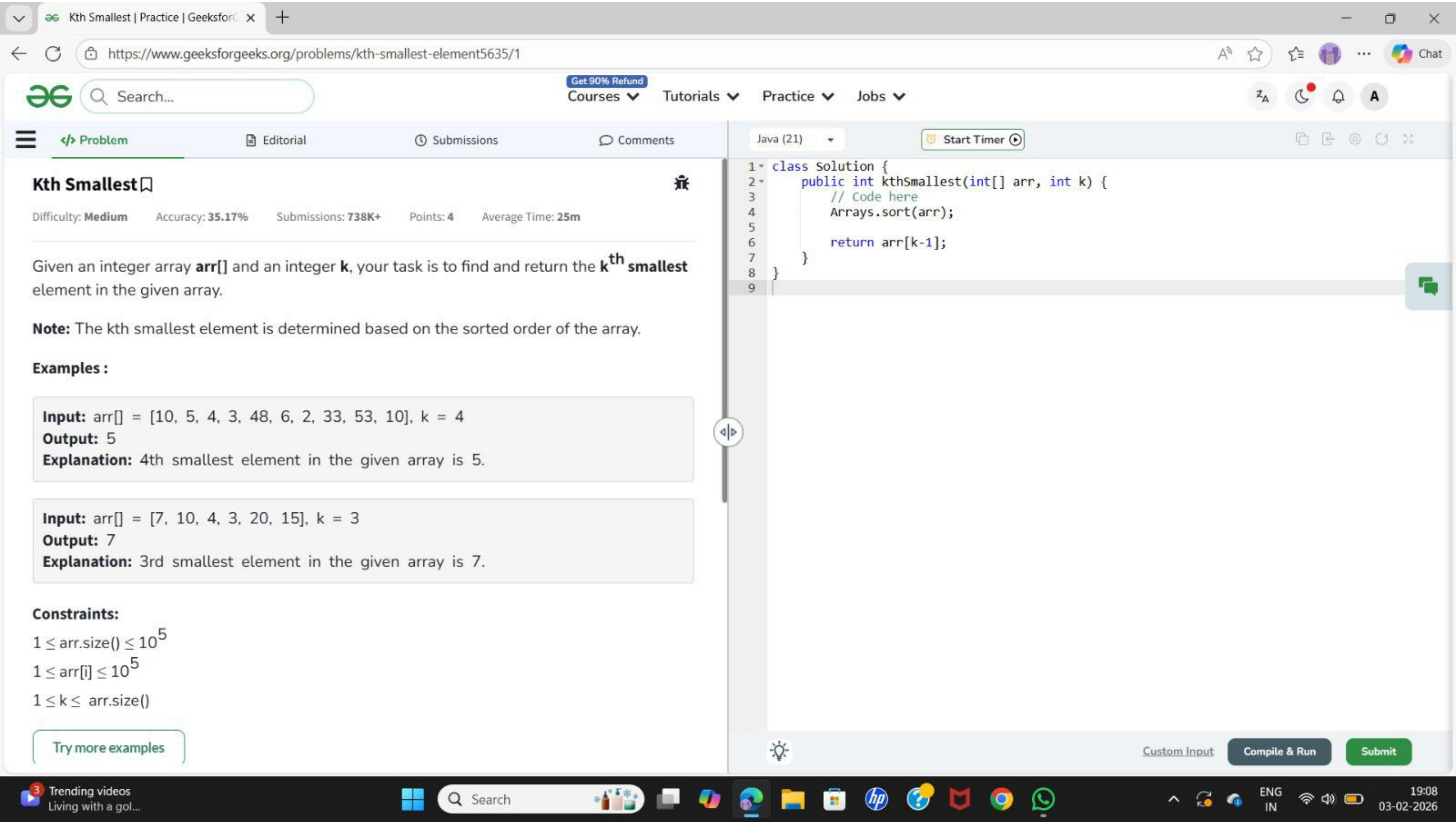


Kth Smallest

Difficulty: **Medium** Accuracy: **35.17%** Submissions: **738K+** Points: **4** Average Time: **25m**

Given an integer array **arr[]** and an integer **k**, your task is to find and return the **k**th **smallest** element in the given array.

Note: The kth smallest element is determined based on the sorted order of the array.

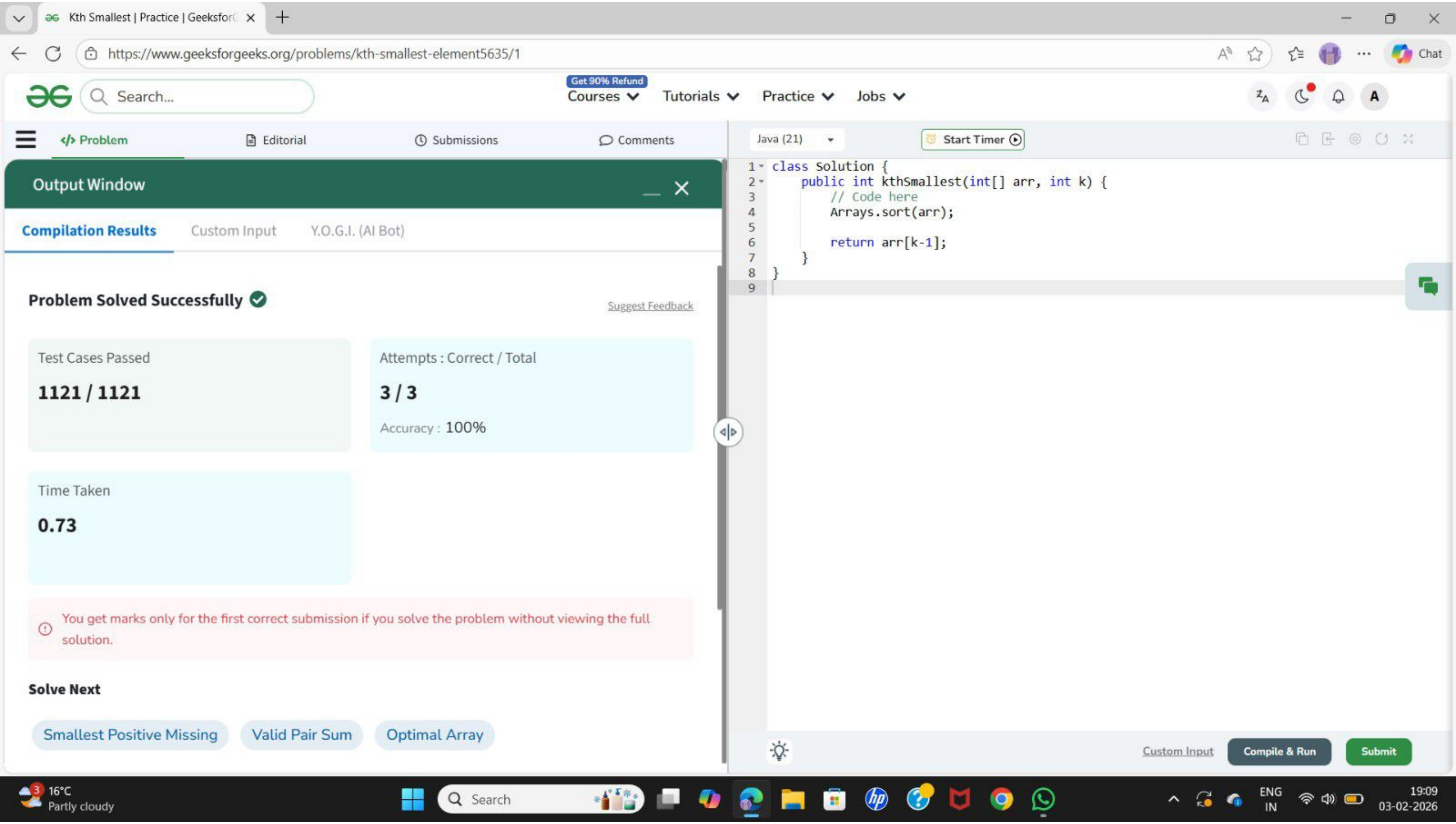
Examples :

Input: arr[] = [10, 5, 4, 3, 48, 6, 2, 33, 53, 10], k = 4
Output: 5
Explanation: 4th smallest element in the given array is 5.

Input: arr[] = [7, 10, 4, 3, 20, 15], k = 3
Output: 7
Explanation: 3rd smallest element in the given array is 7.

Constraints:
$1 \le arr.size() \le 10^5$
$1 \le arr[i] \le 10^5$
$1 \le k \le arr.size()$

```java
class Solution {
    public int kthSmallest(int[] arr, int k) {
        // Code here
        Arrays.sort(arr);

        return arr[k-1];
    }
}
```



Problem Solved Successfully ✓

Test Cases Passed
1121 / 1121

Attempts : Correct / Total
3 / 3
Accuracy : 100%

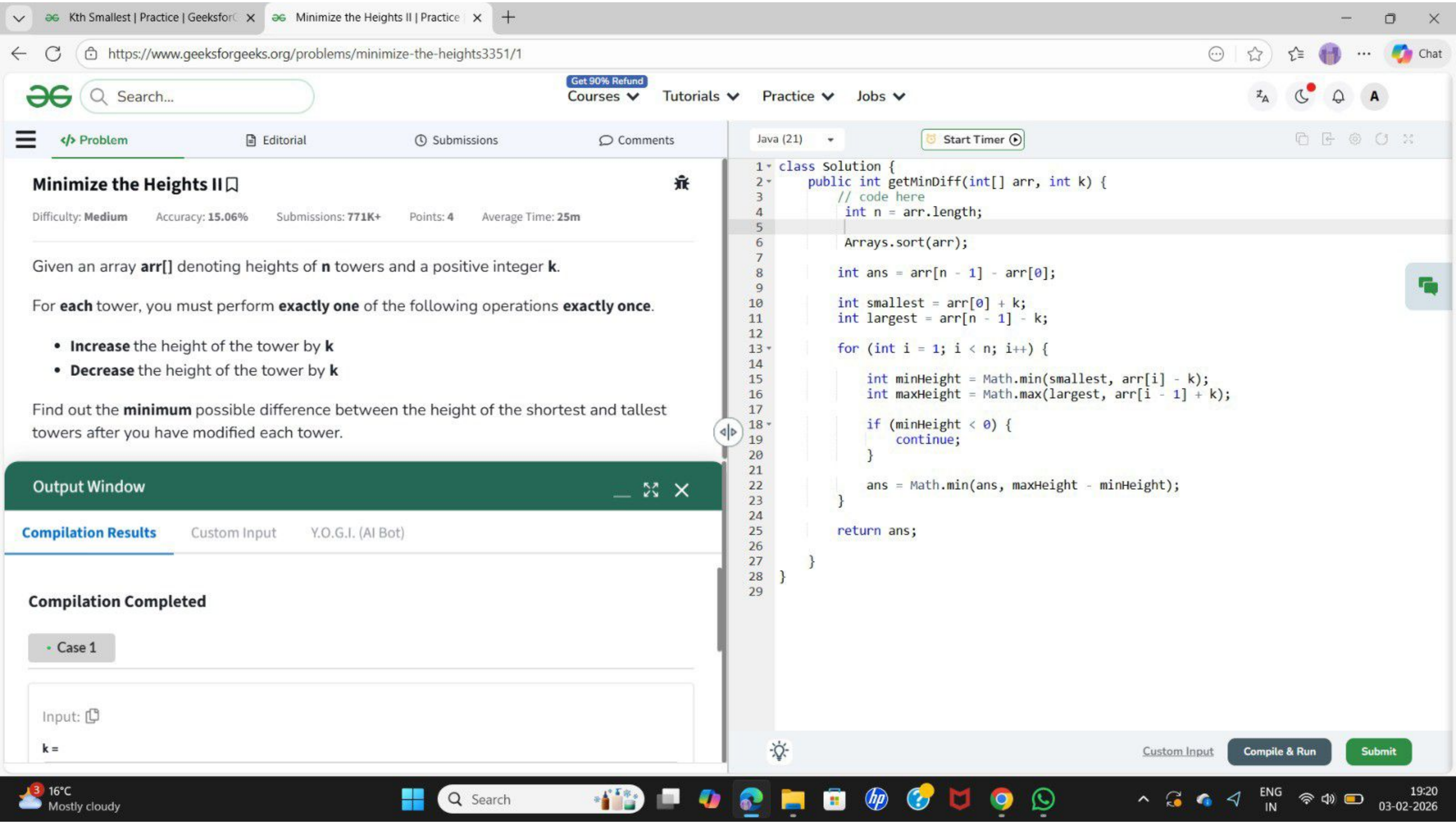
Time Taken
0.73

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

Solve Next

Smallest Positive Missing | Valid Pair Sum | Optimal Array

```java
class Solution {
    public int kthSmallest(int[] arr, int k) {
        // Code here
        Arrays.sort(arr);

        return arr[k-1];
    }
}
```


Kth Smallest | Practice | GeeksforG
Minimize the Heights II | Practice
https://www.geeksforgeeks.org/problems/minimize-the-heights3351/1
Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Java (21)
Start Timer

Minimize the Heights II

Difficulty: **Medium** Accuracy: **15.06%** Submissions: **771K+** Points: **4** Average Time: **25m**

Given an array **arr[]** denoting heights of **n** towers and a positive integer **k**.

