

DSA Practice – 5

1. Stock buy and sell

Problem Solved Successfully ✓

Test Cases Passed: **142 / 142**

Attempts : Correct / Total: **1 / 2**

Accuracy : 50%

Points Scored: **4 / 4**

Time Taken: **0.08**

Your Total Score: 21 ↑

```
1 // Driver Code Ends
42 class Solution {
43     public ArrayList<ArrayList<Integer>> stockBuySell(int A[], int n) {
44         ArrayList<ArrayList<Integer>> result = new ArrayList<>();
45         for (int i = 1; i < n; i++) {
46             if (A[i] > A[i - 1]) {
47                 ArrayList<Integer> pair = new ArrayList<>();
48                 pair.add(i - 1);
49                 pair.add(i);
50                 result.add(pair);
51             }
52         }
53         return result;
54     }
55 }
56
57
```

Time Complexity : $O(N)$

2. Coin Change (Count Ways)

Problem Solved Successfully ✓

Test Cases Passed: **1111 / 1111**

Attempts : Correct / Total: **1 / 1**

Accuracy : 100%

Points Scored: **4 / 4**

Time Taken: **0.17**

Your Total Score: 25 ↑

```
1 // Driver Code Ends
28 class Solution {
29     public int count(int coins[], int sum) {
30         int[] dp = new int[sum + 1];
31         dp[0] = 1;
32         for (int coin : coins) {
33             for (int i = coin; i <= sum; i++) {
34                 dp[i] += dp[i - coin];
35             }
36         }
37         return dp[sum];
38     }
39 }
40
```

Time Complexity : $O(n * \text{sum})$

3. First and Last Occurences

The screenshot shows a coding platform interface with a green header bar containing navigation links: Courses, Tutorials, Jobs, Practice, and Contests. The main area is divided into two panels. The left panel, titled 'Output Window', displays 'Compilation Results' for a submission by 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Performance metrics are shown in four boxes: 'Test Cases Passed' (1120 / 1120), 'Attempts: Correct / Total' (1 / 2), 'Points Scored' (4 / 4), and 'Time Taken' (0.92). A 'Your Total Score: 29' is also visible. The right panel shows the Java code for the solution, which is a class 'GFG' with a method 'find' that returns the first and last indices of a target value 'x' in an array 'arr'.

```
1 // Driver Code Ends
2 class GFG {
3     public static ArrayList<Integer> find(int[] arr, int x) {
4         ArrayList<Integer> result = new ArrayList<>();
5         int first = -1, last = -1;
6         for (int i = 0; i < arr.length; i++) {
7             if (arr[i] == x) {
8                 if (first == -1) {
9                     first = i;
10                }
11                last = i;
12            }
13        }
14        result.add(first);
15        result.add(last);
16        return result;
17    }
18 }
19 // Driver Code Ends
```

Time Complexity : $O(N)$

4. Find Transition Point

The screenshot shows the same coding platform interface as above, but for a different problem. The left panel shows 'Problem Solved Successfully' with 'Test Cases Passed' (1115 / 1115), 'Attempts: Correct / Total' (1 / 1), 'Points Scored' (2 / 2), and 'Time Taken' (0.41). The right panel shows the Java code for the 'Find Transition Point' problem, which is a class 'Solution' with a method 'transitionPoint' that returns the index of the first element greater than 1 in an array 'arr'.

```
1 // Driver Code Ends
2 class Solution {
3     int transitionPoint(int arr[]) {
4         for (int i = 0; i < arr.length; i++) {
5             if (arr[i] == 1) {
6                 return i;
7             }
8         }
9         return -1;
10    }
11 }
12 // Driver Code Ends
```

Time Complexity : $O(n)$

5. Remove Duplicates Sorted Array

The screenshot displays a coding platform interface for the problem "Remove Duplicates Sorted Array". The top navigation bar includes links for Courses, Tutorials, Jobs, Practice, and Contests. The user's profile is visible in the top right corner. The left sidebar shows the "Output Window" with a green header and a close button. Below it, the "Compilation Results" section indicates "Problem Solved Successfully" with a green checkmark. The test results are summarized in four boxes: "Test Cases Passed" (1115 / 1115), "Attempts : Correct / Total" (1 / 1), "Points Scored" (2 / 2), and "Time Taken" (1.36). The "Your Total Score" is 33. The main editor area shows the Java code for the solution, which is a class named "Solution" with a method "remove_duplicate" that takes a list of integers and returns an integer. The code uses a two-pointer approach to remove duplicates in a sorted array.

```
1 // Driver Code Ends
31 class Solution {
32     public int remove_duplicate(List<Integer> arr) {
33         if (arr.size() == 0) {
34             return 0;
35         }
36         int j = 1;
37         for (int i = 1; i < arr.size(); i++) {
38             if (!arr.get(i).equals(arr.get(i - 1))) {
39                 arr.set(j, arr.get(i));
40                 j++;
41             }
42         }
43         return j;
44     }
45 }
46
```

Time Complexity : $O(n)$

6. Wave Array

The screenshot displays a coding platform interface for the problem "Wave Array". The top navigation bar includes links for Courses, Tutorials, Jobs, Practice, and Contests. The user's profile is visible in the top right corner. The left sidebar shows the "Output Window" with a green header and a close button. Below it, the "Compilation Results" section indicates "Problem Solved Successfully" with a green checkmark. The test results are summarized in four boxes: "Test Cases Passed" (1120 / 1120), "Attempts : Correct / Total" (1 / 1), "Points Scored" (2 / 2), and "Time Taken" (0.89). The "Your Total Score" is 35. The main editor area shows the Java code for the solution, which is a class named "Solution" with a method "convertToWave" that takes an array of integers and converts it into a wave array. The code uses a two-pointer approach to swap elements at even and odd indices.

```
1 // Driver Code Ends
53 class Solution {
54     public static void convertToWave(int[] arr) {
55         for (int i = 0; i < arr.length - 1; i += 2) {
56             if (i % 2 == 0) {
57                 int temp = arr[i];
58                 arr[i] = arr[i + 1];
59                 arr[i + 1] = temp;
60             }
61             else {
62                 if (i + 1 < arr.length && arr[i] < arr[i + 1]) {
63                     int temp = arr[i];
64                     arr[i] = arr[i + 1];
65                     arr[i + 1] = temp;
66                 }
67             }
68         }
69     }
70 }
71
72
```

Time Complexity : $O(n)$