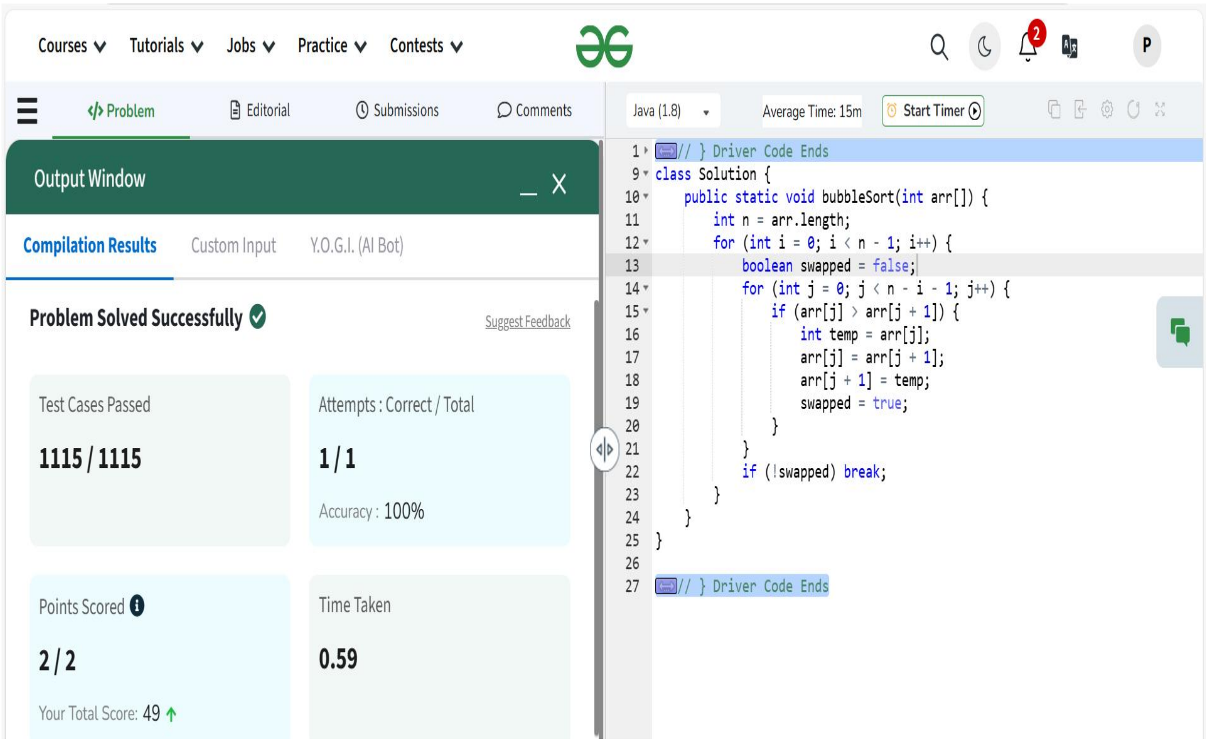
DSA Practice – 6

1.Bubble Sort



Problem Solved Successfully

Test Cases Passed
1115 / 1115

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

Points Scored
2 / 2
Your Total Score: 49

Time Taken
0.59

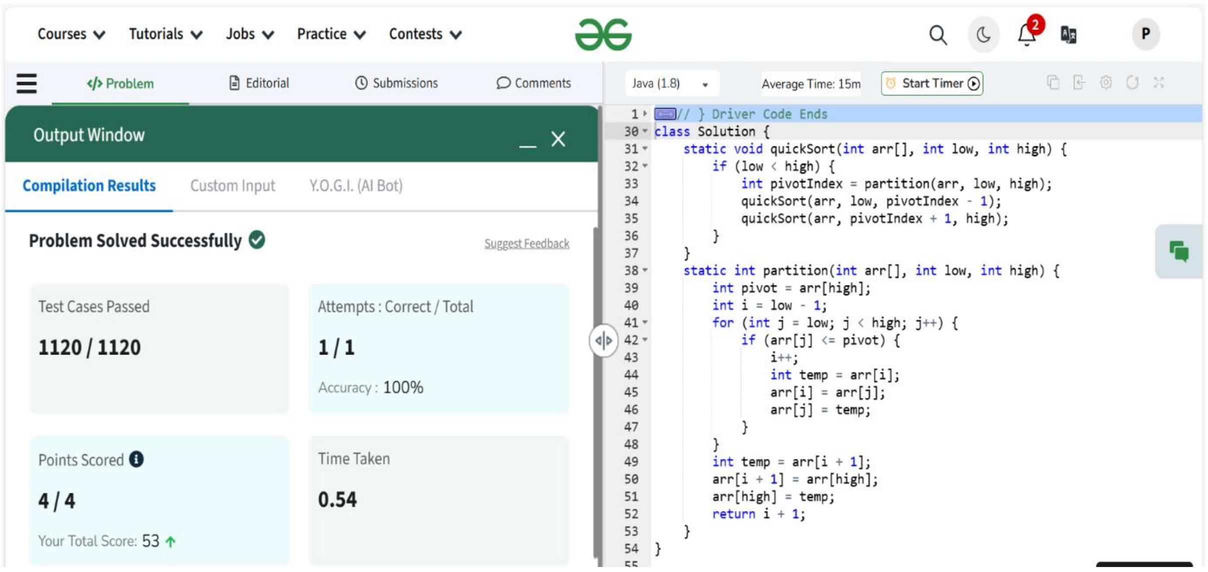
Average Time: 15m

```
// } Driver Code Ends
class Solution {
    public static void bubbleSort(int arr[]) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            boolean swapped = false;
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (!swapped) break;
        }
    }
}

// } Driver Code Ends
```

Time Complexity : O(n)

