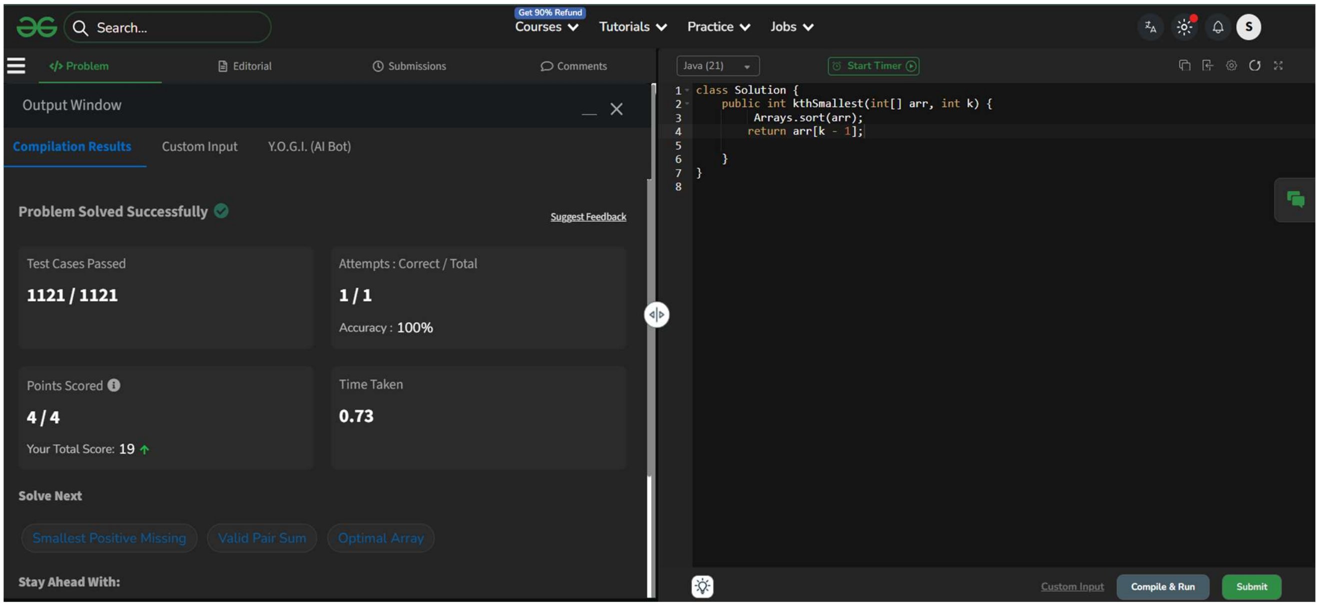


Problem Solved Successfully ✓

Test Cases Passed
1121 / 1121

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

Points Scored
4 / 4
Your Total Score: 19 ↑

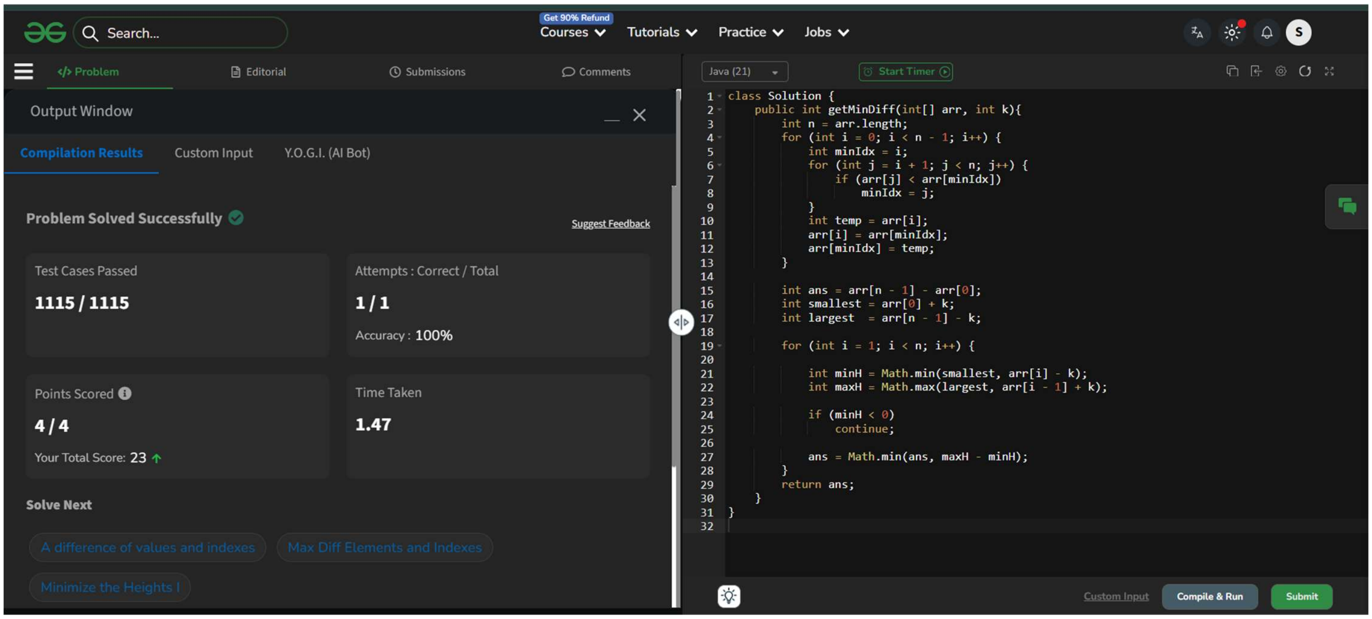
Time Taken
0.73

Solve Next

Smallest Positive Missing | Valid Pair Sum | Optimal Array

Stay Ahead With:

```java
class Solution {
    public int kthSmallest(int[] arr, int k) {
         Arrays.sort(arr);
        return arr[k - 1];

    }
}

```



Problem Solved Successfully ✓

Test Cases Passed
1115 / 1115

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

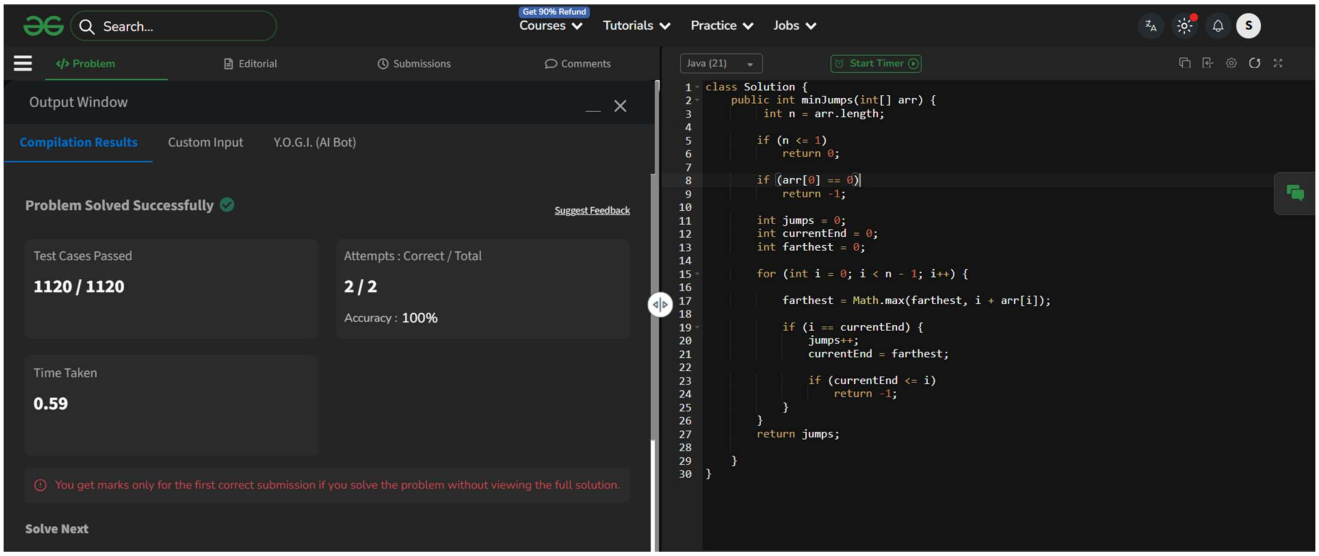
Points Scored
4 / 4
Your Total Score: 23 ↑