For **each** tower, you must perform **exactly one** of the following operations **exactly once**.

- **Increase** the height of the tower by **k**
- **Decrease** the height of the tower by **k**

Find out the **minimum** possible difference between the height of the shortest and tallest towers after you have modified each tower.

```
class Solution {
    public int getMinDiff(int[] arr, int k) {
        // code here
         int n = arr.length;
         
         Arrays.sort(arr);
        
        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 minHeight = Math.min(smallest, arr[i] - k);
            int maxHeight = Math.max(largest, arr[i - 1] + k);
            
            if (minHeight < 0) {
                continue;
            }
            
            ans = Math.min(ans, maxHeight - minHeight);
        }
        
        return ans;
        
    }
}
```

Output Window

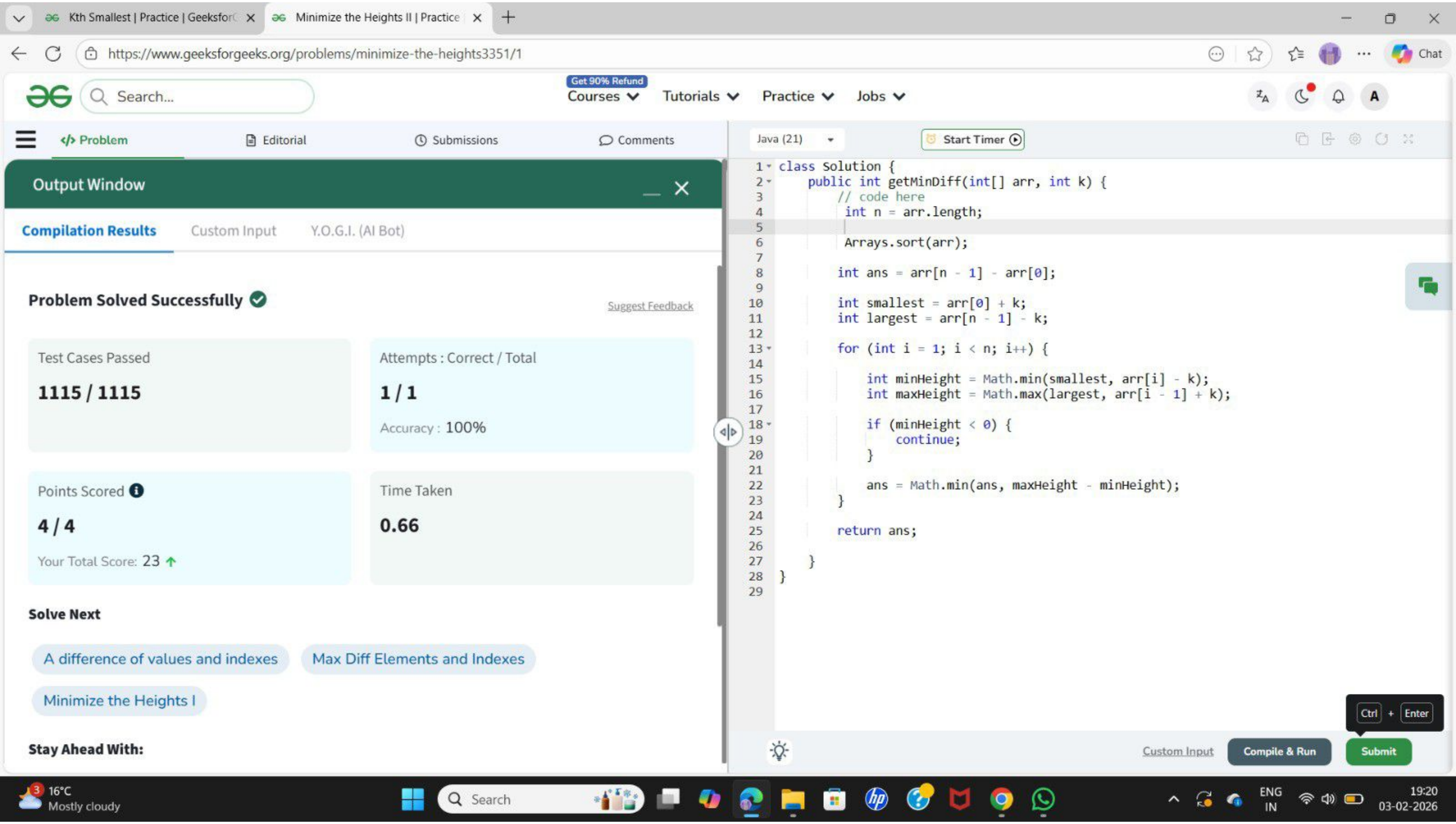
Compilation Results Custom Input Y.O.G.I. (AI Bot)

Compilation Completed

Case 1

Input:

k =



Output Window

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

Problem Solved Successfully

Test Cases Passed
1115 / 1115

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

Points Scored
4 / 4
Your Total Score: 23

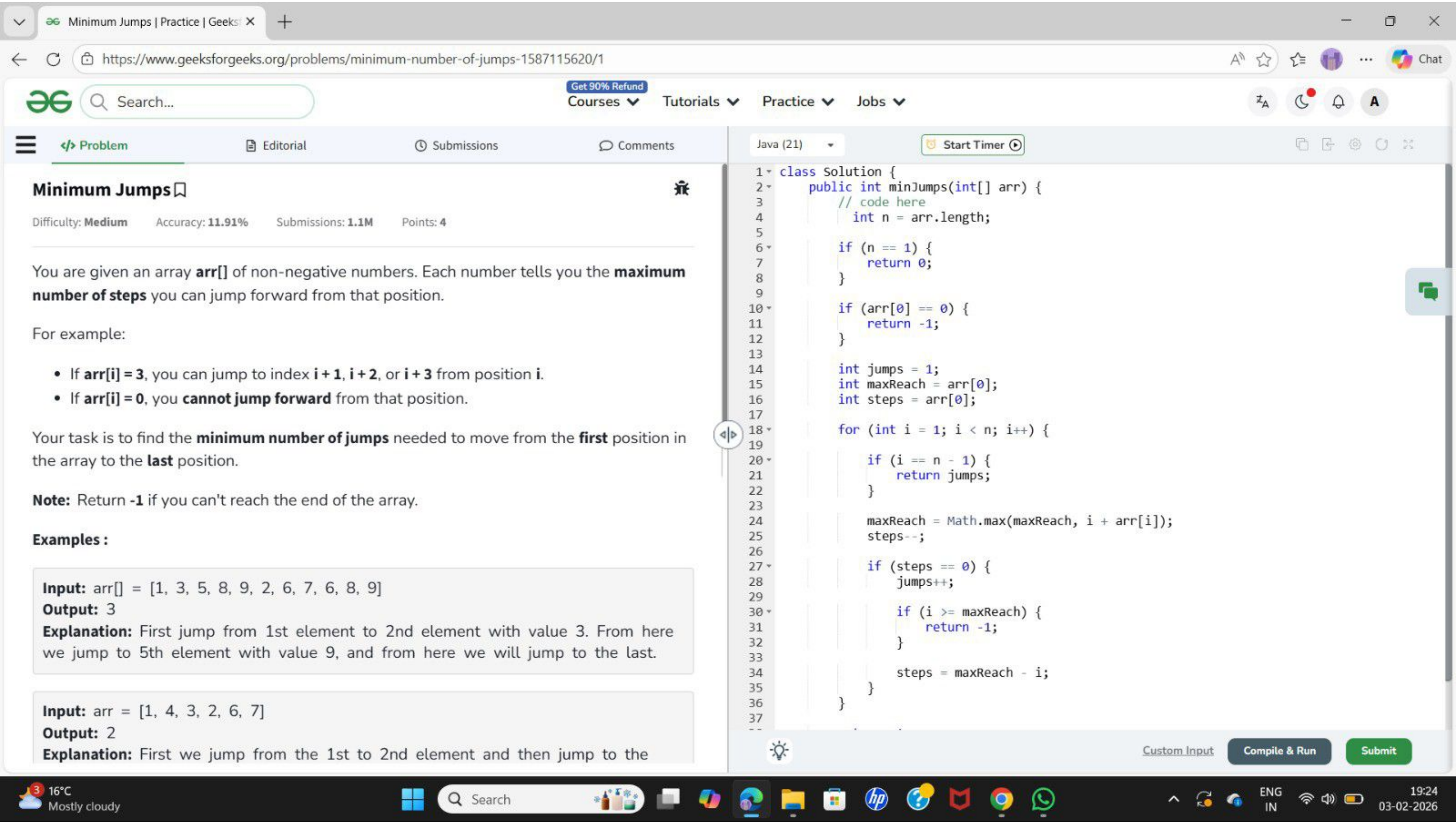
Time Taken
0.66

Solve Next

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

Stay Ahead With:

```java
class Solution {
    public int getMinDiff(int[] arr, int k) {
        // code here
         int n = arr.length;
         
         Arrays.sort(arr);
         
        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 minHeight = Math.min(smallest, arr[i] - k);
            int maxHeight = Math.max(largest, arr[i - 1] + k);
            
            if (minHeight < 0) {
                continue;
            }
            
            ans = Math.min(ans, maxHeight - minHeight);
        }
        
        return ans;
        
    }
}
```



Minimum Jumps

Difficulty: **Medium** Accuracy: **11.91%** Submissions: **1.1M** Points: **4**

You are given an array **arr[]** of non-negative numbers. Each number tells you the **maximum number of steps** you can jump forward from that position.

For example:

- If **arr[i] = 3**, you can jump to index **i + 1**, **i + 2**, or **i + 3** from position **i**.
- If **arr[i] = 0**, you **cannot jump forward** from that position.

Your task is to find the **minimum number of jumps** needed to move from the **first** position in the array to the **last** position.

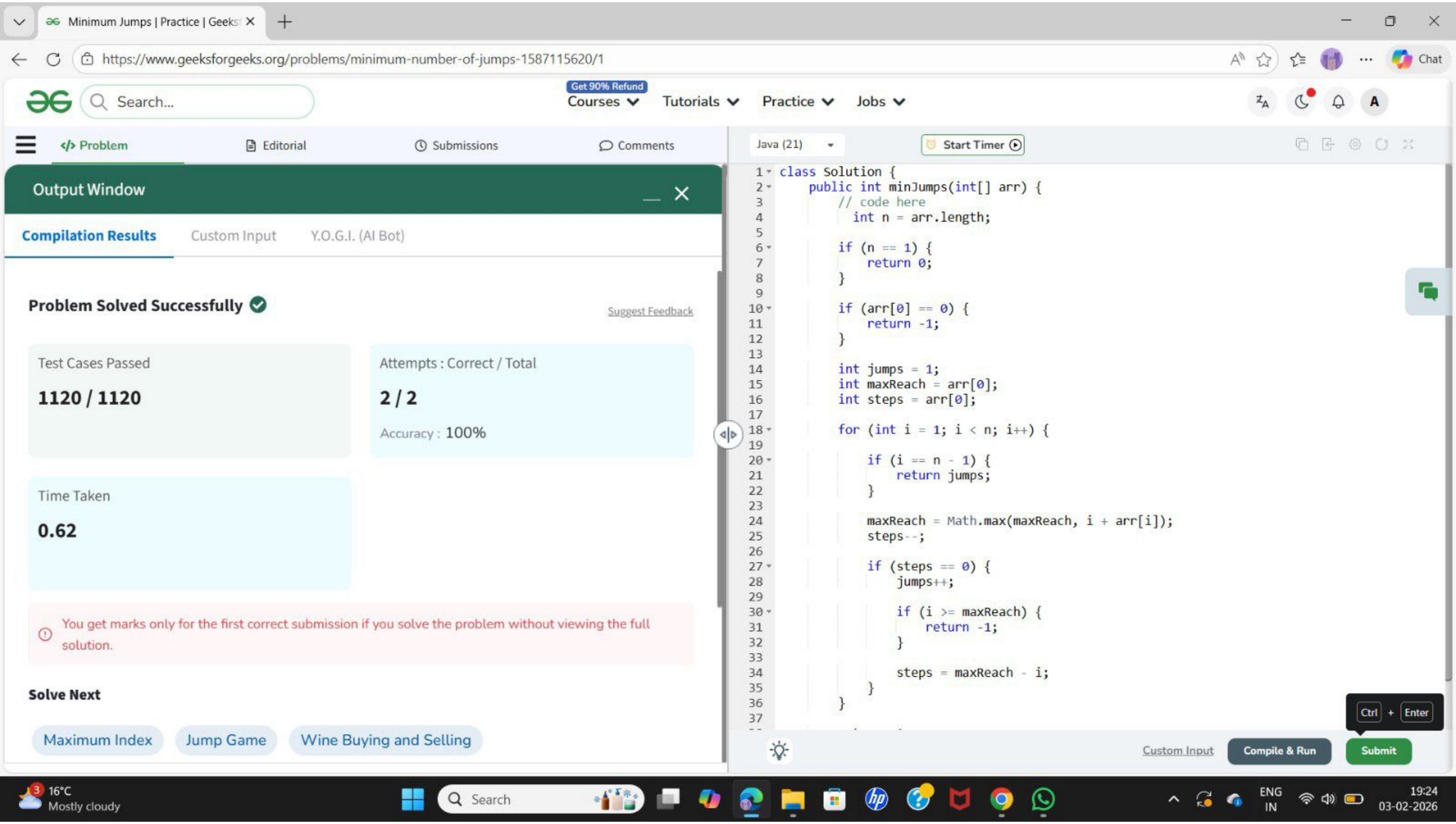
Note: Return **-1** if you can't reach the end of the array.

