DSA PRACTICE – DAY 4

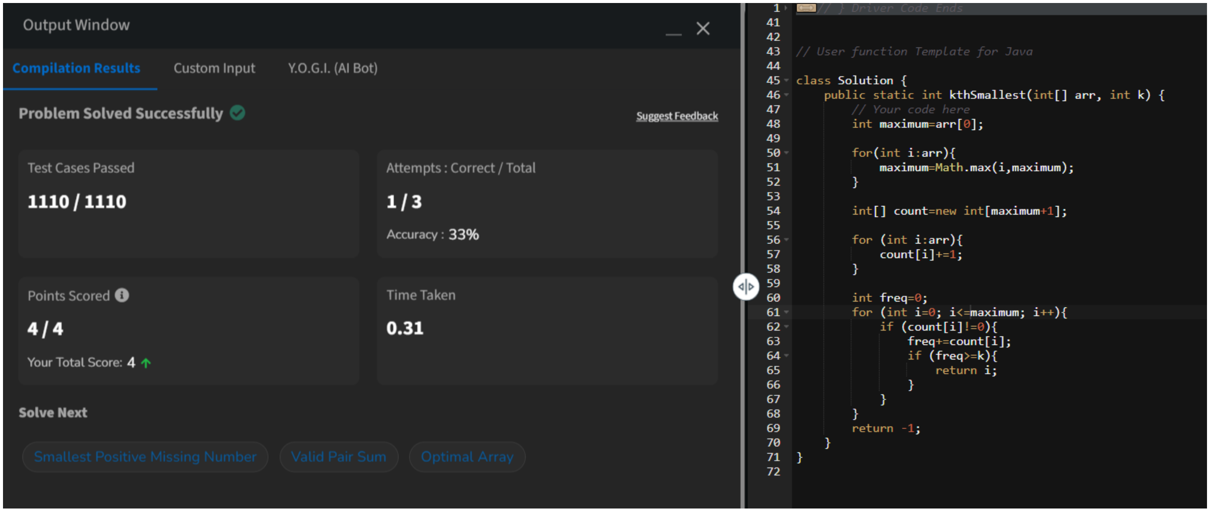
Name: Dhejan R **Reg No:** 22IT022 **Date:** 13/11/2024

1. Kth Smallest Element in an Unsorted Array

Code Solution:

```
class Solution {
    public static int kthSmallest(int[] arr, int k) {
        int maximum=arr[0];
        for(int i:arr){
            maximum=Math.max(i,maximum);
        }
        int[] count=new int[maximum+1];
        for (int i:arr){
            count[i]+=1;
        }
        int freq=0;
        for (int i=0; i<=maximum; i++){
            if (count[i]!=0){
                freq+=count[i];
                if (freq>=k){
                    return i;}
            }
        }
        return -1;
    }
}
```

Output:



Problem Solved Successfully ✓

Test Cases Passed
1110 / 1110

Attempts : Correct / Total
1 / 3
Accuracy : 33%

Points Scored
4 / 4
Your Total Score: 4 ↑

Time Taken
0.31

```
// } Driver Code Ends


// User function Template for Java

class Solution {
    public static int kthSmallest(int[] arr, int k) {
        // Your code here
        int maximum=arr[0];

        for(int i:arr){
            maximum=Math.max(i,maximum);
        }

        int[] count=new int[maximum+1];

        for (int i:arr){
            count[i]+=1;
        }

        int freq=0;
        for (int i=0; i<=maximum; i++){
            if (count[i]!=0){
                freq+=count[i];
                if (freq>=k){
                    return i;
                }
            }
        }
        return -1;
    }
}
```

Time complexity: O (m+n)

Space Complexity: O (max_element)

