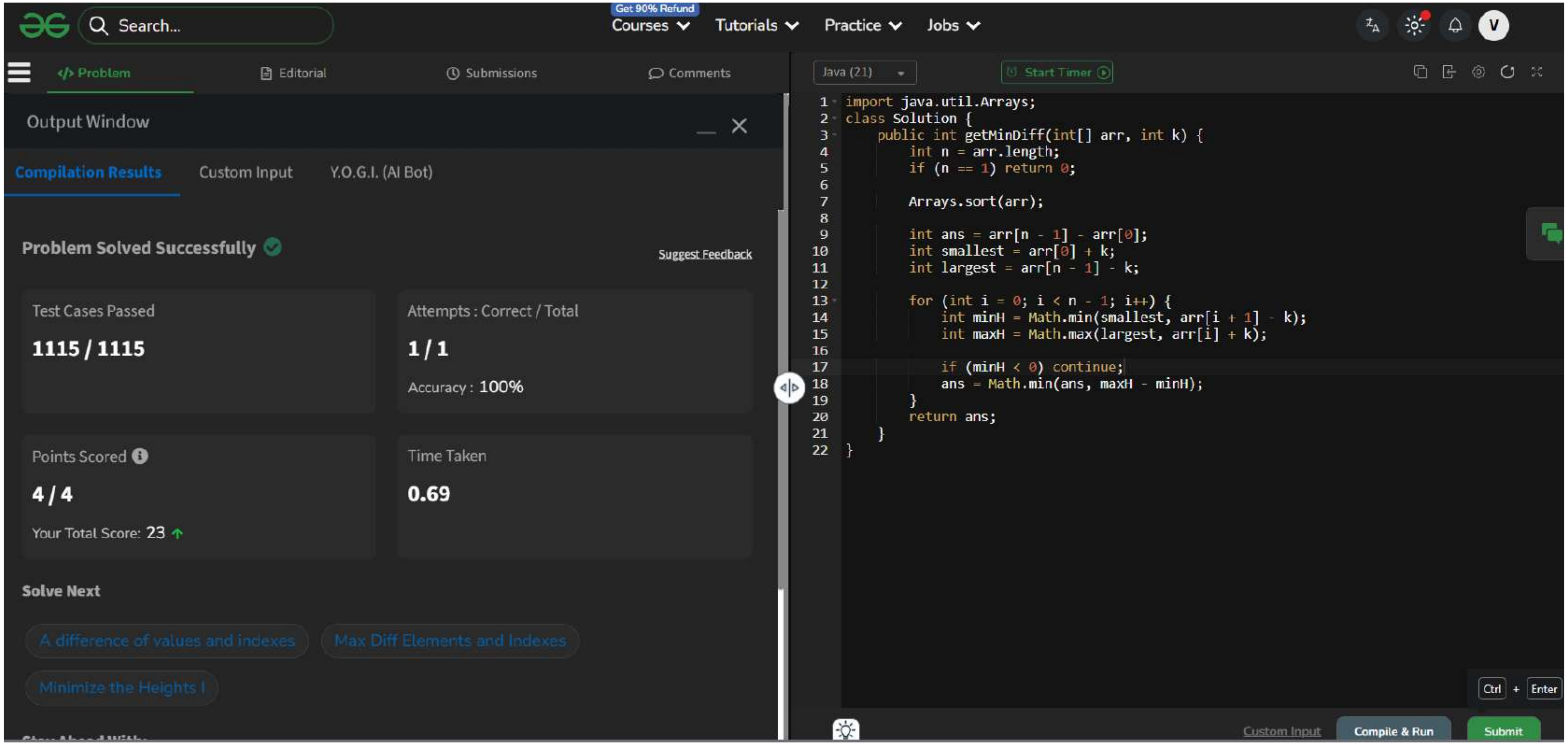


Problem Solved Successfully

Test Cases Passed
1115 / 1115

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

Points Scored
4 / 4
Your Total Score: 23

Time Taken
0.69

Solve Next

A difference of values and indexes

Max Diff Elements and Indexes

Minimize the Heights I

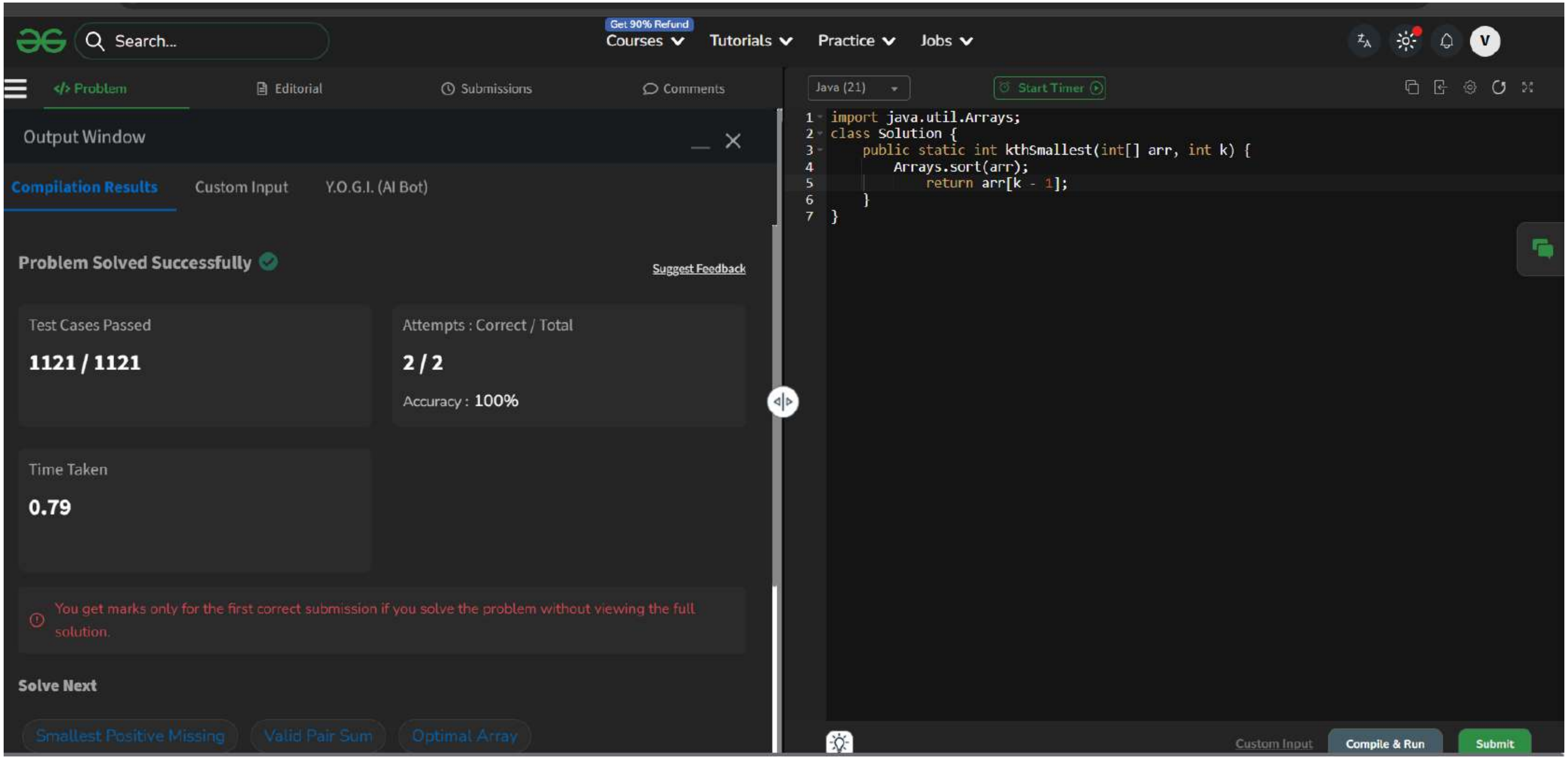
```java
import java.util.Arrays;
class Solution {
    public int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        if (n == 1) return 0;

        Arrays.sort(arr);

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

        for (int i = 0; i < n - 1; i++) {
            int minH = Math.min(smallest, arr[i + 1] - k);
            int maxH = Math.max(largest, arr[i] + k);

            if (minH < 0) continue;
            ans = Math.min(ans, maxH - minH);
        }
        return ans;
    }
}
```



