

DSA PRACTICE

DAY 2

Name: Dhejan R
Reg No.: 22IT022
Date: 11/11/2024

1. 0-1 knapsack problem

Code Solution:

```
public class knapsack{  
    public static int knapsack(int capacity, int[] weight, val[]){  
        int n=val.length;  
        int[] dp=new int[capacity+1];  
  
        for(int i=0; i<n;i++){  
            for(int w=capacity; w>=weights[i]; w--){  
                dp[w]=Math.max(dp[w], dp[w-weights[i]]+val[i]);  
            }  
        }  
        return dp[capacity];  
    }  
}
```

Output:

The screenshot displays a coding environment with two main panels. The left panel, titled 'Output Window', shows 'Compilation Results' for a custom input. It indicates 'Problem Solved Successfully' with a green checkmark. Below this, it shows 'Test Cases Passed: 1115 / 1115', 'Attempts: 3 / 4', 'Accuracy: 75%', and 'Time Taken: 0.42'. The right panel shows the source code for a 'Solution' class, which implements the 0-1 knapsack algorithm using dynamic programming. The code is written in Java and matches the provided code solution.

Time Complexity: $O(n \times \text{capacity})$

Space Complexity: $O(\text{Capacity})$

2. Floor in sorted array

Code Solution:

```
class Solution {  
    static int findFloor(int[] arr, int k) {  
        int low=0;  
        int high=arr.length-1;  
        int floorIndex=-1;
```

```

while (low<=high) {
    int mid=low+(high-low)/2;

    if (arr[mid]==k) {
        return mid;
    } else if (arr[mid]<k) {
        floorIndex=mid;
        low=mid+1;
    } else {
        high=mid-1;
    }
}
return floorIndex;
}
}

```

Output:

The screenshot shows a coding platform interface. On the left, the 'Output Window' displays 'Compilation Results' for 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with a green checkmark. Below this, it shows 'Test Cases Passed: 1111 / 1111', 'Attempts: Correct / Total: 1 / 2', 'Accuracy: 50%', 'Points Scored: 2 / 2', and 'Time Taken: 0.27'. At the bottom, it says 'Your Total Score: 10' and 'Solve Next'. On the right, the code editor shows the following Java code:

```

1 // Driver Code Starts
9 class Solution {
10
11     static int findFloor(int[] arr, int k) {
12         int low=0;
13         int high=arr.length-1;
14         int floorIndex=-1;
15
16         while (low<=high) {
17             int mid=low+(high-low)/2;
18
19             if (arr[mid]==k) {
20                 return mid;
21             } else if (arr[mid]<k) {
22                 floorIndex=mid;
23                 low=mid+1;
24             } else {
25                 high=mid-1;
26             }
27         }
28         return floorIndex;
29     }
30 }
31 // Driver Code Ends
32
33

```

Time Complexity: $O(\text{Logn})$

Space Complexity: $O(1)$

3. Check equal arrays

Code Solution:

```

class Solution {
    public static boolean check(int[] arr1, int[] arr2) {
        if (arr1.length!=arr2.length)
            return false;

        Arrays.sort(arr1);
        Arrays.sort(arr2);
        for (int i=0; i<arr1.length; i++)

```

```

        if (arr1[i]!=arr2[i])
            return false;

        return true;
    }
}

```

Output:

The screenshot shows a coding platform interface. On the left, the 'Output Window' displays 'Compilation Results' for a problem solved successfully. It shows 1116/1116 test cases passed, 1/1 attempts, 100% accuracy, 1/1 points scored, and a time taken of 0.19. On the right, the code editor shows a Java solution for checking if two arrays are equal after sorting.

```

38
39
40 // User function Template for Java
41
42 class Solution {
43     // Function to check if two arrays are equal or not.
44     public static boolean check(int[] arr1, int[] arr2) {
45         if (arr1.length!=arr2.length)
46             return false;
47
48         Arrays.sort(arr1);
49         Arrays.sort(arr2);
50         for (int i=0; i<arr1.length; i++)
51             if (arr1[i]!=arr2[i])
52                 return false;
53
54         return true;
55     }
56 }

```

Time Complexity: $O(n \log n)$

Space Complexity: $O(1)$

4. Palindrome linked list

Code Solution:

```

class Solution {
    // Function to check whether the list is palindrome.
    boolean isPalindrome(Node head) {
        // Your code here
        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 prev=null;
        Node curr=slow;
        while (curr!=null) {
            Node nextNode=curr.next;
            curr.next=prev;
            prev=curr;

```

```

        curr=nextNode;
    }

    Node first=head;
    Node second=prev;
    boolean isPalindrome=true;
    while (second!=null) {
        if (first.data!=second.data) {
            isPalindrome=false;
            break;
        }
        first=first.next;
        second=second.next;
    }

    return isPalindrome;
}
}

```

Output:

The screenshot shows a coding platform interface with two main panels. The left panel, titled 'Output Window', displays the following information:

- Compilation Results:** Custom Input, Y.O.G.I. (AI Bot)
- Problem Solved Successfully:** (Green checkmark icon)
- Test Cases Passed:** 1112 / 1112
- Attempts:** Correct / Total: 1 / 1
- Accuracy:** 100%
- Points Scored:** 4 / 4
- Time Taken:** 1.85
- Your Total Score:** 15 (Green up arrow icon)
- Solve Next:**
 - Intersection Point in Y Shaped Linked Lists
 - Flattening a Linked List
 - Longest Palindrome in Linked List

The right panel shows the Java code for the solution, with line numbers 79 to 118 visible. The code is as follows:

```

79 class Solution {
80     // Function to check whether the List is palindrome.
81     boolean isPalindrome(Node head) {
82         // Your code here
83         if (head==null || head.next==null) return true;
84
85         Node slow=head;
86         Node fast=head;
87
88         while (fast!=null && fast.next!=null) {
89             slow=slow.next;
90             fast=fast.next.next;
91         }
92
93         Node prev=null;
94         Node curr=slow;
95         while (curr!=null) {
96             Node nextNode=curr.next;
97             curr.next=prev;
98             prev=curr;
99             curr=nextNode;
100         }
101
102         Node first=head;
103         Node second=prev;
104         boolean isPalindrome=true;
105         while (second!=null) {
106             if (first.data!=second.data) {
107                 isPalindrome=false;
108                 break;
109             }
110             first=first.next;
111             second=second.next;
112         }
113
114         return isPalindrome;
115     }
116 }
117
118

```

Time Complexity: $O(n)$

Space Complexity: $O(1)$

5. Balanced tree check

Code Solution:

```
class Tree
{
    //Function to check whether a binary tree is balanced or not.
    boolean isBalanced(Node root)
    {
        if (root==null)
            return true;
        int lh=height(root.left);
        if (lh==-1)
            return false;
        int rh=height(root.right);
        if (rh==-1)
            return false;
        if (Math.abs(lh-rh)>1)
            return false;

        return true;
    }

    public static int height(Node root) {
        if (root==null)
            return 0;
        int leftHeight=height(root.left);
        if (leftHeight==-1)
            return -1;
        int rightHeight=height(root.right);
        if (rightHeight==-1)
            return -1;
        if (Math.abs(leftHeight-rightHeight)>1)
            return -1;

        return Math.max(leftHeight, rightHeight)+1;
    }
}
```

Output:

The screenshot shows a coding platform interface. On the left, a sidebar contains navigation links: Courses, Tutorials, Jobs, Practice, and Contests. Below these is a 'Problem' tab with sub-tabs for Editorial, Submissions, and Comments. The main area on the left displays 'Output Window' and 'Compilation Results'. A message states 'Problem Solved Successfully' with a green checkmark. Below this, statistics are shown: Test Cases Passed (1120 / 1120), Attempts: Correct / Total (1 / 1), Accuracy: 100%, Points Scored (2 / 2), and Time Taken (0.44). A 'Solve Next' section lists 'Height of Binary Tree', 'Minimum Depth of a Binary Tree', and 'Array to BST'. The right panel shows the Java code for a balanced binary tree solution, with line numbers 127 to 165. The code includes a class 'Tree' with a method 'isBalanced' and a static method 'height'.

Time Complexity: $O(n)$

Space Complexity: $O(n)$

6. Triplet sum in array

Code Solution:

```
class Solution {
    // Should return true if there is a triplet with sum equal
    // to x in arr[], otherwise false
    public static boolean find3Numbers(int arr[], int n, int x) {
        // Your code Here
        Arrays.sort(arr);
        for (int i=0; i<n-2; i++){
            int l=i+1;
            int r=n-1;
            while(l<r){
                int a=arr[i]+arr[l]+arr[r];
                if(a==x){
                    return true;
                }
                else if (a<x){
                    l++;
                }
                else{
                    r--;
                }
            }
        }
        return false;
    }
}
```

```
}  
}
```

Output:

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed

125 / 125

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Time Taken

0.16

Your Total Score: 19

Solve Next

Sort Elements by Decreasing Frequency

Zero Sum Subarrays

```
30  
31  
32 // User function Template for Java  
33  
34 class Solution {  
35     // Should return true if there is a triplet with sum equal  
36     // to x in arr[], otherwise false  
37     public static boolean find3Numbers(int arr[], int n, int x) {  
38         // Your code here  
39  
40         Arrays.sort(arr);  
41         for (int i=0; i<n-2; i++){  
42             int l=i+1;  
43             int r=n-1;  
44             while(l<r){  
45                 int a=arr[i]+arr[l]+arr[r];  
46                 if(a==x){  
47                     return true;  
48                 }  
49                 else if (a<x){  
50                     l++;  
51                 }  
52                 else{  
53                     r--;  
54                 }  
55             }  
56         }  
57         return false;  
58     }  
59 }
```

Time Complexity: $O(n^2)$

Space Complexity: $O(1)$