

{

cout<<"pairs are : "<<"["<<arr[i]<<","<<arr[i + 1]<<"] ";

}

}

}

void printar(int \* ra,int n)

{

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

{

cout<<ra[i]<<" ";

}

}

int main()

{

int n;

cout<<"enter the size of the array : ";

cin>>n;

cout<<"enter the elements : ";

int \*arr = new int[n];

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

{

cin>>arr[i];

}

cout <<"sorted array is : ";

sort\_arr(arr,n);

printar(arr,n);

cout <<"\n";

abs\_pairs(arr,n);

}

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.

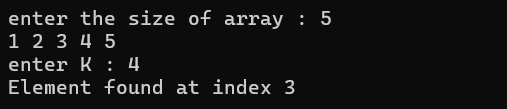
Example

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

Output: 3

Explanation: 4 appears at index 3

Ans :



#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,k;

cout<<"enter the size of array : ";

cin>>n;

int\* arr = new int[n];

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

{

cin>>arr[i];

}

cout<<"enter K : ";

cin>>k;

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

if (index != -1)

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

else

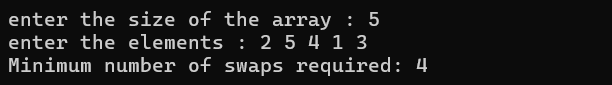
cout << "Element not found." << endl;

return 0;

}

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.

Ans : 

#include <iostream>

#include <algorithm>

using namespace std;

int minSwaps(int arr[], int n)

{

pair<int, int> arrPos[n];

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

{

arrPos[i].first = arr[i];

arrPos[i].second = i;

}

sort(arrPos, arrPos + n);

bool visited[n];

fill(visited, 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()

{

int n;

cout<<"enter the size of the array : ";

cin>>n;

cout<<"enter the elements : ";

int \*arr = new int[n];

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

{

cin>>arr[i];

}

cout << "Minimum number of swaps required: " << minSwaps(arr, n) << 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

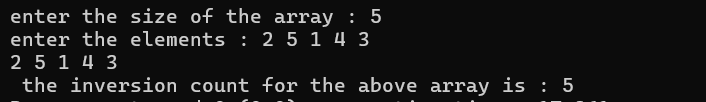
array is from being sorted. If the array is already sorted then the inversion count

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.

Ans :



#include<iostream>

#include<cmath>

using namespace std;

void sort\_arr(int \*arr,int n)

{

int tp;

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

{

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

{

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

{

tp = arr[i];

arr[i] = arr[j];

arr[j] = tp;

}

}

}

}

void inv\_count(int \* arr,int n)

{

int invc = 0;

int tp,swap\_no=0;

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

{

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

{

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

{

tp = arr[i];

arr[i] = arr[j];

arr[j] = tp;

invc++;

}

}

}

cout<<" the inversion count for the above array is : "<< invc;

}

void printar(int \* ra,int n)

{

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

{

cout<<ra[i]<<" ";

}

}

int main()

{

int n;

cout<<"enter the size of the array : ";

cin>>n;

cout<<"enter the elements : ";

int \*arr = new int[n];

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

{

cin>>arr[i];

}

printar(arr,n);

cout <<"\n";

inv\_count(arr,n);

}