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

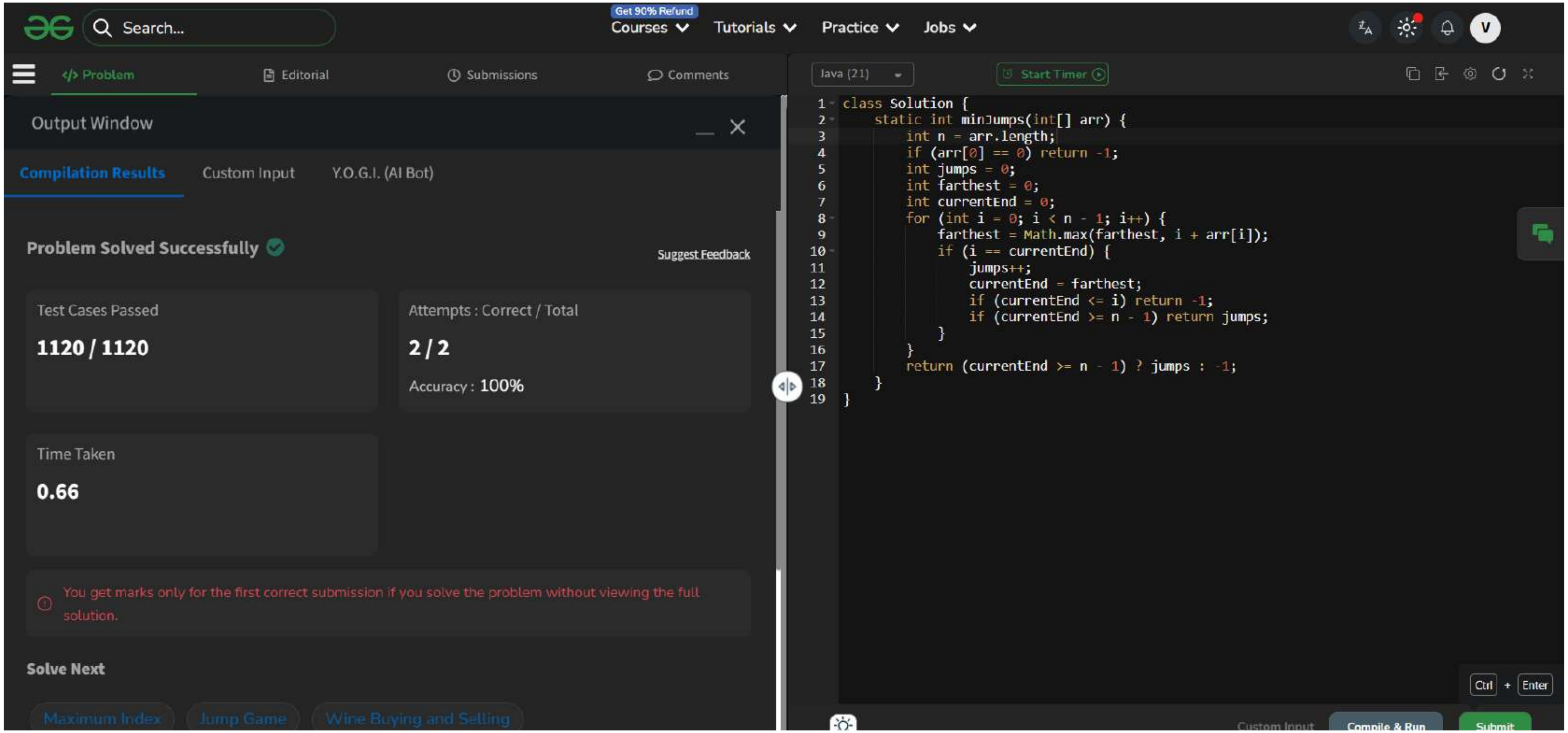
Problem Solved Successfully

Test Cases Passed
1121 / 1121

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

Time Taken
0.79

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



Problem Solved Successfully

Test Cases Passed

1120 / 1120

Attempts : Correct / Total

2 / 2

Accuracy : 100%

Time Taken

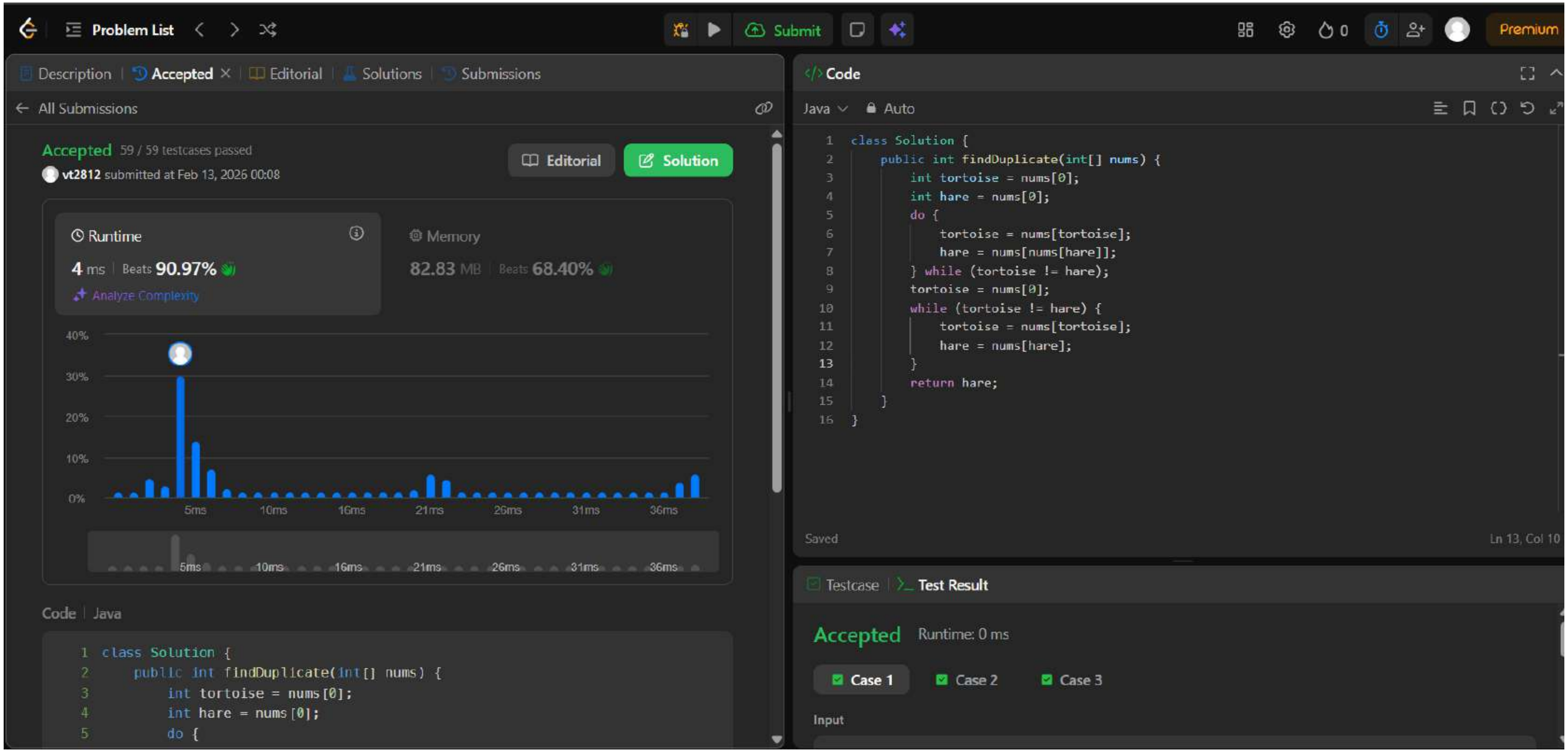
0.66

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

Solve Next

Maximum Index · Jump Game · Wine Buying and Selling

```java
class Solution {
    static int minJumps(int[] arr) {
        int n = arr.length;
        if (arr[0] == 0) return -1;
        int jumps = 0;
        int farthest = 0;
        int currentEnd = 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;
                if (currentEnd >= n - 1) return jumps;
            }
        }
        return (currentEnd >= n - 1) ? jumps : -1;
    }
}
```



