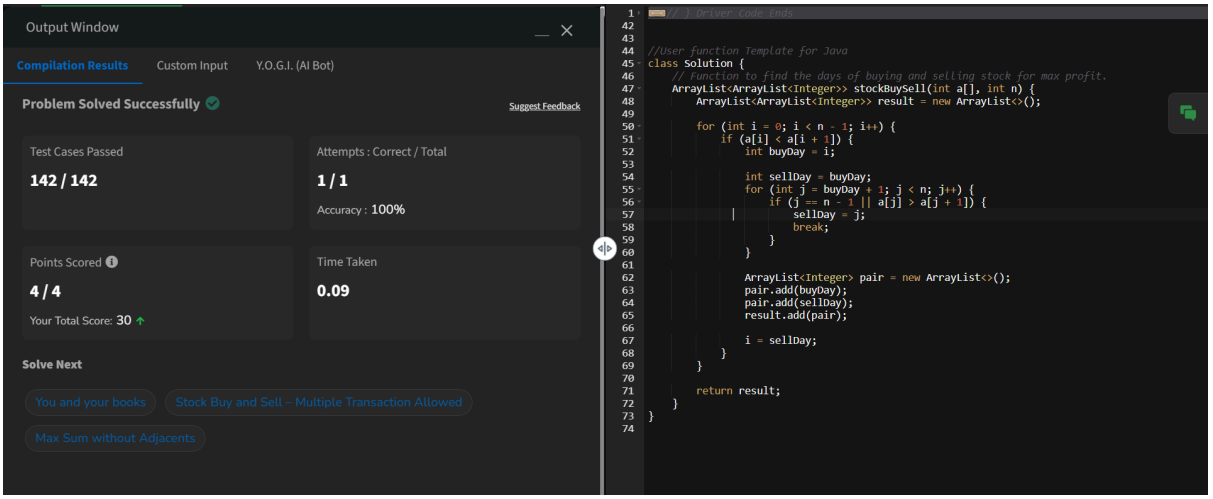
DSA PRACTICE - 5

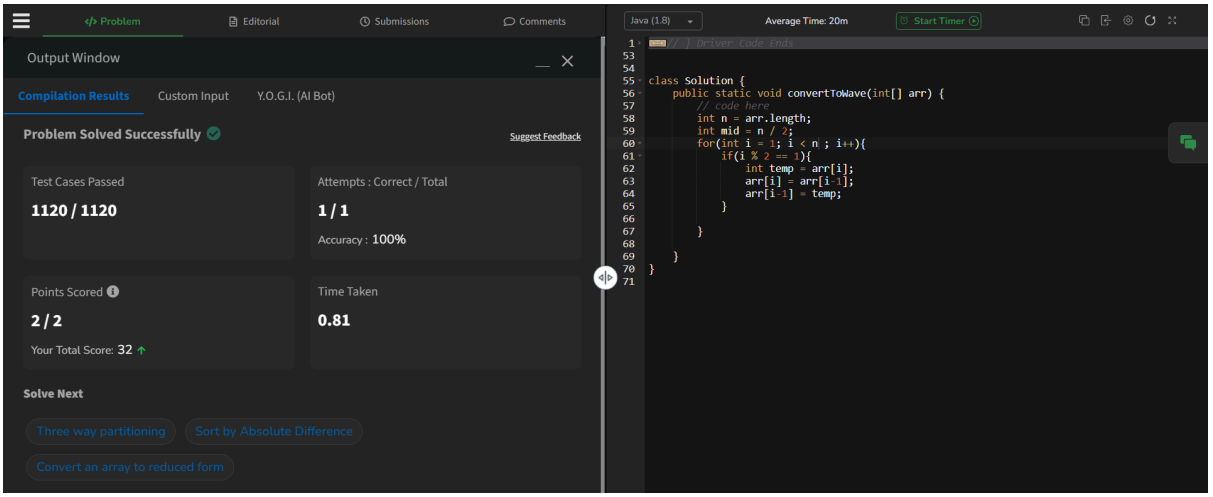
NAME : SIDHARTH N
ROLL NO : 22AD126

1.Buy and Sell Stock



