

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

EXPERIMENT 1.5

Student Name: Praduman Kumar

Branch: CSE

Semester: 06

Subject Code: 20CSP-351

UID: 20BCS9446

Section/Group: 20BCS_DM_714_A

Subject Name: Competitive Coding

1. **AIM:** To demonstrate the concept of trees

2. **OBJECTIVE 1:** Balanced Binary Tree.

3. **CODE:**

```
class Solution {
public:
    bool isBalanced(TreeNode* root) {

        return dfsHeight(root) != -1;

    }

    int dfsHeight(TreeNode* root)
    {
        if(root==NULL) return 0;

        int lh = dfsHeight(root->left);

        if(lh==-1) return -1;

        int rh = dfsHeight(root->right);
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
        if(rh==-1) return -1;

        if(abs(lh-rh) > 1) return -1;

        return max(lh,rh)+1;
    }
};
```

4. OUTPUT:

LeetCode

< Problem List >

Premium

Description Editorial Solutions (4.6K) Submissions

C++ Auto

110. Balanced Binary Tree

Easy 8.5K 485

Companies

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

Example 1:

```
graph TD
    3((3)) --- 9((9))
    3 --- 20((20))
    20 --- 15((15))
    20 --- 7((7))
```

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

```
12 class Solution {
13 public:
14     bool isBalanced(TreeNode* root) {
15         return dfsHeight(root) != -1;
16     }
17 }
18
19
20     int dfsHeight(TreeNode* root)
21     {
22         if(root==NULL) return 0;
23
24         int lh = dfsHeight(root->left);
25
26         if(lh==-1) return -1;
27
28         int rh = dfsHeight(root->right);
29
30         if(rh==-1) return -1;
31
32         if(abs(lh-rh) > 1) return -1;
33
34         return max(lh,rh)+1;
35     }
36 };
37
38
39
40
```

Console ^

Run Submit

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

The screenshot shows the LeetCode interface for the problem '111. Minimum Depth of Binary Tree'. On the left, there are sections for 'Next question' (111. Minimum Depth of Binary Tree), 'More challenges' (129. Sum Root to Leaf Numbers, 1597. Build Binary Expression Tree From Infix Expression, 173. Binary Search Tree Iterator), and a list of accepted solutions. The main area displays the problem title, a C++ solution, and performance metrics: Runtime 22 ms (Beats 17.93%), Memory 21 MB (Beats 29.67%). Below the metrics are sections for 'Notes', 'Related Tags', and a 'Console' area. The solution code is as follows:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 * };
 */

class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {
        if(root==NULL) return false;
        curr+=root->val;
        if(curr==targetSum && root->left==NULL && root->right==NULL) return true;

        bool left =solve(root->left,targetSum,curr);
        bool right=solve(root->right,targetSum,curr);
        curr-=root->val;

        return left || right;
    }
    bool hasPathSum(TreeNode* root, int targetSum)
    { if(root==NULL) return false;
```

5. OBJECTIVE 2: Path Sum

6. CODE:

```
class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {

        if(root==NULL) return false;
        curr+=root->val;
        if(curr==targetSum && root->left==NULL && root->right==NULL) return true;

        bool left =solve(root->left,targetSum,curr);
        bool right=solve(root->right,targetSum,curr);
        curr-=root->val;

        return left || right;
    }
    bool hasPathSum(TreeNode* root, int targetSum)
    { if(root==NULL) return false;
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
int cr=0;
bool ans=solve(root,targetSum,cr);
return ans;
}
};
```

7. OUTPUT

The screenshot displays the LeetCode interface for the 'Path Sum' problem (112). On the left, the problem description states: 'Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals targetSum. A leaf is a node with no children.' Below this, 'Example 1:' shows a binary tree with root 5, left child 4, right child 8, and further children 11, 13, and 4 respectively. On the right, the C++ code is shown in a text editor. The code defines a 'Solution' class with a 'hasPathSum' method that uses a recursive 'solve' function to check for a root-to-leaf path summing to the target. The code is as follows:

```
12 class Solution {
13 public:
14     bool hasPathSum(TreeNode* root, int targetSum) {
15
16         if(root==NULL) return false;
17         curr+=root->val;
18         if(curr==targetSum && root->left==NULL && root->right==NULL) return true;
19
20         bool left =solve(root->left,targetSum,curr);
21         bool right=solve(root->right,targetSum,curr);
22         curr-=root->val;
23
24         return left || right;
25     }
26     bool hasPathSum(TreeNode* root, int targetSum) {
27         if(root==NULL) return false;
28         int cr=0;
29         bool ans=solve(root,targetSum,cr);
30         return ans;
31     }
32 };
33
34
35
36
37
38
39
40
```

The bottom of the image shows a Windows taskbar with the date and time as 10:54 AM on 3/21/2023.

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Next question

113. Path Sum II

More challenges

113. Path Sum II129. Sum Root to Leaf Numbers

437. Path Sum III

All statusesAll languages

Accepted
a minute agoC++

Compile Error
3 minutes agoC++

C++

Runtime7 ms

Beats92.70%

Memory21.2 MB

Beats76.72%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags0/5

/**
 * Definition for a binary tree node.
 * struct TreeNode {
 * int val;
 * TreeNode *left;
 * }

Console

RunSubmit