Accepted 59 / 59 testcases passed

vt2812 submitted at Feb 13, 2026 00:08

Runtime

4 ms | Beats 90.97%

Memory

82.83 MB | Beats 68.40%

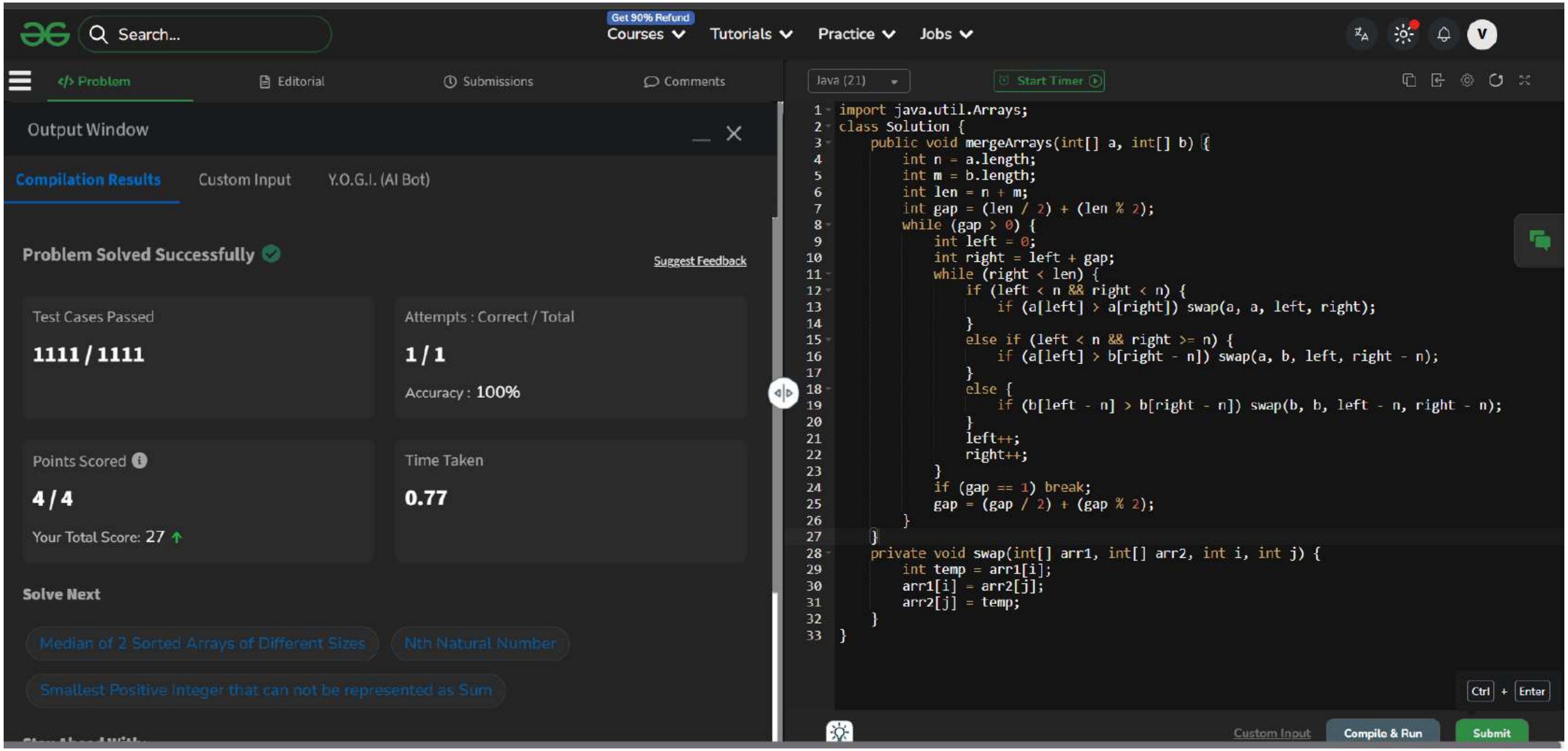
Code | Java

```java
class Solution {
    public int findDuplicate(int[] nums) {
        int tortoise = nums[0];
        int hare = nums[0];
        do {
```

```java
class Solution {
    public int findDuplicate(int[] nums) {
        int tortoise = nums[0];
        int hare = nums[0];
        do {
            tortoise = nums[tortoise];
            hare = nums[nums[hare]];
        } while (tortoise != hare);
        tortoise = nums[0];
        while (tortoise != hare) {
            tortoise = nums[tortoise];
            hare = nums[hare];
        }
        return hare;
    }
}
```