Examples :

Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]
Output: 3
Explanation: First jump from 1st element to 2nd element with value 3. From here we jump to 5th element with value 9, and from here we will jump to the last.

Input: arr = [1, 4, 3, 2, 6, 7]
Output: 2
Explanation: First we jump from the 1st to 2nd element and then jump to the

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

        if (n == 1) {
            return 0;
        }

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

        int jumps = 1;
        int maxReach = arr[0];
        int steps = arr[0];

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

            if (i == n - 1) {
                return jumps;
            }

            maxReach = Math.max(maxReach, i + arr[i]);
            steps--;

            if (steps == 0) {
                jumps++;

                if (i >= maxReach) {
                    return -1;
                }

                steps = maxReach - i;
            }
        }

```



Problem Solved Successfully ✓

Test Cases Passed
1120 / 1120

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

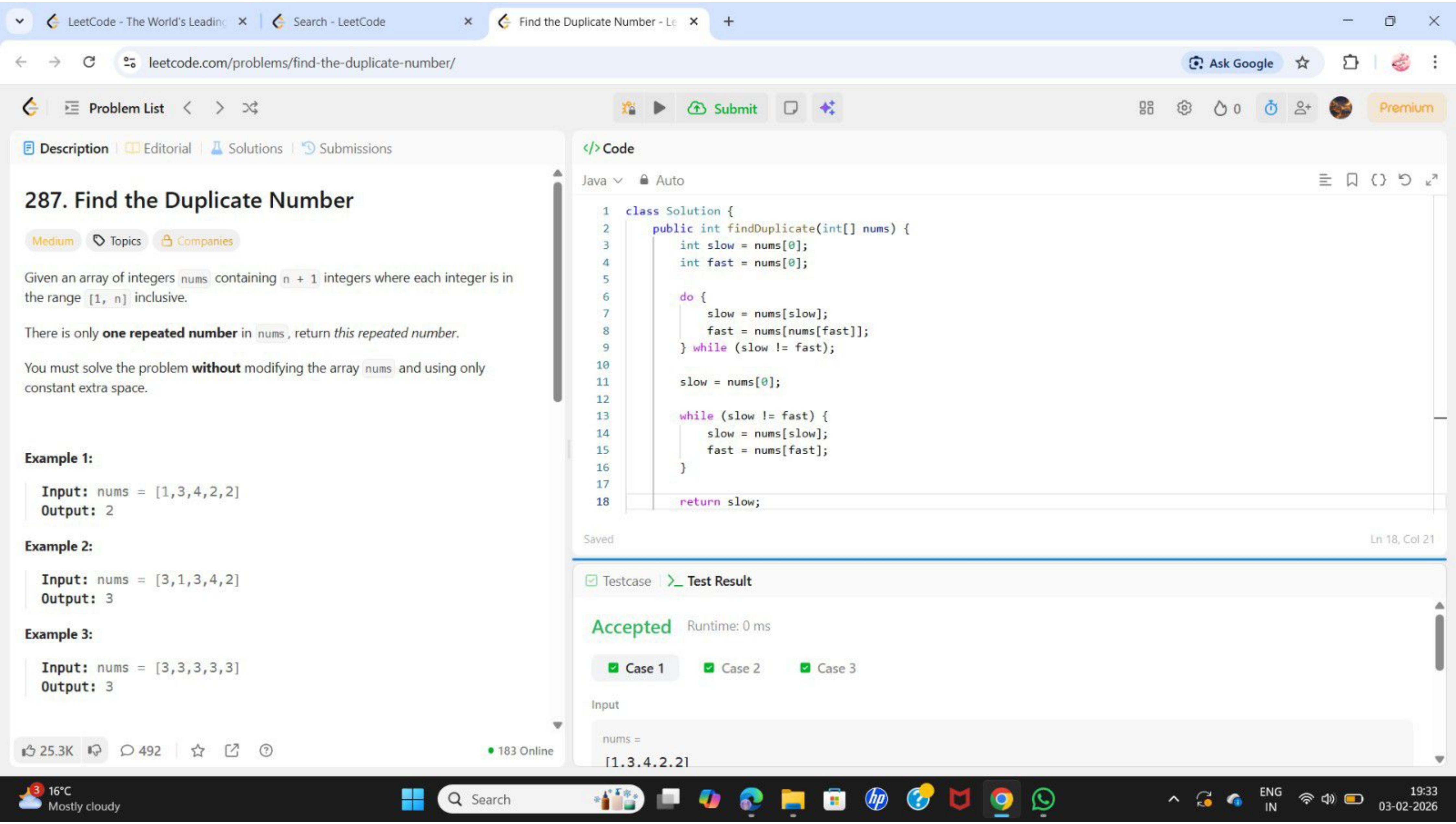
Time Taken
0.62

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 {
    public int minJumps(int[] arr) {
        // code here
          int n = arr.length;

        if (n == 1) {
            return 0;
        }

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

        int jumps = 1;
        int maxReach = arr[0];
        int steps = arr[0];

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

            if (i == n - 1) {
                return jumps;
            }

            maxReach = Math.max(maxReach, i + arr[i]);
            steps--;

            if (steps == 0) {
                jumps++;

                if (i >= maxReach) {
                    return -1;
                }

                steps = maxReach - i;
            }
        }

```



287. Find the Duplicate Number

Medium · Topics · Companies

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.

There is only **one repeated number** in `nums`, return *this repeated number*.

You must solve the problem **without** modifying the array `nums` and using only constant extra space.

Example 1:

```
Input: nums = [1,3,4,2,2]
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
Output: 3
```

Example 3:

```
Input: nums = [3,3,3,3,3]
Output: 3
```

```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;
```

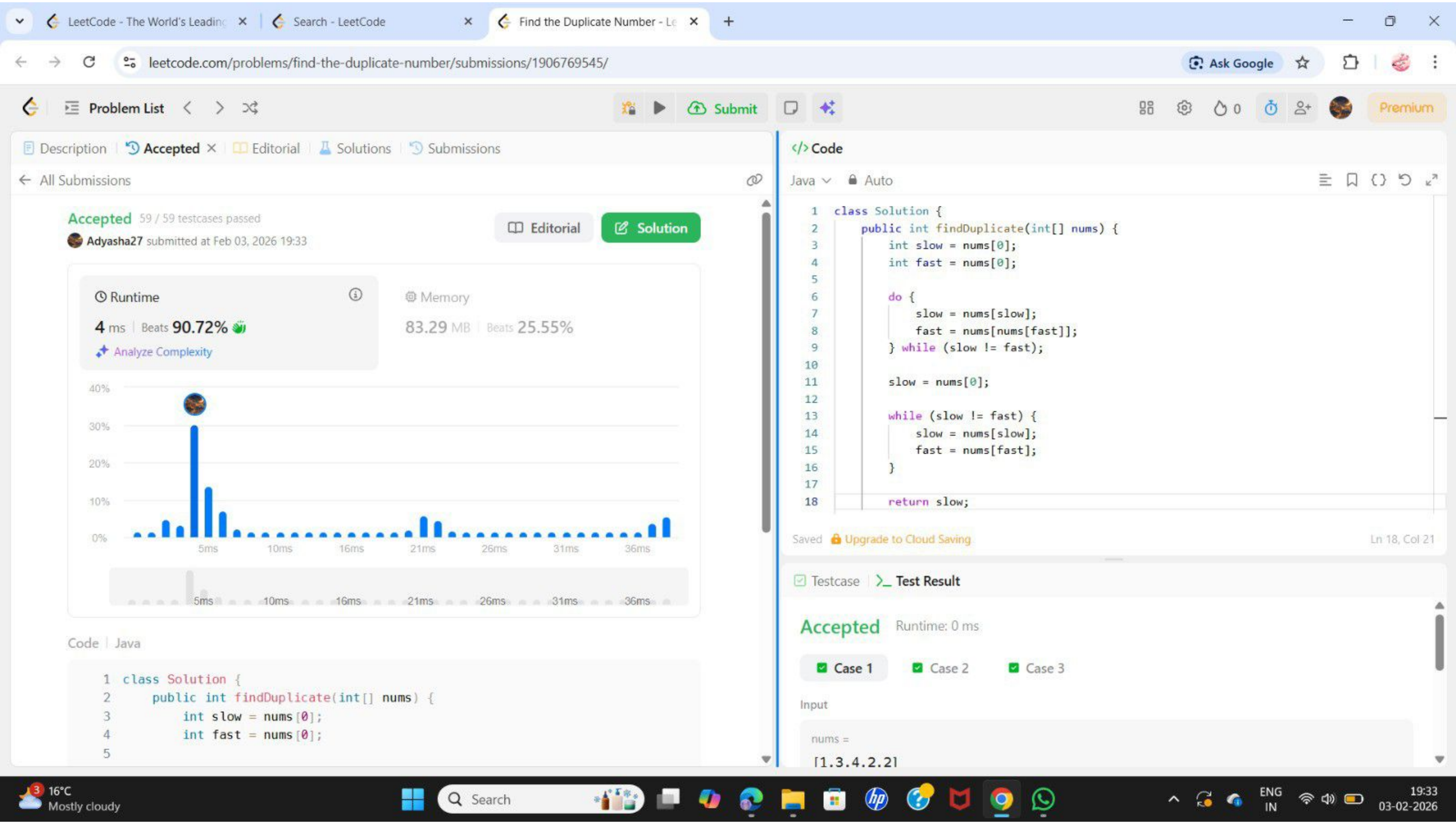
Accepted   Runtime: 0 ms

✓ Case 1   ✓ Case 2   ✓ Case 3

Input

nums =

[1,3,4,2,2]



Accepted 59 / 59 testcases passed
Adyasha27 submitted at Feb 03, 2026 19:33

Runtime
4 ms | Beats 90.72%

Memory
83.29 MB | Beats 25.55%

```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;
```

Accepted Runtime: 0 ms

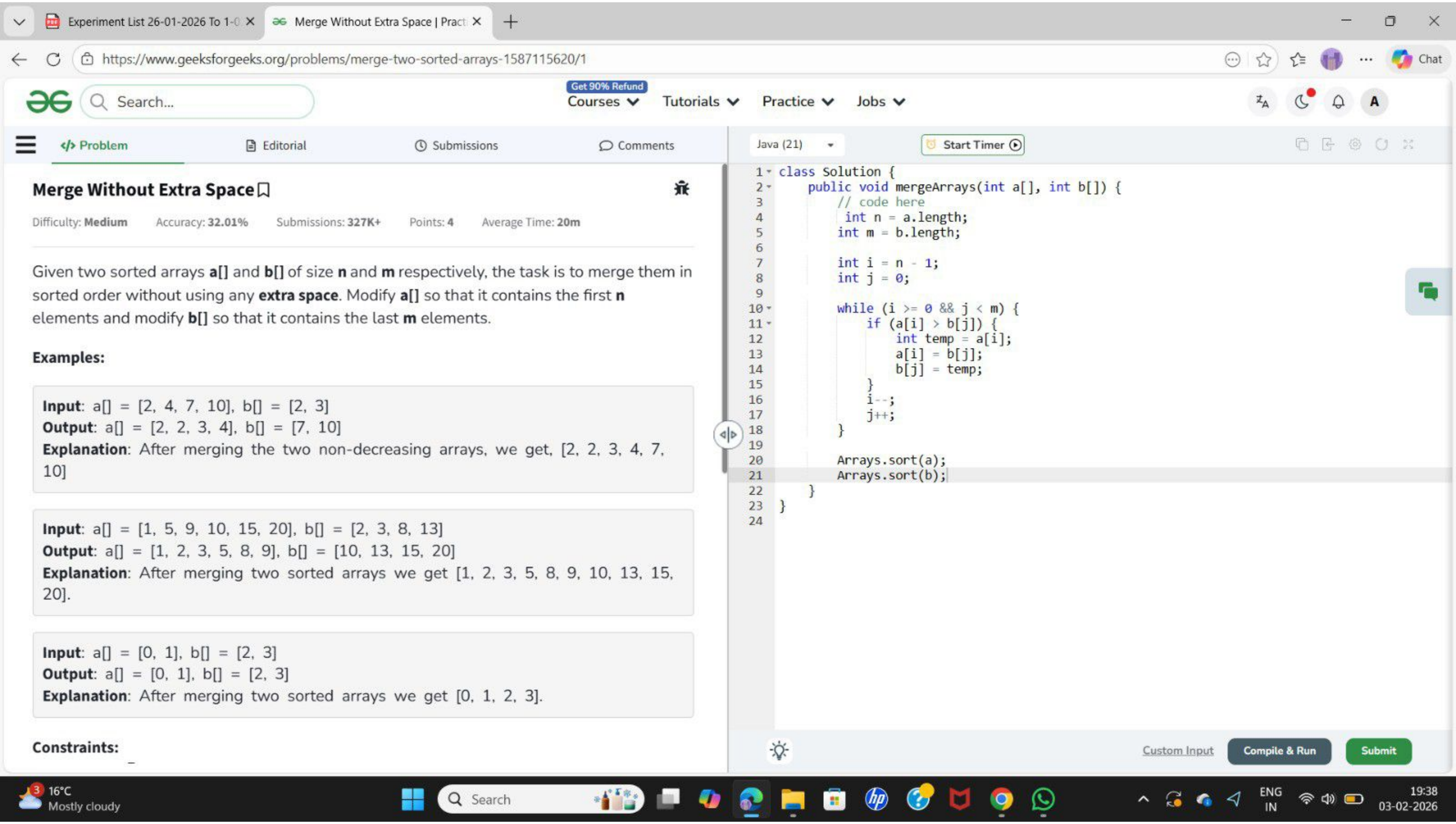
Case 1 Case 2 Case 3

Input

nums =
[1,3,4,2,2]