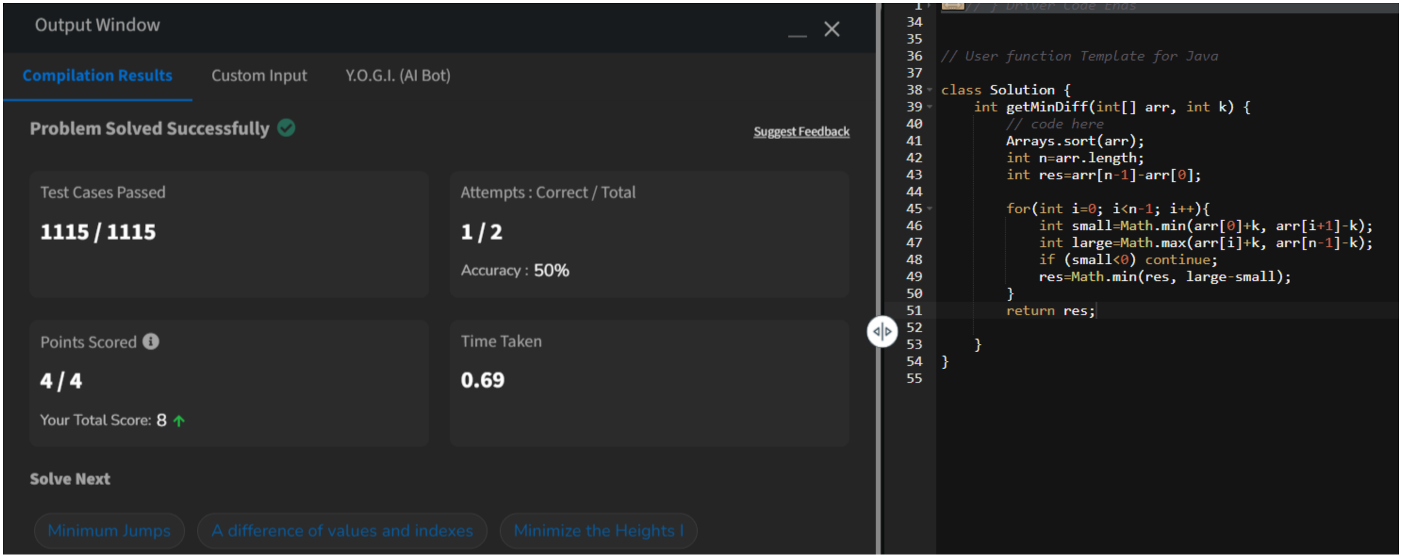
2. Minimize heights II

Code Solution:

```
class Solution {
    int getMinDiff(int[] arr, int k) {
        // code here
        Arrays.sort(arr);
        int n=arr.length;
        int res=arr[n-1]-arr[0];

        for(int i=0; i<n-1; i++){
            int small=Math.min(arr[0]+k, arr[i+1]-k);
            int large=Math.max(arr[i]+k, arr[n-1]-k);
            if (small<0) continue;
            res=Math.min(res, large-small);
        }
        return res;
    }
}
```

Output:



Problem Solved Successfully ✓

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 2
Accuracy : 50%

Points Scored
4 / 4
Your Total Score: 8 ↑

Time Taken
0.69

```
// } Driver Code Ends


// User function Template for Java

class Solution {
    int getMinDiff(int[] arr, int k) {
        // code here
        Arrays.sort(arr);
        int n=arr.length;
        int res=arr[n-1]-arr[0];

        for(int i=0; i<n-1; i++){
            int small=Math.min(arr[0]+k, arr[i+1]-k);
            int large=Math.max(arr[i]+k, arr[n-1]-k);
            if (small<0) continue;
            res=Math.min(res, large-small);
        }
        return res;

    }
}
```

Time Complexity: O (nlogn)
Space Complexity: O (n)