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3



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
4 / 4
Your Total Score: 27

Time Taken
0.77

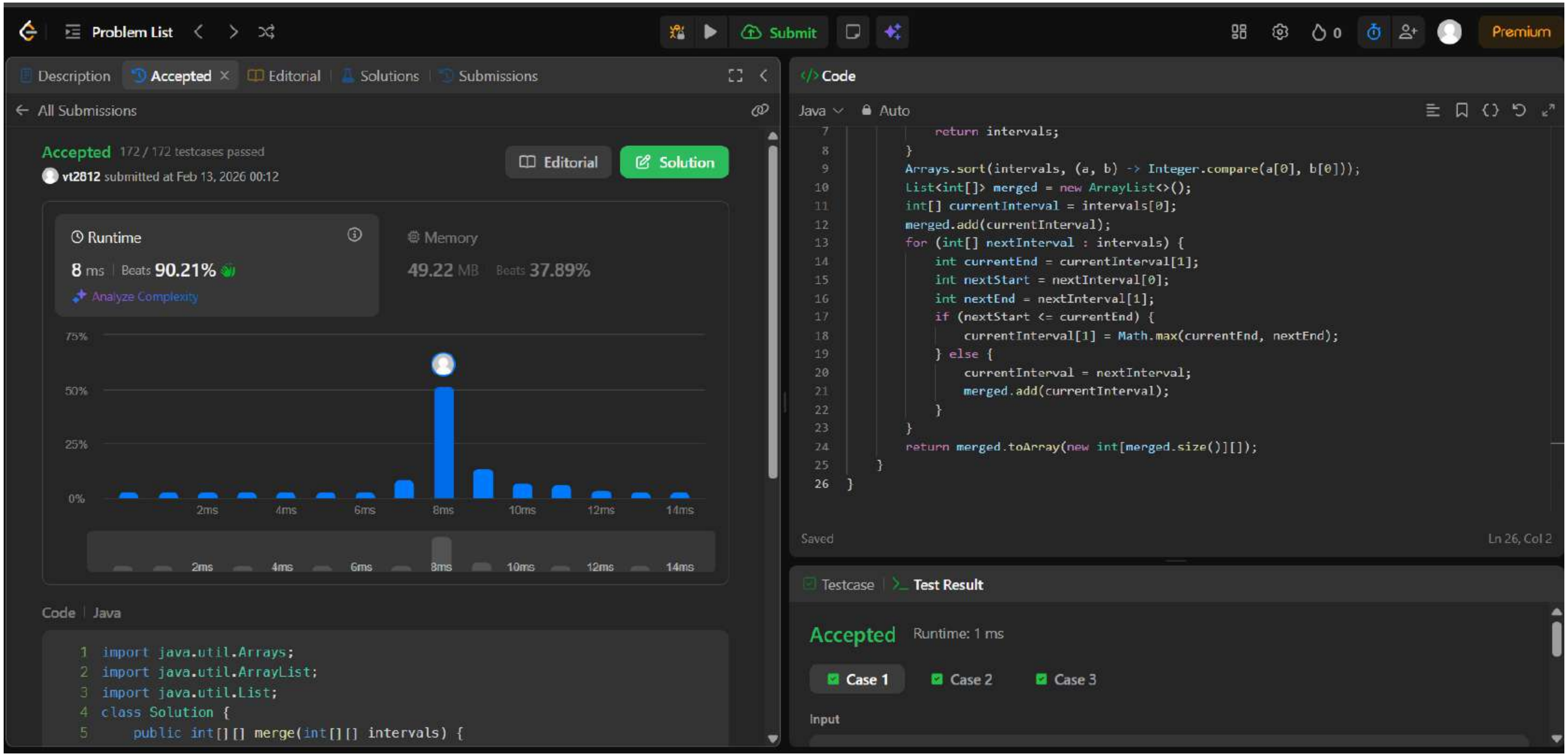
Solve Next

Median of 2 Sorted Arrays of Different Sizes

Nth Natural Number

Smallest Positive Integer that can not be represented as Sum

```java
import java.util.Arrays;
class Solution {
    public void mergeArrays(int[] a, int[] b) {
        int n = a.length;
        int m = b.length;
        int len = n + m;
        int gap = (len / 2) + (len % 2);
        while (gap > 0) {
            int left = 0;
            int right = left + gap;
            while (right < len) {
                if (left < n && right < n) {
                    if (a[left] > a[right]) swap(a, a, left, right);
                }
                else if (left < n && right >= n) {
                    if (a[left] > b[right - n]) swap(a, b, left, right - n);
                }
                else {
                    if (b[left - n] > b[right - n]) swap(b, b, left - n, right - n);
                }
                left++;
                right++;
            }
            if (gap == 1) break;
            gap = (gap / 2) + (gap % 2);
