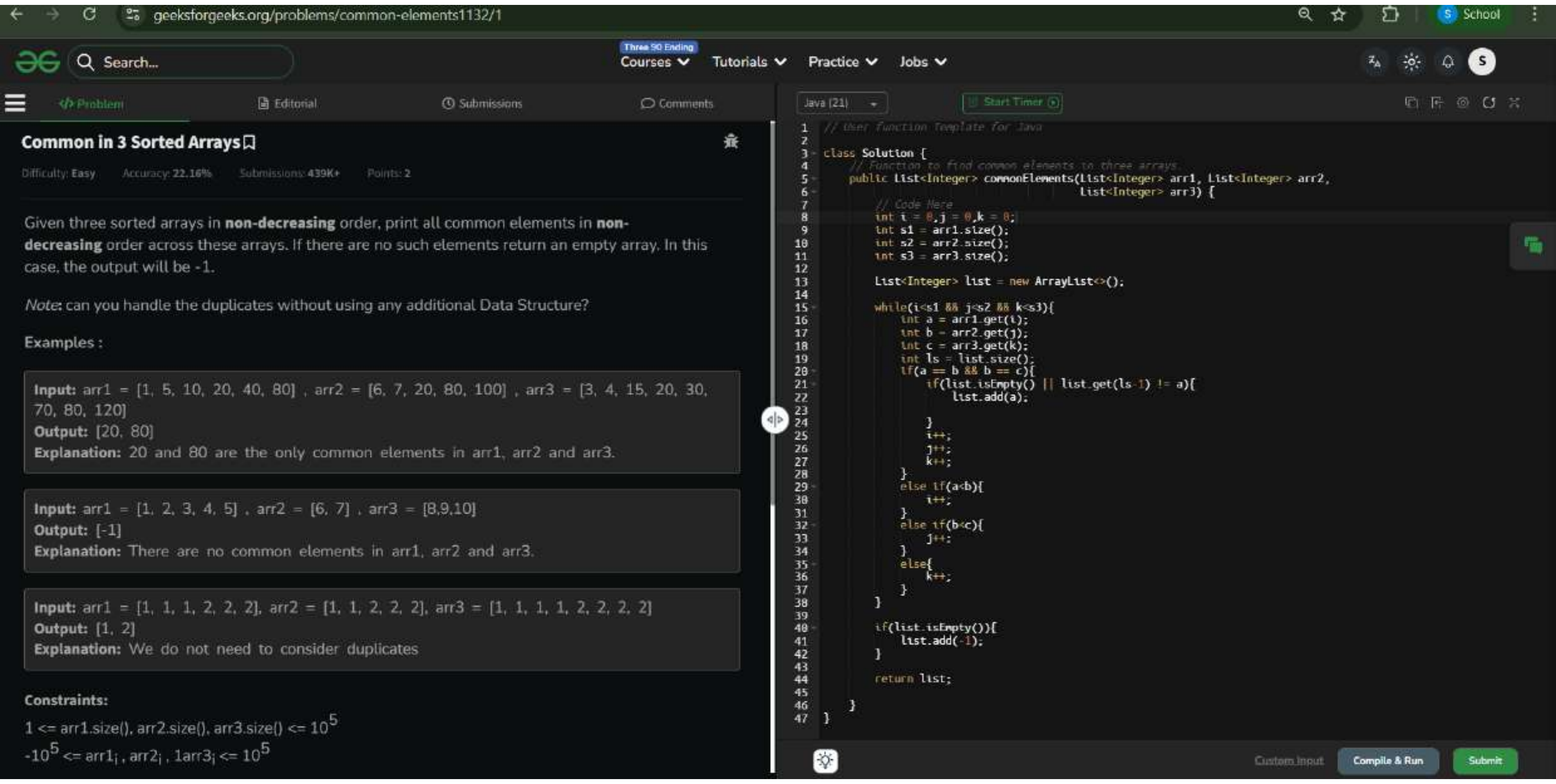


Common in 3 Sorted Arrays

Difficulty: **Easy** Accuracy: **22.16%** Submissions: **439K+** Points: **2**

Given three sorted arrays in **non-decreasing** order, print all common elements in **non-decreasing** order across these arrays. If there are no such elements return an empty array. In this case, the output will be -1.

