**1)Kth Smallest**

Given an array **arr[]** and an integer **k** where k is smaller than the size of the array, the task is to find the **kth smallest** element in the given array.

**Follow up:** Don't solve it using the inbuilt sort function.

**Examples :**

**Input:** arr[] = [7, 10, 4, 3, 20, 15], k = 3

**Output:** 7

**Explanation:** 3rd smallest element in the given array is 7.

**Input:** arr[] = [2, 3, 1, 20, 15], k = 4

**Output:** 15

**Explanation:** 4th smallest element in the given array is 15.

**Expected Time Complexity:**O(n+(max\_element) )

**Expected Auxiliary Space:**O(max\_element)

**Constraints:**  
1 <= arr.size <= 106  
1<= arr[i] <= 1061 <= k <= n

**Code:**  
  
import java.util.\*;

public class KthSmallest {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter array size:");

        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter array elements:");

        for(int i=0; i<n; i++){

            arr[i] = sc.nextInt();

        }

        System.out.println("Enter K:");

        int k = sc.nextInt();

        int res = findKthSmallest(arr, k);

        System.out.println(res);

        sc.close();

    }

    private static int findKthSmallest(int[] arr, int k){

        int n = arr.length;

        int res = -1;

        for(int i=0; i<k; i++){

            int min = Integer.MAX\_VALUE, minInd = -1;

            for(int j=0; j<n; j++){

                if(arr[j] == -1) continue;

                else{

                    if(arr[j] < min){

                        min = arr[j];

                        minInd = j;

                    }

                }

            }

            if(i != k-1)

                arr[minInd] = -1;

            else{

                res = min;

                break;

            }

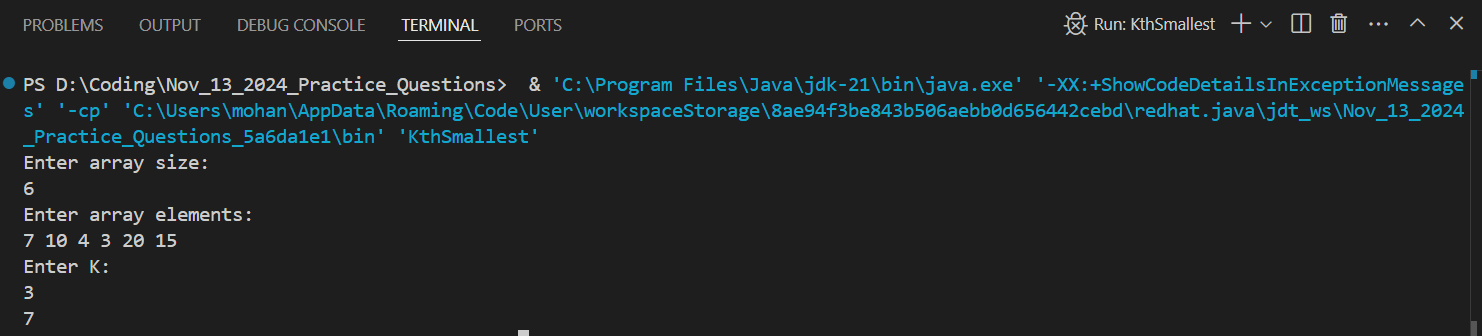
        }

        return res;

