



Experiment No. - 5

Student Name: Vivek Kumar
Branch: BE-CSE(LEET)
Semester: 6th
Subject Name: Competitive coding - II

UID: 21BCS8129
Section/Group: 20BCS-ST-801/B
Date of Performance: 07/03/2023
Subject Code: 20CSP-351

1. Aim/Overview of the practical:

Q.1 Balance Binary Tree.

<https://leetcode.com/problems/balanced-binary-tree/>

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Tree
- To implement the concept of Balance Binary Tree.

4. Code:

```
class Solution {  
  
    public boolean isBalanced(TreeNode root) {  
        return dfsHeight(root) != -1;  
    }  
  
    public static int dfsHeight(TreeNode root){  
        if(root == null){  
            return 0;  
        }  
  
        int leftHeight = dfsHeight(root.left);  
  
        if(leftHeight == -1){  
            return -1;  
        }  
  
        int rightHeight = dfsHeight(root.right);  
  
        if(rightHeight == -1){  
            return -1;  
        }  
  
        if(Math.abs(leftHeight - rightHeight) > 1){  
            return -1;  
        }  
        return Math.max(leftHeight,rightHeight)+1;  
    }  
}
```

5. Result/Output/Writing Summary:

Balanced Binary Tree - LeetCode

<https://leetcode.com/problems/balanced-binary-tree/>

LeetCode Explore Problems Interview Contest Discuss Store

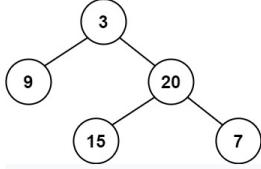
Description Solution Discuss (999+) Submissions

110. Balanced Binary Tree

Easy 861 486 Add to List Share

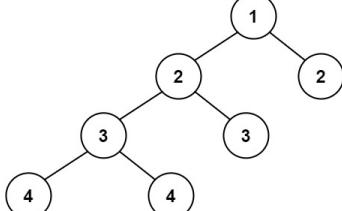
Given a binary tree, determine if it is **height-balanced**.

Example 1:



```
Input: root = [3,9,20,null,null,15,7]
Output: true
```

Example 2:



```
Accepted Runtime: 0 ms
```

Your input: [3,9,20,null,null,15,7]

Output: true

Expected: true

Java Autocomplete

```

4 *     int val;
5 *     TreeNode left;
6 *     TreeNode right;
7 *     TreeNode() {}
8 *     TreeNode(int val) { this.val = val; }
9 *     TreeNode(int val, TreeNode left, TreeNode right) {
10 *         this.val = val;
11 *         this.left = left;
12 *         this.right = right;
13 *     }
14 *
15 *
16 */
class Solution {
17
18     public boolean isBalanced(TreeNode root) {
19         return dfsHeight(root) != -1;
20     }
21
22     public static int dfsHeight(TreeNode root){
23         if(root == null){
24             return 0;
25         }
26
27         int leftHeight = dfsHeight(root.left);
28
29         if(leftHeight == -1){
30             return -1;
31         }
32
33         int rightHeight = dfsHeight(root.right);
34
35         if(rightHeight == -1){
36             return -1;
37         }
38
39         if(Math.abs(leftHeight - rightHeight) > 1){
40             return -1;
41         }
42         return Math.max(leftHeight,rightHeight)+1;
43     }
44 }
```

Testcase Run Code Result Debugger

Problems Pick One < Prev 110/2603 Next > Console Use Example Testcases Run Code Submit

19°C Mostly clear ENG 2:48 AM 3/30/2023

Balanced Binary Tree - LeetCode

<https://leetcode.com/problems/balanced-binary-tree/submissions/>

LeetCode Explore Problems Interview Contest Discuss Store

Description Solution Discuss (999+) Submissions

Success Details >

Runtime: 0 ms, faster than 100.00% of Java online submissions for Balanced Binary Tree.

Memory Usage: 42.9 MB, less than 9.48% of Java online submissions for Balanced Binary Tree.

