**DSA Practice 5 (14-11-24)**

**By Jashwanth SA**

**CSE-C, 22CS066**

1. **Stock buy and sell:**

class Solution{

//Function to find the days of buying and selling stock for max profit.

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

// code here

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

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

if(A[i+1]>A[i]){

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

x.add(i);

x.add(i+1);

res.add(x);

}

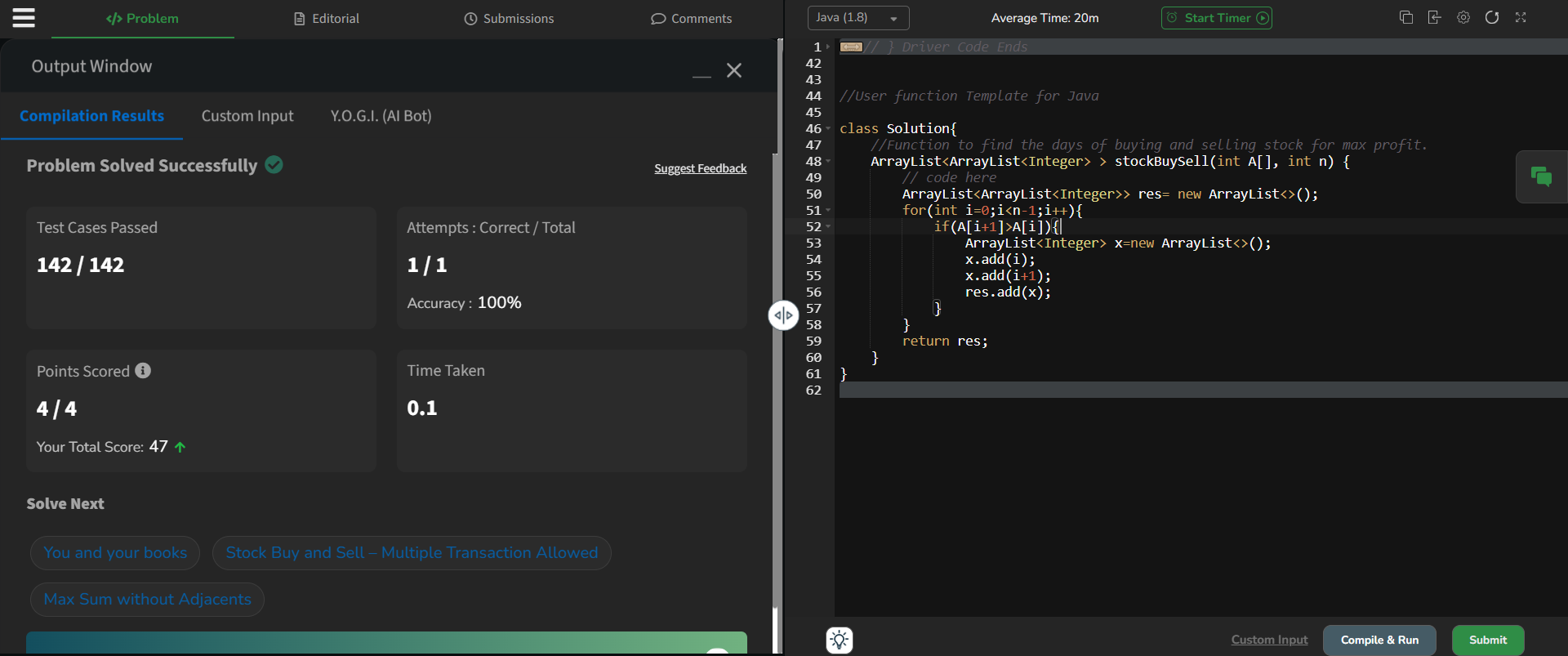
}

return res;

}

}

Output:



Time complexity: O(n)

1. **Coin change (count ways):**

class Solution {

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

// code here.

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

dp[0]=1;