    }

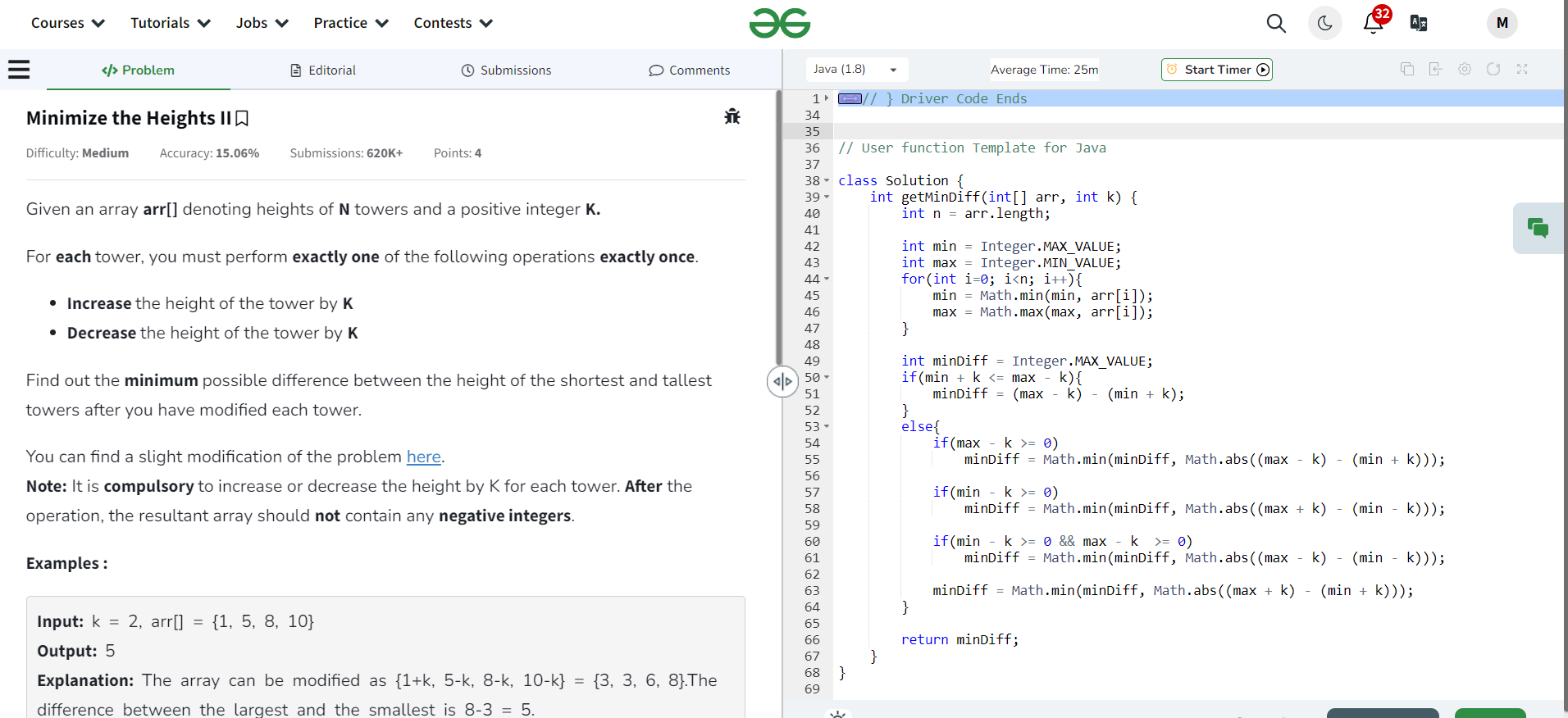
}

**Output:**



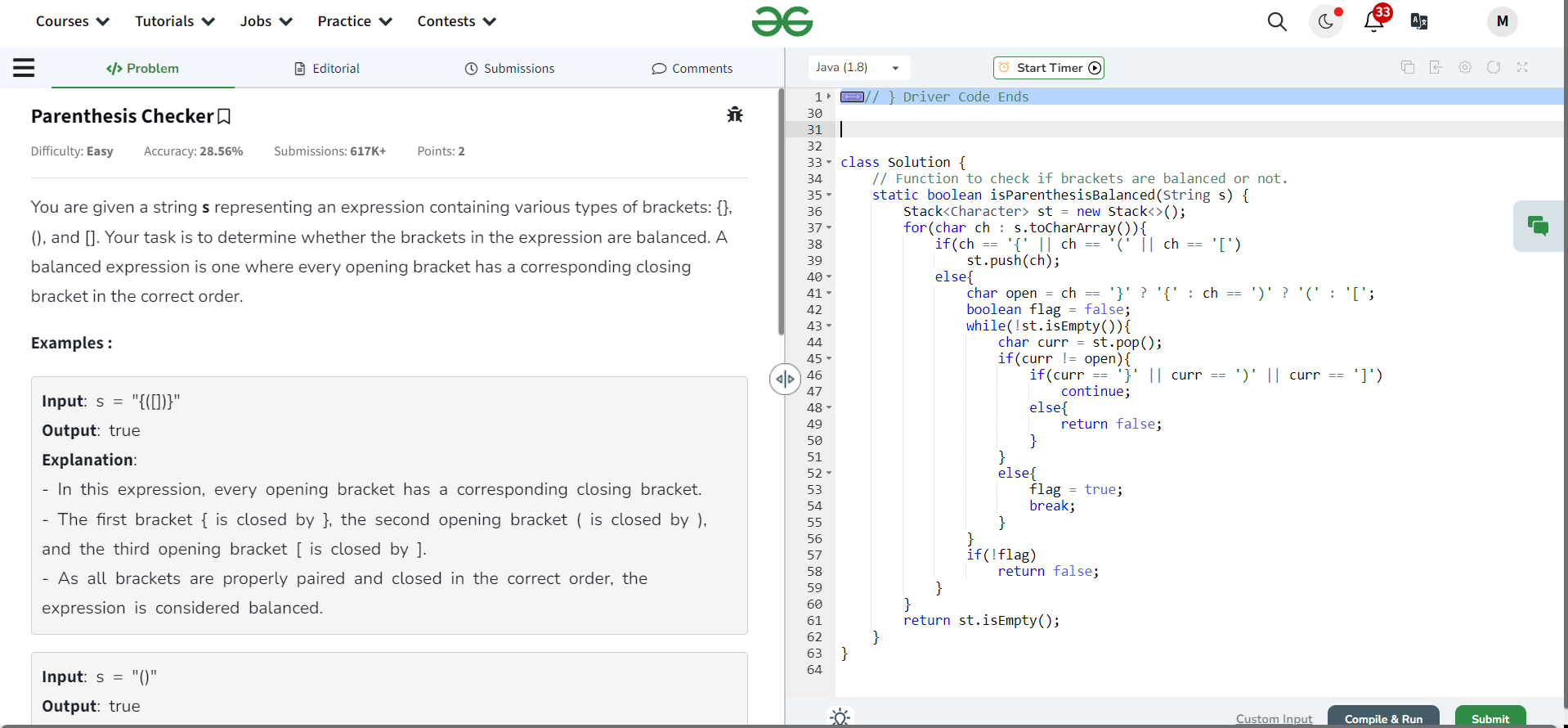
**Time Complexity:** O(k \* n)

