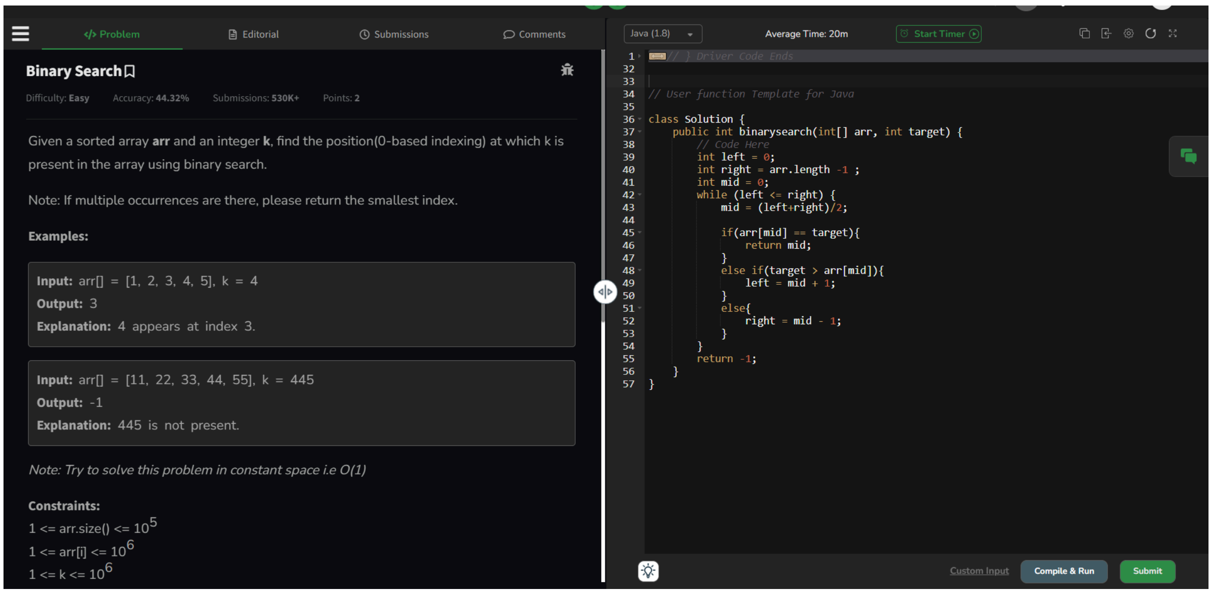
DSA PRACTICE - 3

1.Binary Search

Time Complexity : O(logn)



Binary Search

Difficulty: Easy   Accuracy: 44.32%   Submissions: 530K+   Points: 2

Given a sorted array **arr** and an integer **k**, find the position(0-based indexing) at which k is present in the array using binary search.

Note: If multiple occurrences are there, please return the smallest index.

Examples:

Input: arr[] = [1, 2, 3, 4, 5], k = 4
Output: 3
Explanation: 4 appears at index 3.

Input: arr[] = [11, 22, 33, 44, 55], k = 445
Output: -1
Explanation: 445 is not present.

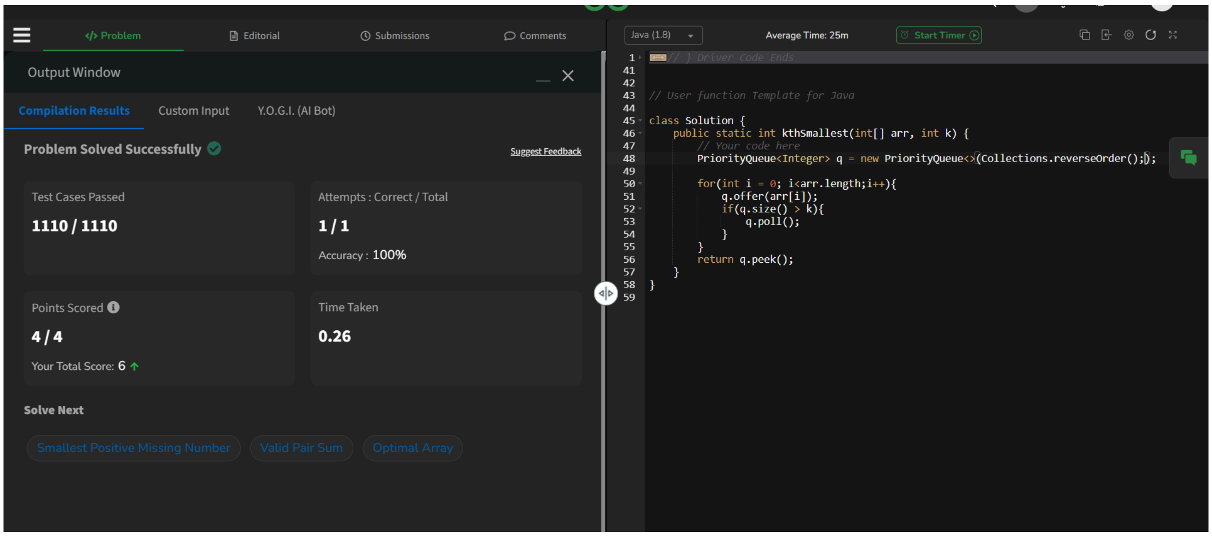
Note: Try to solve this problem in constant space i.e O(1)

