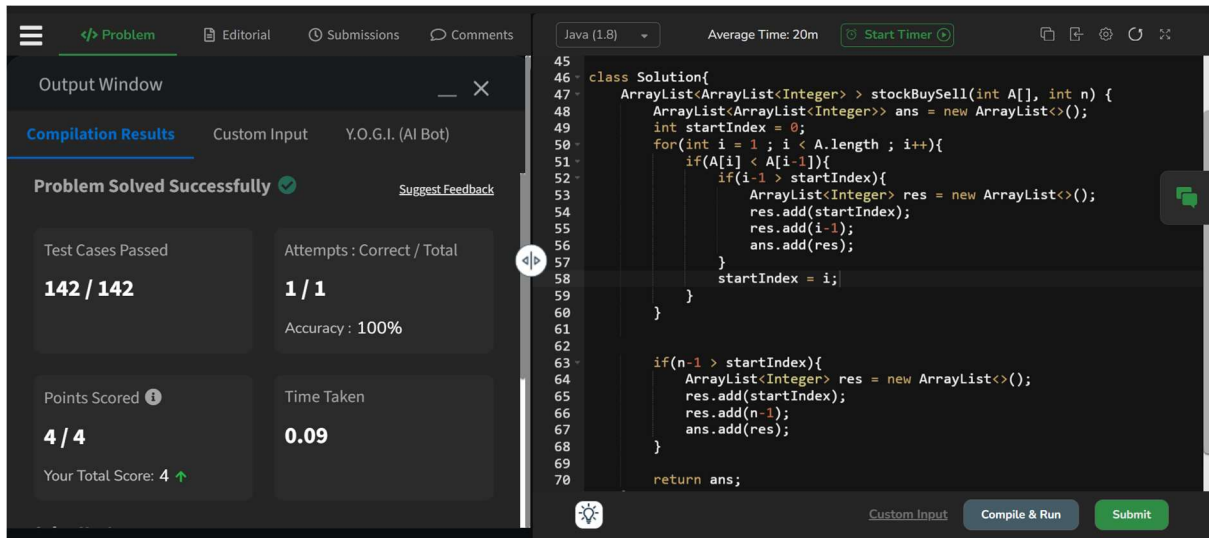


DSA PRACTICE

(NOVEMBER 14, 2024)

1. Stock buy and sell :

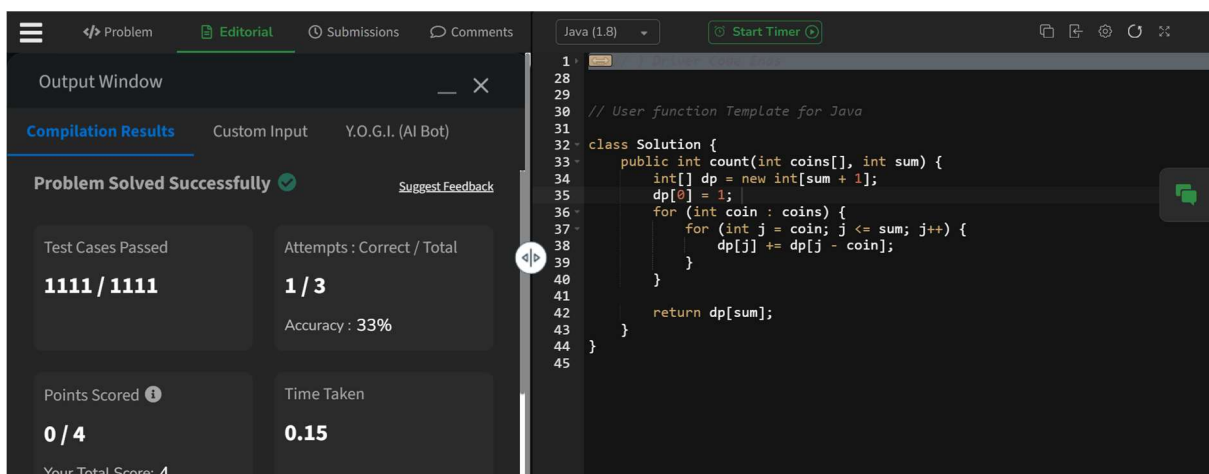


```
45
46 class Solution{
47     ArrayList<ArrayList<Integer>> stockBuySell(int A[], int n) {
48         ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
49         int startIndex = 0;
50         for(int i = 1 ; i < A.length ; i++){
51             if(A[i] < A[i-1]){
52                 if(i-1 > startIndex){
53                     ArrayList<Integer> res = new ArrayList<>();
54                     res.add(startIndex);
55                     res.add(i-1);
56                     ans.add(res);
57                 }
58                 startIndex = i;
59             }
60         }
61
62         if(n-1 > startIndex){
63             ArrayList<Integer> res = new ArrayList<>();
64             res.add(startIndex);
65             res.add(n-1);
66             ans.add(res);
67         }
68
69         return ans;
70     }
}
```