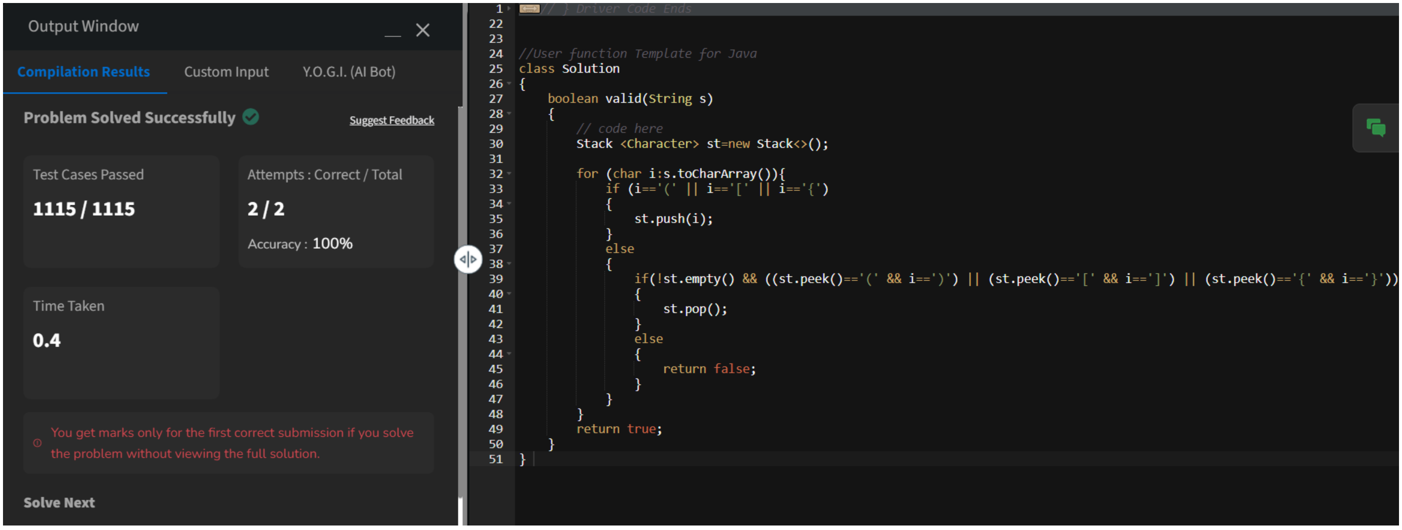
3. Valid Parentheses

Code Solution:

```
class Solution
{
    boolean valid(String s)
    {
        // code here
        Stack <Character> st=new Stack<>();

        for (char i:s.toCharArray()){
            if (i=='(' || i=='[' || i=='{'){
                st.push(i);
            }
            else{
                if(!st.empty() && ((st.peek()=='(' && i==')') || (st.peek()=='[' && i==']') || (st.peek()=='{' && i=='}'))){
                    st.pop();
                }else{
                    return false;
                }
            }
        }
        return true;
    }
}
```

Output:



Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
2 / 2
Accuracy : 100%

Time Taken
0.4

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

```
// } Driver Code Ends


//User function Template for Java
class Solution
{
    boolean valid(String s)
    {
        // code here
        Stack <Character> st=new Stack<>();

        for (char i:s.toCharArray()){
            if (i=='(' || i=='[' || i=='{')
            {
                st.push(i);
            }
            else
            {
                if(!st.empty() && ((st.peek()=='(' && i==')') || (st.peek()=='[' && i==']') || (st.peek()=='{' && i=='}'))
                {
                    st.pop();
                }
                else
                {
                    return false;
                }
            }
        }
        return true;
    }
}
```

Time complexity: O (n)

Space Complexity: O (n)

