

DSA Practice – 4

1.Kth Smallest element

Output Window

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

Problem Solved Successfully ✓ [Suggest Feedback](#)

Test Cases Passed: 1110 / 1110

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 4 / 4

Time Taken: 0.21

Your Total Score: 39 ↑

```
1 // Driver Code Ends
2 class Solution {
3     public static int kthSmallest(int[] arr, int k) {
4         int maxElement = 0;
5         for (int num : arr) {
6             if (num > maxElement) {
7                 maxElement = num;
8             }
9         }
10        int[] frequency = new int[maxElement + 1];
11        for (int num : arr) {
12            frequency[num]++;
13        }
14        int count = 0;
15        for (int i = 0; i <= maxElement; i++) {
16            count += frequency[i];
17            if (count >= k) {
18                return i;
19            }
20        }
21        return -1;
22    }
23 }
```

Time Complexity : $O(n \log n)$

2.Parentheses Checker

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

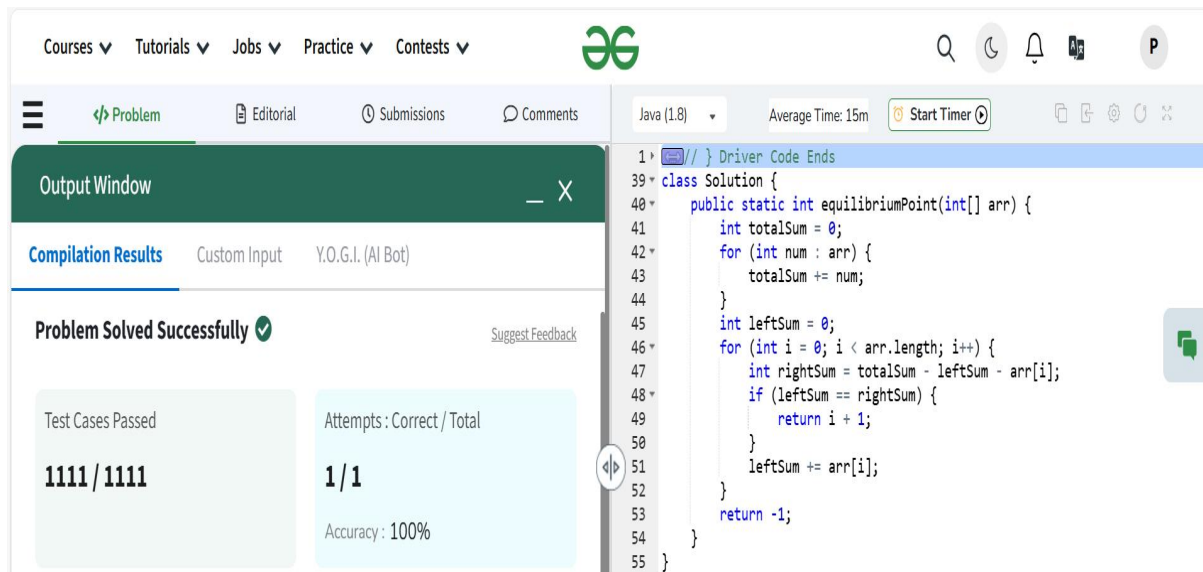
Time Taken: 0.34

Your Total Score: 41 ↑

```
1 // Driver Code Ends
2 class Solution {
3     static boolean isParenthesisBalanced(String s) {
4         Stack<Character> stack = new Stack<>();
5         for (char ch : s.toCharArray()) {
6             if (ch == '(' || ch == '{' || ch == '[') {
7                 stack.push(ch);
8             }
9             else if (ch == ')' || ch == '}' || ch == ']') {
10                if (stack.isEmpty() || !isMatchingPair(stack.pop(), ch)) {
11                    return false;
12                }
13            }
14        }
15        return stack.isEmpty();
16    }
17     private static boolean isMatchingPair(char open, char close) {
18         return (open == '(' && close == ')') ||
19             (open == '{' && close == '}') ||
20             (open == '[' && close == ']');
21     }
22 }
```