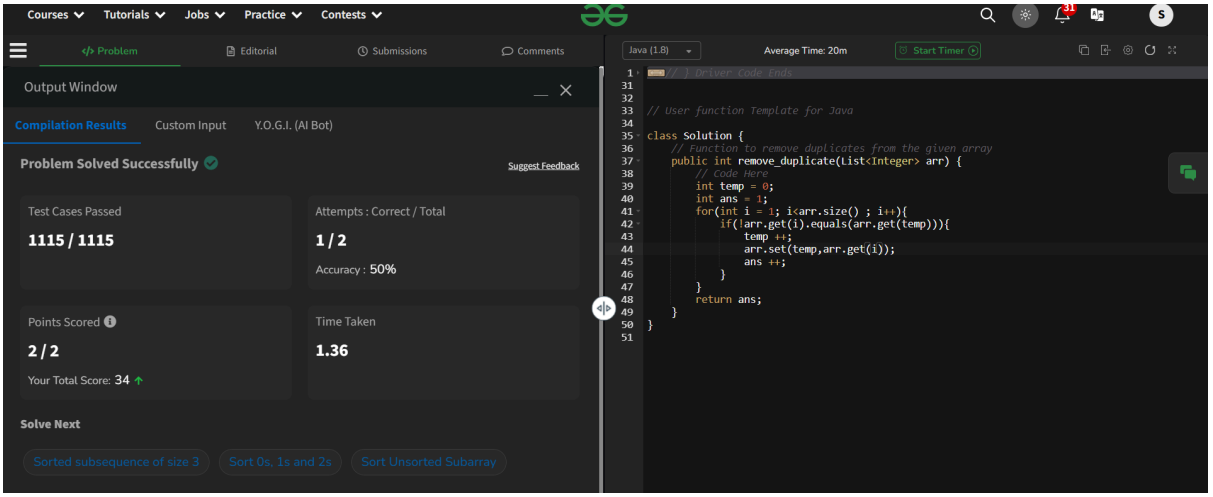
Time Complexity : O(n^2)

2.Wave Array



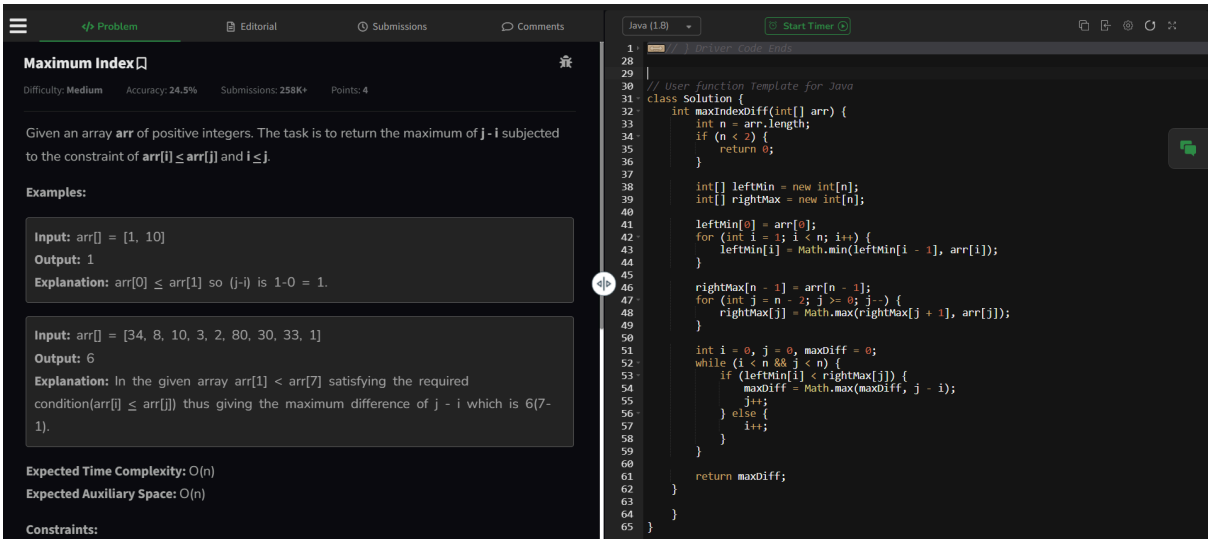
Time Complexity : O(n)

