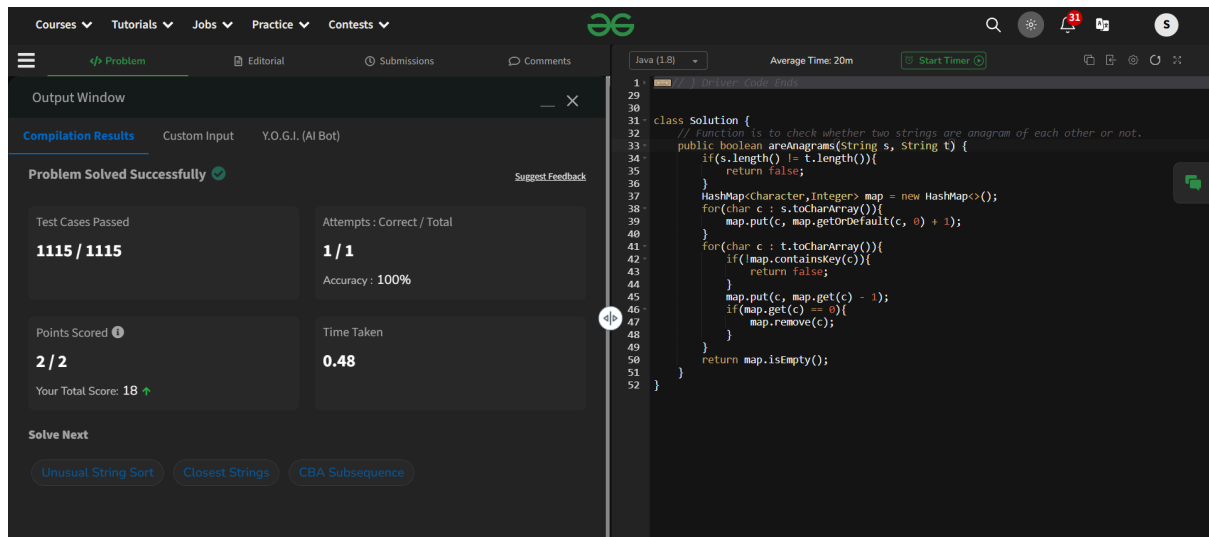


DSA PRACTICE -3

Name : Sidharth.N

Roll No : 22AD126

1. Anagram Program

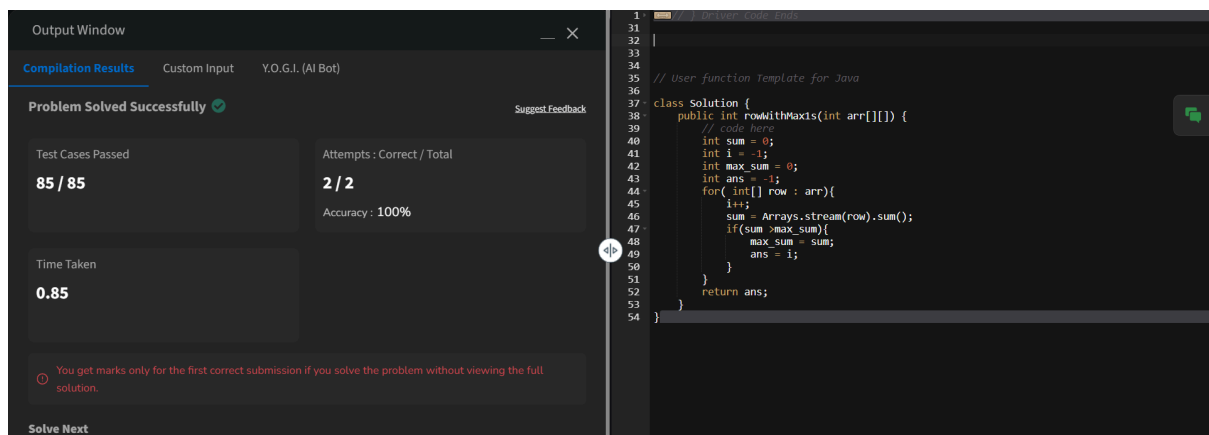


The screenshot shows a coding platform interface with the following details:

- Navigation:** Courses, Tutorials, Jobs, Practice, Contests.
- Problem Status:** Problem Solved Successfully.
- Statistics:**
 - Test Cases Passed: 1115 / 1115
 - Attempts: Correct / Total: 1 / 1
 - Accuracy: 100%
 - Points Scored: 2 / 2
 - Time Taken: 0.48
 - Your Total Score: 18
- Solve Next:** Unusual String Sort, Closest Strings, CBA Subsequence.
- Code:** Java (1.8) solution for the Anagram problem. The code uses a HashMap to check if two strings are anagrams.