Note: can you handle the duplicates without using any additional Data Structure?

Examples :

```
Input: arr1 = [1, 5, 10, 20, 40, 80] , arr2 = [6, 7, 20, 80, 100] , arr3 = [3, 4, 15, 20, 30, 70, 80, 120]
Output: [20, 80]
Explanation: 20 and 80 are the only common elements in arr1, arr2 and arr3.
```

```
Input: arr1 = [1, 2, 3, 4, 5] , arr2 = [6, 7] , arr3 = [8,9,10]
Output: [-1]
Explanation: There are no common elements in arr1, arr2 and arr3.
```

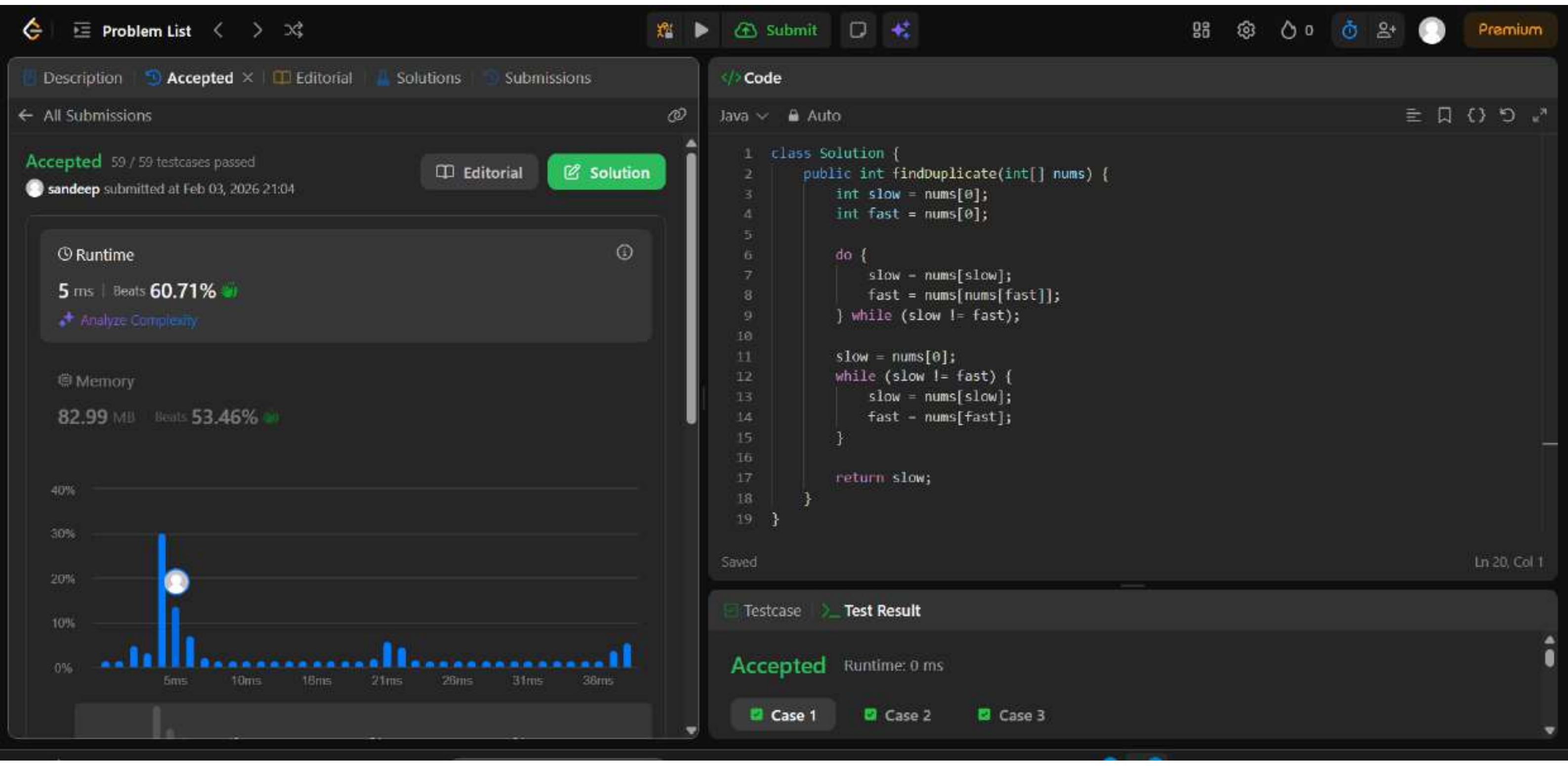
```
Input: arr1 = [1, 1, 1, 2, 2, 2], arr2 = [1, 1, 2, 2, 2], arr3 = [1, 1, 1, 1, 2, 2, 2, 2]
Output: [1, 2]
Explanation: We do not need to consider duplicates
```

