Day-4 DSA Assignment Date: 14/11/2024

1. Stock and Buy Sell: TC: O(n2)

class Solution{  
 ArrayList<ArrayList<Integer> > stockBuySell(int A[], int n) {

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

int i=0;

while(i<n-1)

{

while(i<n-1 && A[i+1]<=A[i])

{

i++;

}

if(i==n-1)

{

break;

}

int buy=i++;

while(i<n && A[i]>=A[i-1])

{

i++;

}

int sell=i-1;

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

pair.add(buy);

pair.add(sell);

a.add(pair);

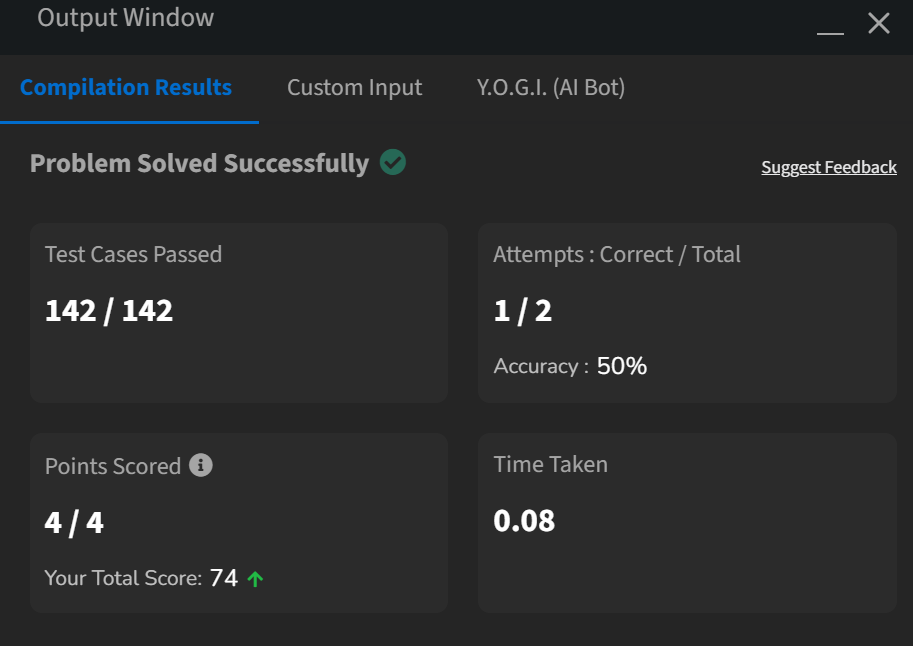
}

return a;

}

}

Output:

  
2. Coin Change: TC: O(n^2)

class Solution {

public int count(int coins[], int sum) {

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

for(int i=0;i<sum+1;i++)

arr[i]=0;

arr[0]=1;

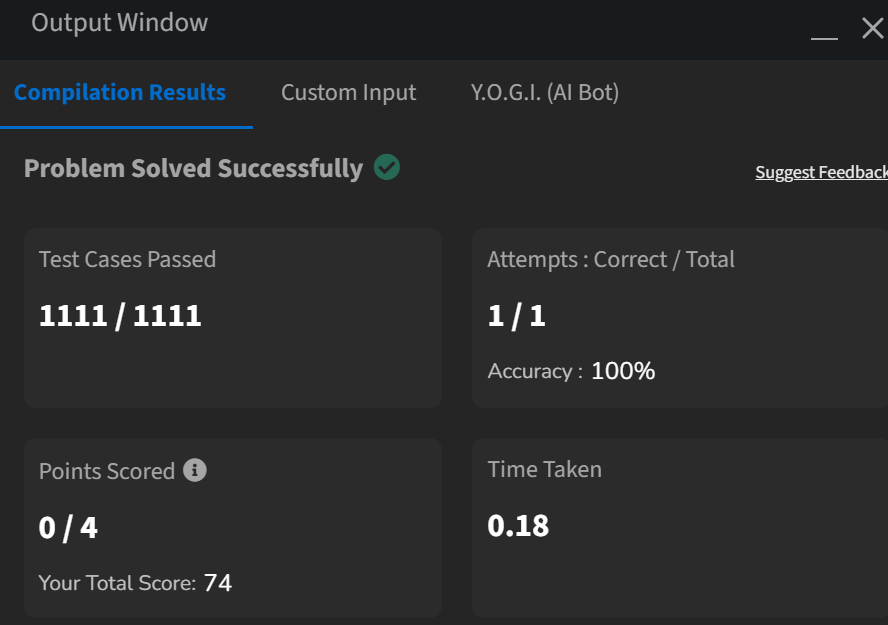
for(int i=0;i<coins.length;i++)