for(int i=1;i<coins.length+1;i++){

for(int j=1;j<sum+1;j++){

if(j-coins[i-1]>=0){ dp[j]+=dp[j-coins[i-1]];}

}

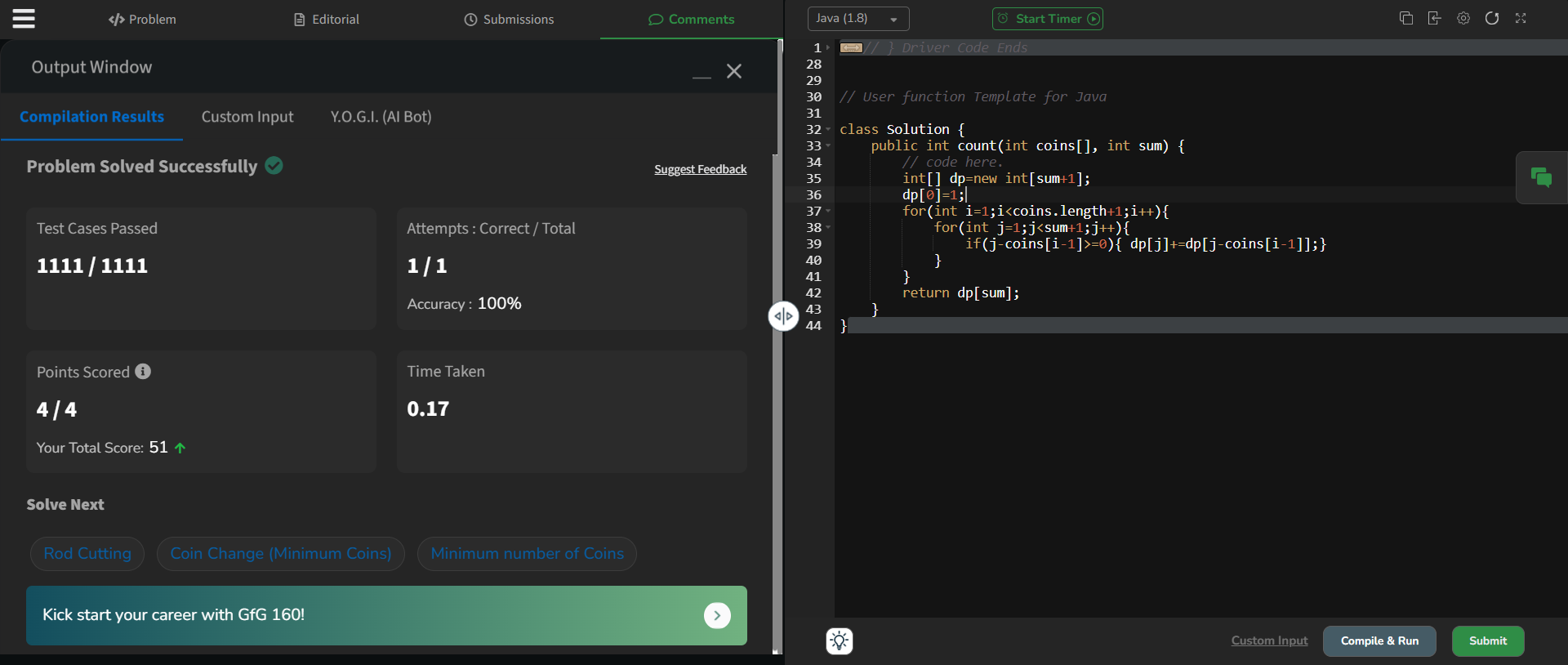
}

return dp[sum];

}

}

Output:



Time complexity: O(n\*sum)

1. **First and last occurences:**

class GFG {

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

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

int f=-1,l=-1;

boolean y=false;

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

if(arr[i]==x){

if(!y){

f=i;

y=true;}

l=i;

}

}

res.add(f);

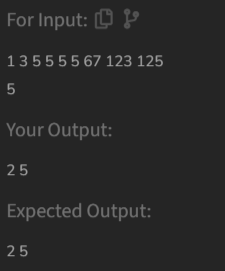
res.add(l);

return res;

}

}

Output:



Time complexity: O(n)

1. **Find transition point:**

class Solution {

int transitionPoint(int arr[]) {

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

if(arr[i]==1){return i;}

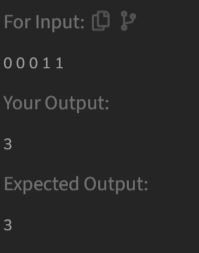
}

return -1;

}

}

Output:



Time complexity: O(n)

1. **Find repeating element:**

class Solution {

// Function to return the position of the first repeating element.

public static int firstRepeated(int[] arr) {

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

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

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

res.put(arr[i],res.get(arr[i])+1);

}else{

res.put(arr[i],1);

}

}

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

if(res.get(arr[i])>1){return i+1;}

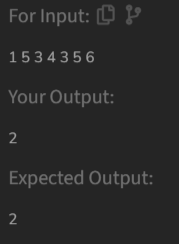
}

return -1;

}

}

Output:



Time complexity: O(n)

1. **Remove duplicates sorted array:**

class Solution {

// Function to remove duplicates from the given array

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

int i=0;

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

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

i++;

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

}

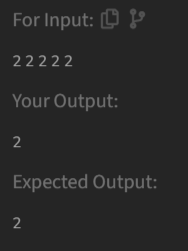
}

return i+1;

}

}

Output:



Time complexity: O(n)

1. **Maximum index:**

class Solution {

// Function to find the maximum index difference.

int maxIndexDiff(int[] arr) {

// Your code here

int n=arr.length;

int ml[]=new int[n];

int mr[]=new int[n];

ml[0]=arr[0];

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

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

}

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

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

mr[i]=Math.max(arr[i],mr[i+1]);

}

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

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

if(ml[i]<=mr[j]){

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

j++;

}else{

i++;

}

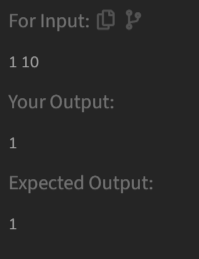
}

return md;

}

}

Output:



Time complexity: O(n)

1. **Wave array:**

class Solution {

public static void convertToWave(int[] arr) {

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

if(i%2==0){

int temp=arr[i];

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

arr[i+1]=temp;

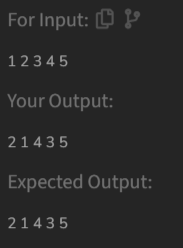
}

}

}

}

Output:



Time complexity: O(n)