DSA PRACTICE PROBLEM DAY 5

PROBLEM 1:

The cost of stock on each day is given in an array A[] of size N. Find all the segments of days on which you buy and sell the stock such that the sum of difference between sell and buy prices is maximized. Each segment consists of indexes of two elements, first is index of day on which you buy stock and second is index of day on which you sell stock.

Note: Since there can be multiple solutions, the driver code will print 1 if your answer is correct, otherwise, it will return 0. In case there's no profit the driver code will print the string "No Profit" for a correct solution.

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] A = new int[n];

System.out.println("Enter the Elements in the Array :");

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

A[i] = scan.nextInt();

}

ArrayList<ArrayList<Integer>> result = new ArrayList<>();

int minIdx = 0;

int maxIdx = 0;

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

if (A[i] >= A[maxIdx]) {

maxIdx = i;

} else {

if (maxIdx > minIdx) {

ArrayList<Integer> list = new ArrayList<>();

list.add(minIdx);

list.add(maxIdx);

result.add(list);

}

minIdx = i;

maxIdx = i;

}

}

if (maxIdx > minIdx) {

ArrayList<Integer> list = new ArrayList<>();

list.add(minIdx);

list.add(maxIdx);

result.add(list);

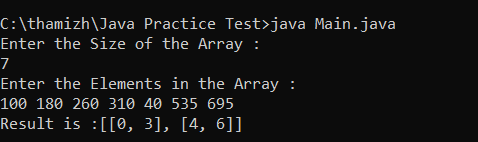
}

System.out.println("Result is :" + result);

}

}

OUTPUT:



PROBLEM 2:

Given an integer array coins[ ] representing different denominations of currency and an integer sum, find the number of ways you can make sum by using different combinations from coins[ ].

Note: Assume that you have an infinite supply of each type of coin. And you can use any coin as many times as you want.

Answers are guaranteed to fit into a 32-bit integer.

CODE :

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] coins = new int[n];

System.out.println("Enter the Elements in the Array :");

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

coins[i] = scan.nextInt();

}

System.out.println("Enter the target value : ");

int sum = scan.nextInt();

int[] dp = new int[sum+1];

dp[0] = 1;

for(int coin : coins){

for(int j = coin ; j<=sum ;j++){

dp[j] += dp[j-coin];

}

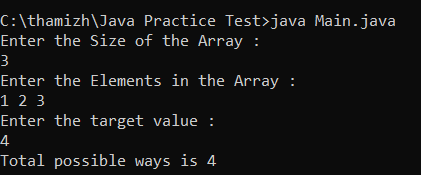
}

System.out.println("Total possible ways is " + dp[sum]);

}

}

OUTPUT :



PROBLEM 3:

Given a sorted array arr with possibly some duplicates, the task is to find the first and last occurrences of an element x in the given array.

Note: If the number x is not found in the array then return both the indices as -1.

CODE :

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array :");

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

arr[i] = scan.nextInt();

}

System.out.println("Enter the Target Element ");

int x = scan.nextInt();

int start = -1;

int end = -1;

for(int i = 0; i<arr.length;i++){

if(arr[i]==x){

start = i;

break;

}

}

for(int i=n-1;i>=0;i--){

if(arr[i]==x){

end = i;

break;

}

}

ArrayList<Integer> result = new ArrayList<>();

if(start==-1 || end==-1){

result.add(-1);

result.add(-1);

}else{

result.add(start);

result.add(end);

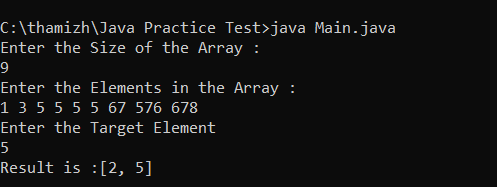
}

System.out.println("Result is :"+ result);

}

}

OUTPUT :



PROBLEM 4:

Given a sorted array, arr[] containing only 0s and 1s, find the transition point, i.e., the first index where 1 was observed, and before that, only 0 was observed. If arr does not have any 1, return -1. If array does not have any 0, return 0.

