

MCS – 253P ADVANCED PROGRAMMING AND PROBLEM SOLVING

HOMEWORK –5 (Binary Tree Maximum Path Sum)

Aswin Sampath
saswin@uci.edu
(53844684)

[Description](#) [Editorial](#) [Solutions \(4.9K\)](#) [Submissions](#)

124. Binary Tree Maximum Path Sum

Hard ✓ 15.8K 683 ☆ ↻

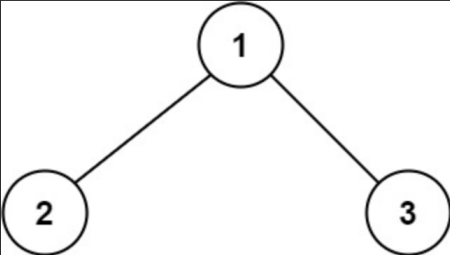
Companies

A **path** in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence **at most once**. Note that the path does not need to pass through the root.

The **path sum** of a path is the sum of the node's values in the path.

Given the `root` of a binary tree, return the *maximum path sum* of any **non-empty** path.

Example 1:



```
Input: root = [1,2,3]
Output: 6
Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 = 6.
```

Understanding the Problem

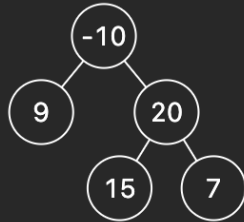
The problem statement asks us to find the maximum path sum in a binary tree. A path in a binary tree is defined as a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. The path sum of a path is the sum of the node values in that path. It's important to note that the path does not need to pass through the root of the tree. Given the root of a binary tree, we need to determine the maximum path sum among all non-empty paths in the tree.

Identifying Edge Cases

- An empty binary tree: When the tree is empty (root is nullptr), the maximum path sum is undefined. The code should handle this case and return an appropriate value.
- A single-node tree: When the tree consists of only one node, the maximum path sum is simply the value of that node itself.

Effective Test Cases

1)



Input
root = [-10,9,20,null,null,15,7]
Output
42
Expected
42

2)

Null Tree returns Null value

3)

Single Node tree returns the node value

Algorithmic Solution

1. The solve function is a recursive function that calculates the maximum path sum starting from a given node.
2. In each recursive call, it checks if the current node is nullptr. If it is, the function returns 0, as there is no path to consider.
3. The function calculates the maximum path sum for the left subtree (ls) and the right subtree (rs). It ensures that negative values are not included in the path sum by taking the maximum of 0 and the values from the left and right subtrees.
4. The maximum path sum that includes the current node is calculated as $ls + rs + \text{root} \rightarrow \text{val}$, and this is compared with the maximum result (res) obtained so far.

5. The function returns the maximum path sum that can extend either from the left subtree or the right subtree plus the value of the current node.
6. In the `maxPathSum` function, it initializes `res` to `INT_MIN`, calls `solve` to calculate the maximum path sum starting from the root, and returns the final result.

Time and Space Complexity Analysis

The time complexity of this algorithm is $O(n)$, where n is the number of nodes in the binary tree. This is because we visit each node once in a bottom-up manner.

The space complexity is $O(h)$, where h is the height of the binary tree. In the worst case, when the binary tree is skewed, the space complexity is $O(n)$, but in a balanced binary tree, it is $O(\log n)$.

Code:

```
1  class Solution {
2  public:
3      int solve(TreeNode root, int &res) {
4          if (!root) return 0;
5          int ls = max(0, solve(root->left, res));
6          int rs = max(0, solve(root->right, res));
7          res = max(res, ls + rs + root->val);
8          return max(ls, rs) + root->val;
9      }
10
11     int maxPathSum(TreeNode root) {
12         int res = INT_MIN;
13         solve(root, res);
14         return res;
15     }
16 };
```

Output:

DescriptionEditorialSolutions (4.9K)Submissions

Accepted

Editorial+ Solution

Runtime

24 ms

Beats 28.32% of users with C++

Memory

27.83 MB

Beats 86.70% of users with C++

More challenges

666. Path Sum IV

687. Longest Univalue Path

1376. Time Needed to Inform All Employees

Status

Language

Runtime

Memory

Notes

Accepted

11 minutes ago

C++

24 ms

27.8 MB

Aswin

Nov 07, 2023 20:50

Details+ Solution

C++

Runtime 24 ms

Beats 28.32%

Memory 27.8 MB

Beats 86.70%

Click the distribution chart to view more details

Related Tags

Select related tags0/5

```
/**
 * 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 {
public:
    int solve(TreeNode* root, int &res) {
        if (!root) return 0;
        int ls = max(0, solve(root->left, res));
        int rs = max(0, solve(root->right, res));
```