for(int j=coins[i];j<=sum;j++)

arr[j]+=arr[j-coins[i]];

return arr[sum]; }}

Output:



1. First and Last Occurrence of an element: TC:O(N)

class GFG {

ArrayList<Integer> find(int arr[], int x) {

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

boolean exists = false;

for (int num : arr) {

if (num == x) {

exists = true;

break;

}

}

if (!exists) {

a.add(-1);

a.add(-1);

return a;

}

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

{

if(arr[i]==x)

{

a.add(i);

break;

}

}

for(int i=arr.length-1;i>=0;i--)

{

if(arr[i]==x)

{

a.add(i);

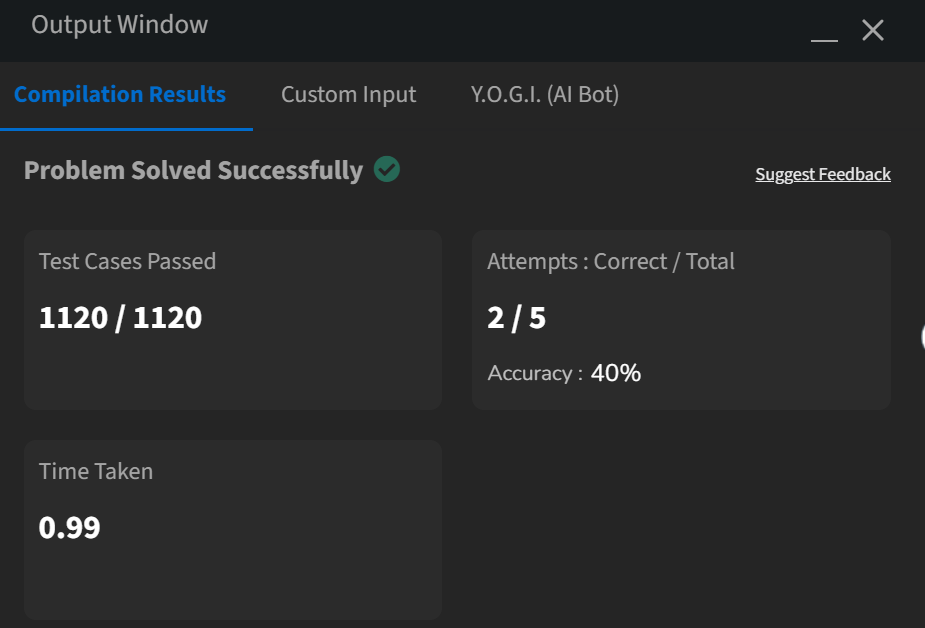
break;

}

return a;

}

}



4.

Program: TC: O(N)

class Solution {

int transitionPoint(int arr[])