        }
    }
    private void swap(int[] arr1, int[] arr2, int i, int j) {
        int temp = arr1[i];
        arr1[i] = arr2[j];
        arr2[j] = temp;
    }
}
```



Accepted 172 / 172 testcases passed

vt2812 submitted at Feb 13, 2026 00:12

Runtime

8 ms | Beats 90.21%

Memory

49.22 MB | Beats 37.89%

Code | Java

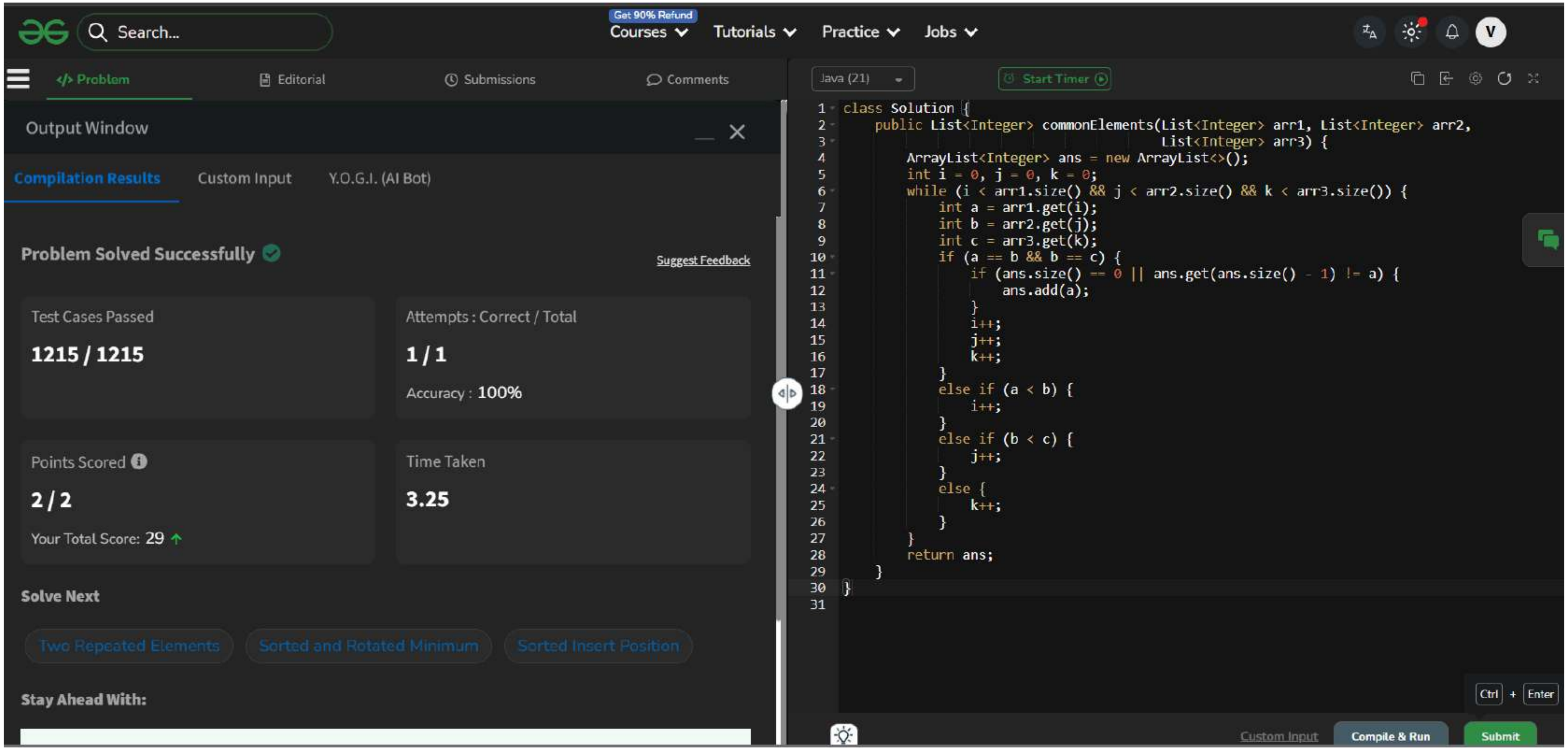
```
import java.util.Arrays;
import java.util.ArrayList;
import java.util.List;
class Solution {
    public int[][] merge(int[][] intervals) {
```

```
            return intervals;
        }
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> merged = new ArrayList<>();
        int[] currentInterval = intervals[0];
        merged.add(currentInterval);
        for (int[] nextInterval : intervals) {
            int currentEnd = currentInterval[1];
            int nextStart = nextInterval[0];
            int nextEnd = nextInterval[1];
            if (nextStart <= currentEnd) {
                currentInterval[1] = Math.max(currentEnd, nextEnd);
            } else {
                currentInterval = nextInterval;
                merged.add(currentInterval);
            }
        }
        return merged.toArray(new int[merged.size()][]);
    }
}
```