Time Taken
1.47

Solve Next

A difference of values and indexes | Max Diff Elements and Indexes | Minimize the Heights I

```java
class Solution {
    public int getMinDiff(int[] arr, int k){
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            int minIdx = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j] < arr[minIdx])
                    minIdx = j;
            }
            int temp = arr[i];
            arr[i] = arr[minIdx];
            arr[minIdx] = temp;
        }

        int ans = arr[n - 1] - arr[0];
        int smallest = arr[0] + k;
        int largest  = arr[n - 1] - k;

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

            int minH = Math.min(smallest, arr[i] - k);
            int maxH = Math.max(largest, arr[i - 1] + k);

            if (minH < 0)
                continue;

            ans = Math.min(ans, maxH - minH);
        }
        return ans;
    }
}

```



Problem Solved Successfully ✓

Test Cases Passed
1120 / 1120

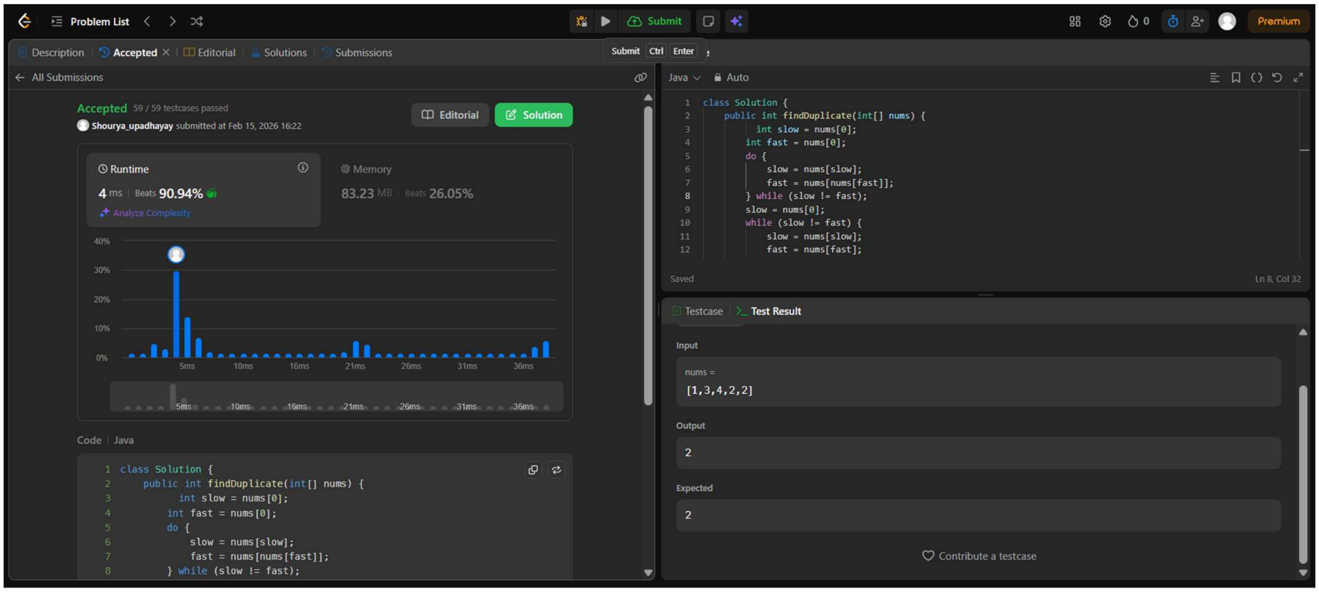
Attempts : Correct / Total
2 / 2
Accuracy : 100%

Time Taken
0.59

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

Solve Next

```java
class Solution {
    public int minJumps(int[] arr) {
         int n = arr.length;

        if (n <= 1)
            return 0;

        if (arr[0] == 0)
            return -1;

        int jumps = 0;
        int currentEnd = 0;
        int farthest = 0;

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

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

            if (i == currentEnd) {
                jumps++;
                currentEnd = farthest;

                if (currentEnd <= i)
                    return -1;
            }
        }
        return jumps;

    }
}
```