Constraints:
1 <= arr.size() <= 10^5
1 <= arr[i] <= 10^6
1 <= k <= 10^6

```java
// } Driver Code Ends


// User function Template for Java

class Solution {
    public int binarysearch(int[] arr, int target) {
        // Code Here
        int left = 0;
        int right = arr.length -1 ;
        int mid = 0;
        while (left <= right) {
            mid = (left+right)/2;

            if(arr[mid] == target){
                return mid;
            }
            else if(target > arr[mid]){
                left = mid + 1;
            }
            else{
                right = mid - 1;
            }
        }
        return -1;
    }
}
```

2. Kth smallest Element

Time Complexity : O(n)



Output Window

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

Problem Solved Successfully

```
Test Cases Passed        Attempts : Correct / Total
1110 / 1110              1 / 1
                         Accuracy : 100%

Points Scored            Time Taken
4 / 4                    0.26
Your Total Score: 6
```

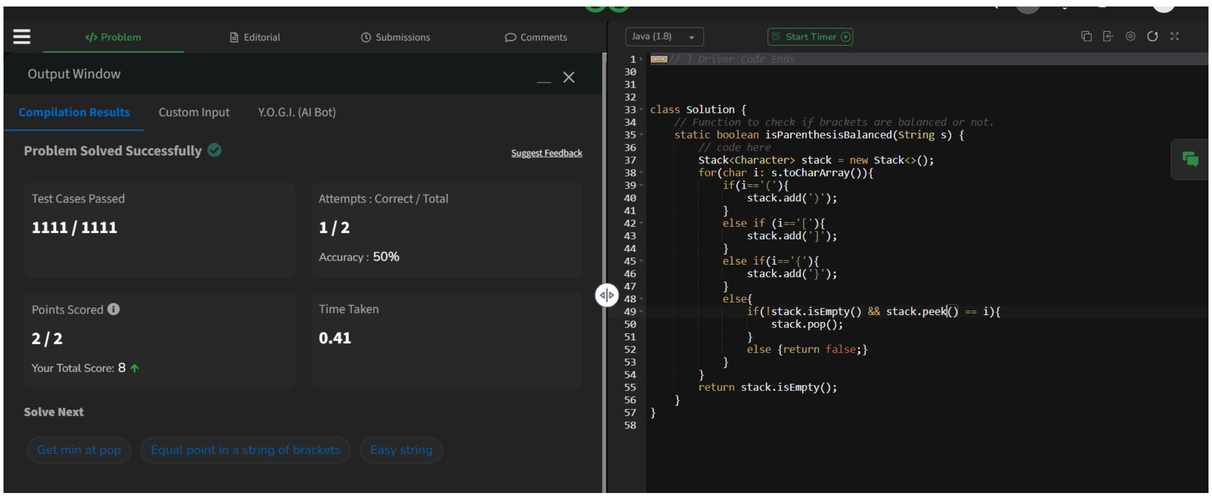
Solve Next

Smallest Positive Missing Number   Valid Pair Sum   Optimal Array

```java
// } Driver Code Ends


// User function Template for Java

class Solution {
    public static int kthSmallest(int[] arr, int k) {
        // Your code here
        PriorityQueue<Integer> q = new PriorityQueue<>(Collections.reverseOrder(););

        for(int i = 0; i<arr.length;i++){
            q.offer(arr[i]);
            if(q.size() > k){
                q.poll();
            }
        }
        return q.peek();
    }
}