Time Complexity : $O(N)$

Space Complexity : $O(N)$

2. Coin exchange



```
1
28
29
30 // User function Template for Java
31
32 class Solution {
33     public int count(int coins[], int sum) {
34         int[] dp = new int[sum + 1];
35         dp[0] = 1;
36         for (int coin : coins) {
37             for (int j = coin; j <= sum; j++) {
38                 dp[j] += dp[j - coin];
39             }
40         }
41         return dp[sum];
42     }
43 }
44
45 }
```

Time Complexity : $O(N*M)$

Space Complexity: $O(M)$

3. First and last occurrence

The screenshot displays a coding platform interface. On the left, the 'Output Window' shows 'Problem Solved Successfully' with a green checkmark. It includes statistics: 'Test Cases Passed: 1120 / 1120', 'Attempts: Correct / Total: 1 / 1', 'Accuracy: 100%', 'Points Scored: 4 / 4', and 'Time Taken: 0.91'. The 'Your Total Score' is 8. On the right, the code editor shows a Java solution for finding the first and last occurrence of an element in an array using binary search. The code is as follows:

```
1 // User function Template for Java
2
3 class GFG {
4     ArrayList<Integer> find(int arr[], int x) {
5         ArrayList<Integer> answer = new ArrayList<>(Arrays.asList(-1, -1));
6
7         int left = 0, right = 0;
8         int start = 0, end = arr.length - 1;
9         boolean found = false;
10
11         while(start <= end) {
12             int mid = start + (end - start) / 2;
13             if(arr[mid] == x) {
14                 left = right = mid;
15                 found = true;
16                 break;
17             }
18             if(arr[mid] > x) end = mid - 1;
19             if(arr[mid] < x) start = mid + 1;
20         }
21
22         if (!found) return answer;
23
24         for(int i = left; i <= right; i++) {
25             // ... (code is partially obscured)
26         }
27     }
28 }
```

Time Complexity : $O(N)$

Space Complexity : $O(1)$

4. First Index Transition

The screenshot displays a coding platform interface. On the left, the 'Output Window' shows 'Problem Solved Successfully' with a green checkmark. It includes statistics: 'Test Cases Passed: 1115 / 1115', 'Attempts: Correct / Total: 1 / 3', 'Accuracy: 33%', 'Points Scored: 2 / 2', and 'Time Taken: 0.43'. On the right, the code editor shows a Java solution for finding the first index transition point in an array. The code is as follows:

```
27 class Solution {
28     int transitionPoint(int arr[]) {
29         int n = arr.length;
30         int start = 0, end = n - 1;
31         int result = -1;
32
33         while (start <= end) {
34             int mid = start + (end - start) / 2;
35
36             if (arr[mid] == 1) {
37                 result = mid;
38                 end = mid - 1;
39             } else {
40                 start = mid + 1;
41             }
42         }
43
44         if (result == -1 && arr[0] == 1) {
45             return 0;
46         }
47
48         return result;
49     }
50 }
```

Time Complexity : $O(\log n)$

Space Complexity : $O(1)$

5. First Repeating String

Output Window

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

Problem Solved Successfully [Suggest Feedback](#)

Test Cases Passed
1115 / 1115

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

Points Scored
2 / 2

Time Taken
1.97

```
1 // } Driver Code Ends
39
40
41 / User function Template for Java
42
43 class Solution {
44     public static int firstRepeated(int[] arr) {
45         int max = Integer.MIN_VALUE;
46         for(int i : arr) {
47             max = Math.max(max, i);
48         }
49
50         int[][] freq = new int[max + 1][2];
51         for(int i = 0; i < arr.length; i++){
52             if (freq[arr[i]][1] == 0) {
53                 freq[arr[i]][0] = i + 1;
54                 freq[arr[i]][1] = 1;
55             } else {
56                 freq[arr[i]][1]++;
57             }
58         }
59
60 }
```

Time complexity: $O(N)$

Space Complexity: $O(M)$

6. Remove Duplicates from the List

Output Window

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

Problem Solved Successfully [Suggest Feedback](#)

Test Cases Passed
1115 / 1115

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

Points Scored
2 / 2
Your Total Score: 14

Time Taken
1.45

Solve Next

[Sorted subsequence of size 3](#) [Sort 0s, 1s and 2s](#) [Sort Unsorted Subarray](#)