3.Remove duplicates sorted array



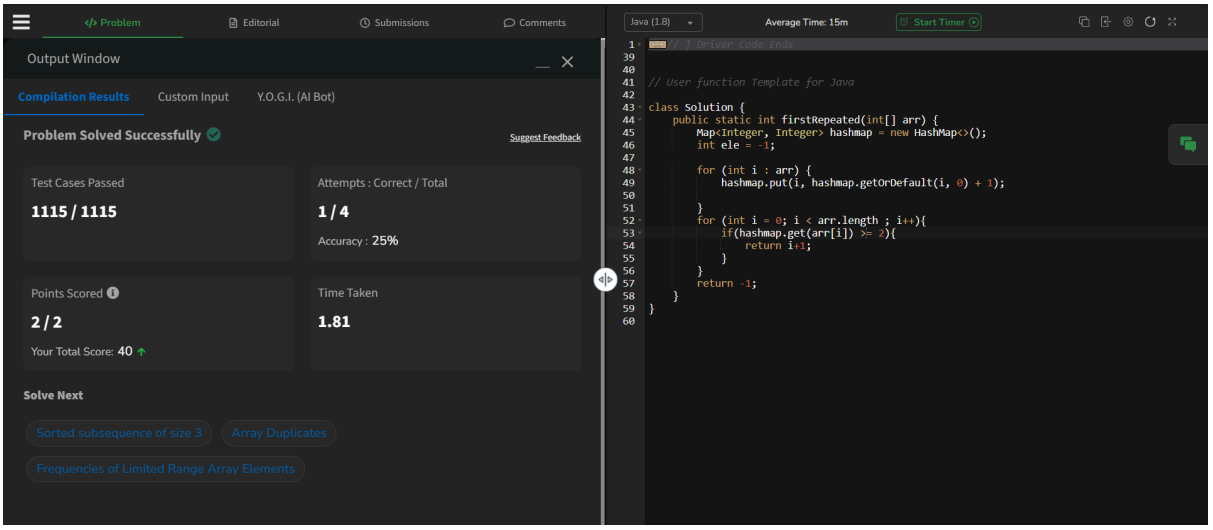
Time complexity : O(n)

4.Maximum Index



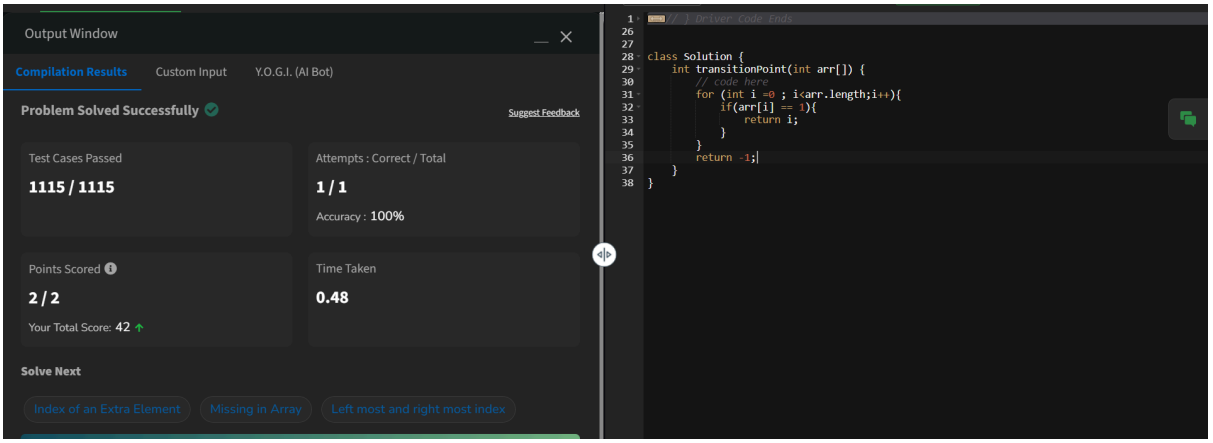
Time Complexity : O(n)