```

3. Parenthesis Checker

Time Complexity : O(n)



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

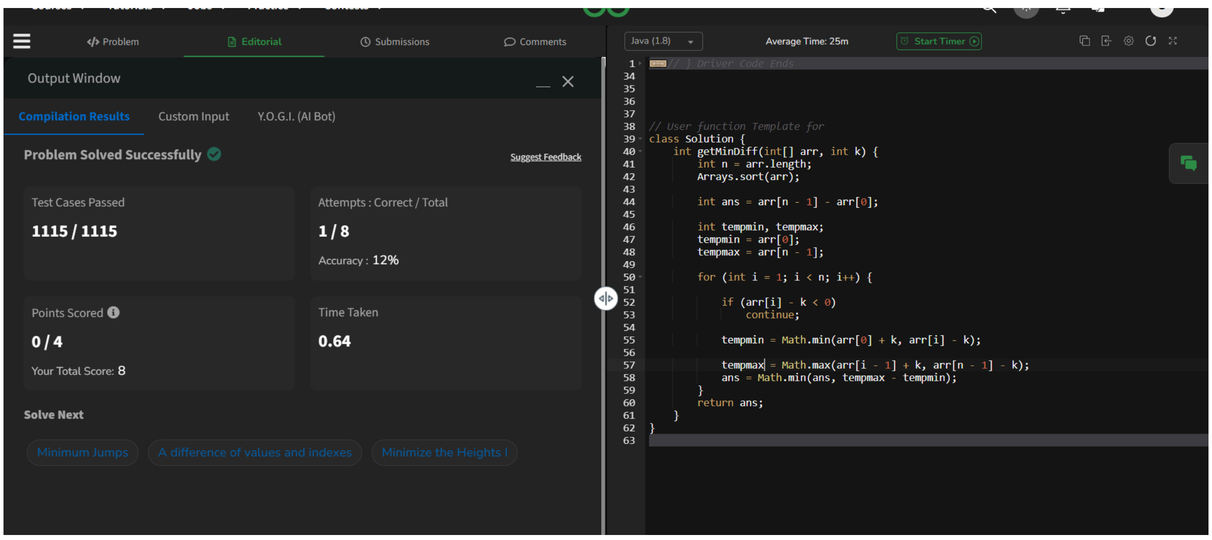
Points Scored
2 / 2
Your Total Score: 8

Time Taken
0.41

Solve Next
Get min at pop | Equal point in a string of brackets | Easy string

```java
// } Driver Code Ends



class Solution {
    // Function to check if brackets are balanced or not.
    static boolean isParenthesisBalanced(String s) {
        // code here
        Stack<Character> stack = new Stack<>();
        for(char i: s.toCharArray()){
            if(i=='('){
                stack.add(')');
            }
            else if (i=='['){
                stack.add(']');
            }
            else if(i=='{'){
                stack.add('}');
            }
            else{
                if(!stack.isEmpty() && stack.peek() == i){
                    stack.pop();
                }
                else {return false;}
            }
        }
        return stack.isEmpty();
    }
}
```