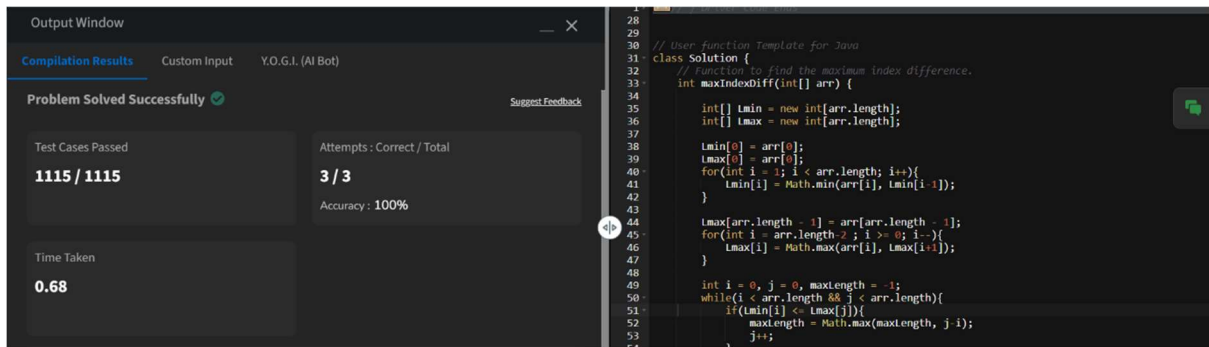
Kick start your career with GfG 160!

```
1 // } Driver Code Ends
31
32
33 class Solution {
34     public int remove_duplicate(List<Integer> arr) {
35         int i = 0;
36         for(int j = 1; j < arr.size(); j++){
37             if(!arr.get(i).equals(arr.get(j))){
38                 i++;
39                 arr.set(i, arr.get(j));
40             }
41         }
42         return i+1;
43     }
44 }
45 }
```

Time Complexity : $O(N)$

Space Complexity : $O(1)$

7. Maximum Index

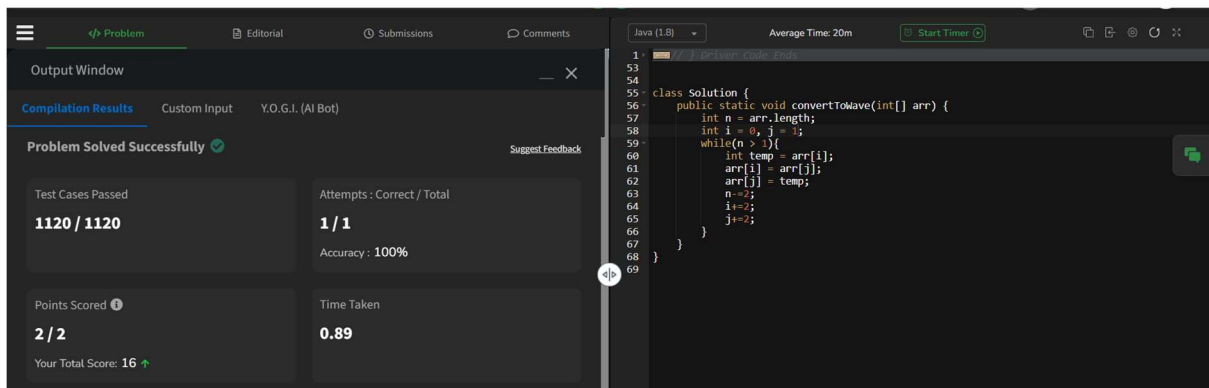


```
1
28
29
30 // User function Template for Java
31 class Solution {
32 // Function to find the maximum index difference.
33 int maxIndexDiff(int[] arr) {
34
35     int[] lmin = new int[arr.length];
36     int[] lmax = new int[arr.length];
37
38     lmin[0] = arr[0];
39     lmax[0] = arr[0];
40     for(int i = 1; i < arr.length; i++){
41         lmin[i] = Math.min(arr[i], lmin[i-1]);
42     }
43
44     lmax[arr.length - 1] = arr[arr.length - 1];
45     for(int i = arr.length-2; i >= 0; i--){
46         lmax[i] = Math.max(arr[i], lmax[i+1]);
47     }
48
49     int i = 0, j = 0, maxlength = -1;
50     while(i < arr.length && j < arr.length){
51         if(lmin[i] <= lmax[j]){
52             maxlength = Math.max(maxlength, j-i);
53             j++;
54         }
```

Time Complexity : $O(N)$

Space Complexity : $O(N)$

8. Wave Array



```
1 // Driver code starts
53
54
55 class Solution {
56 public static void convertToWave(int[] arr) {
57     int n = arr.length;
58     int i = 0, j = 1;
59     while(n > 1){
60         int temp = arr[i];
61         arr[i] = arr[j];
62         arr[j] = temp;
63         n-=2;
64         i+=2;
65         j+=2;
66     }
67 }
68
69 }
```

Time Complexity : $O(N)$

Space Complexity : $O(1)$