Saved

Ln 26, Col 2

Accepted Runtime: 1 ms



Problem Solved Successfully

Test Cases Passed
1215 / 1215

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

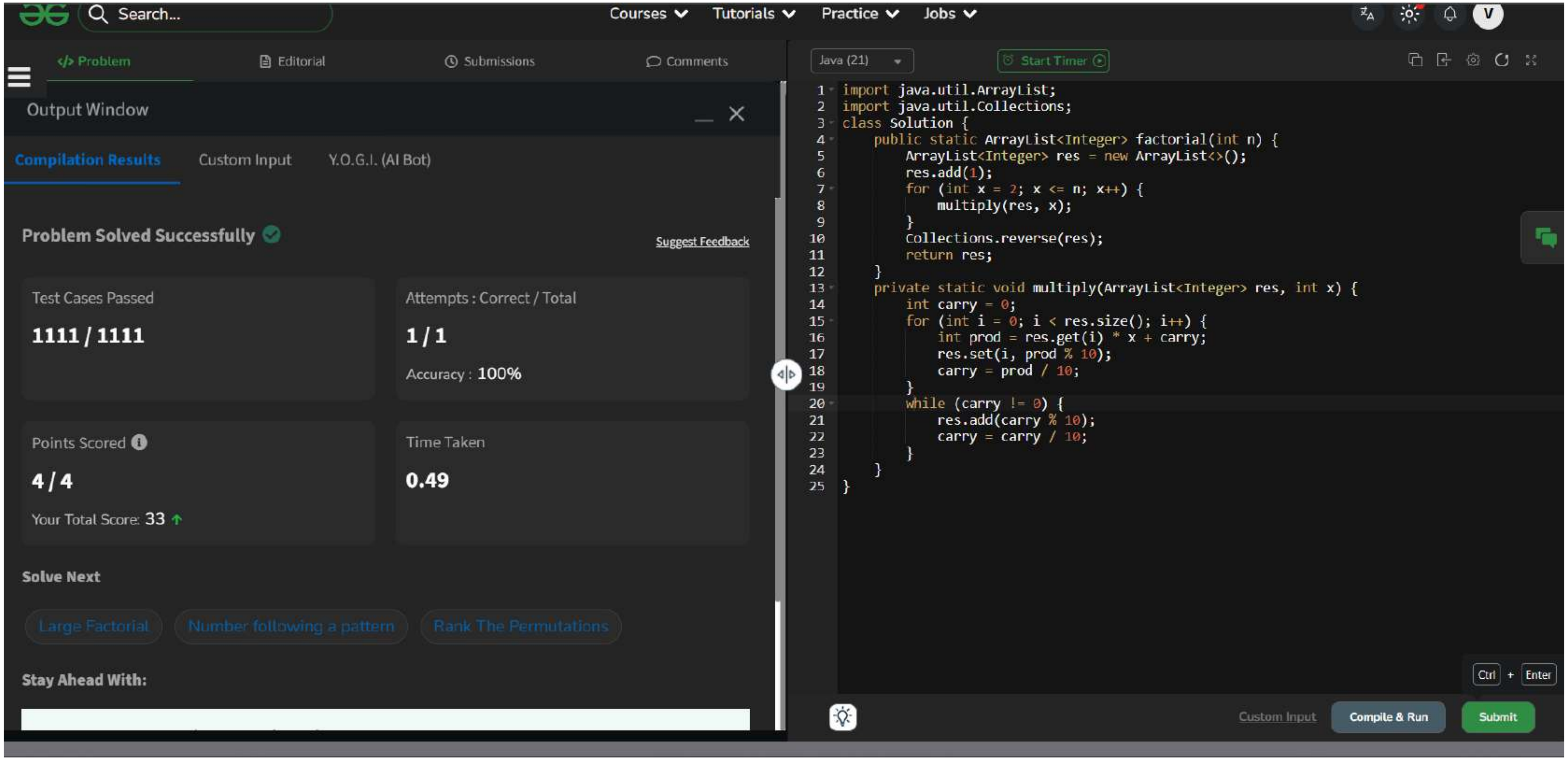
Points Scored
2 / 2
Your Total Score: 29

Time Taken
3.25

Solve Next
Two Repeated Elements
Sorted and Rotated Minimum
Sorted Insert Position

Stay Ahead With:

```java
class Solution {
    public List<Integer> commonElements(List<Integer> arr1, List<Integer> arr2,
                                        List<Integer> arr3) {
        ArrayList<Integer> ans = new ArrayList<>();
        int i = 0, j = 0, k = 0;
        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 (ans.size() == 0 || ans.get(ans.size() - 1) != a) {
                    ans.add(a);
                }
                i++;
                j++;
                k++;
            }
            else if (a < b) {
                i++;
            }
            else if (b < c) {
                j++;
            }
            else {
                k++;
            }
        }
        return ans;
    }
}
```