Code | Java

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

```



Merge Without Extra Space

Difficulty: **Medium** Accuracy: **32.01%** Submissions: **327K+** Points: **4** Average Time: **20m**

Given two sorted arrays **a[]** and **b[]** of size **n** and **m** respectively, the task is to merge them in sorted order without using any **extra space**. Modify **a[]** so that it contains the first **n** elements and modify **b[]** so that it contains the last **m** elements.

Examples:

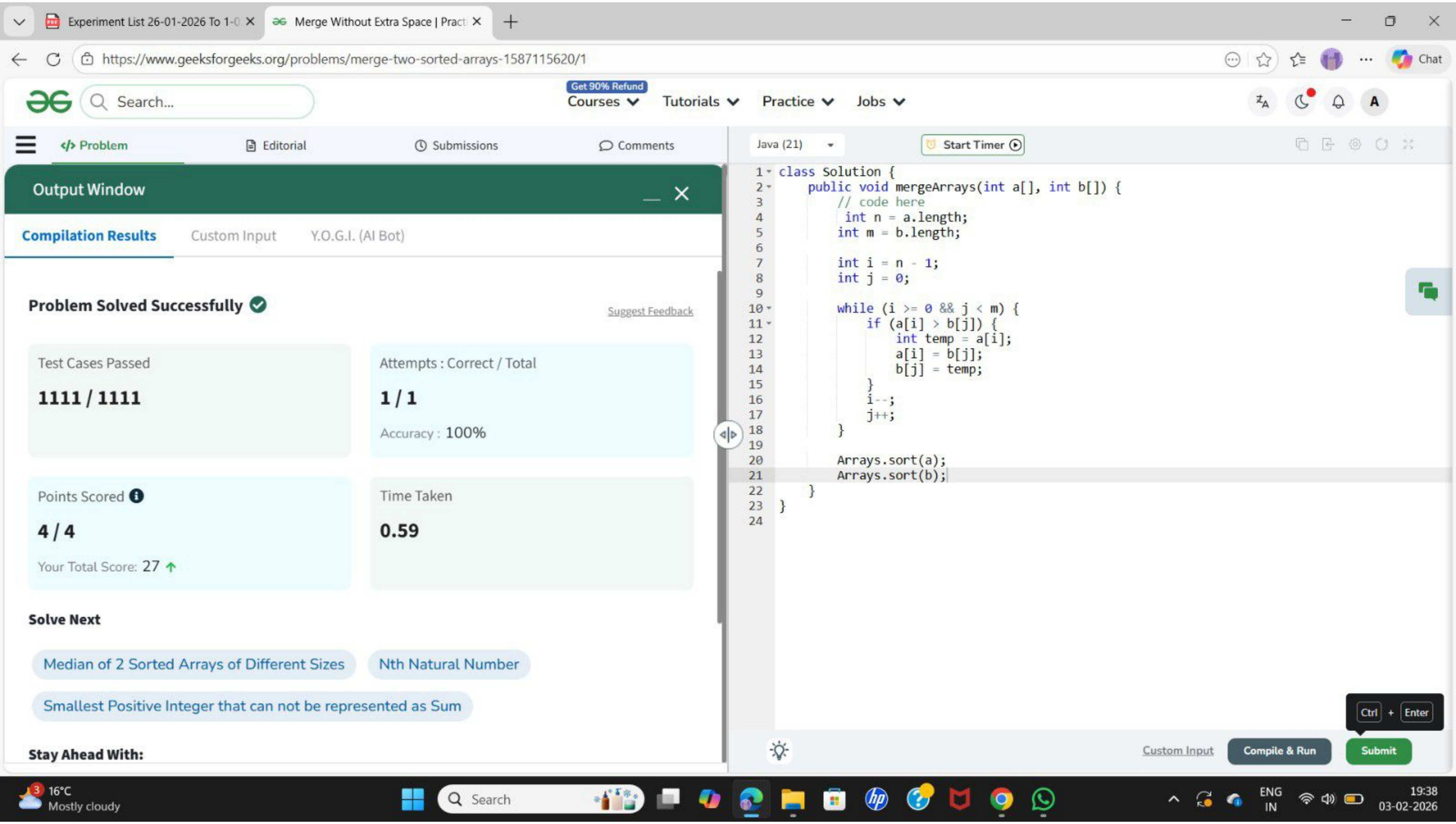
Input: a[] = [2, 4, 7, 10], b[] = [2, 3]
Output: a[] = [2, 2, 3, 4], b[] = [7, 10]
Explanation: After merging the two non-decreasing arrays, we get, [2, 2, 3, 4, 7, 10]

Input: a[] = [1, 5, 9, 10, 15, 20], b[] = [2, 3, 8, 13]
Output: a[] = [1, 2, 3, 5, 8, 9], b[] = [10, 13, 15, 20]
Explanation: After merging two sorted arrays we get [1, 2, 3, 5, 8, 9, 10, 13, 15, 20].

Input: a[] = [0, 1], b[] = [2, 3]
Output: a[] = [0, 1], b[] = [2, 3]
Explanation: After merging two sorted arrays we get [0, 1, 2, 3].

Constraints:

```java
class Solution {
    public void mergeArrays(int a[], int b[]) {
        // code here
         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);
    }
}
```



https://www.geeksforgeeks.org/problems/merge-two-sorted-arrays-1587115620/1

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 **0.59**

Solve Next

Median of 2 Sorted Arrays of Different Sizes

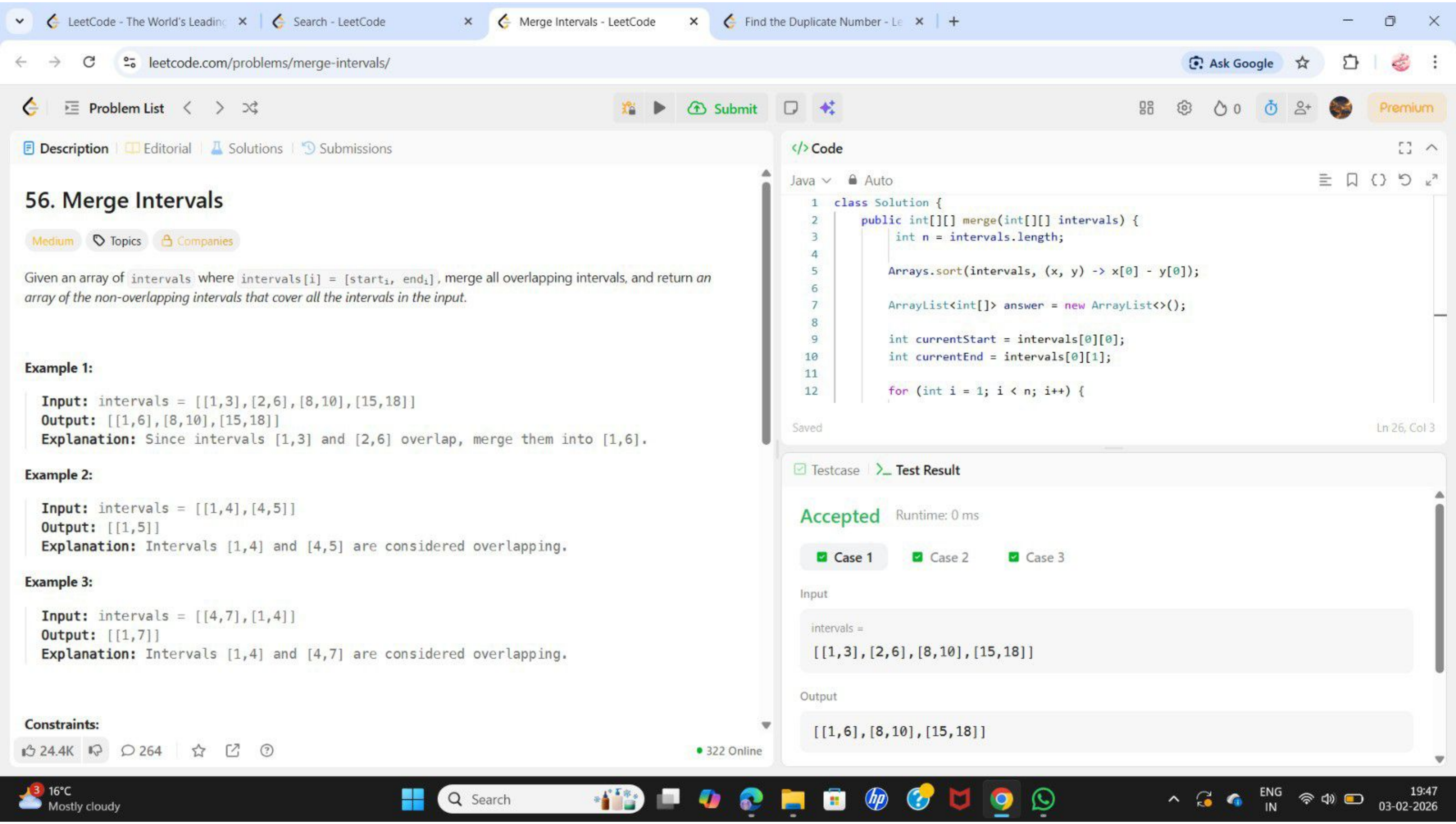
Nth Natural Number

Smallest Positive Integer that can not be represented as Sum

Stay Ahead With:

Java (21)

```java
class Solution {
    public void mergeArrays(int a[], int b[]) {
        // code here
         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);
    }
}
```



56. Merge Intervals

Given an array of `intervals` where `intervals[i] = [start_i, end_i]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input.*

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

Example 3:

```
Input: intervals = [[4,7],[1,4]]
Output: [[1,7]]
Explanation: Intervals [1,4] and [4,7] are considered overlapping.
```

Constraints:

24.4K 264 • 322 Online

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

        Arrays.sort(intervals, (x, y) -> x[0] - y[0]);

        ArrayList<int[]> answer = new ArrayList<>();

        int currentStart = intervals[0][0];
        int currentEnd = intervals[0][1];

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

Saved Ln 26, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

✓ Case 1 ✓ Case 2 ✓ Case 3

Input

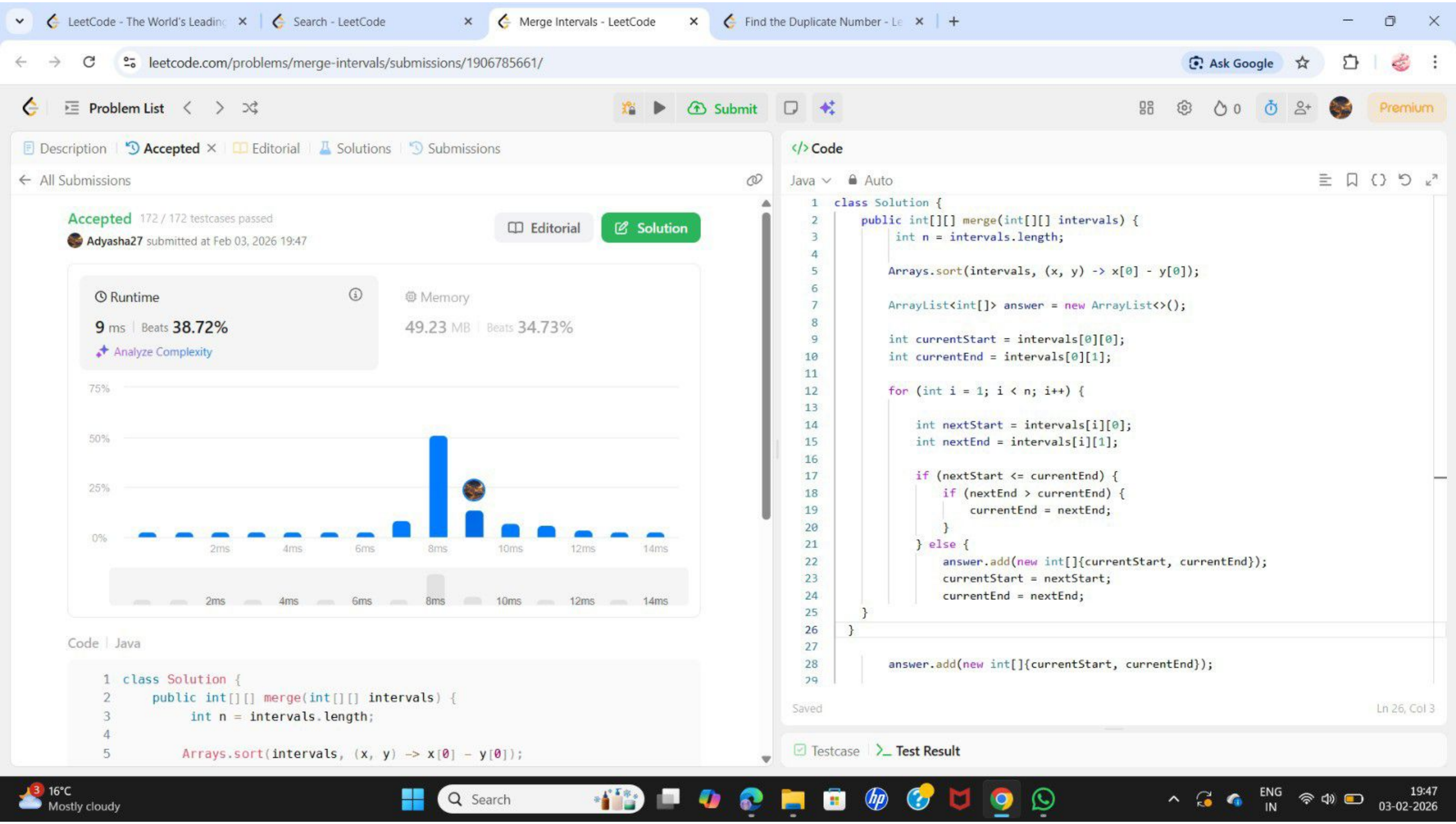
intervals =

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

Output

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

16°C Mostly cloudy Search ENG IN 19:47 03-02-2026



Accepted 172 / 172 testcases passed

Adyasha27 submitted at Feb 03, 2026 19:47

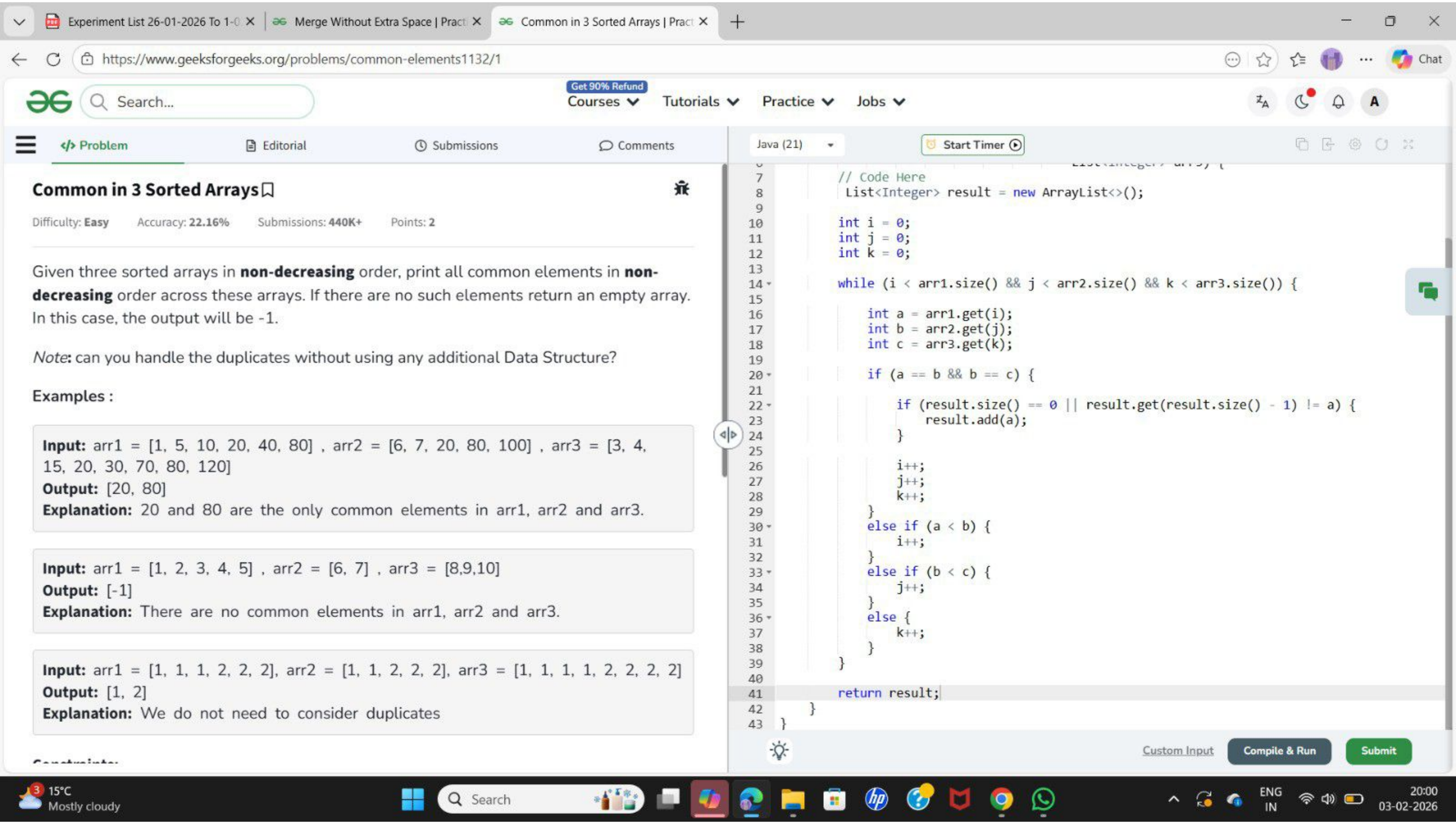
Runtime
9 ms | Beats 38.72%

Memory
49.23 MB | Beats 34.73%

Code | Java

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

       Arrays.sort(intervals, (x, y) -> x[0] - y[0]);
```

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

    Arrays.sort(intervals, (x, y) -> x[0] - y[0]);

    ArrayList<int[]> answer = new ArrayList<>();

    int currentStart = intervals[0][0];
    int currentEnd = intervals[0][1];

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

      int nextStart = intervals[i][0];
      int nextEnd = intervals[i][1];

      if (nextStart <= currentEnd) {
        if (nextEnd > currentEnd) {
          currentEnd = nextEnd;
        }
      } else {
        answer.add(new int[]{currentStart, currentEnd});
        currentStart = nextStart;
        currentEnd = nextEnd;
  }
 }

    answer.add(new int[]{currentStart, currentEnd});
```



Common in 3 Sorted Arrays

Difficulty: **Easy** Accuracy: **22.16%** Submissions: **440K+** 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?

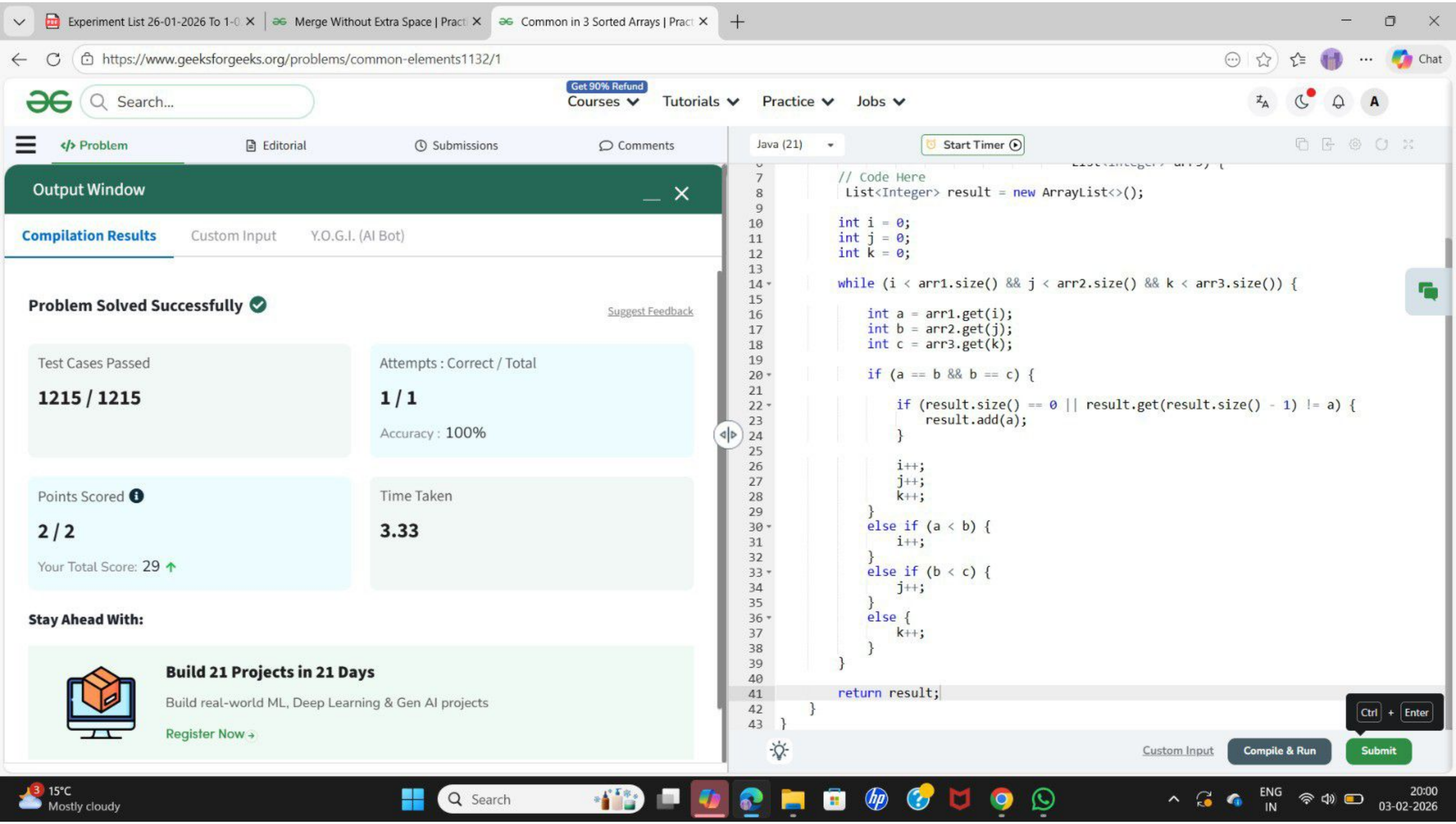
Examples :

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

```java
        // Code Here
         List<Integer> result = new ArrayList<>();

        int i = 0;
        int j = 0;
        int 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 (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