2.Quick Sort



Problem Solved Successfully

Test Cases Passed
1120 / 1120

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

Points Scored
4 / 4
Your Total Score: 53

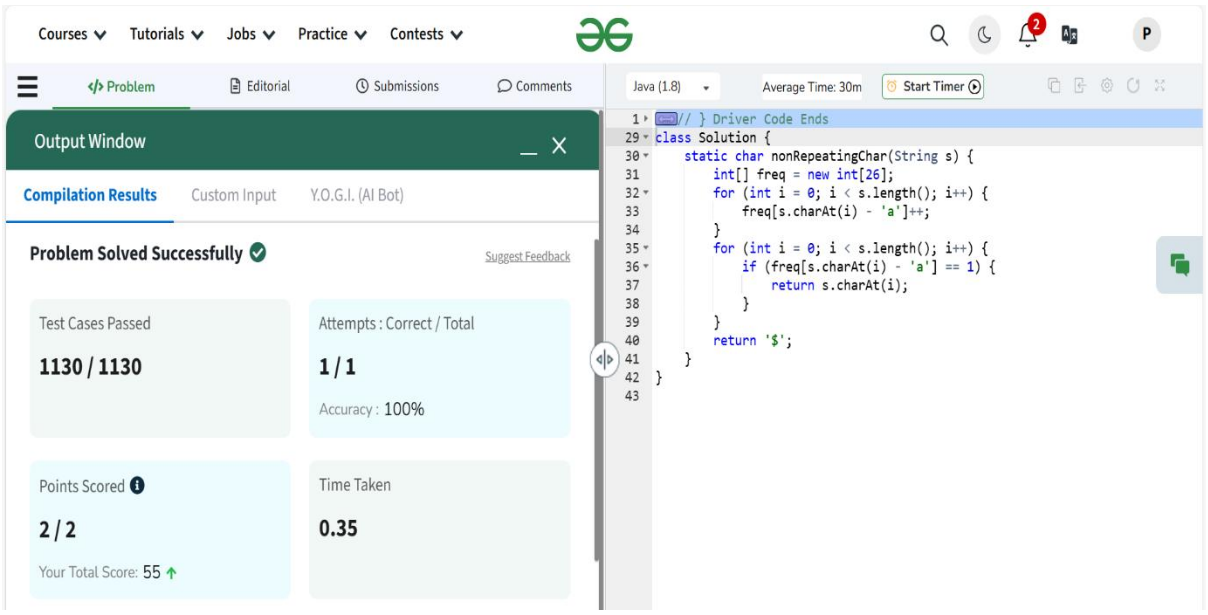
Time Taken
0.54

Average Time: 15m

```
// } Driver Code Ends
class Solution {
    static void quickSort(int arr[], int low, int high) {
        if (low < high) {
            int pivotIndex = partition(arr, low, high);
            quickSort(arr, low, pivotIndex - 1);