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

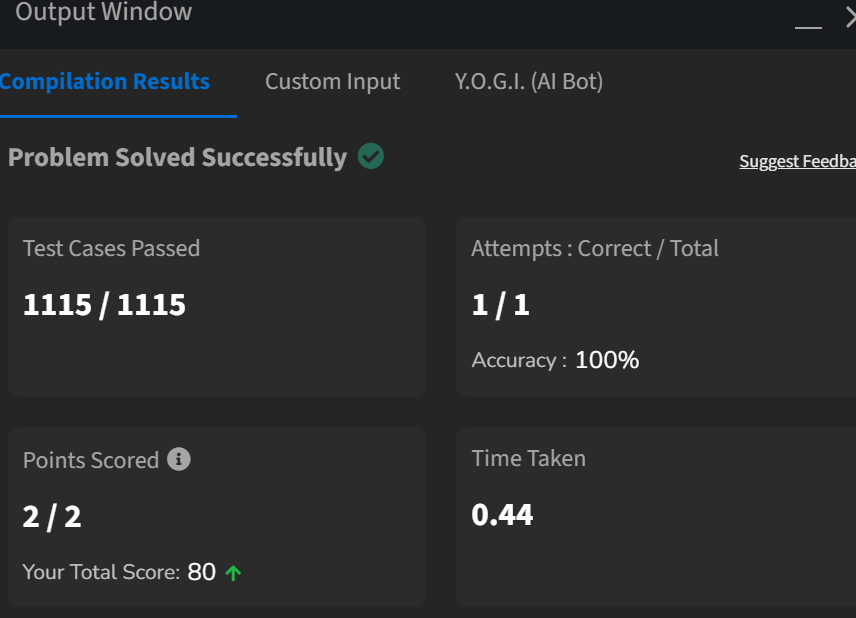
if(arr[i] == 1)

return i;

return -1;

}

}



5. First Repeating Element:TC: O(N)

class Solution {

public static int firstRepeated(int[] arr) {

HashMap<Integer, Integer> firstOccurrenceMap = new HashMap<>();

Set<Integer> repeatingElements = new HashSet<>();

int firstRepeatingIndex = Integer.MAX\_VALUE;

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

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

repeatingElements.add(arr[i]);

} else {

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

}

}

for (int element : repeatingElements) {

int index = firstOccurrenceMap.get(element);

if (index < firstRepeatingIndex) {

firstRepeatingIndex = index;

}

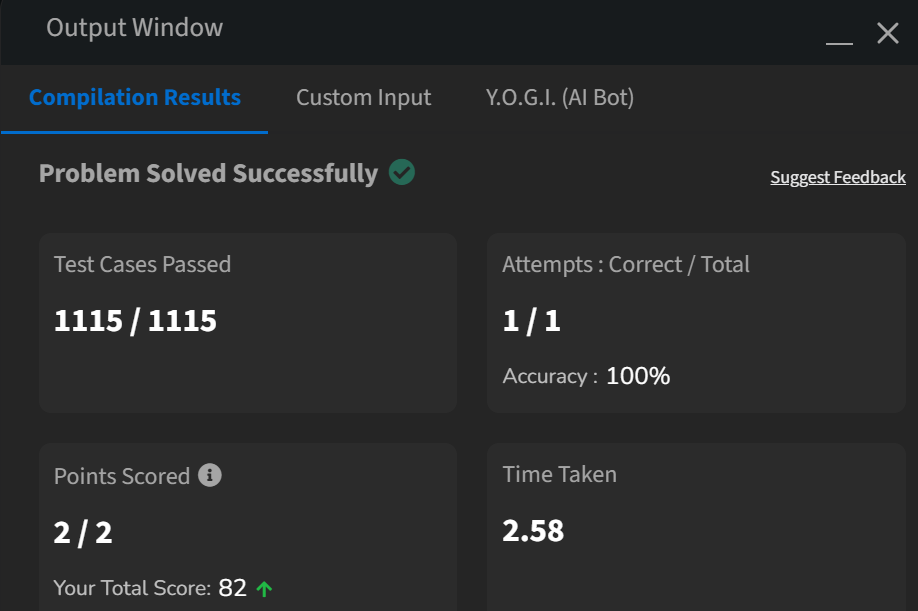
}

return firstRepeatingIndex == Integer.MAX\_VALUE ? -1 : firstRepeatingIndex + 1;

}

}

Output:



6.Remove Duplicates in an sorted array: TC(O(N))

class Solution {

public int remove\_duplicate(List<Integer> arr) {

if (arr.size() == 0) return 0

int uniqueIndex = 0;

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

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

uniqueIndex++;

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

}

}

for (int i = arr.size() - 1; i > uniqueIndex; i--) {

arr.remove(i);