1215 / 1215

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

2 / 2

Your Total Score: 29

Time Taken

3.33

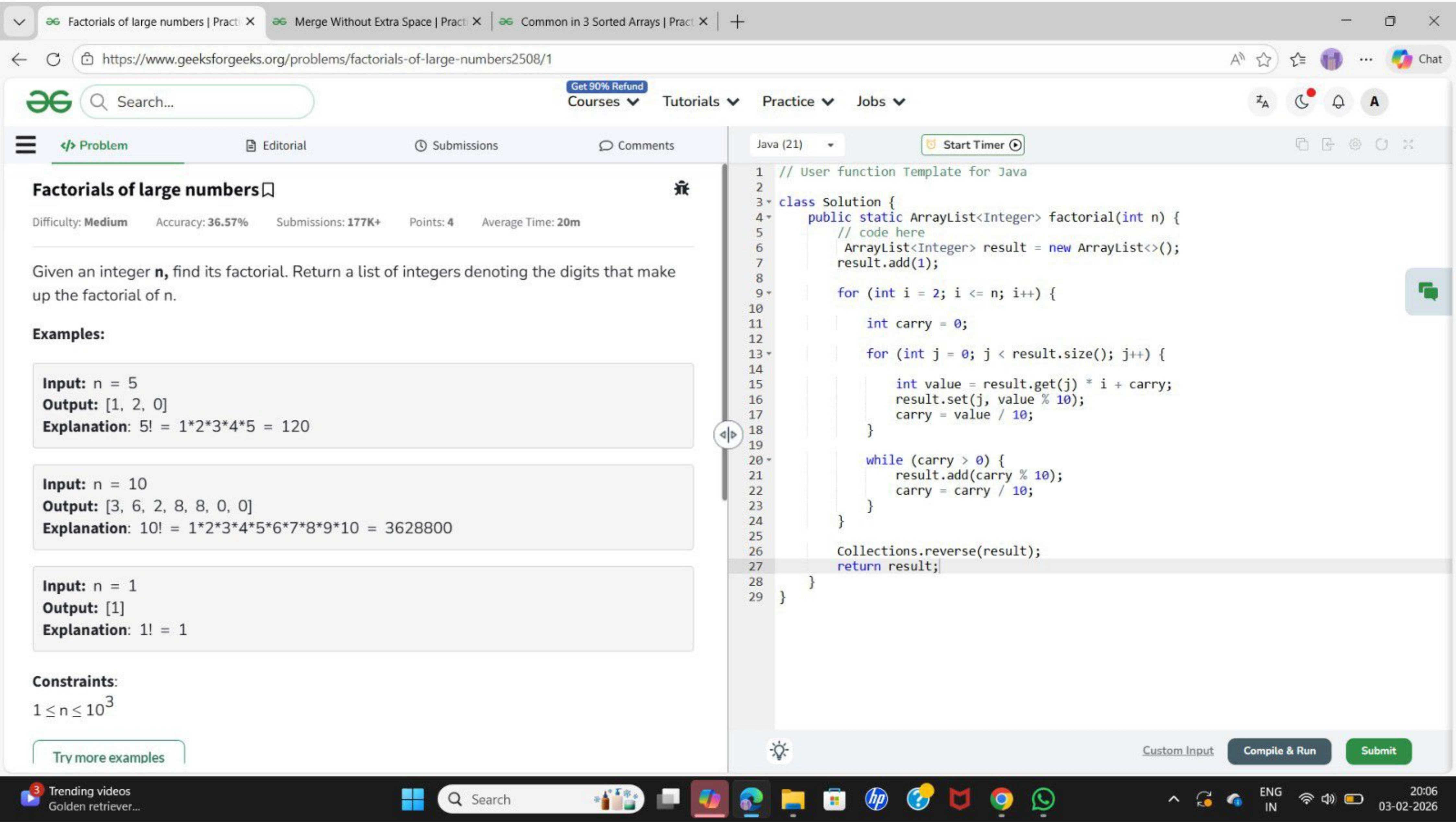
Stay Ahead With:

Build 21 Projects in 21 Days

Build real-world ML, Deep Learning & Gen AI projects

Register Now

```java
                                                        List<Integer> arr3) {
        // Code Here
         List<Integer> result = new ArrayList<>();

        int i = 0;
        int j = 0;
        int 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 (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;
    }
}
```



Factorials of large numbers | Practice

https://www.geeksforgeeks.org/problems/factorials-of-large-numbers2508/1

Factorials of large numbers

Difficulty: **Medium** Accuracy: **36.57%** Submissions: **177K+** Points: **4** Average Time: **20m**

Given an integer **n,** find its factorial. Return a list of integers denoting the digits that make up the factorial of n.

Examples:

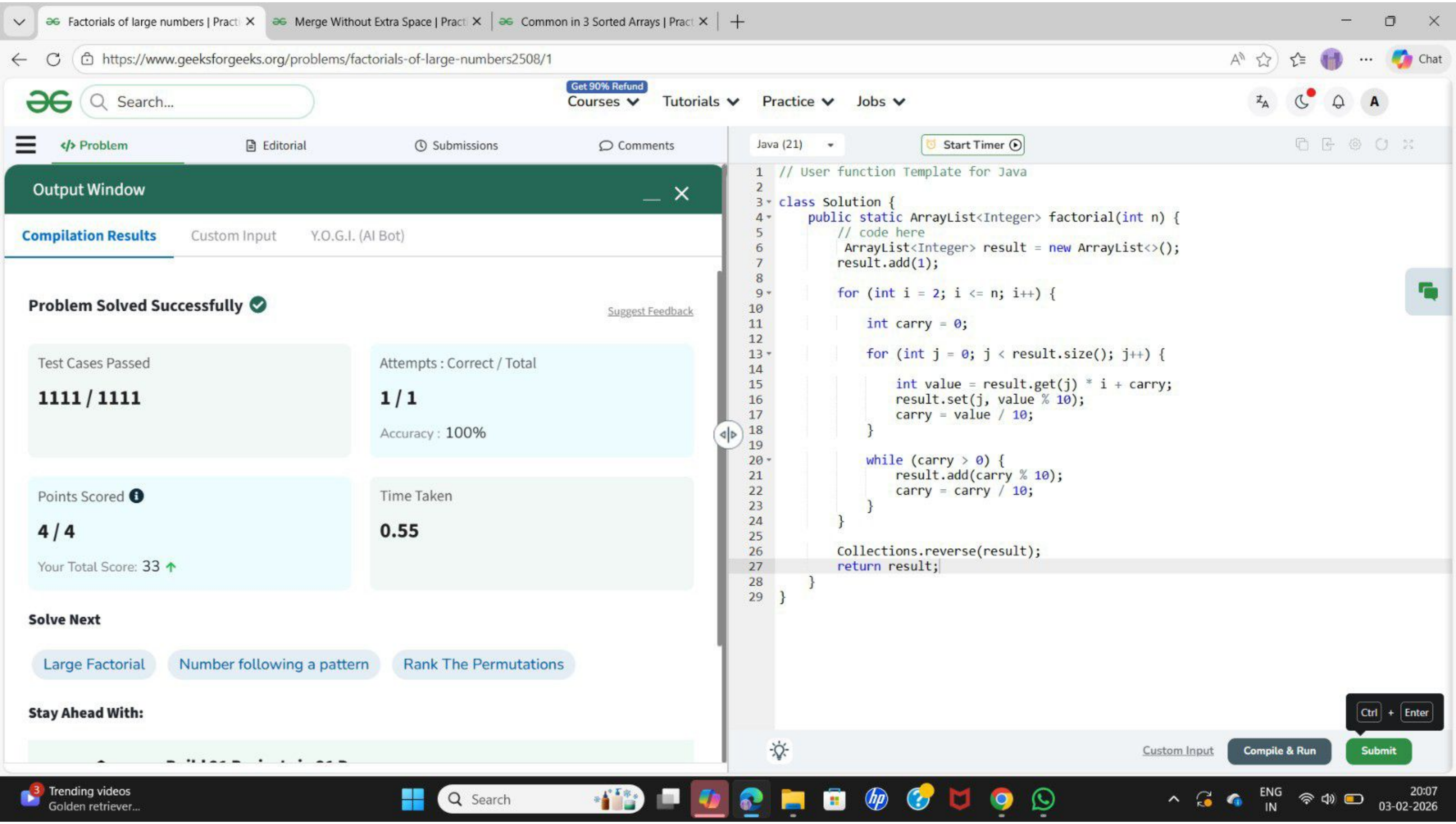
Input: n = 5
Output: [1, 2, 0]
Explanation: 5! = 1*2*3*4*5 = 120

Input: n = 10
Output: [3, 6, 2, 8, 8, 0, 0]
Explanation: 10! = 1*2*3*4*5*6*7*8*9*10 = 3628800

Input: n = 1
Output: [1]
Explanation: 1! = 1

Constraints:
$1 \le n \le 10^3$

```java
// User function Template for Java

class Solution {
    public static ArrayList<Integer> factorial(int n) {
        // code here
         ArrayList<Integer> result = new ArrayList<>();
        result.add(1);

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

            int carry = 0;

            for (int j = 0; j < result.size(); j++) {

                int value = result.get(j) * i + carry;
                result.set(j, value % 10);
                carry = value / 10;
            }

            while (carry > 0) {
                result.add(carry % 10);
                carry = carry / 10;
            }
        }

        Collections.reverse(result);
        return result;
    }
}
```


Factorials of large numbers | Pract
Merge Without Extra Space | Pract
Common in 3 Sorted Arrays | Pract
https://www.geeksforgeeks.org/problems/factorials-of-large-numbers2508/1
Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Java (21)
Start Timer

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

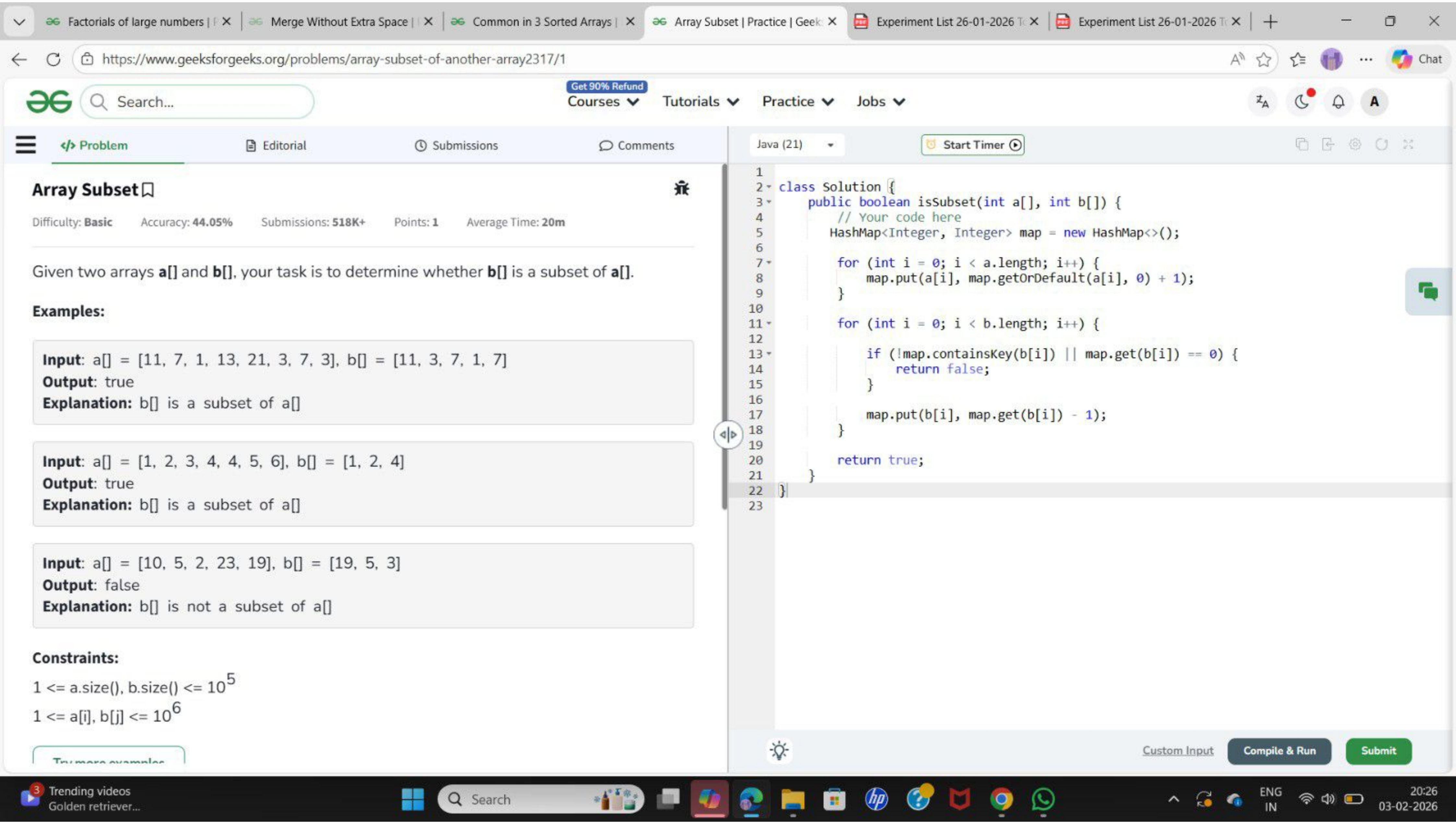
Time Taken
0.55

Solve Next

Large Factorial   Number following a pattern   Rank The Permutations

Stay Ahead With:

```java
// User function Template for Java

class Solution {
    public static ArrayList<Integer> factorial(int n) {
        // code here
         ArrayList<Integer> result = new ArrayList<>();
        result.add(1);

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

            int carry = 0;

            for (int j = 0; j < result.size(); j++) {

                int value = result.get(j) * i + carry;
                result.set(j, value % 10);
                carry = value / 10;
            }

            while (carry > 0) {
                result.add(carry % 10);
                carry = carry / 10;
            }
        }

        Collections.reverse(result);
        return result;
    }
}
```



https://www.geeksforgeeks.org/problems/array-subset-of-another-array2317/1

Array Subset

Difficulty: **Basic** Accuracy: **44.05%** Submissions: **518K+** Points: **1** Average Time: **20m**

Given two arrays **a[]** and **b[]**, your task is to determine whether **b[]** is a subset of **a[]**.