            quickSort(arr, pivotIndex + 1, high);
        }
    }
    static int partition(int arr[], int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        return i + 1;
    }
}
```

Time Complexity : O(n log n)

3.Non Repeating Character



Problem Solved Successfully

Test Cases Passed
1130 / 1130

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

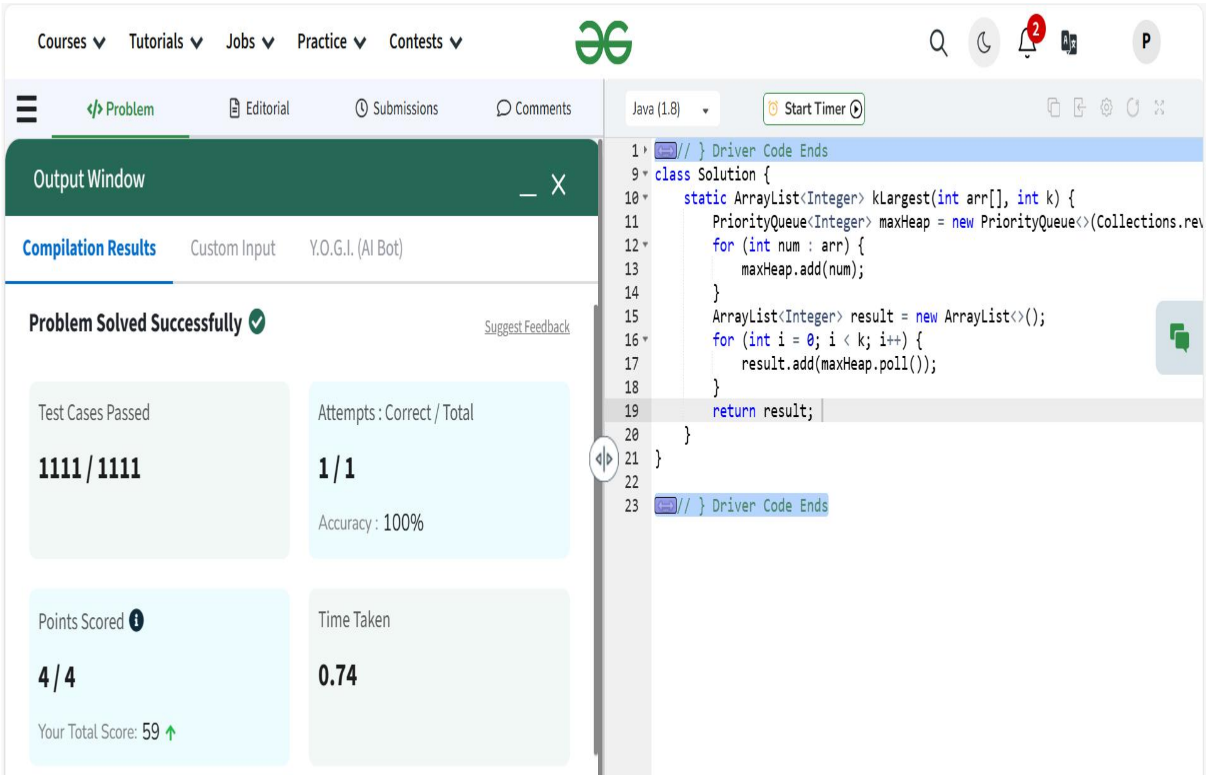
Points Scored
2 / 2
Your Total Score: 55

