Binary Tree Maximum Path Sum

Problem Description

Overview

A path in a binary tree is a sequence of nodes where an edge connects each pair of adjacent nodes in the sequence. A node can only appear once in any given path. Importantly, the path does not need to pass through the tree's root. The goal is to find any non-empty path's maximum sum in the binary tree.

Path Sum

The path sum is the sum of the values of the nodes in the path. Given the root of a binary tree, the task is to return the maximum path sum possible for any path within the tree.

Constraints

- The number of nodes in the tree is in the range [1, 30,000].
- Each node's value is between -1,000 and 1,000 inclusive.

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.

Example 2

Input: root = [-10,9,20,null,null,15,7]

Output: 42

Explanation: The optimal path is 15 -> 20 -> 7 with a path sum of 15 + 20 + 7 = 42.

Solution Approach

Key Points

1. *Recursive Helper Function:* The solution uses a helper function to compute the maximum path sum starting from each node recursively.

2. *Non-local Maximum Sum:* A non-local variable max_sum is used to keep track of the global maximum path sum.

3. *Path Gains:* For each node, the maximum path sums from the left and right children are computed, ensuring that they are non-negative (by taking the maximum with 0).

4. *New Path Price:* The potential new path sum that includes the current node and its maximum left and right gains is calculated.

5. *Update Maximum Sum:* The global maximum sum is updated if the new path sum is greater.

6. *Return Path Gain:* The function returns the maximum gain if the current node is included in the path.

Process

- *Base Case:* If the current node is None, return 0.
- *Compute Left and Right Gains:* Recursively compute the maximum path sums for the left and right children.
- *Calculate New Path Price:* Add the current node's value to the sum of the left and right gains.
- *Update Maximum Sum:* Compare the new path sum with the global maximum and update if necessary.
- *Return the Best Gain for Recursion:* Return the current node's value plus the maximum of the left and right gains to be used in the parent node's calculation.

Complexity

- *Time Complexity:* O(n), where n is the number of nodes in the tree. Each node is visited once.
- *Space Complexity:* O(h), where h is the height of the tree. This is due to the recursion stack.

This solution efficiently finds the maximum path sum by leveraging recursion and dynamic programming principles to ensure each node's contribution to the path sum is computed and considered optimally.