Constraints:

$1 <= arr1.size(), arr2.size(), arr3.size() <= 10^5$

$-10^5 <= arr1_i , arr2_i , 1arr3_i <= 10^5$

```java
// User function Template for Java

class Solution {
    // Function to find common elements in three arrays.
    public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
                                        List<Integer> arr3) {
        // Code Here
        int i = 0,j = 0,k = 0;
        int s1 = arr1.size();
        int s2 = arr2.size();
        int s3 = arr3.size();

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

        while(i<s1 && j<s2 && k<s3){
            int a = arr1.get(i);
            int b = arr2.get(j);
            int c = arr3.get(k);
            int ls = list.size();
            if(a == b && b == c){
                if(list.isEmpty() || list.get(ls-1) != a){
                    list.add(a);

                }
                i++;
                j++;
                k++;
            }
            else if(a<b){
                i++;
            }
            else if(b<c){
                j++;
            }
            else{
                k++;
            }
        }

        if(list.isEmpty()){
            list.add(-1);
        }

        return list;

    }
}
```



Accepted 59 / 59 testcases passed
sandeep submitted at Feb 03, 2026 21:04

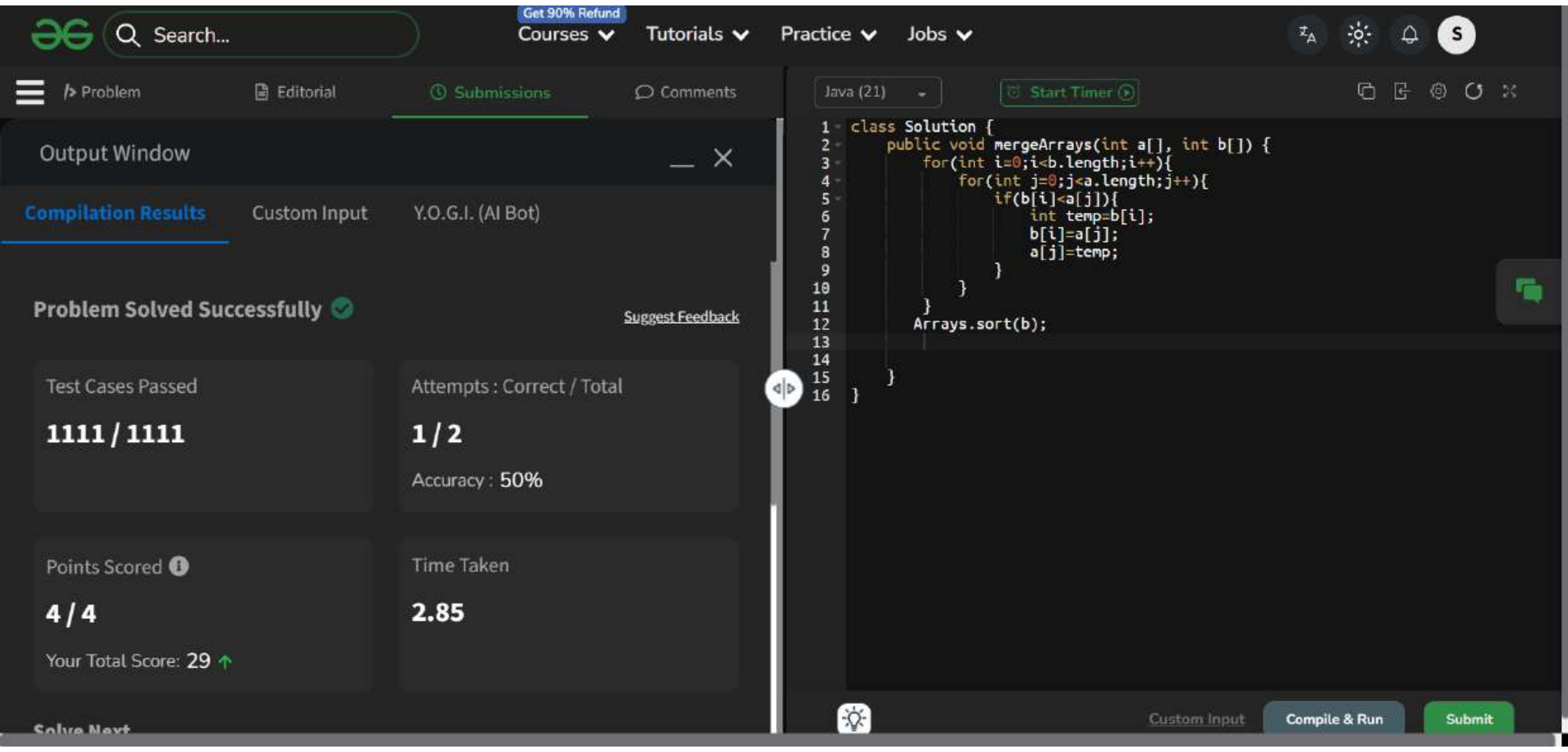