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

4 / 4

Your Total Score: 33

Time Taken

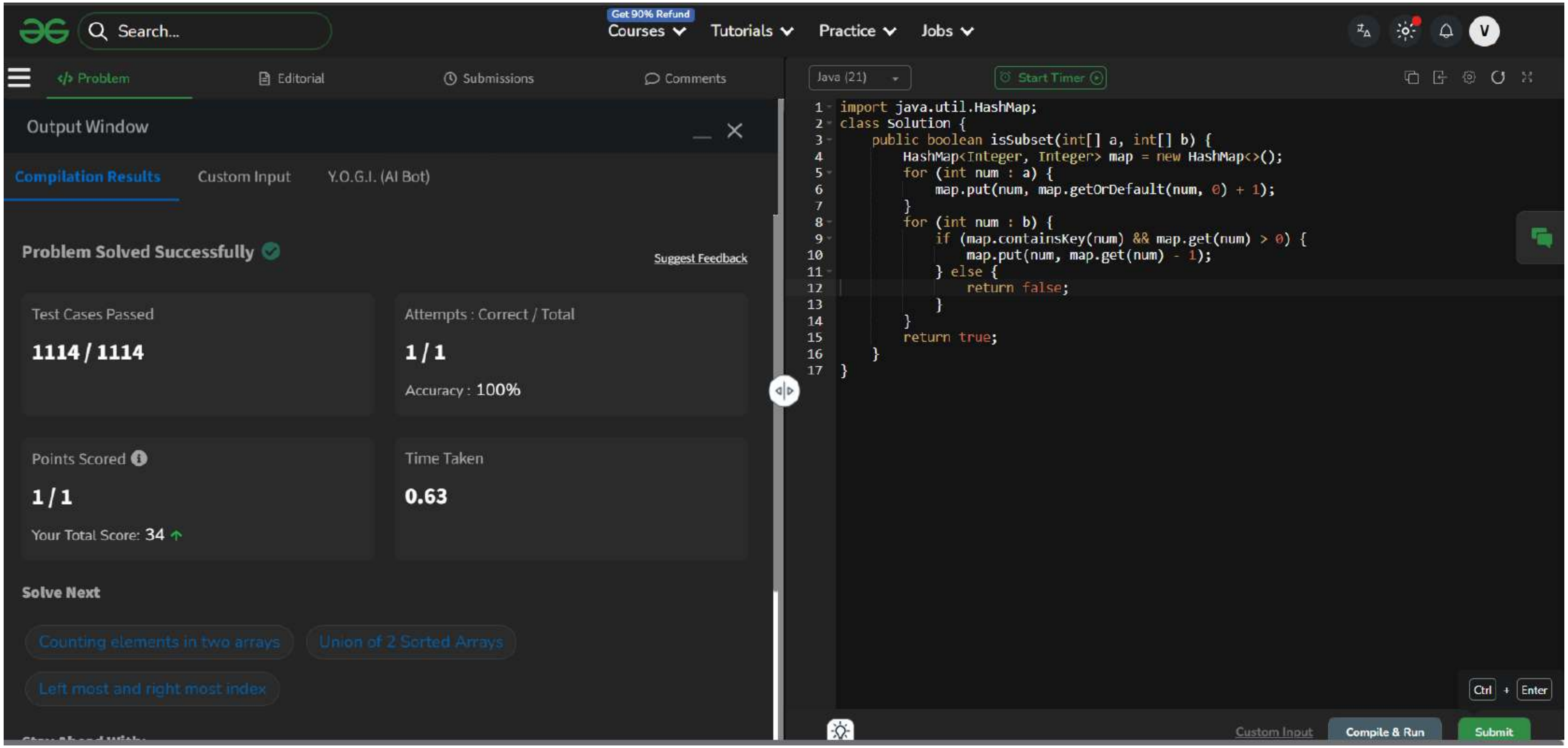
0.49

Solve Next

Large Factorial | Number following a pattern | Rank The Permutations

Stay Ahead With:

```java
import java.util.ArrayList;
import java.util.Collections;
class Solution {
    public static ArrayList<Integer> factorial(int n) {
        ArrayList<Integer> res = new ArrayList<>();
        res.add(1);
        for (int x = 2; x <= n; x++) {
            multiply(res, x);
        }
        Collections.reverse(res);
        return res;
    }
    private static void multiply(ArrayList<Integer> res, int x) {
        int carry = 0;
        for (int i = 0; i < res.size(); i++) {
            int prod = res.get(i) * x + carry;
            res.set(i, prod % 10);
            carry = prod / 10;
        }
        while (carry != 0) {
            res.add(carry % 10);
            carry = carry / 10;
        }
    }
}
```



Problem Solved Successfully

Test Cases Passed
1114 / 1114

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

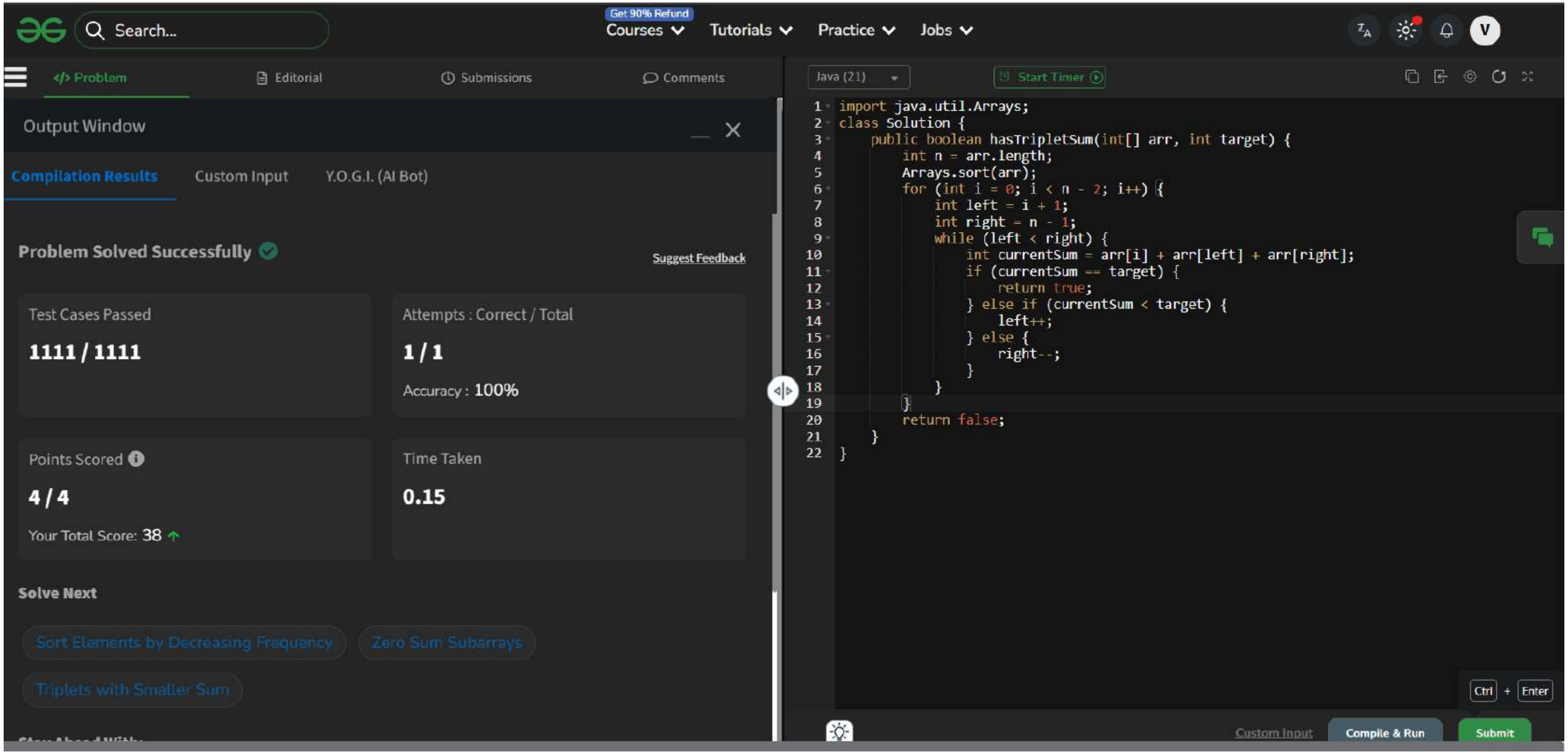
Points Scored
1 / 1
Your Total Score: 34

