

DSA Practice – 2

1. Floor in sorted array

The screenshot displays a coding interface for the problem 'Floor in sorted array'. The 'Output Window' on the left shows 'Problem Solved Successfully' with 1111 test cases passed out of 1111, 3 correct attempts out of 3, and an accuracy of 100%. The time taken is 0.29 seconds. The code on the right is a Java solution using binary search to find the floor of a given number in a sorted array.

```
1 // Driver Code Ends
2 class Solution {
3     public int findFloor(int[] arr, int k) {
4         int n = arr.length;
5         int left = 0;
6         int right = n - 1;
7         int floorIndex = -1;
8         while (left <= right) {
9             int mid = left + (right - left) / 2;
10            if (arr[mid] <= k) {
11                floorIndex = mid;
12                left = mid + 1;
13            } else {
14                right = mid - 1;
15            }
16        }
17        return floorIndex;
18    }
19 }
20 // Driver Code Ends
```

Time Complexity : $O(\log n)$

2. Check equal arrays

The screenshot displays a coding interface for the problem 'Check equal arrays'. The 'Output Window' on the left shows 'Problem Solved Successfully' with 1116 test cases passed out of 1116, 4 correct attempts out of 4, and an accuracy of 100%. The time taken is 0.15 seconds. The code on the right is a Java solution that sorts both arrays and compares them element by element to check if they are equal.

```
1 // Driver Code Ends
2 class Solution {
3     public static boolean check(int[] arr1, int[] arr2) {
4         if (arr1.length != arr2.length) {
5             return false;
6         }
7         Arrays.sort(arr1);
8         Arrays.sort(arr2);
9         for (int i = 0; i < arr1.length; i++) {
10            if (arr1[i] != arr2[i]) {
11                return false;
12            }
13        }
14        return true;
15    }
16 }
```

Time Complexity : $O(n)$

3. Palindrome linked list

```
class Solution {
    Node reverse(Node head) {
        Node prev = null;
        Node current = head;
        Node nextNode = null;
        while (current != null) {
            nextNode = current.next;
            current.next = prev;
            prev = current;
            current = nextNode;
        }
        return prev;
    }

    boolean areIdentical(Node list1, Node list2) {
        while (list1 != null && list2 != null) {
            if (list1.data != list2.data) {
                return false;
            }
            list1 = list1.next;
            list2 = list2.next;
        }
        return list1 == null && list2 == null;
    }

    boolean isPalindrome(Node head) {
        if (head == null || head.next == null) {
            return true;
        }
    }
```

```

    }

    Node slow = head;
    Node fast = head;
    while (fast != null && fast.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }

    Node secondHalf = reverse(slow);
    Node firstHalf = head;
    while (secondHalf != null) {
        if (firstHalf.data != secondHalf.data) {
            return false;
        }
        firstHalf = firstHalf.next;
        secondHalf = secondHalf.next;
    }

    return true;
}
}

```

The screenshot displays a coding platform interface with a green header bar containing navigation links: Courses, Tutorials, Jobs, Practice, and Contests. Below the header, a toolbar includes icons for Problem, Editorial, Submissions, and Comments. The main content area is divided into two panels. The left panel, titled 'Output Window', shows 'Compilation Results' with a green checkmark indicating 'Problem Solved Successfully'. It also displays 'Test Cases Passed' as 1112 / 1112, 'Attempts: Correct / Total' as 3 / 8, and 'Accuracy: 37%'. The 'Time Taken' is listed as 2.06. The right panel shows the Java code for the solution, which includes a `reverse` method and an `areIdentical` method. The code is written in Java 1.8 and includes a `Start Timer` button. The code is as follows:

```

1 // Driver Code Ends
65 class Solution {
66     Node reverse(Node head) {
67         Node prev = null;
68         Node current = head;
69         Node nextNode = null;
70         while (current != null) {
71             nextNode = current.next;
72             current.next = prev;
73             prev = current;
74             current = nextNode;
75         }
76         return prev;
77     }
78     boolean areIdentical(Node list1, Node list2) {
79         while (list1 != null && list2 != null) {
80             if (list1.data != list2.data) {
81                 return false;
82             }
83             list1 = list1.next;
84             list2 = list2.next;
85         }
86         return list1 == null && list2 == null;
87     }
88     boolean isPalindrome(Node head) {
89         if (head == null || head.next == null) {
90             return true;
91         }

```

Time Complexity : $O(n)$

4. Triplet sum in array

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

- Navigation Bar:** Courses, Tutorials, Jobs, Practice, Contests.
- Problem Bar:** Problem, Editorial, Submissions, Comments.
- Output Window:** Compilation Results, Custom Input, Y.O.G.I. (AI Bot).
- Problem Status:** Problem Solved Successfully (checkmark).
- Test Cases Passed:** 125 / 125.
- Attempts:** Correct / Total: 2 / 2.
- Accuracy:** 100%.
- Time Taken:** 0.13.
- Code Editor:** Java (1.8), Average Time: 15m, Start Timer.
- Code:**

```
1 // } Driver Code Ends
30 class Solution {
31     public boolean find3Numbers(int[] arr, int n, int x) {
32         Arrays.sort(arr);
33         for (int i = 0; i < n - 2; i++) {
34             int left = i + 1;
35             int right = n - 1;
36             while (left < right) {
37                 int sum = arr[i] + arr[left] + arr[right];
38                 if (sum == x) {
39                     return true;
40                 }
41                 if (sum < x) {
42                     left++;
43                 } else {
44                     right--;
45                 }
46             }
47         }
48         return false;
49     }
50 }
51
52
53
54
```

Time Complexity : $O(n^2)$