Time Taken
0.35

```
// } Driver Code Ends
class Solution {
    static char nonRepeatingChar(String s) {
        int[] freq = new int[26];
        for (int i = 0; i < s.length(); i++) {
            freq[s.charAt(i) - 'a']++;
        }
        for (int i = 0; i < s.length(); i++) {
            if (freq[s.charAt(i) - 'a'] == 1) {
                return s.charAt(i);
            }
        }
        return '$';
    }
}
```

Time Complexity : O(m×n)

4.K Largest Element



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

Points Scored
4 / 4
Your Total Score: 59

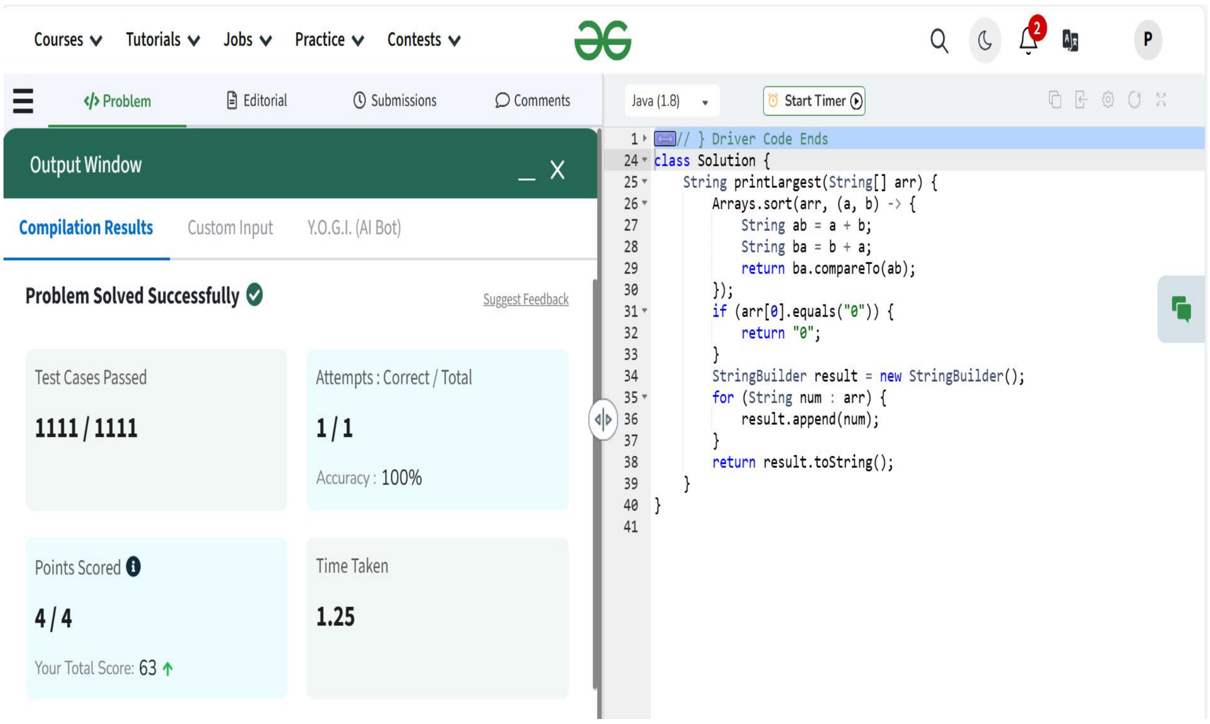
Time Taken
0.74

```
// } Driver Code Ends
class Solution {
    static ArrayList<Integer> kLargest(int arr[], int k) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.rev
        for (int num : arr) {
            maxHeap.add(num);
        }
        ArrayList<Integer> result = new ArrayList<>();
        for (int i = 0; i < k; i++) {
            result.add(maxHeap.poll());
        }
        return result;
    }
}

// } Driver Code Ends
```

Time Complexity : O(n log n)