Accepted 59 / 59 testcases passed

Shourya_upadhayay submitted at Feb 15, 2026 16:22

Runtime

4 ms | Beats 90.94%

Memory

83.23 MB | Beats 26.05%

Code | Java

```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);
```

```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];
```

Saved

Ln 8, Col 32

Input

nums =

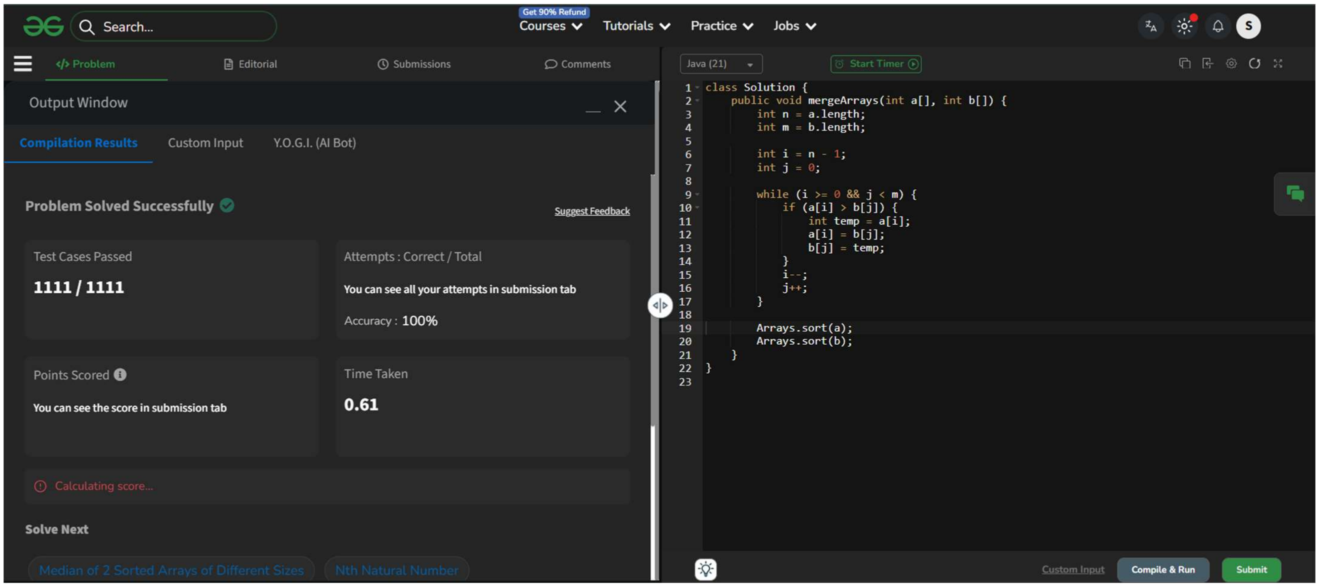
```
[1,3,4,2,2]
```

Output

```
2
```

Expected

```
2
```



Output Window

Problem Solved Successfully

Test Cases Passed

1111 / 1111

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

0.61

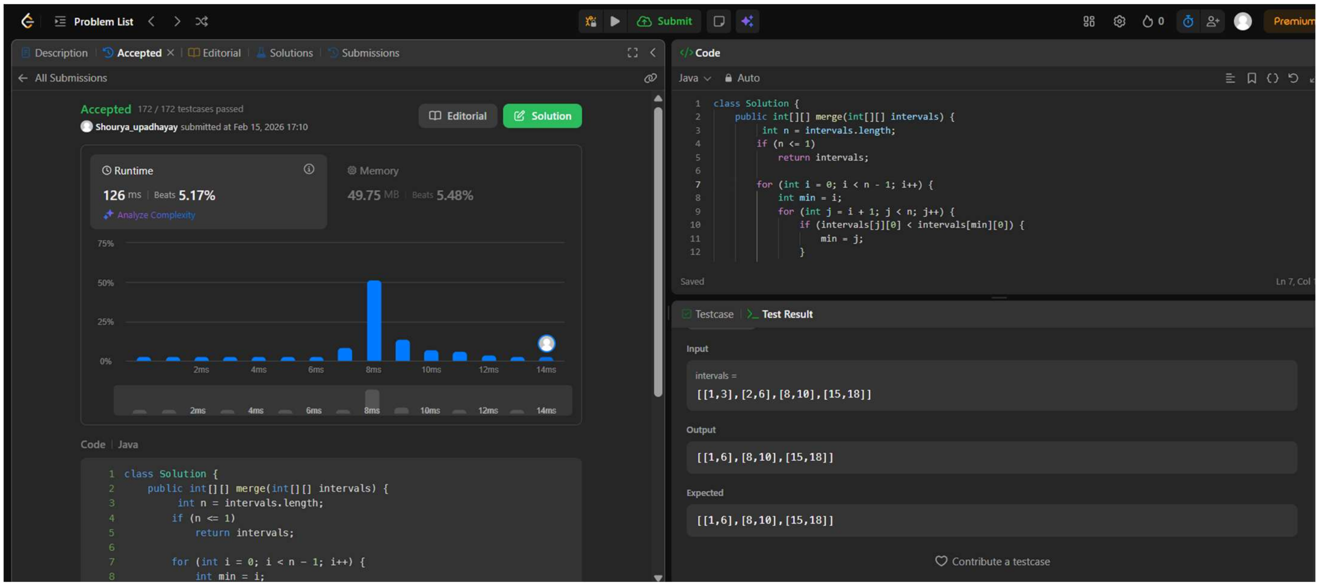
Calculating score...

Solve Next

Median of 2 Sorted Arrays of Different Sizes

Nth Natural Number

```java
class Solution {
    public void mergeArrays(int a[], int b[]) {
        int n = a.length;
        int m = b.length;

        int i = n - 1;
        int j = 0;

        while (i >= 0 && j < m) {
            if (a[i] > b[j]) {
                int temp = a[i];
                a[i] = b[j];
                b[j] = temp;
            }
            i--;
            j++;
        }

        Arrays.sort(a);
        Arrays.sort(b);
    }
}
```



Accepted 172 / 172 testcases passed

Shourya_upadhayay submitted at Feb 15, 2026 17:10

Runtime

126 ms | Beats 5.17%

Memory

49.75 MB | Beats 5.48%

Code | Java

```java
class Solution {
    public int[][] merge(int[][] intervals) {
        int n = intervals.length;
        if (n <= 1)
            return intervals;

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

```java
class Solution {
    public int[][] merge(int[][] intervals) {
        int n = intervals.length;
        if (n <= 1)
            return intervals;