2) Minimize Heights II

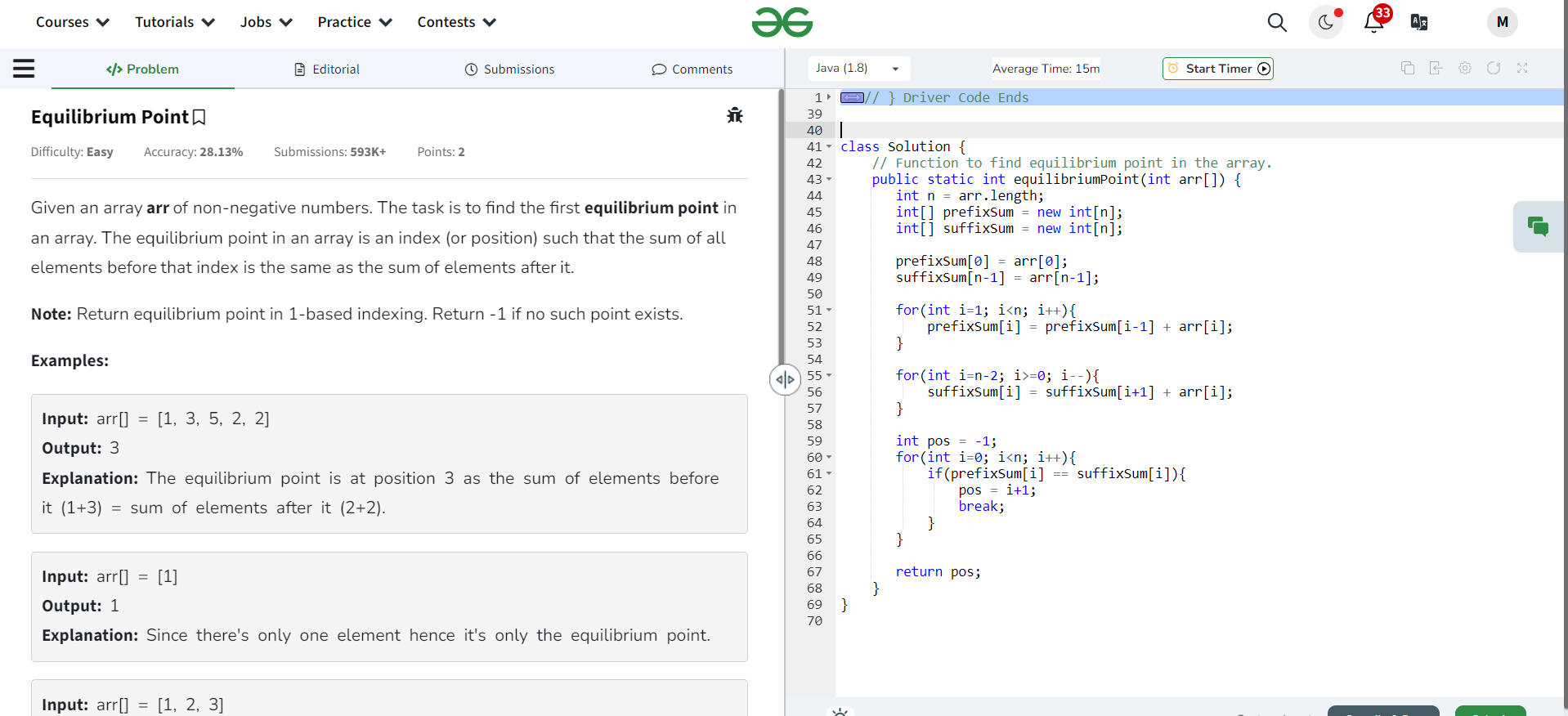


Time Complexity: O(n log n)