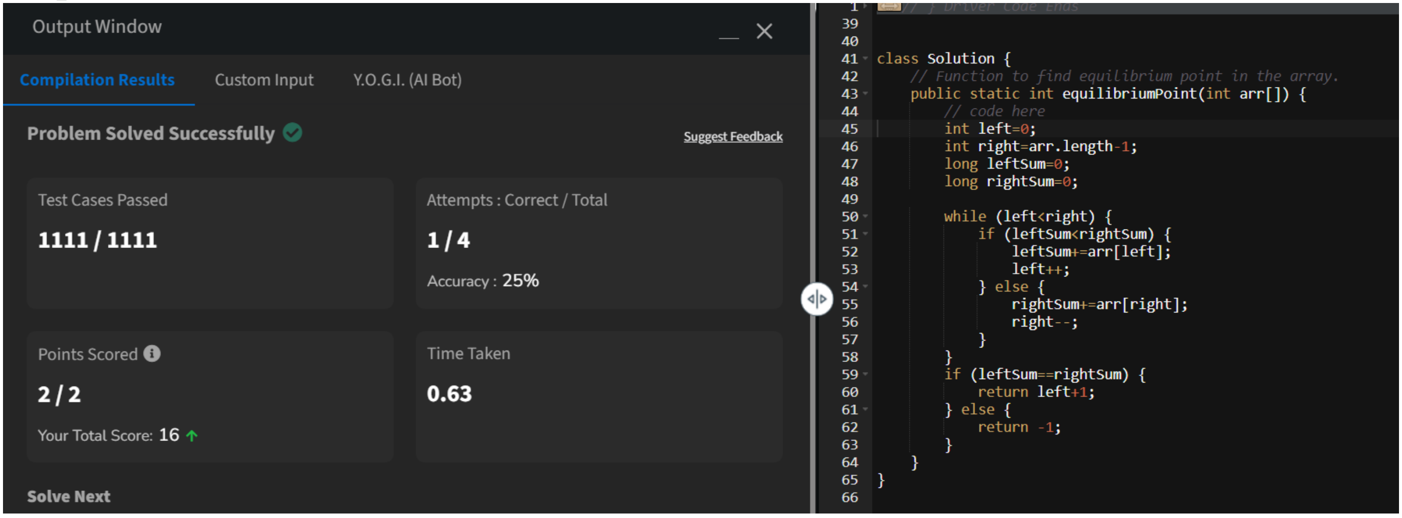
4. Equilibrium Point

Code Solution:

```
class Solution {
    // Function to find equilibrium point in the array.
    public static int equilibriumPoint(int arr[]) {
        // code here
        int left=0;
        int right=arr.length-1;
        long leftSum=0;
        long rightSum=0;

        while (left<right) {
            if (leftSum<rightSum) {
                leftSum+=arr[left];
                left++;
            } else {
                rightSum+=arr[right];
                right--;
            }
        }
        if (leftSum==rightSum) {
            return left+1;
        } else {
            return -1;
        }
    }
}
```

Output:



Problem Solved Successfully ✓

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 4
Accuracy : 25%

Points Scored
2 / 2
Your Total Score: 16 ↑

Time Taken
0.63

```
// } Driver Code Ends


class Solution {
    // Function to find equilibrium point in the array.
    public static int equilibriumPoint(int arr[]) {
        // code here
        int left=0;
        int right=arr.length-1;
        long leftSum=0;
        long rightSum=0;

        while (left<right) {
            if (leftSum<rightSum) {
                leftSum+=arr[left];
                left++;
            } else {
                rightSum+=arr[right];
                right--;
            }
        }
        if (leftSum==rightSum) {
            return left+1;
        } else {
            return -1;
        }
    }
}
```

Time Complexity: O (n)
Space Complexity: O (1)

