

# DSA Practice

Date: 11/11/2024

## Problems:

0-1 knapsack problem

Floor in sorted array

Check equal arrays

Palindrome linked list

Balanced tree check

Triplet sum in array

## Problem 1:

```
class Solution {  
    // Function to return max value that can be put in knapsack of capacity.  
    static int knapSack(int capacity, int val[], int wt[]) {  
        // code here  
        int n = val.length;  
        int value=0;  
        int[] max = new int[]{-1};  
        func(capacity,val,wt,value,max,0);  
        return max[0];  
    }  
    static void func(int capacity,int[] val,int[] wt,int value,int[] max,int index){  
        if(capacity==0 || index==val.length){  
            if(value>max[0]){  
                max[0]=value;  
            }  
            return;  
        }  
    }  
}
```

```

    }

    if(wt[index]<=capacity){

        func(capacity-wt[index],val,wt,value+val[index],max,index+1);

    }

    func(capacity,val,wt,value,max,index+1);

}

```

## Output:

The screenshot displays the GeeksforGeeks submission interface for the '0-1 Knapsack Problem'. The left sidebar indicates a successful solution with 1115/1115 test cases passed, 2/5 attempts, and 40% accuracy. The main area shows the Java code for the solution, which includes a recursive function 'func' and a 'knapSack' method. The code is as follows:

```

// Driver Code Ends
37
38
39 class Solution {
40     // Function to return max value that can be put in knapsack of capacity.
41     static int knapSack(int capacity, int val[], int wt[]) {
42         // code here
43         int n = val.length;
44         int value=0;
45         int[] max = new int[]{-1};
46         func(capacity, val, wt, value, max, 0);
47         return max[0];
48     }
49     static void func(int capacity, int[] val, int[] wt, int value, int[] max, int index){
50         if(capacity==0 || index==val.length){
51             if(value>max[0]){
52                 max[0]=value;
53             }
54             return;
55         }
56         if(wt[index]<=capacity){
57             func(capacity-wt[index], val, wt, value+val[index], max, index+1);
58         }
59         func(capacity, val, wt, value, max, index+1);
60     }
61 }
62

```

## Problem 2:

```

Class Solution{

    static int findFloor(int[] arr, int k) {

        // write code here

        int n = arr.length;

        int max = -1;

        int maxindex = -1;

        for(int i=0;i<n;i++){

```

```

        if(arr[i]<=k && arr[i]>max){
            max=arr[i];
            maxindex=i;
        }
    }

    return maxindex;
}
}

```

### Output:

The screenshot shows the GeeksforGeeks online IDE interface. On the left, the 'Output Window' displays the following information:

- Problem Solved Successfully** (with a green checkmark)
- Test Cases Passed:** 1111 / 1111
- Attempts:** Correct / Total: 1 / 2
- Accuracy:** 50%
- Points Scored:** 2 / 2
- Your Total Score:** 6 (with a green up arrow)
- Time Taken:** 0.32
- Solve Next:**
  - Index of an Extra Element
  - Minimum in a Sorted and Rotated
  - Search insert position of K in a sorted array

On the right, the code editor shows the following Java code:

```

1 // } Driver Code Ends
2
3 class Solution {
4
5     static int findFloor(int[] arr, int k) {
6         // write code here
7         int n = arr.length;
8         int max = -1;
9         int maxindex = -1;
10        for(int i=0; i<n; i++){
11            if(arr[i]<=k && arr[i]>max){
12                max=arr[i];
13                maxindex=i;
14            }
15        }
16        return maxindex;
17    }
18 }
19 // } Driver Code Ends

```

### Problem 3:

```

class Solution {

    // Function to check if two arrays are equal or not.
    public static boolean check(int[] arr1, int[] arr2) {

        // Your code here

        Map<Integer,Integer> map1 = new HashMap<>();
        Map<Integer,Integer> map2 = new HashMap<>();
    }
}

```

```

for(int i=0;i<arr1.length;i++){
    map1.put(arr1[i],map1.getOrDefault(arr1[i],0)+1);
    map2.put(arr2[i],map2.getOrDefault(arr2[i],0)+1);
}

for(Map.Entry<Integer,Integer> entry : map1.entrySet()){
    int key = entry.getKey();
    int val = entry.getValue();
    if(val != map2.getOrDefault(key,0)){
        return false;
    }
}

return true;
}
}

```

## Output:

The screenshot shows the GeeksforGeeks online IDE interface. The left sidebar displays the 'Output Window' with the following details:

- Problem Solved Successfully** (with a green checkmark icon)
- Test Cases Passed:** 1116 / 1116
- Attempts:** Correct / Total: 1 / 3
- Accuracy:** 33%
- Points Scored:** 1 / 1
- Your Total Score:** 7 (with a green upward arrow icon)
- Solve Next:** Count pair sum, Permutations in array, The problem of identical arrays

The right sidebar shows the Java code editor with the following code:

```

1 // Driver Code Ends
2
3 // User function Template for Java
4
5 class Solution {
6     // Function to check if two arrays are equal or not.
7     public static boolean check(int[] arr1, int[] arr2) {
8         // Your code here
9         Map<Integer,Integer> map1 = new HashMap<>();
10        Map<Integer,Integer> map2 = new HashMap<>();
11        for(int i=0;i<arr1.length;i++){
12            map1.put(arr1[i],map1.getOrDefault(arr1[i],0)+1);
13            map2.put(arr2[i],map2.getOrDefault(arr2[i],0)+1);
14        }
15        for(Map.Entry<Integer,Integer> entry : map1.entrySet()){
16            int key = entry.getKey();
17            int val = entry.getValue();
18            if(val != map2.getOrDefault(key,0)){
19                return false;
20            }
21        }
22        return true;
23    }
24 }

```

The bottom status bar shows the system clock as 20:34 on 11-11-2024, and the language is set to Java (1.8).