Time Complexity : $O(n)$

2. Row with max 1's



The screenshot shows a coding platform interface with the following details:

- Problem Status:** Problem Solved Successfully.
- Statistics:**
 - Test Cases Passed: 85 / 85
 - Attempts: Correct / Total: 2 / 2
 - Accuracy: 100%
 - Time Taken: 0.85
- Solve Next:** You get marks only for the first correct submission if you solve the problem without viewing the full solution.
- Code:** Java solution for the Row with max 1's problem. The code iterates through each row of a 2D array and calculates the sum of 1s to find the row with the maximum sum.

Time Complexity : $O(n^2)$

3. Longest Consecutive Subsequence

The screenshot shows a coding problem titled "Longest consecutive subsequence". The problem description states: "Given an array `arr` of non-negative integers. Find the **length** of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers** can be in **any order**." Examples provided are: Input: `arr[] = [2, 6, 1, 9, 4, 5, 3]` Output: 6; and Input: `arr[] = [1, 9, 3, 10, 4, 20, 2]` Output: 4. The right side of the image shows a Java solution using a HashSet to track elements and a loop to find the longest consecutive sequence.

Longest consecutive subsequence

Difficulty: Medium Accuracy: 33.0% Submissions: 309K+ Points: 4

Given an array `arr` of non-negative integers. Find the **length** of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers** can be in **any order**.

Examples:

Input: `arr[] = [2, 6, 1, 9, 4, 5, 3]`
Output: 6
Explanation: The consecutive numbers here are 1, 2, 3, 4, 5, 6. These 6 numbers form the longest consecutive subsequence.

Input: `arr[] = [1, 9, 3, 10, 4, 20, 2]`
Output: 4

```
1 // Driver Code Starts
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 public class Solution {
28     public int findLongestConseqSubseq(int[] arr) {
29         HashSet<Integer> set = new HashSet<>();
30         for (int num : arr) {
31             set.add(num);
32         }
33         int ans = 0;
34         for (int i : set) {
35             if (!set.contains(i - 1)) {
36                 int num = i;
37                 int streak = 1;
38                 while (set.contains(num + 1)) {
39                     num++;
40                     streak++;
41                 }
42                 ans = Math.max(ans, streak);
43             }
44         }
45         return ans;
46     }
47 }
48 // Driver Code Ends
```

Time Complexity : $O(n)$

4. Longest Palindromic Substring

The screenshot shows a coding problem titled "Longest Palindromic Substring". The left panel displays the problem status: "Problem Solved Successfully", "Test Cases Passed: 1111 / 1111", "Attempts: 1 / 2", "Accuracy: 50%", "Points Scored: 4 / 4", and "Time Taken: 0.13". The right panel shows a Java solution using a helper function `pal` to check for palindromes and a loop to find the longest one.

Output Window

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

Problem Solved Successfully

Test Cases Passed: 1111 / 1111

Attempts: Correct / Total: 1 / 2

Accuracy: 50%

Points Scored: 4 / 4

Time Taken: 0.13

Your Total Score: 22

Solve Next

Longest Common Substring Longest Palindromic Subsequence

Longest repeating and non-overlapping substring

```
11 while (t-- > 0) {
12     String S = read.readLine();
13
14     Solution ob = new Solution();
15     System.out.println(ob.longestPalindrome(S));
16 }
17
18 // Driver Code Ends
19
20
21
22
23 // User function Template for Java
24
25
26 class Solution {
27     // Static method to find the longest palindromic substring
28     // code here
29     String ans = "";
30     for (int i = 0; i < s.length(); i++) {
31         String temp = pal(s, i, i);
32         if (temp.length() > ans.length()) {
33             ans = temp;
34         }
35         String temp2 = pal(s, i, i + 1);
36         if (temp2.length() > ans.length()) {
37             ans = temp2;
38         }
39     }
40     return ans;
41 }
42
43 static String pal(String s, int l, int r) {
44     while (l >= 0 && r < s.length() && s.charAt(l) == s.charAt(r)) {
45         l--;
46         r++;
47     }
48     return s.substring(l + 1, r);
49 }
50 }
```

Time Complexity : $O(n^2)$

5. Rat in a maze problem

The screenshot shows a coding platform interface with the following details:

- Navigation Bar:** Courses, Tutorials, Jobs, Practice, Contests.
- Problem Status:** Problem Solved Successfully (with a green checkmark).
- Statistics:**
 - Test Cases Passed: 162 / 162
 - Attempts: Correct / Total: 1 / 1
 - Accuracy: 100%
 - Points Scored: 4 / 4
 - Time Taken: 0.52
 - Your Total Score: 26
- Solve Next:** Tower Of Hanoi, Black and White, Rat Maze With Multiple Jumps.
- Code Editor:** Java (1.8) language. The code implements a backtracking solution for finding a path in a maze. The maze is represented by a 2D array 'mat' where 0 is an open cell and 1 is a wall. The solution starts at (0,0) and explores four directions (D, L, R, U) until it reaches the destination (n-1, n-1).

Time Complexity : $O(4^{(n^2)})$