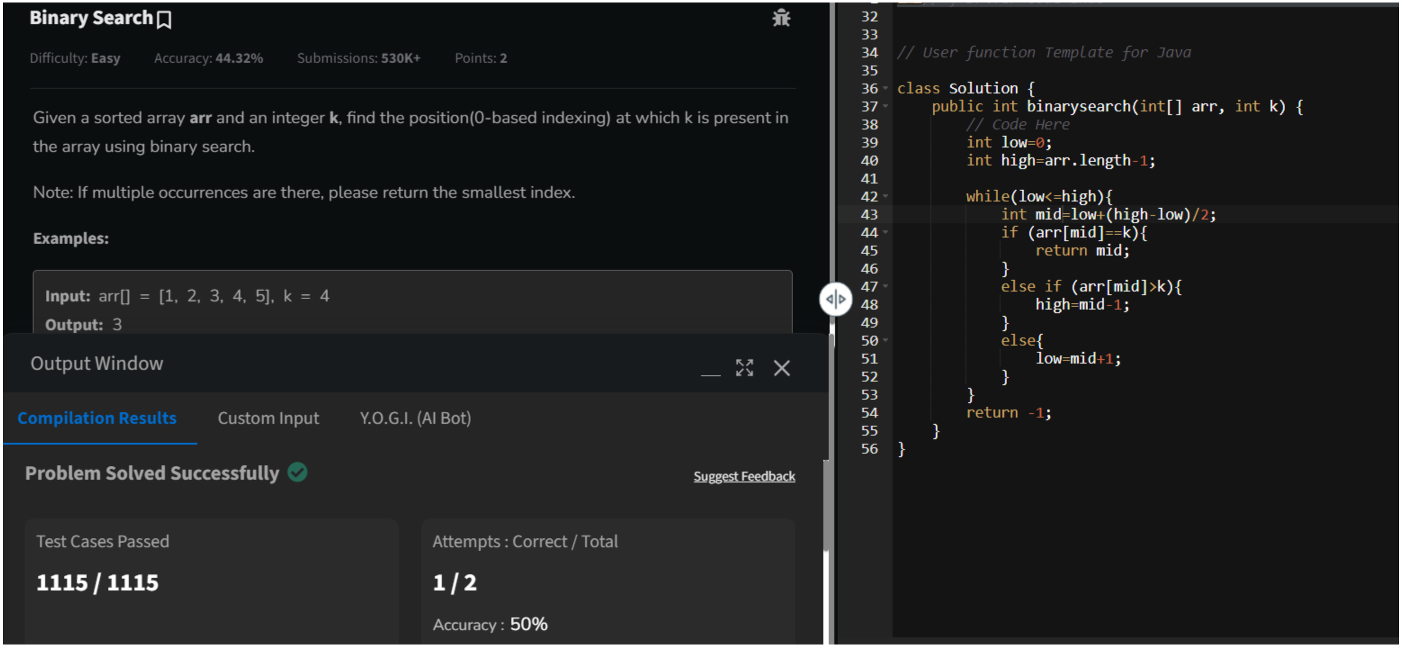
5. Binary Search

Code Solution:

```
class Solution {
    public int binarysearch(int[] arr, int k) {
        // Code Here
        int low=0;
        int high=arr.length-1;

        while(low<=high){
            int mid=low+(high-low)/2;
            if (arr[mid]==k){
                return mid;
            }
            else if (arr[mid]>k){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return -1;
    }
}
```

Output:



Binary Search

Difficulty: **Easy** Accuracy: **44.32%** Submissions: **530K+** Points: **2**

Given a sorted array **arr** and an integer **k**, find the position(0-based indexing) at which k is present in the array using binary search.

Note: If multiple occurrences are there, please return the smallest index.

Examples:

```
Input: arr[] = [1, 2, 3, 4, 5], k = 4
Output: 3
```

Output Window

Problem Solved Successfully ✓

Test Cases Passed

1115 / 1115

Attempts : Correct / Total

1 / 2

Accuracy : 50%

```
// User function Template for Java

class Solution {
    public int binarysearch(int[] arr, int k) {
        // Code Here
        int low=0;
        int high=arr.length-1;

        while(low<=high){
            int mid=low+(high-low)/2;
            if (arr[mid]==k){
                return mid;
            }
            else if (arr[mid]>k){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return -1;
    }
}
```

Time Complexity: O(logn)

Space Complexity: O (1)

6. Next Greater Element

Code Solution:

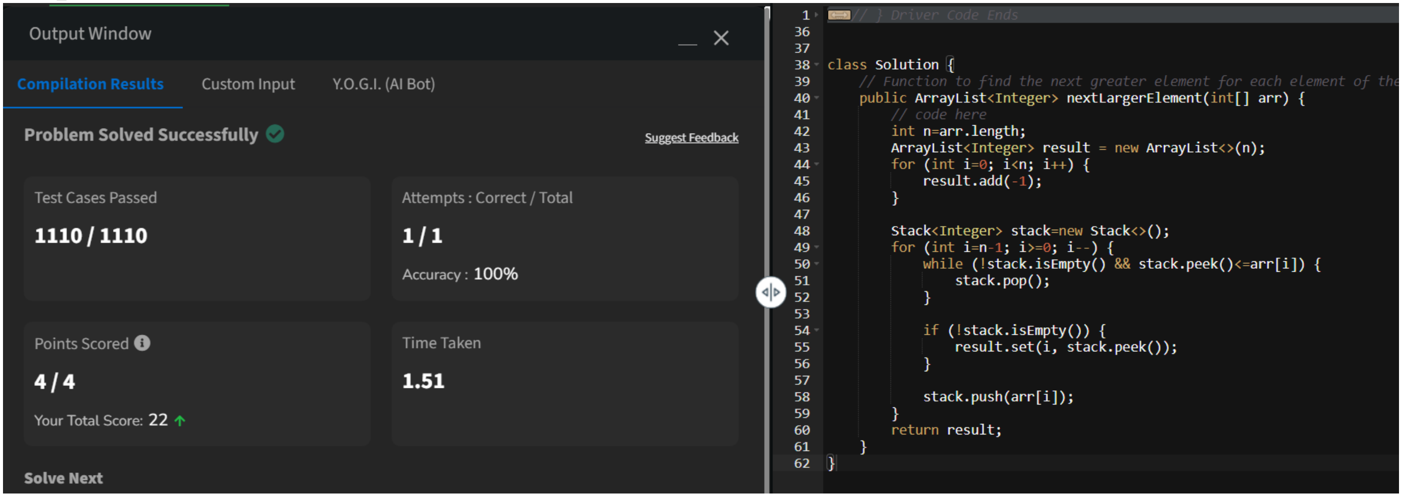
```
class Solution {
    // Function to find the next greater element for each element of the array.
    public ArrayList<Integer> nextLargerElement(int[] arr) {
        // code here
        int n=arr.length;
        ArrayList<Integer> result = new ArrayList<>(n);
        for (int i=0; i<n; i++) {
            result.add(-1);
        }

        Stack<Integer> stack=new Stack<>();
        for (int i=n-1; i>=0; i--) {
            while (!stack.isEmpty() && stack.peek()<=arr[i]) {
                stack.pop();
            }

            if (!stack.isEmpty()) {
                result.set(i, stack.peek());
            }

            stack.push(arr[i]);
        }
        return result;
    }
}
```

Output:



Problem Solved Successfully ✓

Test Cases Passed
1110 / 1110

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 22 ↑

Time Taken
1.51

```
// } Driver Code Ends


class Solution {
    // Function to find the next greater element for each element of the
    public ArrayList<Integer> nextLargerElement(int[] arr) {
        // code here
        int n=arr.length;
        ArrayList<Integer> result = new ArrayList<>(n);
        for (int i=0; i<n; i++) {
            result.add(-1);
        }

        Stack<Integer> stack=new Stack<>();
        for (int i=n-1; i>=0; i--) {
            while (!stack.isEmpty() && stack.peek()<=arr[i]) {
                stack.pop();
            }

            if (!stack.isEmpty()) {
                result.set(i, stack.peek());
            }

            stack.push(arr[i]);
        }
        return result;
    }
}
```

Time Complexity: O (n)