5.Form the Largest Element



Problem Solved Successfully

Test Cases Passed
1111 / 1111

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

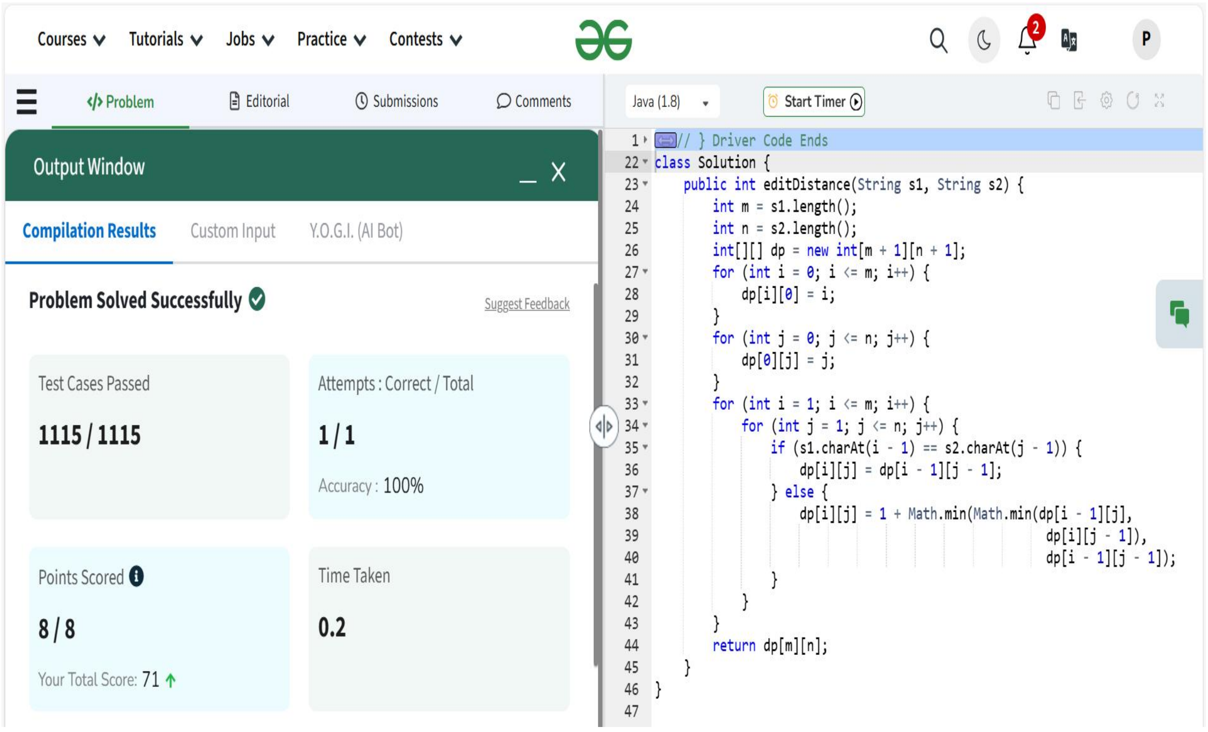
Points Scored
4 / 4
Your Total Score: 63

Time Taken
1.25

```java
// } Driver Code Ends
class Solution {
    String printLargest(String[] arr) {
        Arrays.sort(arr, (a, b) -> {
            String ab = a + b;
            String ba = b + a;
            return ba.compareTo(ab);
        });
        if (arr[0].equals("0")) {
            return "0";
        }
        StringBuilder result = new StringBuilder();
        for (String num : arr) {
            result.append(num);
        }
        return result.toString();
    }
}
```

Time Complexity : O(n log n * m)

6.Edit distance



Problem Solved Successfully

Test Cases Passed
1115 / 1115

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

Points Scored
8 / 8
Your Total Score: 71

Time Taken
0.2

```java
// } Driver Code Ends
class Solution {
    public int editDistance(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();
        int[][] dp = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
            dp[i][0] = i;
        }
        for (int j = 0; j <= n; j++) {
            dp[0][j] = j;
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = 1 + Math.min(Math.min(dp[i - 1][j],
                                                      dp[i][j - 1]),
                                                      dp[i - 1][j - 1]);
                }
            }
        }
        return dp[m][n];
    }
}
```

Time Complexity : O(m×n)