        for (int i = 0; i < n - 1; i++) {
            int min = i;
            for (int j = i + 1; j < n; j++) {
                if (intervals[j][0] < intervals[min][0]) {
                    min = j;
                }
```

Saved

Ln 7, Col

Input

intervals =

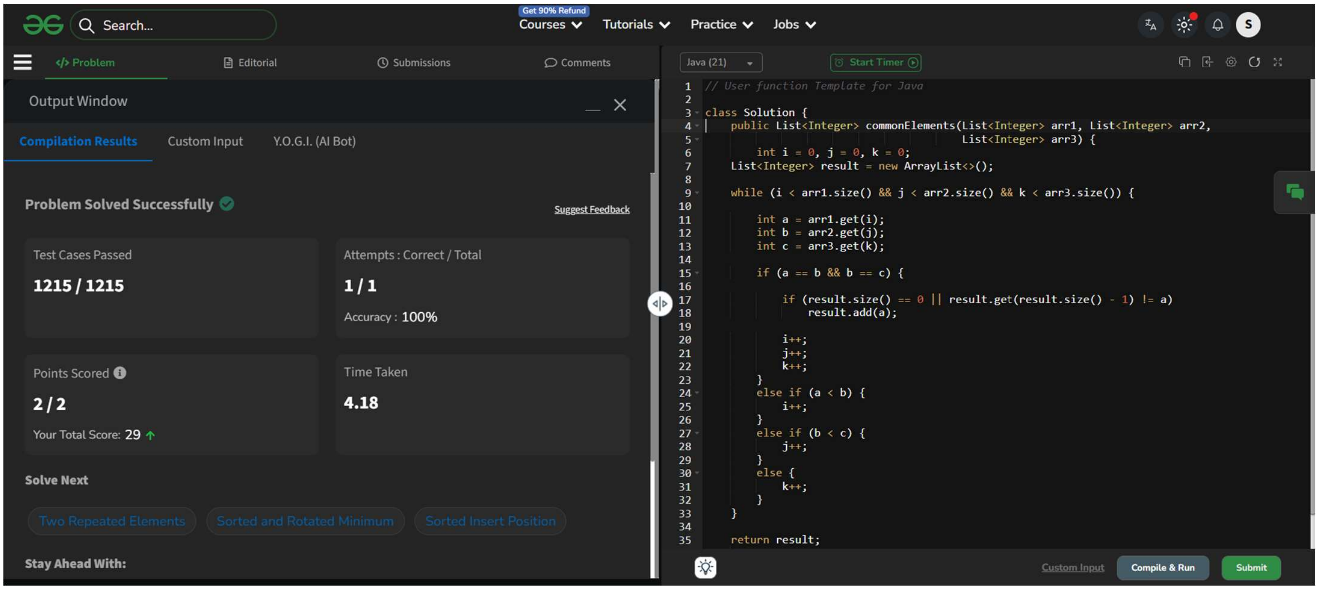
```
[[1,3],[2,6],[8,10],[15,18]]
```

Output

```
[[1,6],[8,10],[15,18]]
```

Expected

```
[[1,6],[8,10],[15,18]]
```



Problem Solved Successfully

Test Cases Passed
1215 / 1215

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

Points Scored
2 / 2
Your Total Score: 29

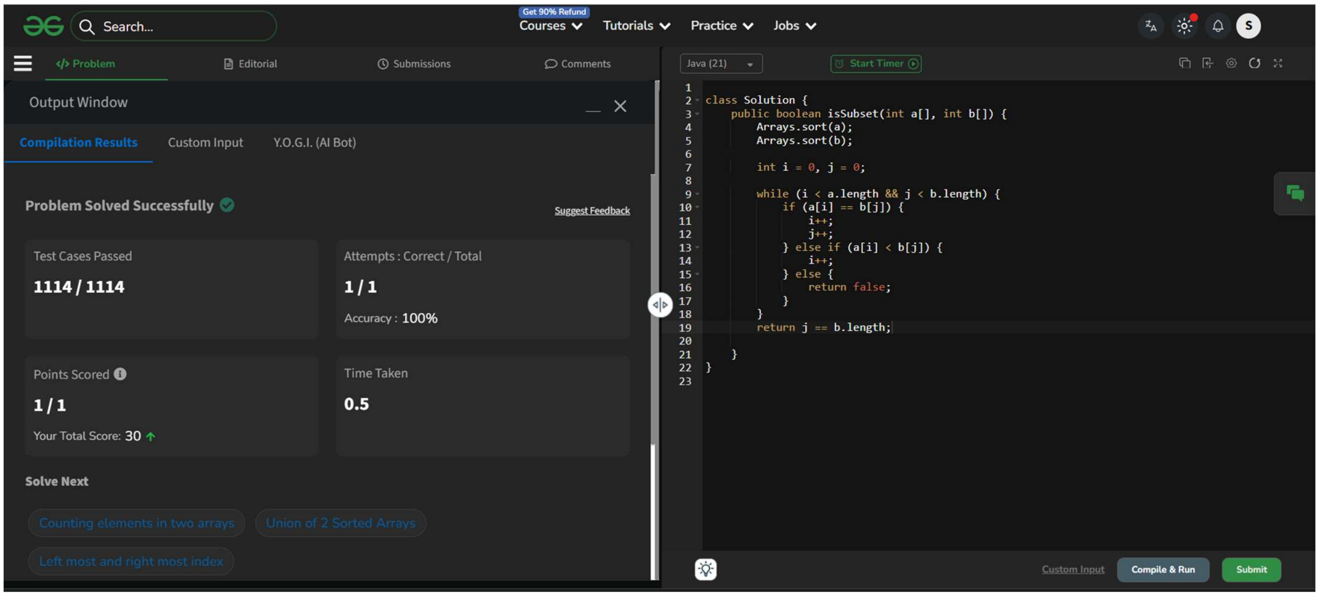
Time Taken
4.18

Solve Next

Two Repeated Elements · Sorted and Rotated Minimum · Sorted Insert Position

Stay Ahead With:

```java
// User function Template for Java

class Solution {
    public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
                                        List<Integer> arr3) {
        int i = 0, j = 0, k = 0;
    List<Integer> result = new ArrayList<>();

    while (i < arr1.size() && j < arr2.size() && k < arr3.size()) {

        int a = arr1.get(i);
        int b = arr2.get(j);
        int c = arr3.get(k);

        if (a == b && b == c) {

            if (result.size() == 0 || result.get(result.size() - 1) != a)
                result.add(a);

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

    return result;
```



Problem Solved Successfully

Test Cases Passed
1114 / 1114

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

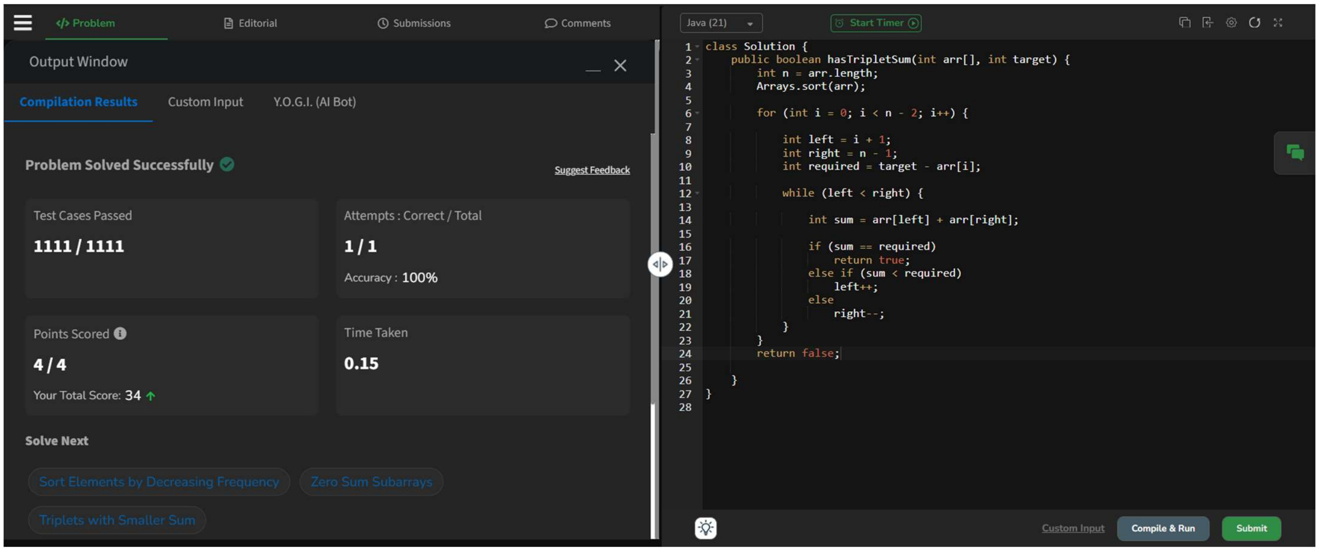
Points Scored
1 / 1
Your Total Score: 30

Time Taken
0.5

Solve Next

Counting elements in two arrays · Union of 2 Sorted Arrays · Left most and right most index

```java

class Solution {
    public boolean isSubset(int a[], int b[]) {
        Arrays.sort(a);
        Arrays.sort(b);

        int i = 0, j = 0;

        while (i < a.length && j < b.length) {
            if (a[i] == b[j]) {
                i++;
                j++;
            } else if (a[i] < b[j]) {
                i++;
            } else {
                return false;
            }
        }
        return j == b.length;