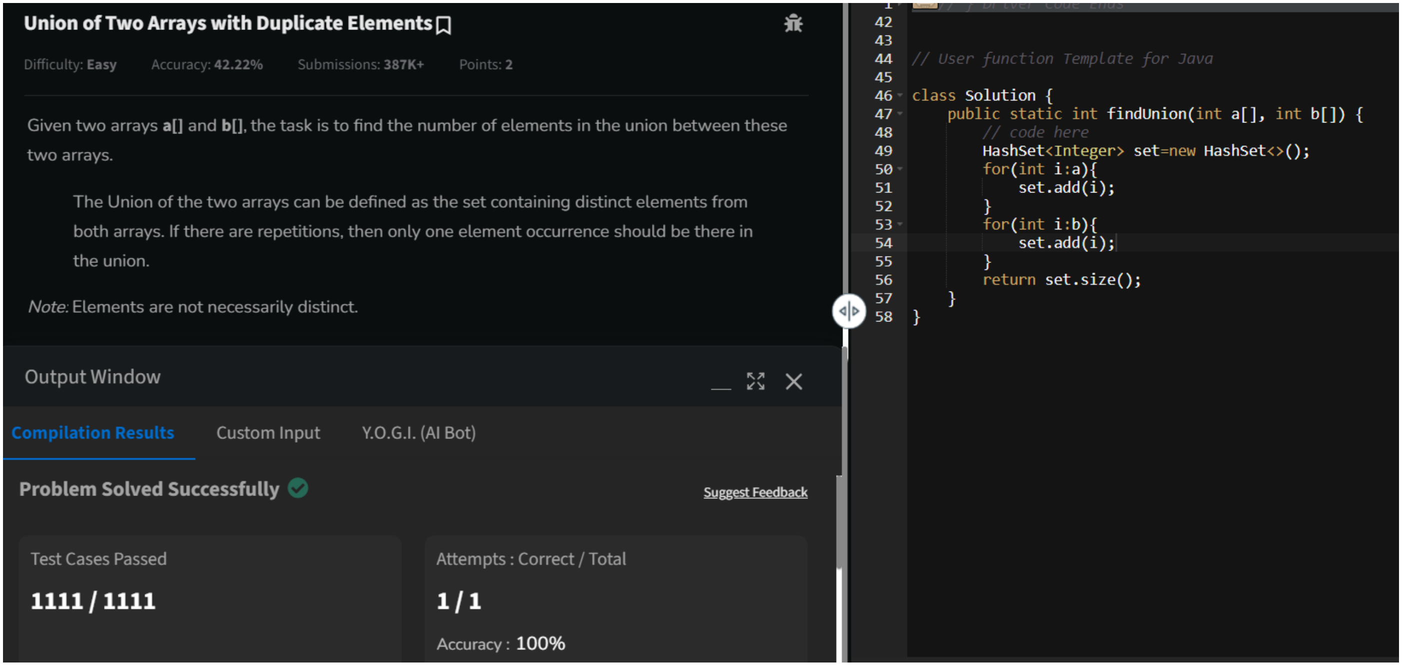
Space Complexity: O (n)

7. Union of 2 Arrays with Duplicate elements

Code Solution

```
class Solution {
    public static int findUnion(int a[], int b[]) {
        // code here
        HashSet<Integer> set=new HashSet<>();
        for(int i:a){
            set.add(i);
        }
        for(int i:b){
            set.add(i);
        }
        return set.size();
    }
}
```

Output:



Union of Two Arrays with Duplicate Elements

Difficulty: **Easy** Accuracy: **42.22%** Submissions: **387K+** Points: **2**

Given two arrays **a[]** and **b[]**, the task is to find the number of elements in the union between these two arrays.

> The Union of the two arrays can be defined as the set containing distinct elements from both arrays. If there are repetitions, then only one element occurrence should be there in the union.

Note: Elements are not necessarily distinct.

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully ✓

Suggest Feedback

Test Cases Passed

1111 / 1111

Attempts : Correct / Total

1 / 1

Accuracy : **100%**

```
// } Driver Code Ends


// User function Template for Java

class Solution {
    public static int findUnion(int a[], int b[]) {
        // code here
        HashSet<Integer> set=new HashSet<>();
        for(int i:a){
            set.add(i);
        }
        for(int i:b){
            set.add(i);
        }
        return set.size();
    }
}
```

Time Complexity: O (n+m)

Space Complexity: O (n+m)