3) Parenthesis checker

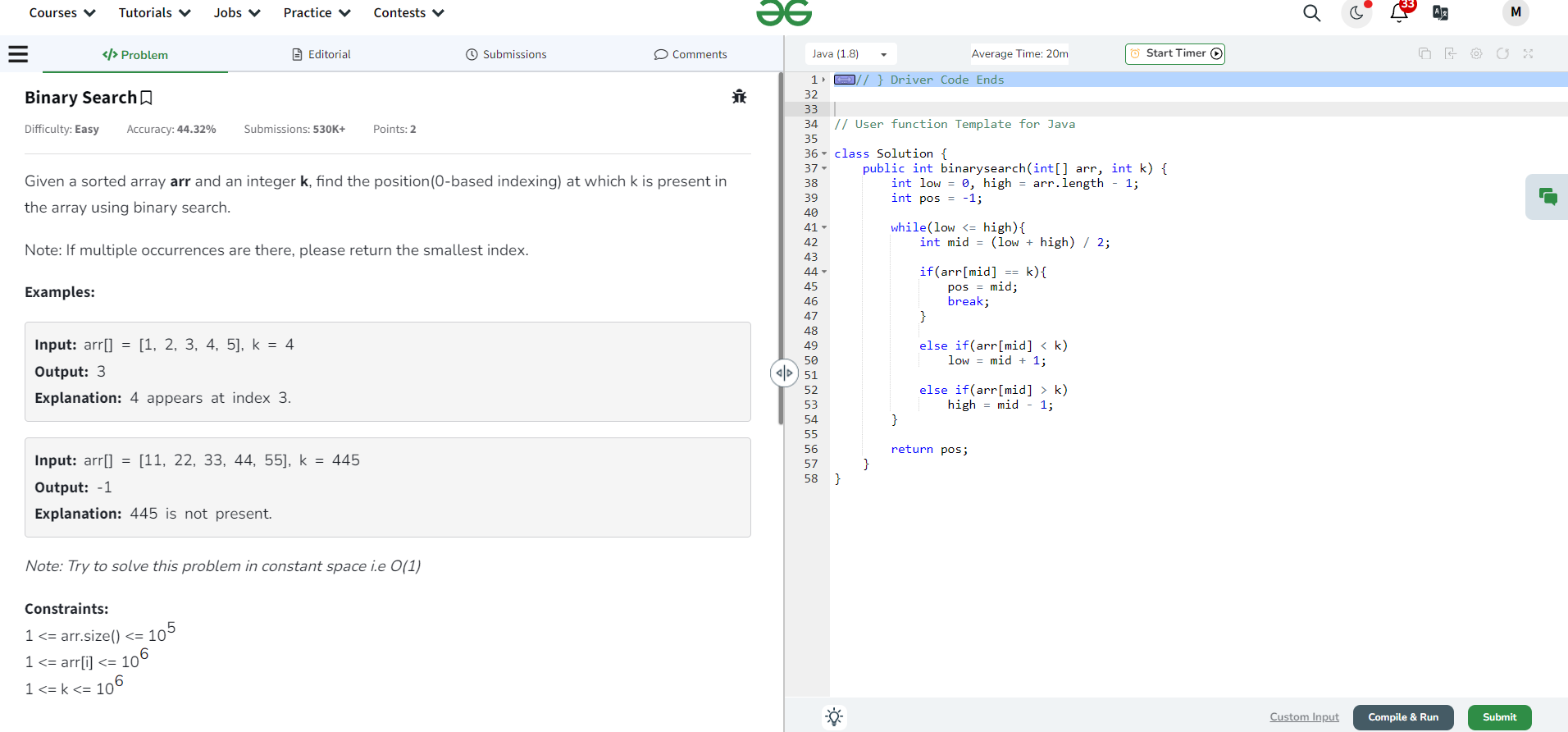
  
  
Time Complexity: O(n)