Examples:

Input: a[] = [11, 7, 1, 13, 21, 3, 7, 3], b[] = [11, 3, 7, 1, 7]
Output: true
Explanation: b[] is a subset of a[]

Input: a[] = [1, 2, 3, 4, 4, 5, 6], b[] = [1, 2, 4]
Output: true
Explanation: b[] is a subset of a[]

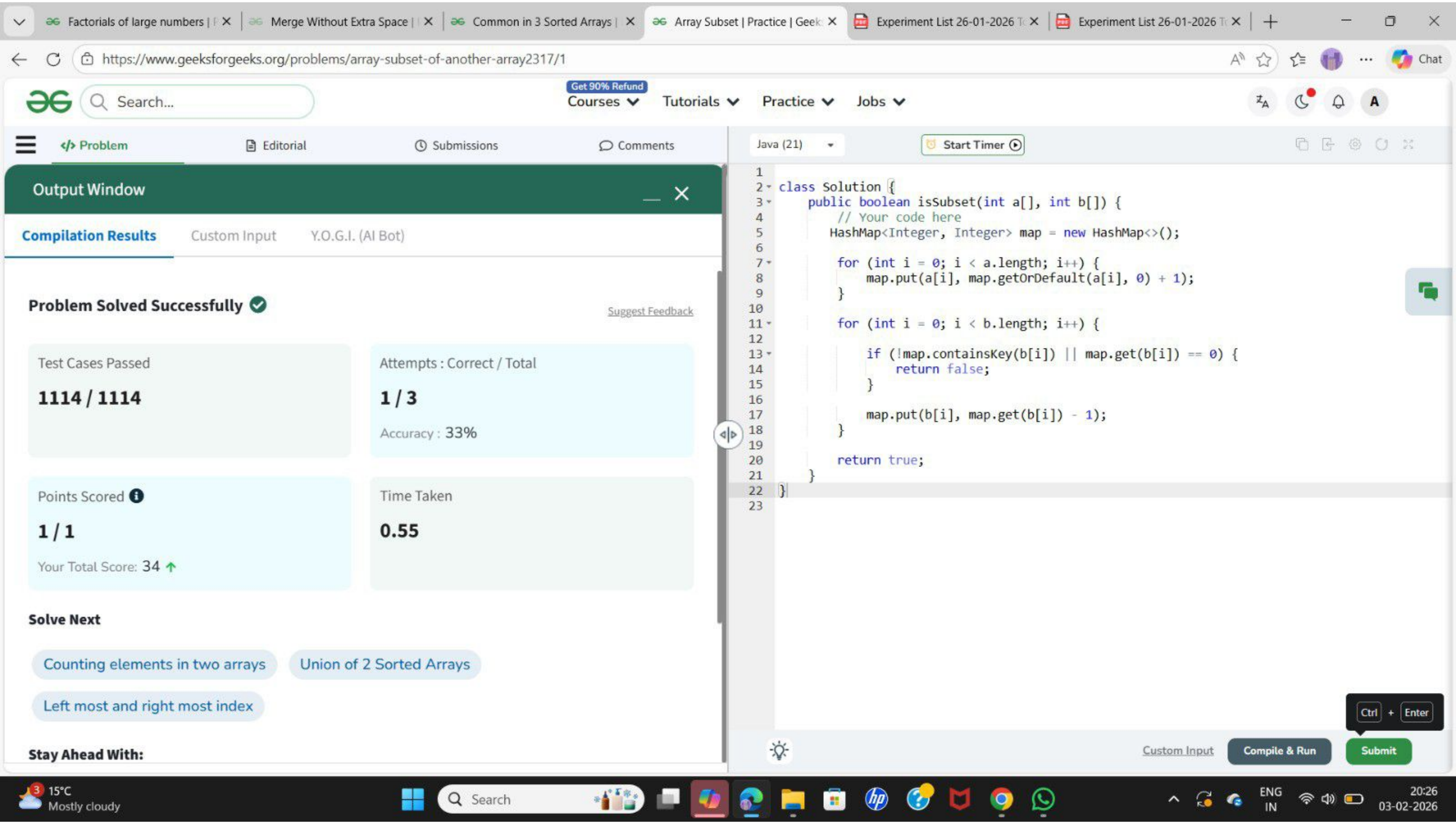
Input: a[] = [10, 5, 2, 23, 19], b[] = [19, 5, 3]
Output: false
Explanation: b[] is not a subset of a[]

Constraints:

1 <= a.size(), b.size() <= 10^5

1 <= a[i], b[j] <= 10^6

```java

class Solution {
    public boolean isSubset(int a[], int b[]) {
      // Your code here
       HashMap<Integer, Integer> map = new HashMap<>();

        for (int i = 0; i < a.length; i++) {
            map.put(a[i], map.getOrDefault(a[i], 0) + 1);
        }

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

            if (!map.containsKey(b[i]) || map.get(b[i]) == 0) {
                return false;
            }

            map.put(b[i], map.get(b[i]) - 1);
        }

        return true;
    }
}

```



https://www.geeksforgeeks.org/problems/array-subset-of-another-array2317/1

Output Window

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

Problem Solved Successfully

Test Cases Passed
1114 / 1114

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

Points Scored
1 / 1
Your Total Score: 34

Time Taken
0.55

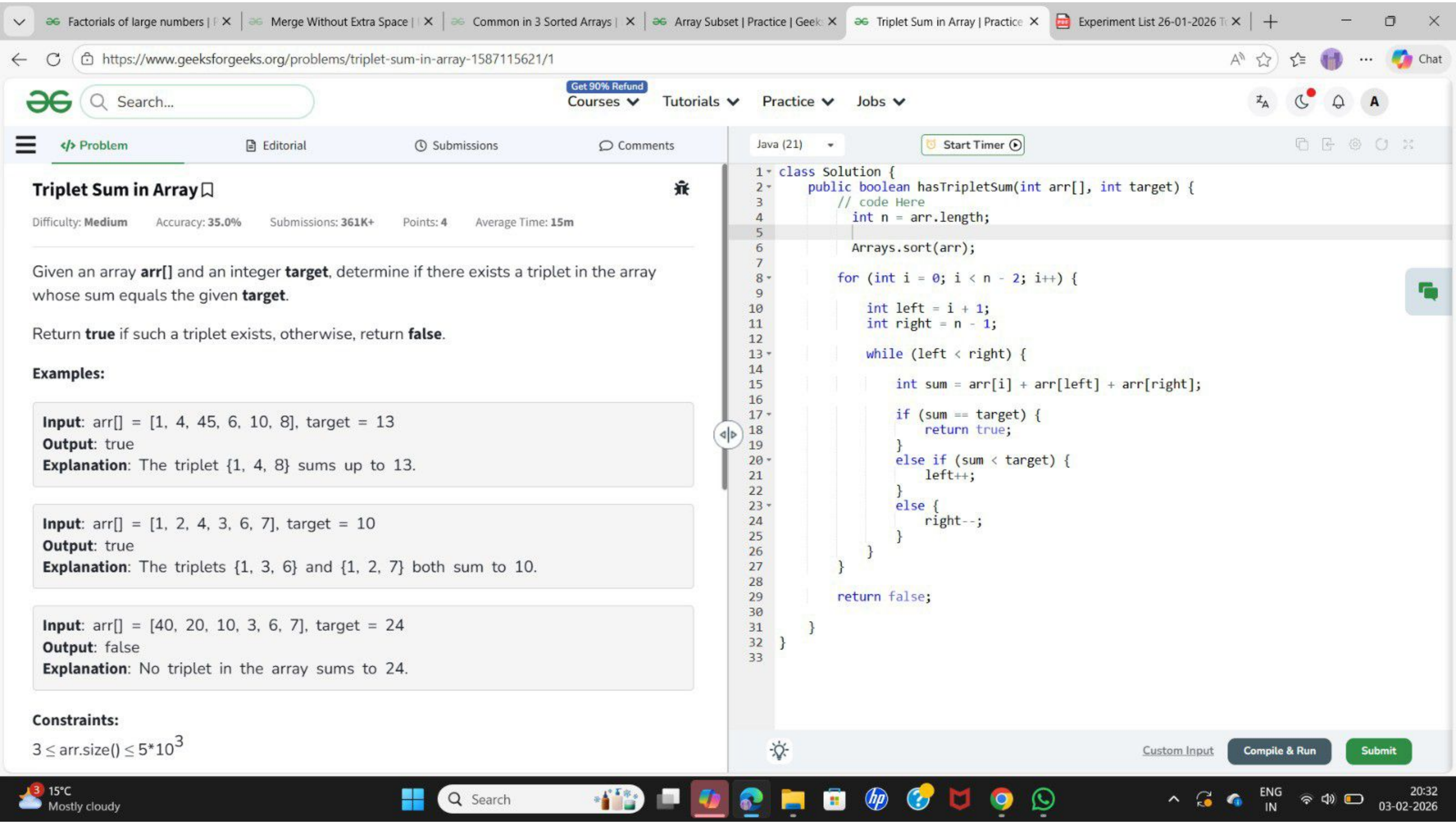
Solve Next

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

Stay Ahead With:

Java (21)

```java
class Solution {
    public boolean isSubset(int a[], int b[]) {
      // Your code here
     HashMap<Integer, Integer> map = new HashMap<>();

      for (int i = 0; i < a.length; i++) {
          map.put(a[i], map.getOrDefault(a[i], 0) + 1);
      }

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

          if (!map.containsKey(b[i]) || map.get(b[i]) == 0) {
              return false;
          }

          map.put(b[i], map.get(b[i]) - 1);
      }

      return true;
    }
}
```


Factorials of large numbers |
Merge Without Extra Space |
Common in 3 Sorted Arrays |
Array Subset | Practice | Geek
Triplet Sum in Array | Practice
Experiment List 26-01-2026
https://www.geeksforgeeks.org/problems/triplet-sum-in-array-1587115621/1
Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Java (21)
Start Timer
Custom Input
Compile & Run
Submit
15°C
Mostly cloudy
Search
ENG
IN
20:32
03-02-2026

Triplet Sum in Array

Difficulty: **Medium** Accuracy: **35.0%** Submissions: **361K+** Points: **4** Average Time: **15m**

Given an array **arr[]** and an integer **target**, determine if there exists a triplet in the array whose sum equals the given **target**.

Return **true** if such a triplet exists, otherwise, return **false**.

Examples:

Input: arr[] = [1, 4, 45, 6, 10, 8], target = 13
Output: true
Explanation: The triplet {1, 4, 8} sums up to 13.

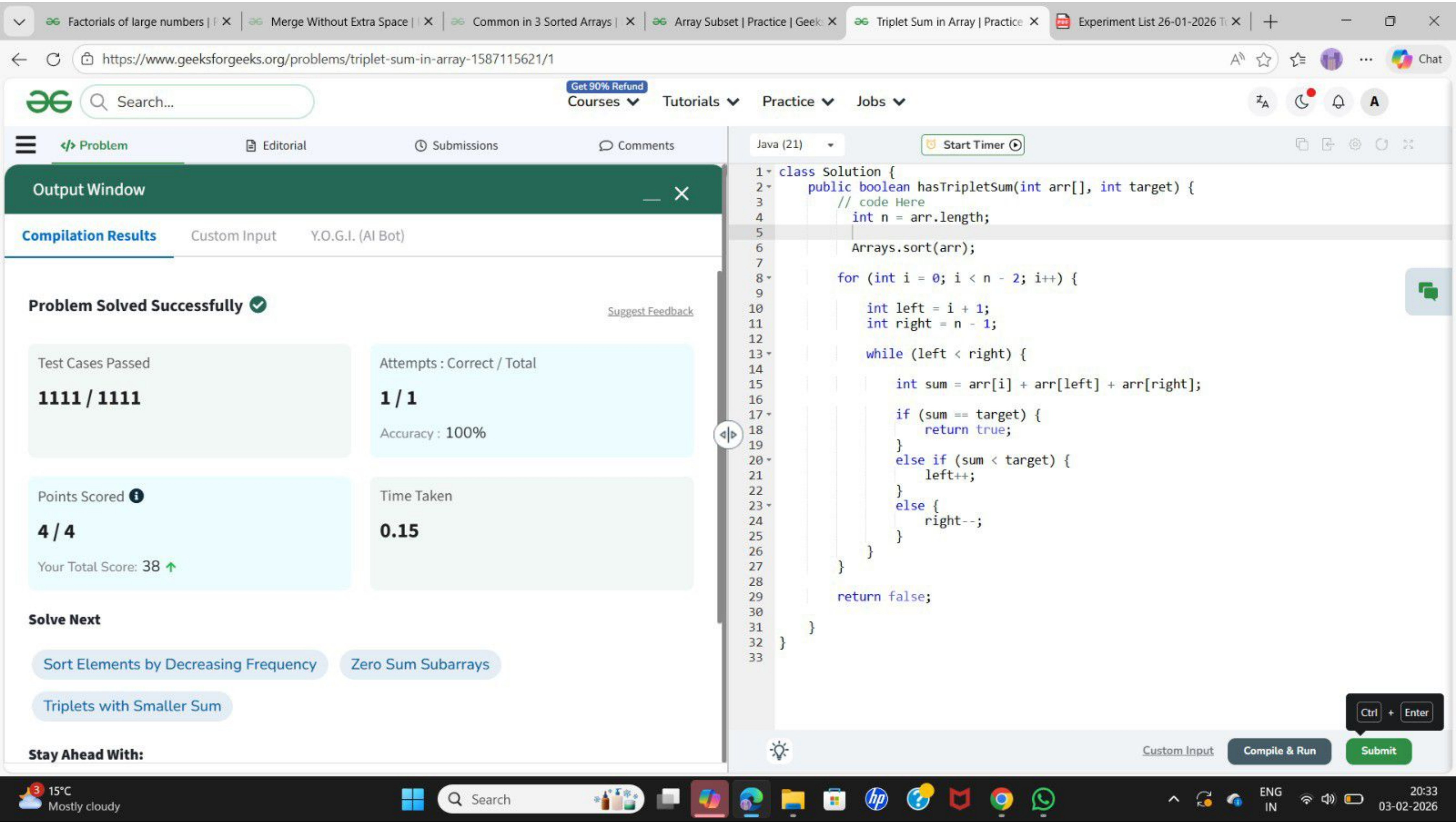
Input: arr[] = [1, 2, 4, 3, 6, 7], target = 10
Output: true
Explanation: The triplets {1, 3, 6} and {1, 2, 7} both sum to 10.