4. Minimize the heights - 2



Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 8
Accuracy : 12%

Points Scored
0 / 4
Your Total Score: 8

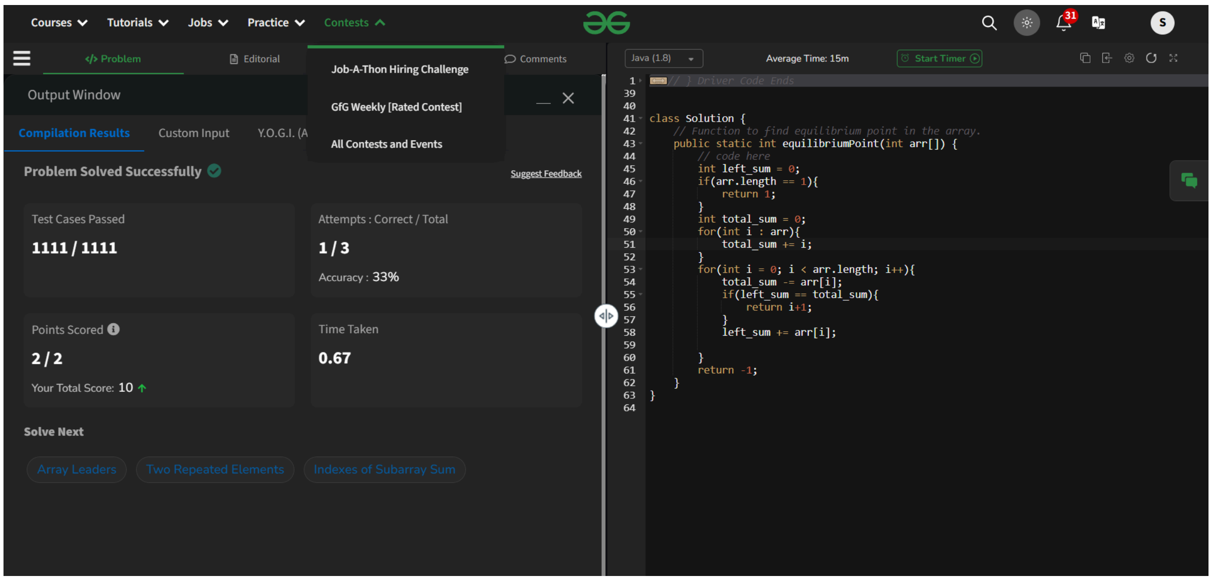
Time Taken
0.64

Solve Next
Minimum Jumps | A difference of values and indexes | Minimize the Heights I

```java
// } Driver Code Ends




// User function Template for
class Solution {
    int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        Arrays.sort(arr);

        int ans = arr[n - 1] - arr[0];

        int tempmin, tempmax;
        tempmin = arr[0];
        tempmax = arr[n - 1];

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

            if (arr[i] - k < 0)
                continue;

            tempmin = Math.min(arr[0] + k, arr[i] - k);

            tempmax = Math.max(arr[i - 1] + k, arr[n - 1] - k);
            ans = Math.min(ans, tempmax - tempmin);
        }
        return ans;
    }
}
```

Time Complexity : O(nlogn)