## Problem 4:

```

class Solution {
    // Function to check whether the list is palindrome.
    boolean isPalindrome(Node head) {
        // Your code here
        if(head==null ){
            return false;
        }
        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;
    }
    Node reverse(Node head){
        Node prev=null;

```

```

Node curr = head;

while(curr!=null){

    Node nextnode = curr.next;

    curr.next=prev;

    prev=curr;

    curr=nextnode;

}

return prev;

}

}

```

## Output:

The screenshot shows the GeeksforGeeks website interface for the problem "Check if linked list is palindrome". The "Output Window" on the left indicates that the problem was solved successfully with 1112/1112 test cases passed, 1/12 attempts correct, and a time taken of 1.83. The code editor on the right shows the Java solution, which uses a two-pointer approach to check for a palindrome in a linked list. The code includes a `Node` class, a `reverse` method, and a `isPalindrome` method.

```

// Java code to check if a linked list is a palindrome
// Function to check whether the list is palindrome.
// Your code here
// if(head==null){
//     return false;
// }
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;
}

Node reverse(Node head){
    Node prev=null;
    Node curr = head;
    while(curr!=null){
        Node nextnode = curr.next;
        curr.next=prev;
        prev=curr;
        curr=nextnode;
    }
    return prev;
}

```

## Problem 5:

```

class Tree
{

```

//Function to check whether a binary tree is balanced or not.

```

boolean isBalanced(Node root){
    // Your code here
    int val = check(root);
    if(val==-1){
        return false;
    }
    return true;
}

int check (Node node){
    if(node==null){
        return 0;
    }
    int leftheight = check(node.left);
    int rightheight = check(node.right);
    if(leftheight == -1 || rightheight == -1){
        return -1;
    }
    if(Math.abs(leftheight-rightheight) > 1){
        return -1;
    }
    return Math.max(leftheight,rightheight)+1;
}
}

```

Output:

Sign in | Practice | GeeksforGeeks | A.com | X | Balanced Tree Check | Practice | X |

https://www.geeksforgeeks.org/problems/check-for-balanced-tree/1

Courses | Tutorials | Jobs | Practice | Contests

Problem | Editorial | Submissions | Comments

Output Window

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

Problem Solved Successfully

Test Cases Passed: 1120 / 1120

Attempts: Correct / Total: 1 / 1

Accuracy: 100%

Points Scored: 2 / 2

Your Total Score: 13

Solve Next

Height of Binary Tree | Minimum Depth of a Binary Tree | Array to BST

Java (1.8) | Average Time: 20m | Start Timer

```

123 Node left, right;
124
125 Node(int d)
126 {
127     data = d;
128     left = right = null;
129 }
130
131 class Tree
132 {
133     //Function to check whether a binary tree is balanced or not.
134     boolean isbalanced(Node root){
135         // Your code here
136         int val = check(root);
137         if(val == -1){
138             return false;
139         }
140         return true;
141     }
142
143     int check (Node node){
144         if(node == null){
145             return 0;
146         }
147         int leftheight = check(node.left);
148         int rightheight = check(node.right);
149         if(leftheight == -1 || rightheight == -1){
150             return -1;
151         }
152         if(Math.abs(leftheight - rightheight) > 1){
153             return -1;
154         }
155         return Math.max(leftheight, rightheight) + 1;
156     }
157 }
158
159
160

```

Custom Input | Compile & Run | Submit

28°C Mostly cloudy | Search | 22:00 11-11-2024

## Problem 6:

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 left=i+1;

int right=n-1;

while(left<right){

int currentsum = arr[i]+arr[left]+arr[right];

if(currentsum==x){

return true;

}else if(currentsum<x){



```

        left++;
    }else{
        right--;
    }
}

return false;
}
}

```

## Output:

The screenshot shows the GeeksforGeeks website interface for the 'Triplet Sum in Array' problem. The left sidebar indicates that the problem has been solved successfully, with 125 out of 125 test cases passed, 1 out of 3 attempts correct, 33% accuracy, 4 out of 4 points scored, and a time taken of 0.15 seconds. The main area displays the Java code for the solution, which sorts the array and uses a two-pointer technique to find triplets that sum to x.

```

1 // Driver Code Ends
2
3 // User function Template for Java
4
5 class Solution {
6     // Should return true if there is a triplet with sum equal
7     // to x in arr[], otherwise false
8     public static boolean findNumbers(int arr[], int n, int x) {
9         // Your code Here
10        Arrays.sort(arr);
11        for(int i=0; i<n-2; i++){
12            int left=i+1;
13            int right=n-1;
14            while(left<right){
15                int currentsum = arr[i]+arr[left]+arr[right];
16                if(currentsum==x){
17                    return true;
18                }else if(currentsum<x){
19                    left++;
20                }else{
21                    right--;
22                }
23            }
24        }
25        return false;
26    }
27 }
28

```