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array only 0 and 1 in Sorted Order:");

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

arr[i] = scan.nextInt();

}

int result = -1;

int low = 0;

int high = arr.length-1;

while(low<=high){

int mid = low + (high-low)/2;

if(arr[mid]==1){

result = mid;

high = mid-1;

}else{

low = mid+1;

}

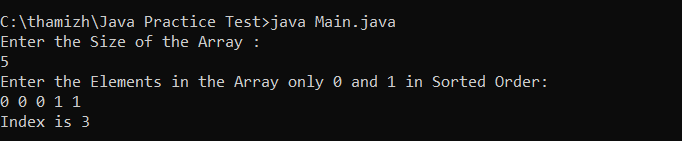
}

System.out.println("Index is " + result);

}

}

OUTPUT :



PROBLEM 5 :

Given an array arr[], find the first repeating element. The element should occur more than once and the index of its first occurrence should be the smallest.

Note:- The position you return should be according to 1-based indexing.

CODE :

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array :");

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

arr[i] = scan.nextInt();

}

Map<Integer,Integer> map = new HashMap<>();

int min = Integer.MAX\_VALUE;

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

if(map.containsKey(arr[i])){

min = Math.min(min,map.get(arr[i]));

}else{

map.put(arr[i],i);

}

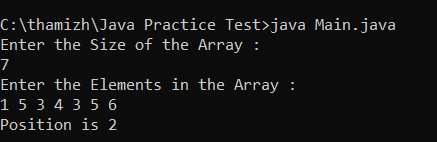
}

System.out.println("Position is " + (min==Integer.MAX\_VALUE?-1:min+1));

}

}

OUTPUT :



PROBLEM 6:

Given a sorted array arr. Return the size of the modified array which contains only distinct elements.

Note:

1. Don't use set or HashMap to solve the problem.

2. You must return the modified array size only where distinct elements are present and modify the original array such that all the distinct elements come at the beginning of the original array.

CODE:

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

ArrayList<Integer> arr = new ArrayList<>();

System.out.println("Enter the Elements in the Array :");

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

int num = scan.nextInt();

arr.add(num);

}

int k=1;

int c=0;

for(int i=1;i<arr.size();i++){

if(!arr.get(c).equals(arr.get(i))){

c++;

arr.set(c,arr.get(i));

k++;

}

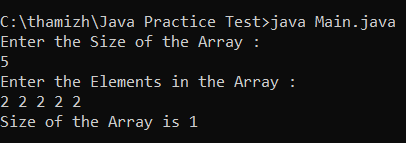
}

System.out.println("Size of the Array is "+k);

}

}

OUTPUT :



PROBLEM 7:

Given an array arr of positive integers. The task is to return the maximum of j - i subjected to the constraint of arr[i] < arr[j] and i < j.

CODE :

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array :");

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

arr[i] = scan.nextInt();

}

int[] LeftMin = new int[n];

int[] RightMax = new int[n];

LeftMin[0] = arr[0];

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

LeftMin[i] = Math.min(arr[i], LeftMin[i - 1]);

}

RightMax[n - 1] = arr[n - 1];

for (int j = n - 2; j >= 0; j--) {

RightMax[j] = Math.max(arr[j], RightMax[j + 1]);

}

int i = 0, j = 0, maxDiff = -1;

while (i < n && j < n) {

if (LeftMin[i] <= RightMax[j]) {

maxDiff = Math.max(maxDiff, j - i);

j++;

} else {

i++;

}

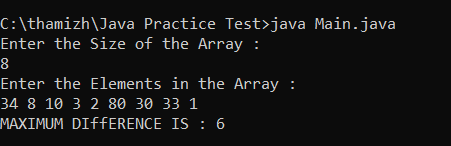
}

System.out.println("MAXIMUM DIffERENCE IS : " + maxDiff);

}

}

OUTPUT :



PROBLEM 8 :

Given a sorted array arr[] of distinct integers. Sort the array into a wave-like array(In Place). In other words, arrange the elements into a sequence such that arr[1] >= arr[2] <= arr[3] >= arr[4] <= arr[5].....

If there are multiple solutions, find the lexicographically smallest one.

Note: The given array is sorted in ascending order, and you don't need to return anything to change the original array.

CODE :

import java.util.\*;

public class Main {

public static void main(String... argv) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the Size of the Array :");

int n = scan.nextInt();

int[] arr = new int[n];

System.out.println("Enter the Elements in the Array :");

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

arr[i] = scan.nextInt();

}

for(int i=0;i<arr.length-1;i+=2){

int temp = arr[i];

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

arr[i+1] = temp;

}

For(int i : arr){

System.out.print(i+” “);

}

}

}

OUTPUT :