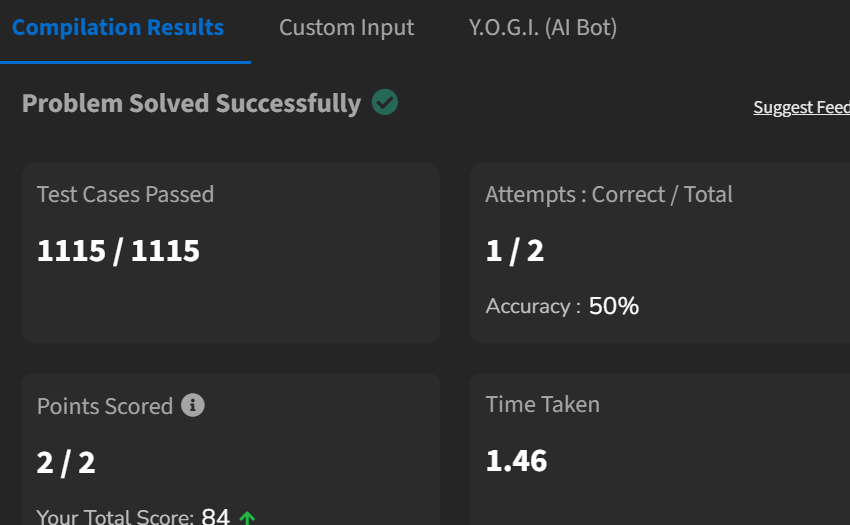
}

return arr.size();

}

}

Output:



7.Maximum Index: TC:O(N)

class Solution {

int maxIndexDiff(int[] arr) {

int n=arr.length;

int[] lMin = new int[n];

int[] rMax = new int[n];

lMin[0] = arr[0];

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

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

}

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

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

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

}

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

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

if (lMin[i] <= rMax[j]) {

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

j++;

} else {

i++;

}

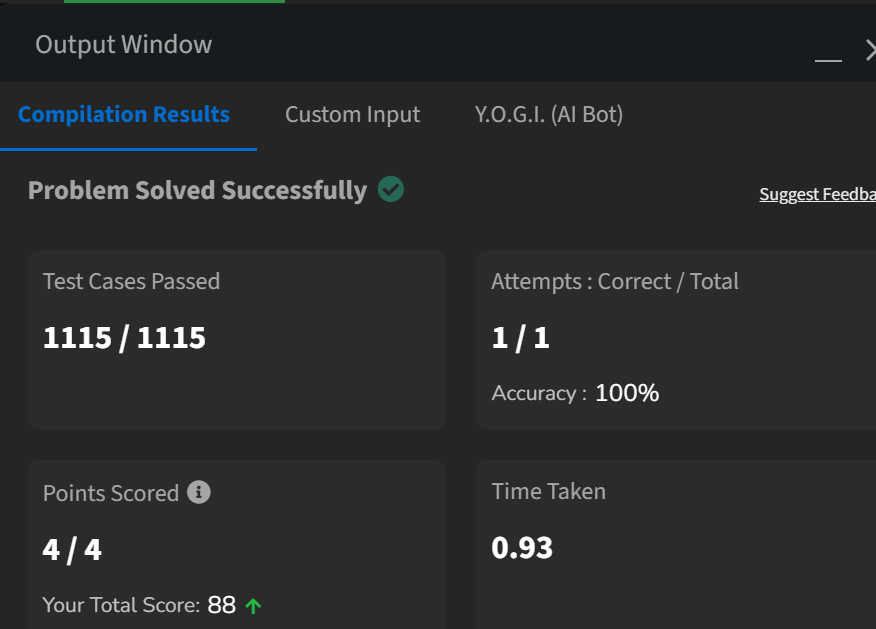
}

return maxDiff;

}

}

Output:



8.Wave Array:TC : O(N)

class Solution {

static void swap(int arr[], int a, int b)

{

int temp = arr[a];

arr[a] = arr[b];

arr[b] = temp;

}

public static void convertToWave(int[] arr)

{

int n=arr.length;

Arrays.sort(arr);

for (int i=0; i<n-1; i += 2)

swap(arr, i, i+1);

} }

Output:

