

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## Experiment-1.5

**Student Name:** Kanishk Soni  
**Branch:** BE-CSE  
**Semester:** 6<sup>th</sup>  
**Subject Code:** 20CSP-351

**UID:** 20BCS9398  
**Section/Group:** 20BCS\_DM-708B  
**Subject Name:** Competitive Coding-II

**AIM:** To demonstrate the concept of Trees.

**Problem1:** Path Sum

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

**Program Code:**

```
class Solution {  
public:  
    bool hasPathSum(TreeNode* root, int targetSum) {  
        if (!root) return false;  
        if (!root->left && !root->right) return root->val == targetSum;  
        return  
            hasPathSum(root->left, targetSum - root->val) ||  
            hasPathSum(root->right, targetSum - root->val);  
    }  
};
```

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Output:

The screenshot displays a LeetCode interface for problem 113, "Path Sum II". On the left, a sidebar shows the problem is "Accepted" and lists other challenges like "113. Path Sum II", "124. Binary Tree Maximum Path Sum", and "129. Sum Root to Leaf Numbers". The main area shows the problem title, a C++ solution, and performance metrics: Runtime 8 ms, Beats 85.78%, Memory 21.1 MB, and Beats 93.68%. The solution code is as follows:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {
        if (!root) return false;

        if (!root->left && !root->right) return root->val == targetSum;
    }
};
```

## Problem2: Symmetric Tree

<https://leetcode.com/problems/symmetric-tree/>

## Program Code:

```
class Solution {
private:
    bool isEquivalent(TreeNode* left, TreeNode* right) {
        if (!left || !right) {
            return left == right;
        } else {
            return left->val==right->val && isEquivalent(left->left, right->right) &&
isEquivalent(right->left, left->right);
        }
    }
}
```

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

public:

```
bool isSymmetric(TreeNode* root) {  
    return isEquivalent(root->left, root->right);  
}  
};
```

**Output:**

Accepted

Next question

102. Binary Tree Level Order Traversal

More challenges

1602. Find Nearest Right Node in Binary Tree

222. Count Complete Tree Nodes

733. Flood Fill

All statuses

All languages

Accepted  
a few seconds ago

C++

Runtime 0 ms

Beats 100%

Memory 16.4 MB

Beats 84.22%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags

0/5

```
/**  
 * Definition for a binary tree node.  
 * struct TreeNode {  
 *     int val;  
 *     TreeNode *left;  
 *     TreeNode *right;  
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}  
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}  
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}  
 * };  
 */  
class Solution {  
private:  
    bool isEquivalent(TreeNode* left, TreeNode* right) {  
        if (!left || !right) {  
            return left == right;  
        } else {  
            return left->val == right->val && isEquivalent(left->left, right->left) && isEquivalent(left->right, right->right);  
        }  
    }  
public:  
    bool isSymmetric(TreeNode* root) {  
        return isEquivalent(root->left, root->right);  
    }  
};
```

Console

Run

Submit