4) Equilibrium Point



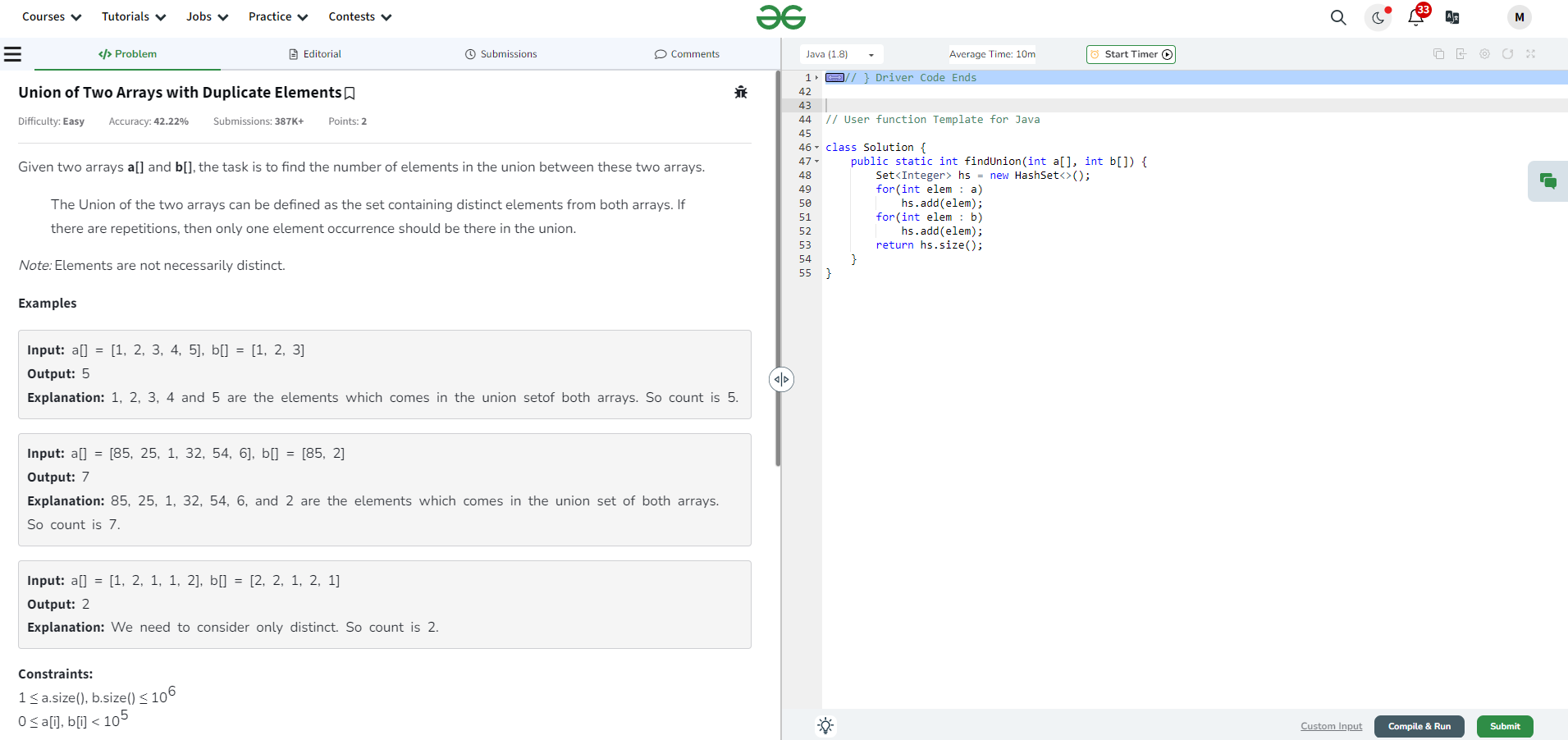
Time Complexity: O(n)

5) Binary Search



Time Complexity: O(log n)

6) Next Greater Element has already completed in practice set 1

7) Union of two arrays with duplicate numbers  
  
Time Complexity: O(n)