Input: arr[] = [40, 20, 10, 3, 6, 7], target = 24
Output: false
Explanation: No triplet in the array sums to 24.

Constraints:

$3 \le arr.size() \le 5*10^3$

```java
class Solution {
    public boolean hasTripletSum(int arr[], int target) {
        // code Here
         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 sum = arr[i] + arr[left] + arr[right];

                if (sum == target) {
                    return true;
                }
                else if (sum < target) {
                    left++;
                }
                else {
                    right--;
                }
            }
        }

        return false;

    }
}
```



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

Points Scored
4 / 4
Your Total Score: 38

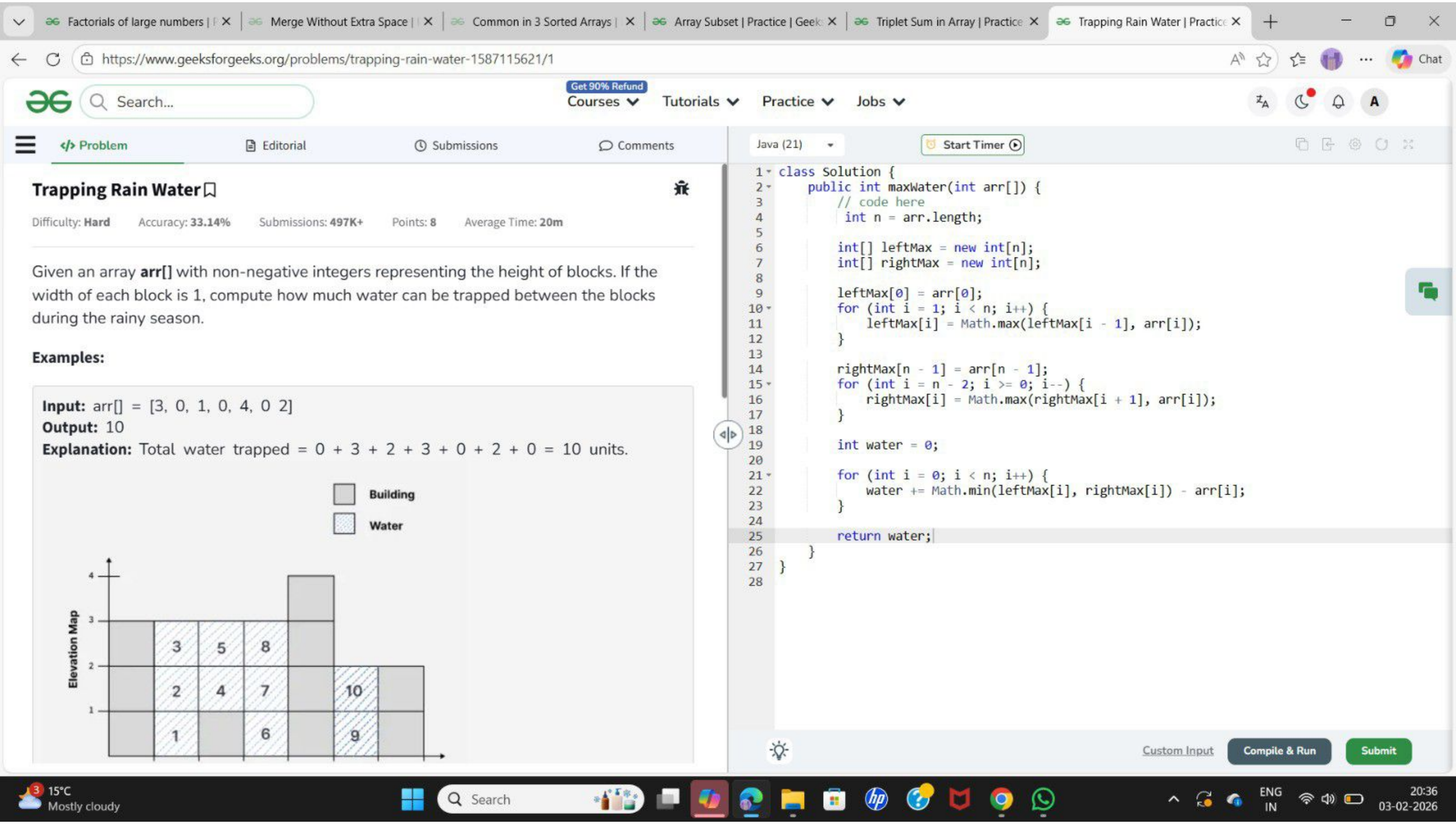
Time Taken
0.15

Solve Next

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

Stay Ahead With:

```java
class Solution {
    public boolean hasTripletSum(int arr[], int target) {
        // code Here
         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 sum = arr[i] + arr[left] + arr[right];

                if (sum == target) {
                    return true;
                }
                else if (sum < target) {
                    left++;
                }
                else {
                    right--;
                }
            }
        }

        return false;

    }
}
```



https://www.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1

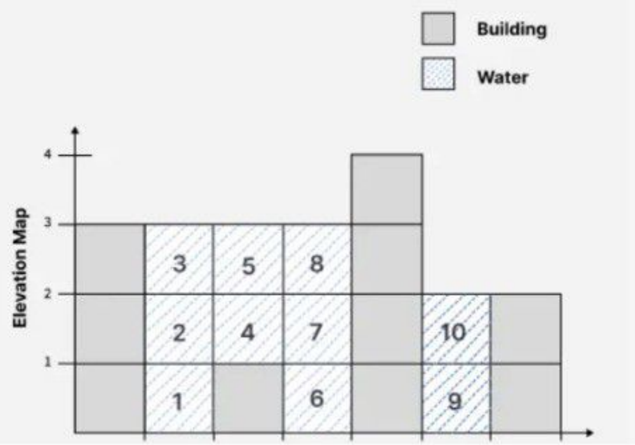
Trapping Rain Water

Difficulty: **Hard** Accuracy: **33.14%** Submissions: **497K+** Points: **8** Average Time: **20m**

Given an array **arr[]** with non-negative integers representing the height of blocks. If the width of each block is 1, compute how much water can be trapped between the blocks during the rainy season.

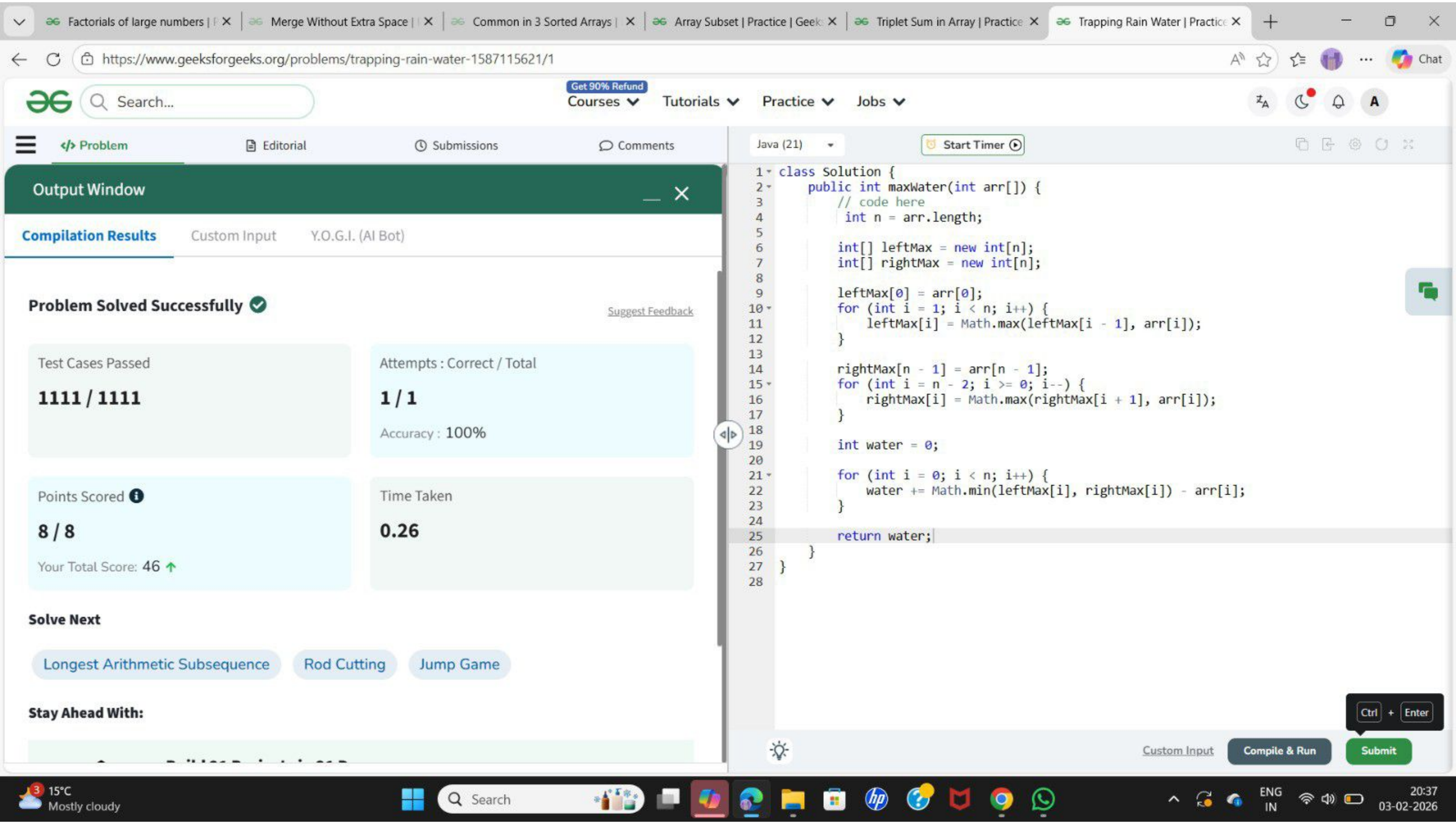
Examples:

Input: arr[] = [3, 0, 1, 0, 4, 0 2]
Output: 10
Explanation: Total water trapped = 0 + 3 + 2 + 3 + 0 + 2 + 0 = 10 units.



Java (21)

```java
class Solution {
    public int maxWater(int arr[]) {
        // code here
         int n = arr.length;

        int[] leftMax = new int[n];
        int[] rightMax = new int[n];

        leftMax[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMax[i] = Math.max(leftMax[i - 1], arr[i]);
        }

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

        int water = 0;

        for (int i = 0; i < n; i++) {
            water += Math.min(leftMax[i], rightMax[i]) - arr[i];
        }

        return water;
    }
}
```



Factorials of large numbers
Merge Without Extra Space
Common in 3 Sorted Arrays
Array Subset | Practice | Geek
Triplet Sum in Array | Practice
Trapping Rain Water | Practice
https://www.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1
Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Java (21)
Start Timer
Custom Input
Compile & Run
Submit
Ctrl + Enter

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: 0.26

Solve Next

Longest Arithmetic Subsequence · Rod Cutting · Jump Game

Stay Ahead With:

```java
class Solution {
    public int maxWater(int arr[]) {
        // code here
         int n = arr.length;

        int[] leftMax = new int[n];
        int[] rightMax = new int[n];

        leftMax[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMax[i] = Math.max(leftMax[i - 1], arr[i]);
        }

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

        int water = 0;

        for (int i = 0; i < n; i++) {
            water += Math.min(leftMax[i], rightMax[i]) - arr[i];
        }

        return water;
    }
}
```

15°C Mostly cloudy · Search · ENG IN · 20:37 03-02-2026