Next challenges:

Maximum Depth of Binary Tree

Show off your acceptance:   

Time Submitted	Status	Runtime	Memory	Language
03/30/2023 02:48	Accepted	0 ms	42.9 MB	java

Java Autocomplete

```

4 *     int val;
5 *     TreeNode left;
6 *     TreeNode right;
7 *     TreeNode() {}
8 *     TreeNode(int val) { this.val = val; }
9 *     TreeNode(int val, TreeNode left, TreeNode right) {
10 *         this.val = val;
11 *         this.left = left;
12 *         this.right = right;
13 *     }
14 *
15 *
16 */
class Solution {
17
18     public boolean isBalanced(TreeNode root) {
19         return dfsHeight(root) != -1;
20     }
21
22     public static int dfsHeight(TreeNode root){
23         if(root == null){
24             return 0;
25         }
26
27         int leftHeight = dfsHeight(root.left);
28
29         if(leftHeight == -1){
30             return -1;
31         }
32
33         int rightHeight = dfsHeight(root.right);
34
35         if(rightHeight == -1){
36             return -1;
37         }
38
39         if(Math.abs(leftHeight - rightHeight) > 1){
40             return -1;
41         }
42         return Math.max(leftHeight,rightHeight)+1;
43     }
44 }
```

Testcase Run Code Result Debugger

Accepted Runtime: 0 ms

Your input: [3,9,20,null,null,15,7]

Output: true

Expected: true

Problems Pick One < Prev 110/2603 Next > Console Use Example Testcases Run Code Submit

19°C Mostly clear ENG 2:48 AM 3/30/2023

1. Aim/Overview of the practical:

Q.2 Path Sum

<https://leetcode.com/problems/path-sum/>

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

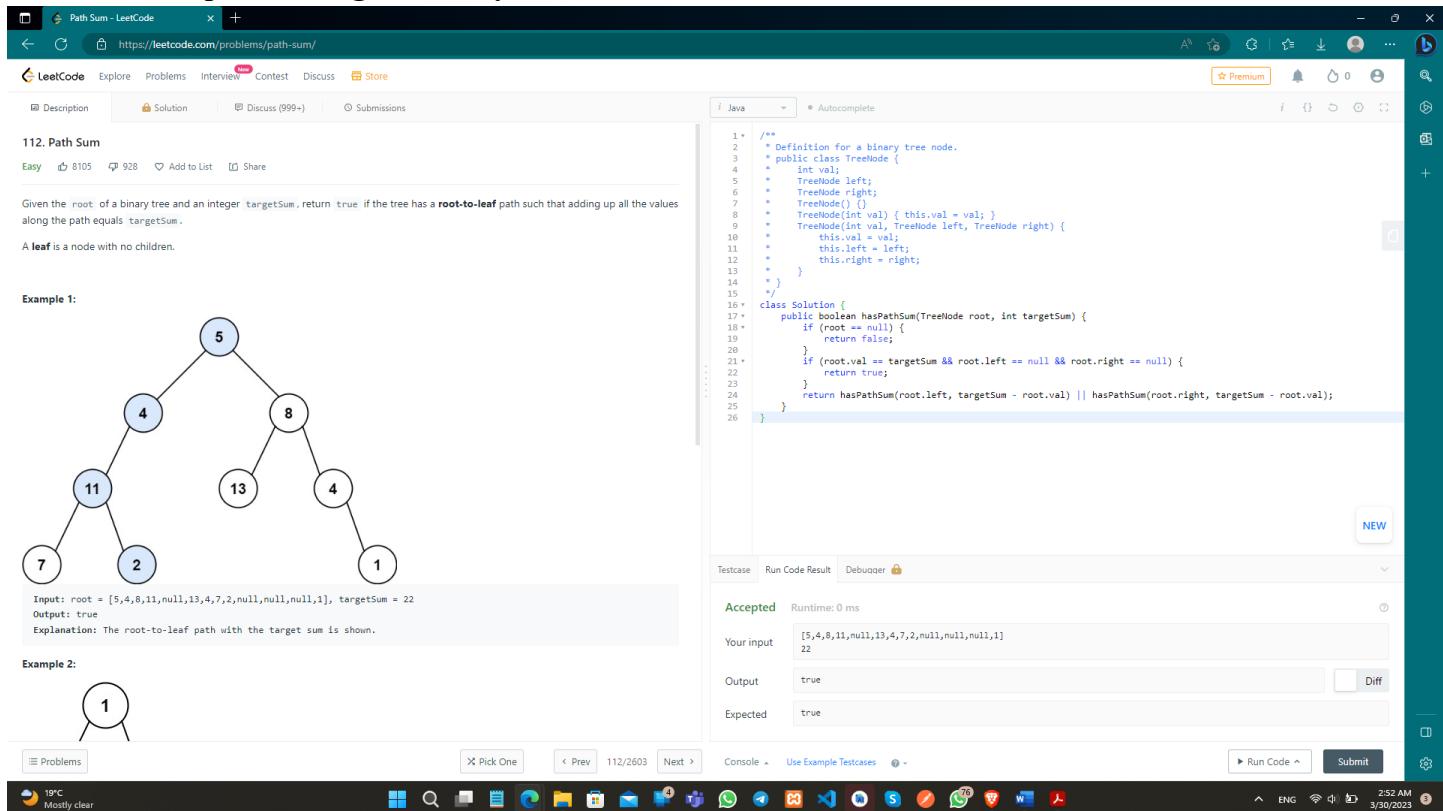
3. Objective:

- To understand the concept of Tree traversal.
- To implement the concept of calculate the path sum.

4. Code:

```
class Solution {
    public boolean hasPathSum(TreeNode root, int targetSum) {
        if (root == null) {
            return false;
        }
        if (root.val == targetSum && root.left == null && root.right == null) {
            return true;
        }
        return hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val);
    }
}
```

5. Result/Output/Writing Summary:



The screenshot shows the LeetCode interface for problem 112. Path Sum. The problem description and example 1 are visible, along with a binary tree diagram. The Java code solution is pasted into the code editor. The code defines a `TreeNode` class and a `Solution` class with a `hasPathSum` method. The tree diagram shows a root node with value 5, left child 4, right child 8. Node 4 has children 11 and 2. Node 8 has children 13 and 4, which in turn has a child 1. Example 1 shows the path from root to leaf node 2, which sums to 22. The Java code uses recursion to check if the current node's value plus the sum of its children's paths equals the target sum.



DEPARTMENT OF ACADEMIC AFFAIRS

Discover. Learn. Empower.

NAAC
GRADE A+
ACCREDITED UNIVERSITY

The screenshot shows a LeetCode submission for the 'Path Sum' problem. The code is a Java solution for a binary tree node and a hasPathSum method. It uses recursion to check if the sum of values from the root to any leaf node equals a target sum. The submission was accepted with 0 ms runtime and 42.5 MB memory usage. The test case passed with input [5,4,8,11,null,13,4,7,2,null,null,null,1] and output true. The interface includes navigation buttons, a search bar, and a weather widget at the bottom.

```
1* /**
2*  * Definition for a binary tree node.
3*  */
4* public class TreeNode {
5*     int val;
6*     TreeNode left;
7*     TreeNode right;
8*     TreeNode() {}
9*     TreeNode(int val) { this.val = val; }
10*    TreeNode(int val, TreeNode left, TreeNode right) {
11*        this.val = val;
12*        this.left = left;
13*        this.right = right;
14*    }
15* }
16*
17* class Solution {
18*     public boolean hasPathSum(TreeNode root, int targetSum) {
19*         if (root == null) {
20*             return false;
21*         }
22*         if (root.val == targetSum && root.left == null && root.right == null) {
23*             return true;
24*         }
25*         return hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val);
26*     }
27* }
```

Time Submitted	Status	Runtime	Memory	Language
03/30/2023 02:53	Accepted	0 ms	42.5 MB	java

Testcase: Accepted Runtime: 0 ms

Your input: [5,4,8,11,null,13,4,7,2,null,null,null,1]
Output: true
Expected: true

Problems | Pick One | < Prev | 112/2603 | Next > | Console | Use Example Testcases | Run Code | Submit | 19°C | 253 AM | 3/30/2023

Learning outcomes (What I have learnt):

- Learned the concept of Balanced Binary Tree.
- Learnt about Tree and Path Sum.