    }
}

```



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

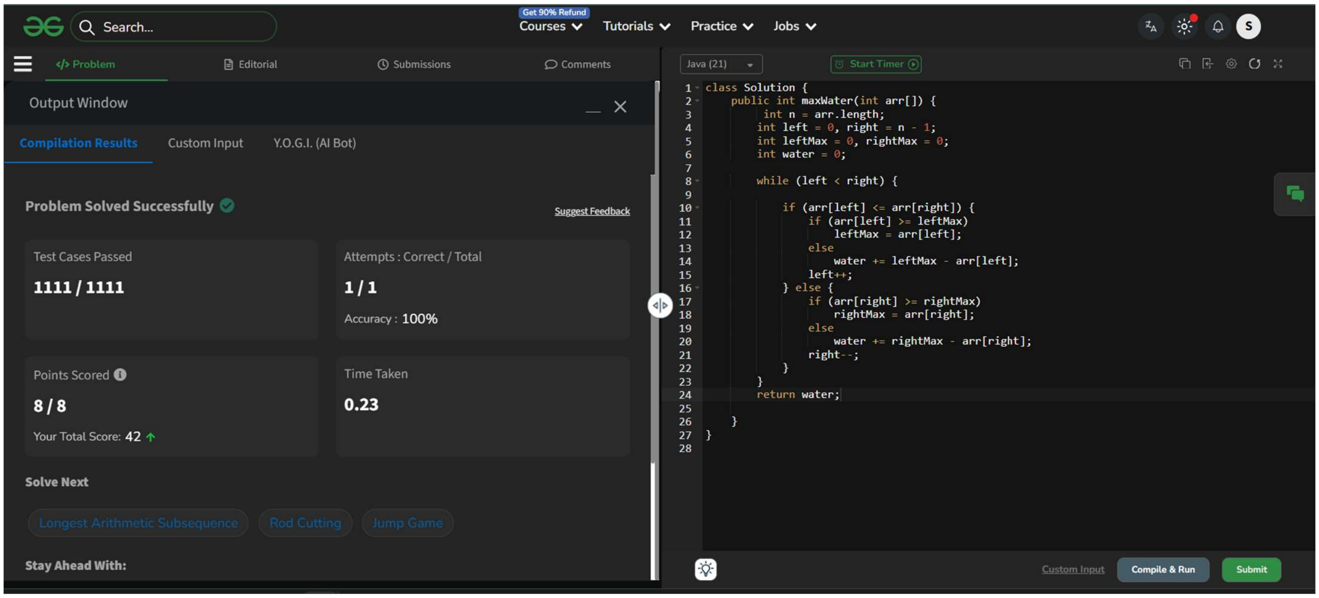
Points Scored
4 / 4
Your Total Score: 34

Time Taken
0.15

Solve Next

Sort Elements by Decreasing Frequency
Zero Sum Subarrays
Triplets with Smaller Sum

```java
class Solution {
    public boolean hasTripletSum(int arr[], int target) {
        int n = arr.length;
        Arrays.sort(arr);

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

            int left = i + 1;
            int right = n - 1;
            int required = target - arr[i];

            while (left < right) {

                int sum = arr[left] + arr[right];

                if (sum == required)
                    return true;
                else if (sum < required)
                    left++;
                else
                    right--;
            }
        }
        return false;

    }
}
```



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

Points Scored
8 / 8
Your Total Score: 42

Time Taken
0.23

Solve Next

Longest Arithmetic Subsequence
Rod Cutting
Jump Game

Stay Ahead With:

```java
class Solution {
    public int maxWater(int arr[]) {
         int n = arr.length;
        int left = 0, right = n - 1;
        int leftMax = 0, rightMax = 0;
        int water = 0;

        while (left < right) {

            if (arr[left] <= arr[right]) {
                if (arr[left] >= leftMax)
                    leftMax = arr[left];
                else
                    water += leftMax - arr[left];
                left++;
            } else {
                if (arr[right] >= rightMax)
                    rightMax = arr[right];
                else
                    water += rightMax - arr[right];
                right--;
            }
        }
        return water;

    }
}
```