Runtime
5 ms | Beats 60.71%

Memory
82.99 MB Beats 53.46%

```java
class Solution {
    public int findDuplicate(int[] nums) {
        int slow = nums[0];
        int fast = nums[0];

        do {
            slow = nums[slow];
            fast = nums[nums[fast]];
        } while (slow != fast);

        slow = nums[0];
        while (slow != fast) {
            slow = nums[slow];
            fast = nums[fast];
        }

        return slow;
    }
}
```

Saved
Ln 20, Col 1

Accepted Runtime: 0 ms



Problem Solved Successfully

Test Cases Passed
1111 / 1111

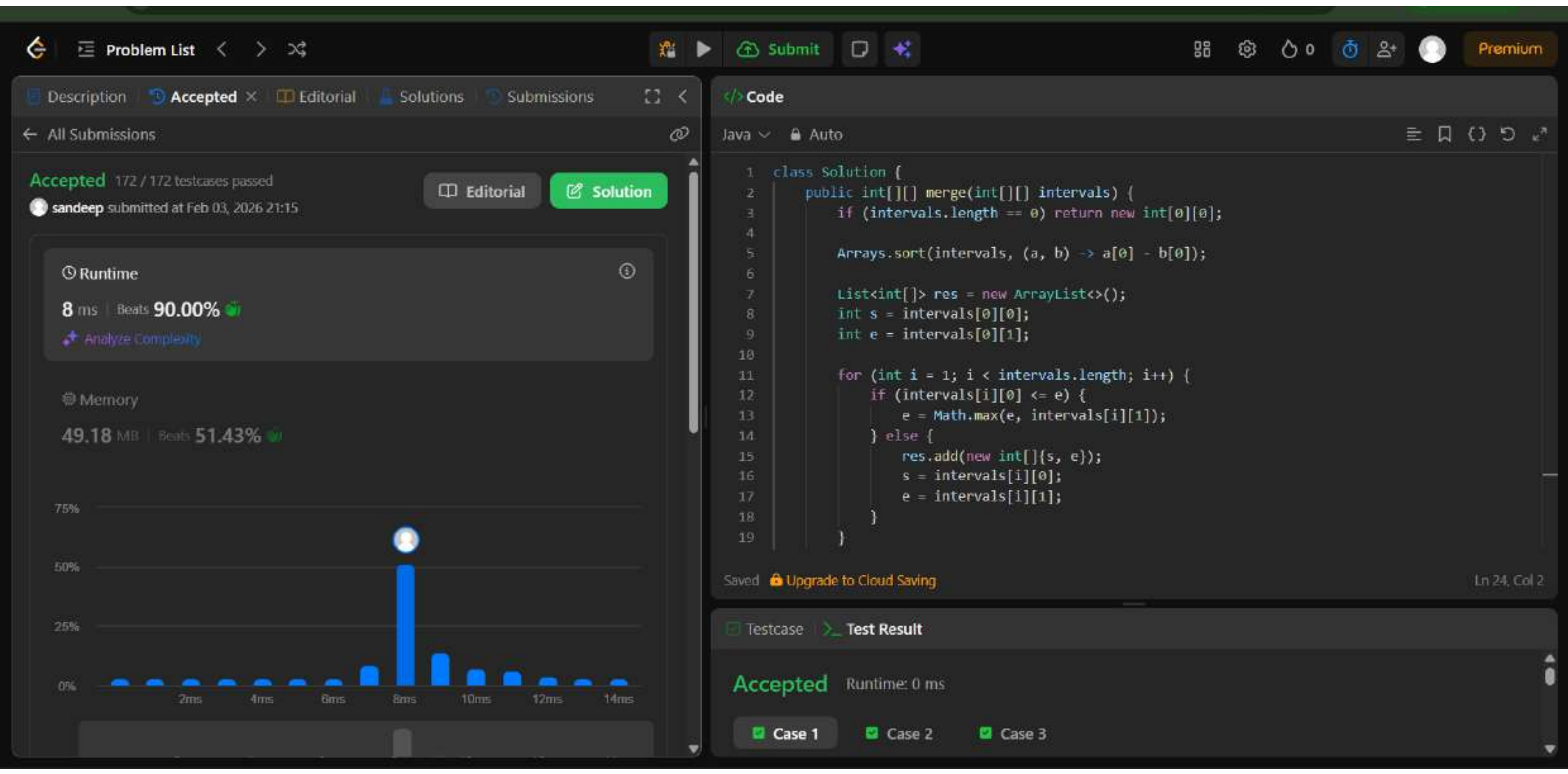
Attempts : Correct / Total
1 / 2
Accuracy : 50%