5.First repeating element



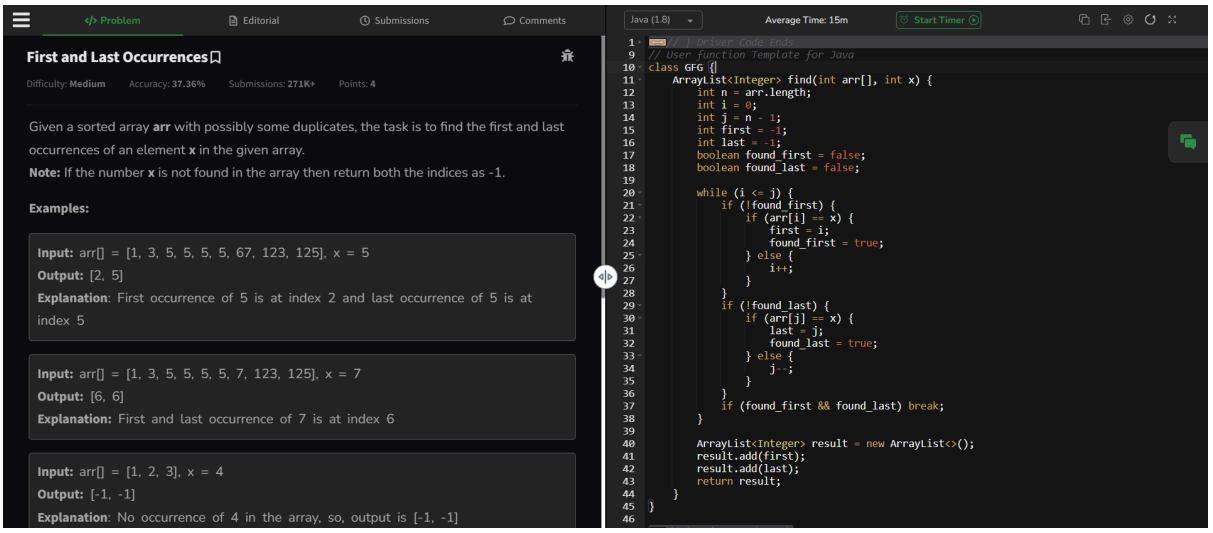
Time complexity : O(n)

6.First Transition point



Time Complexity : O(n)

7.First and Last Occurences



Problem | Editorial | Submissions | Comments

First and Last Occurrences

Difficulty: **Medium** Accuracy: **37.36%** Submissions: **271K+** Points: **4**

Given a sorted array **arr** with possibly some duplicates, the task is to find the first and last occurrences of an element **x** in the given array.

Note: If the number **x** is not found in the array then return both the indices as -1.

Examples:

Input: arr[] = [1, 3, 5, 5, 5, 5, 67, 123, 125], x = 5
Output: [2, 5]
Explanation: First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5

Input: arr[] = [1, 3, 5, 5, 5, 5, 7, 123, 125], x = 7
Output: [6, 6]
Explanation: First and last occurrence of 7 is at index 6

Input: arr[] = [1, 2, 3], x = 4
Output: [-1, -1]
Explanation: No occurrence of 4 in the array, so, output is [-1, -1]

Java (1.8) | Average Time: 15m | Start Timer

```
1  // } Driver Code Ends
9  // User function Template for Java
10 class GFG {
11     ArrayList<Integer> find(int arr[], int x) {
12         int n = arr.length;
13         int i = 0;
14         int j = n - 1;
15         int first = -1;
16         int last = -1;
17         boolean found_first = false;
18         boolean found_last = false;
19
20         while (i <= j) {
21             if (!found_first) {
22                 if (arr[i] == x) {
23                     first = i;
24                     found_first = true;
25                 } else {
26                     i++;
27                 }
28             }
29             if (!found_last) {
30                 if (arr[j] == x) {
31                     last = j;
32                     found_last = true;
33                 } else {
34                     j--;
35                 }
36             }
37             if (found_first && found_last) break;
38         }
39
40         ArrayList<Integer> result = new ArrayList<>();
41         result.add(first);
42         result.add(last);
43         return result;
44     }
45 }
46
```

Time complexity : O(n)