Time Complexity : $O(n)$

3. Equilibrium Point

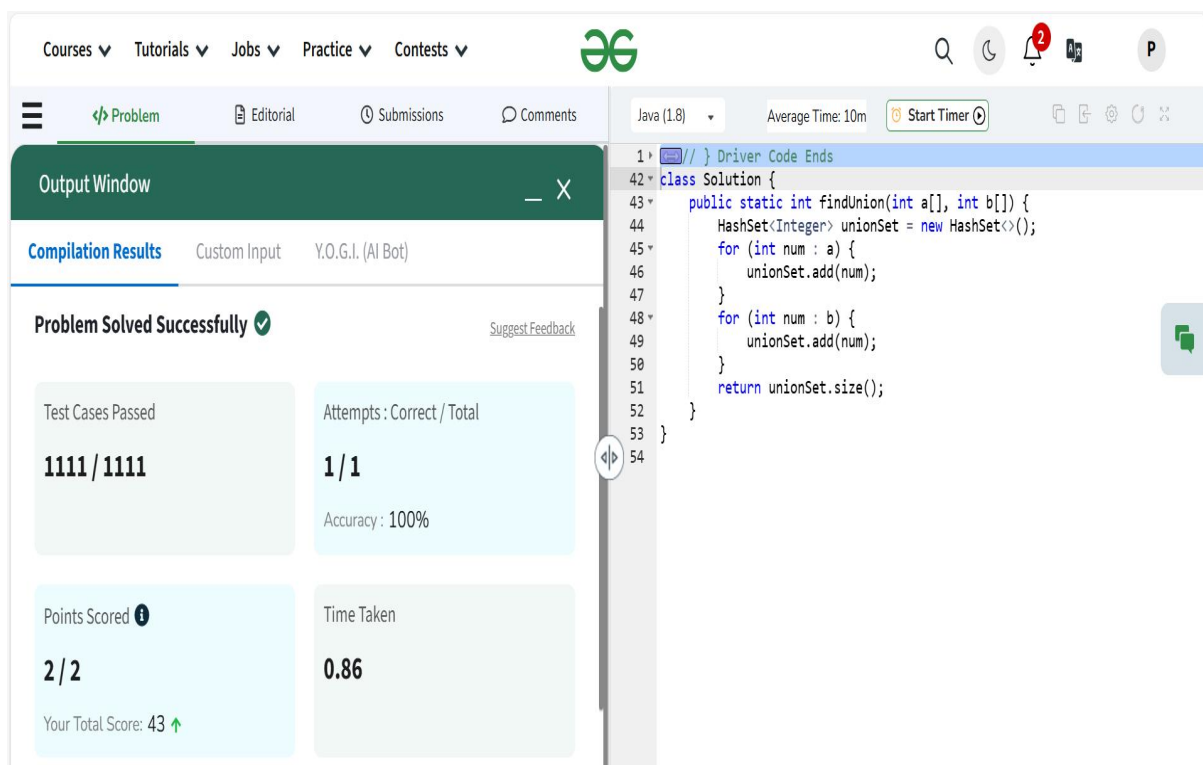


The screenshot shows a coding platform interface. The top navigation bar includes 'Courses', 'Tutorials', 'Jobs', 'Practice', and 'Contests'. The main header has a logo and search, refresh, and user profile icons. Below the header, there's a sub-header with 'Problem', 'Editorial', 'Submissions', and 'Comments' tabs. The 'Problem' tab is active, showing the 'Equilibrium Point' problem. The 'Output Window' on the left displays 'Problem Solved Successfully' with a green checkmark. It also shows 'Test Cases Passed: 1111 / 1111', 'Attempts: Correct / Total: 1 / 1', and 'Accuracy: 100%'. The code editor on the right shows a Java solution for finding the equilibrium point in an array.

```
1 // } Driver Code Ends
39 class Solution {
40     public static int equilibriumPoint(int[] arr) {
41         int totalSum = 0;
42         for (int num : arr) {
43             totalSum += num;
44         }
45         int leftSum = 0;
46         for (int i = 0; i < arr.length; i++) {
47             int rightSum = totalSum - leftSum - arr[i];
48             if (leftSum == rightSum) {
49                 return i + 1;
50             }
51             leftSum += arr[i];
52         }
53         return -1;
54     }
55 }
```

Time Complexity : $O(n)$

4. Union of arrays with duplicate



The screenshot shows a coding platform interface. The top navigation bar includes 'Courses', 'Tutorials', 'Jobs', 'Practice', and 'Contests'. The main header has a logo and search, refresh, and user profile icons. Below the header, there's a sub-header with 'Problem', 'Editorial', 'Submissions', and 'Comments' tabs. The 'Problem' tab is active, showing the 'Union of arrays with duplicate' problem. The 'Output Window' on the left displays 'Problem Solved Successfully' with a green checkmark. It also shows 'Test Cases Passed: 1111 / 1111', 'Attempts: Correct / Total: 1 / 1', and 'Accuracy: 100%'. The code editor on the right shows a Java solution for finding the union of two arrays using a HashSet.

```
1 // } Driver Code Ends
42 class Solution {
43     public static int findUnion(int a[], int b[]) {
44         HashSet<Integer> unionSet = new HashSet<>();
45         for (int num : a) {
46             unionSet.add(num);
47         }
48         for (int num : b) {
49             unionSet.add(num);
50         }
51         return unionSet.size();
52     }
53 }
54 }
```

Time Complexity : $O((n + m)\log(n + m))$