Points Scored
4 / 4
Your Total Score: 29

Time Taken
2.85

Solve Next

```java
class Solution {
    public void mergeArrays(int a[], int b[]) {
        for(int i=0;i<b.length;i++){
            for(int j=0;j<a.length;j++){
                if(b[i]<a[j]){
                    int temp=b[i];
                    b[i]=a[j];
                    a[j]=temp;
                }
            }
        }
       Arrays.sort(b);
        

    }
}
```



Accepted 172 / 172 testcases passed

sandeep submitted at Feb 03, 2026 21:15

Runtime

8 ms | Beats 90.00%

Analyze Complexity

Memory

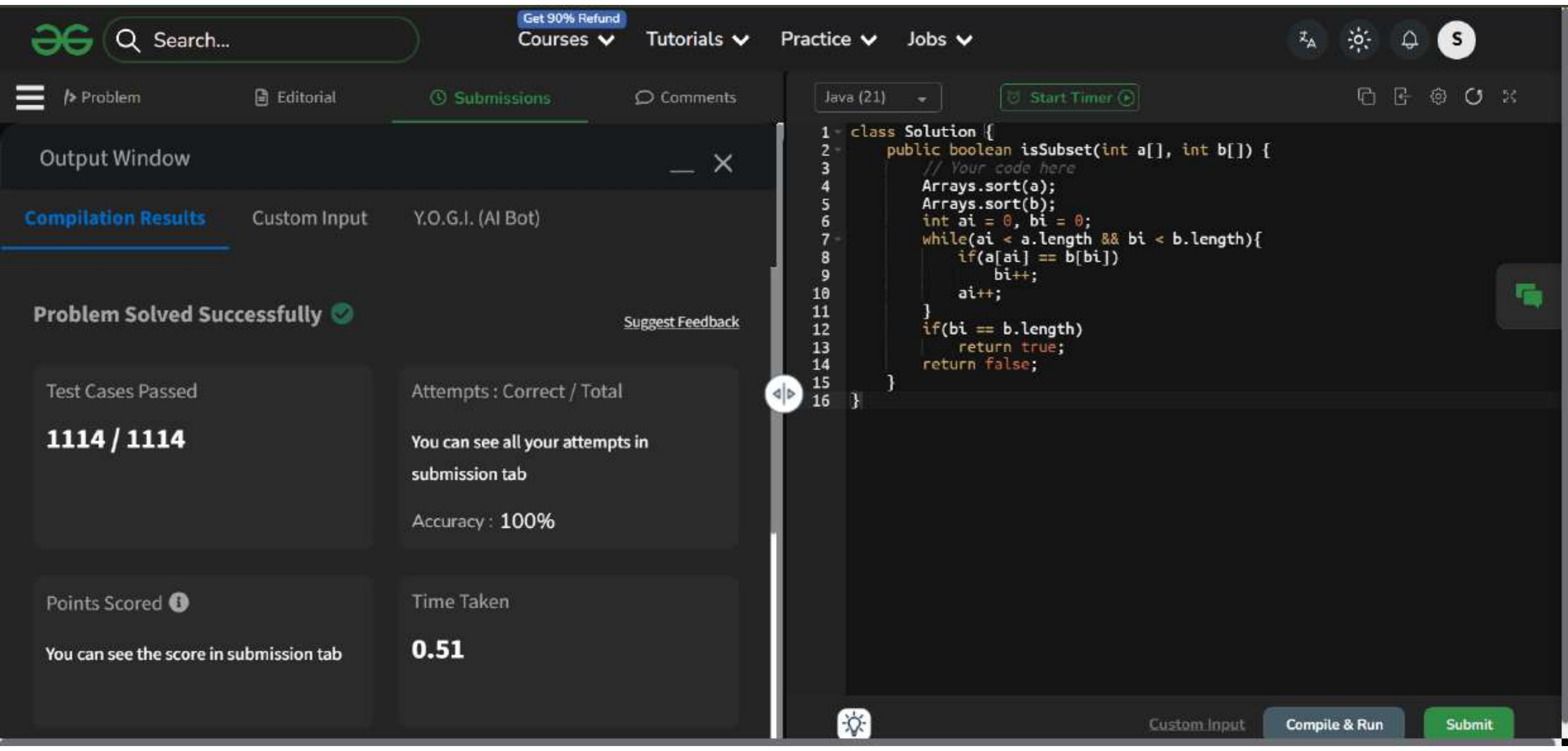
49.18 MB | Beats 51.43%

```java
class Solution {
    public int[][] merge(int[][] intervals) {
        if (intervals.length == 0) return new int[0][0];

        Arrays.sort(intervals, (a, b) -> a[0] - b[0]);

        List<int[]> res = new ArrayList<>();
        int s = intervals[0][0];
        int e = intervals[0][1];

        for (int i = 1; i < intervals.length; i++) {
            if (intervals[i][0] <= e) {
                e = Math.max(e, intervals[i][1]);
            } else {
                res.add(new int[]{s, e});
                s = intervals[i][0];
                e = intervals[i][1];
            }
        }
```