Time Taken
0.63

Solve Next

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

```java
import java.util.HashMap;
class Solution {
    public boolean isSubset(int[] a, int[] b) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int num : a) {
            map.put(num, map.getOrDefault(num, 0) + 1);
        }
        for (int num : b) {
            if (map.containsKey(num) && map.get(num) > 0) {
                map.put(num, map.get(num) - 1);
            } else {
                return false;
            }
        }
        return true;
    }
}
```



Problem Solved Successfully

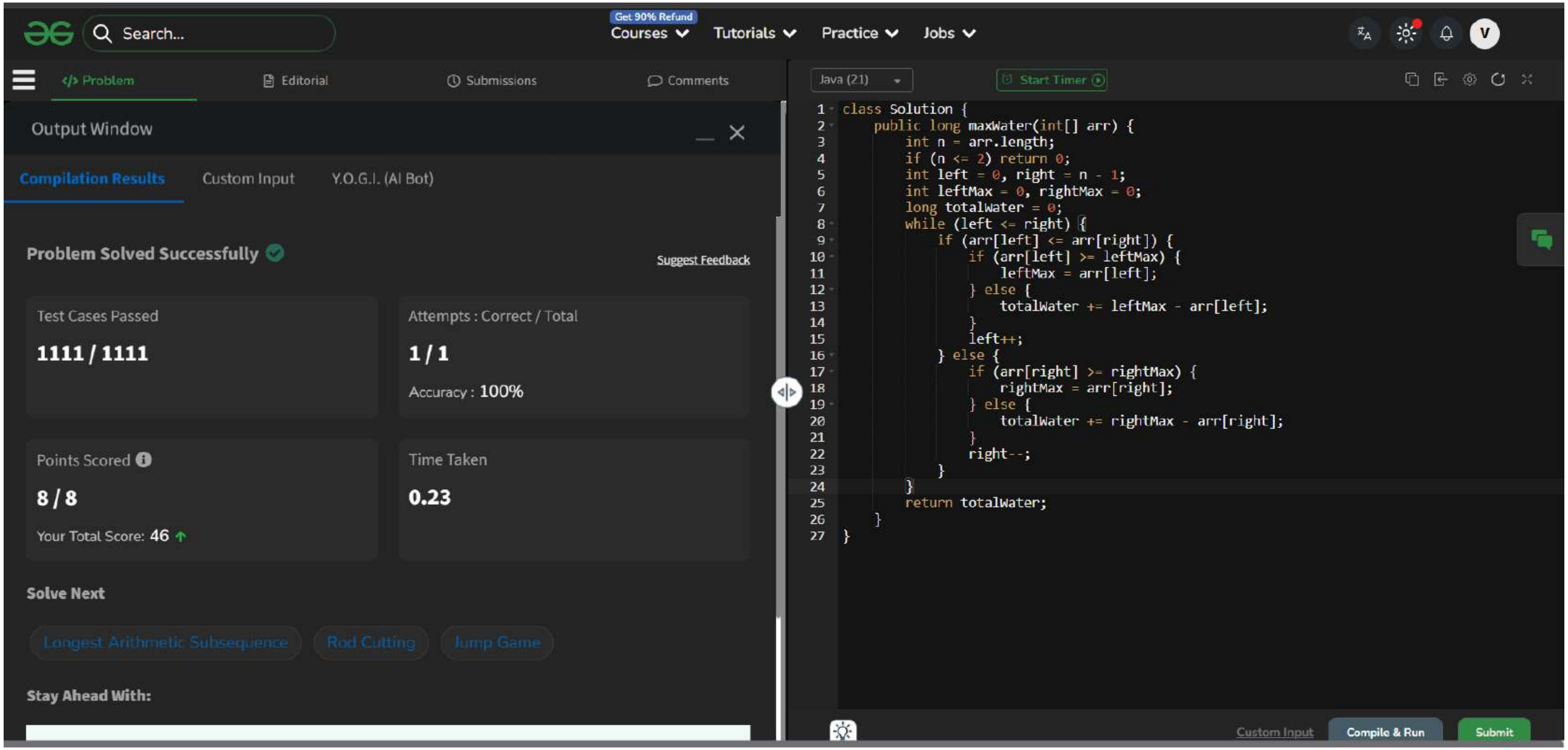
Test Cases Passed
1111 / 1111

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

Points Scored
4 / 4
Your Total Score: 38

Time Taken
0.15

```java
import java.util.Arrays;
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;
            while (left < right) {
                int currentSum = arr[i] + arr[left] + arr[right];
                if (currentSum == target) {
                    return true;
                } else if (currentSum < 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

0.23

Solve Next

Longest Arithmetic Subsequence   Rod Cutting   Jump Game

Stay Ahead With:

```java
class Solution {
    public long maxWater(int[] arr) {
        int n = arr.length;
        if (n <= 2) return 0;
        int left = 0, right = n - 1;
        int leftMax = 0, rightMax = 0;
        long totalWater = 0;
        while (left <= right) {
            if (arr[left] <= arr[right]) {
                if (arr[left] >= leftMax) {
                    leftMax = arr[left];
                } else {
                    totalWater += leftMax - arr[left];
                }
                left++;
            } else {
                if (arr[right] >= rightMax) {
                    rightMax = arr[right];
                } else {
                    totalWater += rightMax - arr[right];
                }
                right--;
            }
        }
        return totalWater;
    }
}
```