

educative

13% completed

Search Course

Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

We'll cover the following

- Path with Maximum Sum (hard)
 - Solution
 - Code
 - Time complexity
 - Space complexity

Solution Review: Problem Challenge 2

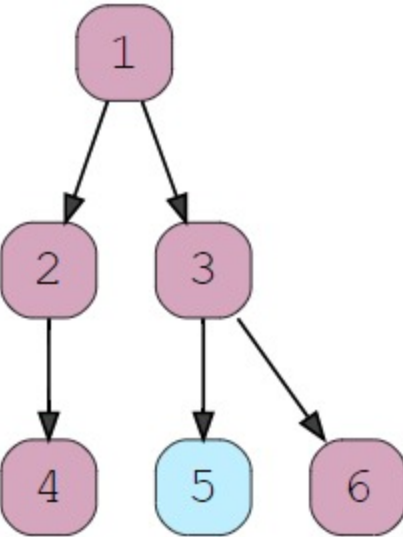
Path with Maximum Sum (hard)

Find the path with the maximum sum in a given binary tree. Write a function that returns the maximum sum.

A path can be defined as a **sequence of nodes between any two nodes** and doesn't necessarily pass through the root. The path must contain at least one node.

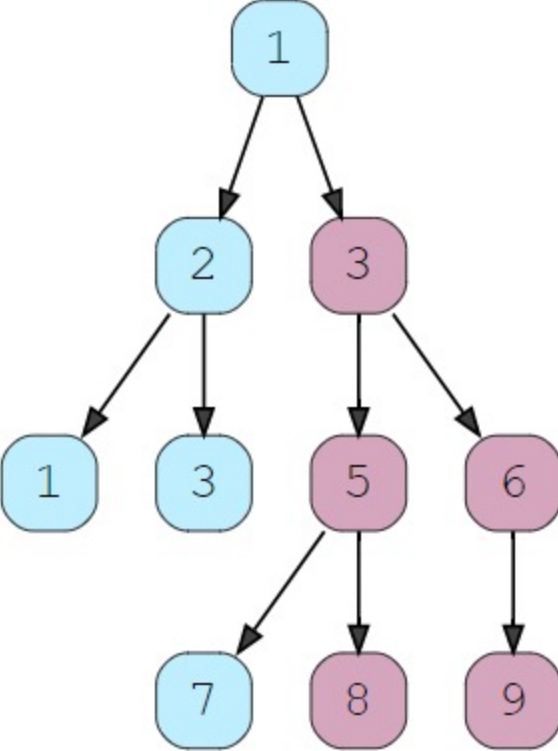
Example 1:

Output: 16
Explanation: The path with maximum sum is: [4, 2, 1, 3, 6]



Example 2:

Output: 31
Explanation: The path with maximum sum is: [8, 5, 3, 6, 9]



Solution

This problem follows the [Binary Tree Path Sum](#) pattern and shares the algorithmic logic with [Tree Diameter](#). We can follow the same **DFS** approach. The only difference will be to ignore the paths with negative sums. Since we need to find the overall maximum sum, we should ignore any path which has an overall negative sum.

Code

Here is what our algorithm will look like, the most important changes are in the highlighted lines:

Java

Python3

C++

JS

```
1 import math
2
3
4 class TreeNode:
5     def __init__(self, val, left=None, right=None):
6         self.val = val
7         self.left = left
8         self.right = right
9
10
11 class MaximumPathSum:
12
13     def find_maximum_path_sum(self, root):
14         self.globalMaximumSum = -math.inf
15         self.find_maximum_path_sum_recursive(root)
16         return self.globalMaximumSum
17
18     def find_maximum_path_sum_recursive(self, currentNode):
19         if currentNode is None:
20             return 0
21
22         maxPathSumFromLeft = self.find_maximum_path_sum_recursive(
23             currentNode.left)
24         maxPathSumFromRight = self.find_maximum_path_sum_recursive(
25             currentNode.right)
26
27         # ignore paths with negative sums, since we need to find the maximum sum we should
28         # ignore any path which has an overall negative sum.
29         maxPathSumFromLeft = max(maxPathSumFromLeft, 0)
30         maxPathSumFromRight = max(maxPathSumFromRight, 0)
31
```

Run

Save

Reset

Time complexity

The time complexity of the above algorithm is $O(N)$, where 'N' is the total number of nodes in the tree. This is due to the fact that we traverse each node once.

Space complexity

The space complexity of the above algorithm will be $O(N)$ in the worst case. This space will be used to store the recursion stack. The worst case will happen when the given tree is a linked list (i.e., every node has only one child).

← Back

Problem Challenge 2

Next →

Introduction

✔ Mark as Completed

🚩 Report an Issue

🔍 Ask a Question