Saved Upgrade to Cloud Saving

Ln 24, Col 2

Accepted Runtime: 0 ms



Problem Solved Successfully

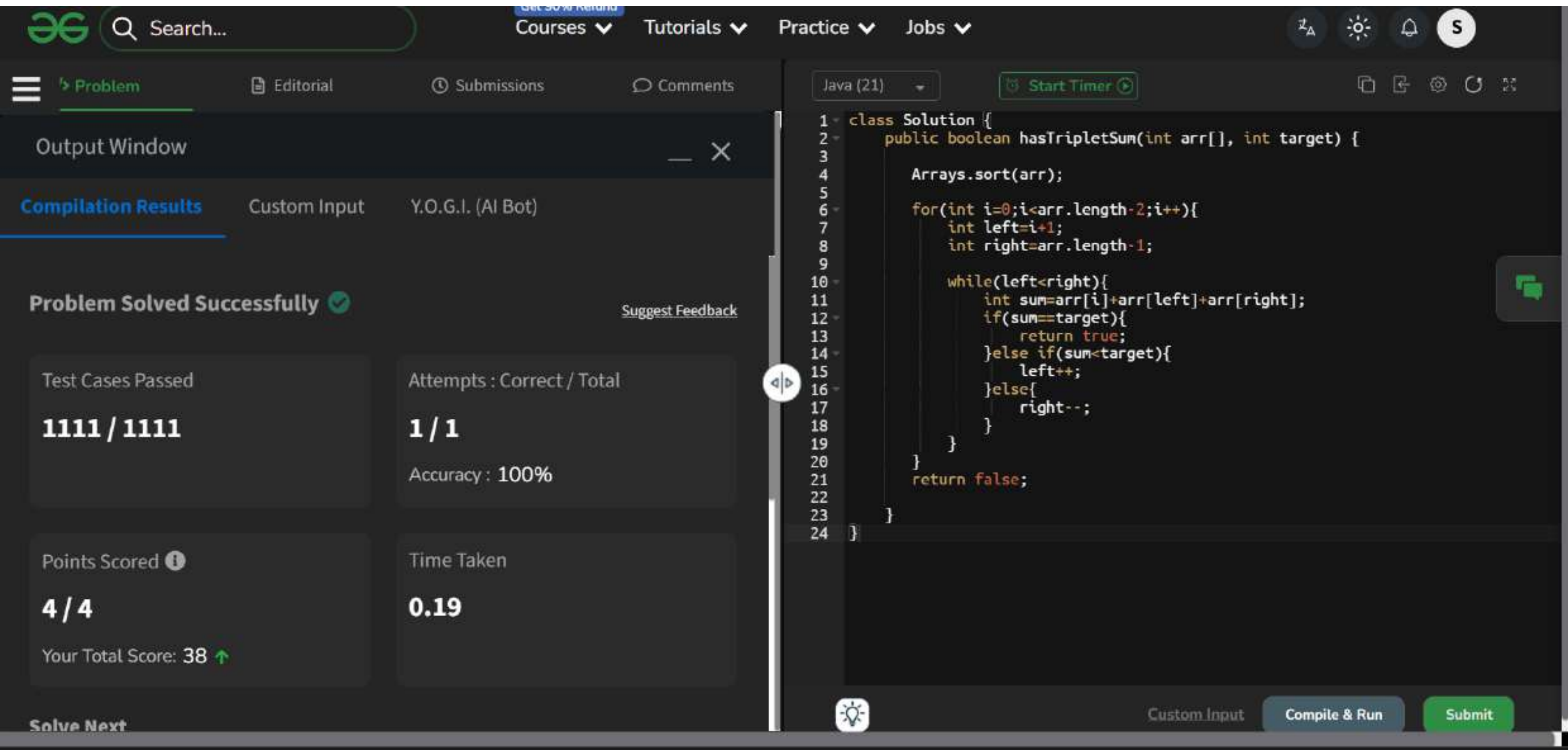
Test Cases Passed
1114 / 1114

Attempts : Correct / Total
You can see all your attempts in submission tab
Accuracy : 100%

Points Scored
You can see the score in submission tab

Time Taken
0.51

```java
class Solution {
    public boolean isSubset(int a[], int b[]) {
        // Your code here
        Arrays.sort(a);
        Arrays.sort(b);
        int ai = 0, bi = 0;
        while(ai < a.length && bi < b.length){
            if(a[ai] == b[bi])
                bi++;
            ai++;
        }
        if(bi == b.length)
            return true;
        return false;
    }
}
```



Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1111 / 1111

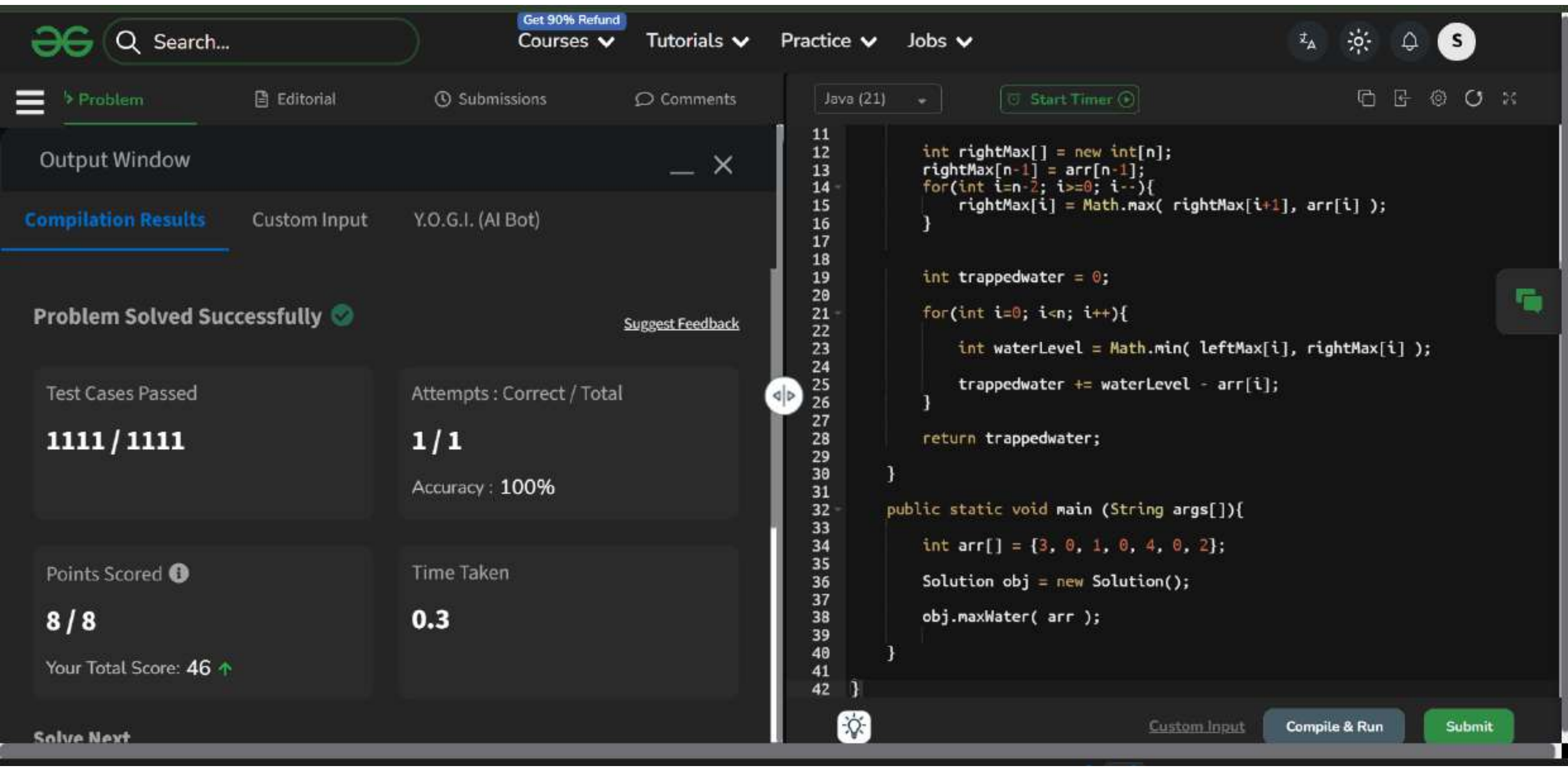
Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 38

Time Taken
0.19

Solve Next

```java
class Solution {
    public boolean hasTripletSum(int arr[], int target) {

       Arrays.sort(arr);

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

           while(left<right){
               int sum=arr[i]+arr[left]+arr[right];
               if(sum==target){
                   return true;
               }else if(sum<target){
                   left++;
               }else{
                   right--;
               }
           }
       }
       return false;

    }
}
```



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%

Points Scored
8 / 8
Your Total Score: 46

Time Taken
0.3

Solve Next

```java

        int rightMax[] = new int[n];
        rightMax[n-1] = arr[n-1];
        for(int i=n-2; i>=0; i--){
            rightMax[i] = Math.max( rightMax[i+1], arr[i] );
        }


        int trappedwater = 0;

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

            int waterLevel = Math.min( leftMax[i], rightMax[i] );

            trappedwater += waterLevel - arr[i];
        }

        return trappedwater;

    }

    public static void main (String args[]){

        int arr[] = {3, 0, 1, 0, 4, 0, 2};

        Solution obj = new Solution();

        obj.maxWater( arr );

    }

}
```