5. Equilibrium point



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

Points Scored
2 / 2
Your Total Score: 10

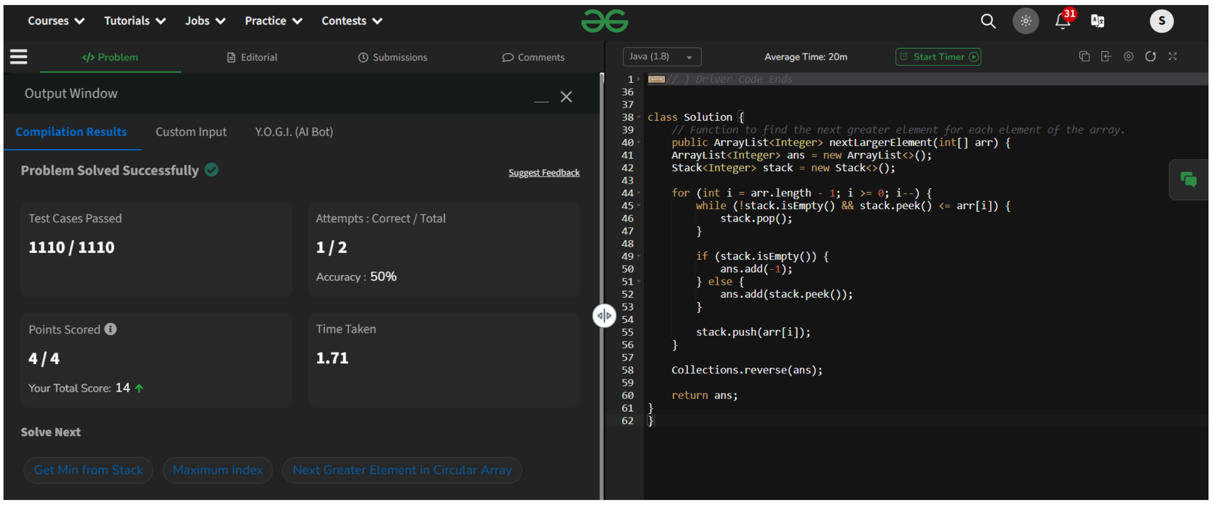
Time Taken
0.67

Solve Next
Array Leaders | Two Repeated Elements | Indexes of Subarray Sum

```java
// } Driver Code Ends


class Solution {
    // Function to find equilibrium point in the array.
    public static int equilibriumPoint(int arr[]) {
        // code here
        int left_sum = 0;
        if(arr.length == 1){
            return 1;
        }
        int total_sum = 0;
        for(int i : arr){
            total_sum += i;
        }
        for(int i = 0; i < arr.length; i++){
            total_sum -= arr[i];
            if(left_sum == total_sum){
                return i+1;
            }
            left_sum += arr[i];

        }
        return -1;
    }
}
```

Time Complexity : O(n)

6.Next Greater Element



Problem Solved Successfully

Test Cases Passed
1110 / 1110

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

Points Scored
4 / 4
Your Total Score: 14

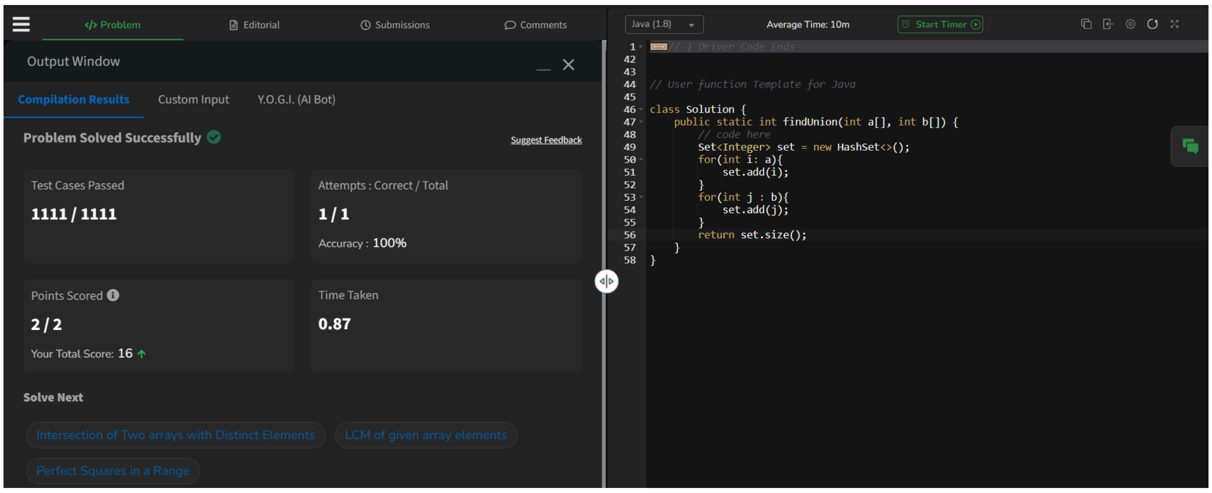
Time Taken
1.71

Solve Next
Get Min from Stack | Maximum Index | Next Greater Element in Circular Array

```java
// } Driver Code Ends


class Solution {
    // Function to find the next greater element for each element of the array.
    public ArrayList<Integer> nextLargerElement(int[] arr) {
    ArrayList<Integer> ans = new ArrayList<>();
    Stack<Integer> stack = new Stack<>();

    for (int i = arr.length - 1; i >= 0; i--) {
        while (!stack.isEmpty() && stack.peek() <= arr[i]) {
            stack.pop();
        }

        if (stack.isEmpty()) {
            ans.add(-1);
        } else {
            ans.add(stack.peek());
        }

        stack.push(arr[i]);
    }

    Collections.reverse(ans);

    return ans;
}
}
```

Time Complexity : O(n)

7.Union of two arrays with duplicate elements



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
2 / 2
Your Total Score: 16

Time Taken
0.87

Solve Next

Intersection of Two arrays with Distinct Elements | LCM of given array elements | Perfect Squares in a Range

```java
// } Driver Code Ends


// User function Template for Java

class Solution {
    public static int findUnion(int a[], int b[]) {
        // code here
        Set<Integer> set = new HashSet<>();
        for(int i: a){
            set.add(i);
        }
        for(int j : b){
            set.add(j);
        }
        return set.size();
    }
}
```

Time Complexity : O(n+m)