Solution: Binary Tree Maximum Path Sum

Let's solve the Binary Tree Maximum Path Sum problem using the Tree Depth-first Search pattern.

We'll cover the following

Statement

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

A path in a binary tree is defined as follows:

- A sequence of nodes in which each pair of adjacent nodes must have an edge connecting them.
  - A node can only be included in a path once at most.
  - Including the root in the path is not compulsory.

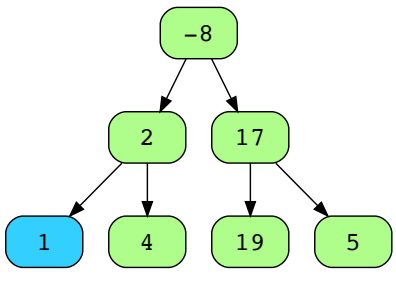
You can calculate the path sum by adding up all node values in the path. To solve this problem, calculate the maximum path sum given the root of a binary tree so that there won't be any greater path than it in the tree.

Constraints:

- $1 \leq$ Number of nodes in the tree $\leq 3 \times 10^4$.
- $-1000 \leq$ Node value ≤ 1000

Solution

We'll start by simplifying the problem. We look at the implementation of the `maxContrib(node)` function, which takes a node as an argument and computes a maximum contribution that this node and one or none of its subtrees can add, for example:



maxContrib(1) = 1 as it doesn't have children.

Now, if we are sure that the `root` is also included in the **maximum path**, the problem can be solved by using `maxContrib(root)`, and then adding both the `maxContrib` of the left child if its greater than zero, and, doing the same for the right child. Unfortunately, this isn't always the case, and the **maximum path** can exclude `root`. Therefore, we need to make a few additions to the solution so we can check whether to continue with the current path or to start a new path with the current node as the highest node at each step.

First, we need to initialize a `maxSum` as a global variable of negative infinity. We'll start by calling `maxContrib` with `root` as its parameter. Second, we need to implement the `maxContrib(node)` function so we can decide whether to continue with the old path or start a new path. Here's how we'll implement it:

1. This is a recursive function, so the base case states that the `maxContrib` is `0` if the node is NULL.
2. Call `maxContrib` recursively for the node's children to calculate the max gain for the left and right subtrees. We also need to check if this value is greater than zero or not so we can avoid adding any negative values.
3. Now, check whether to continue with the old path or start a new path. To start a new path, the sum of this path will be `node.val + maxContrib(root.left) + maxContrib(root.right)`. Update `maxSum` if it's better to start a new path.
4. We'll return the max gain to the node, and one or none of its subtrees can be added to the current path—`node.val + max(leftSubtree, rightSubtree)`.

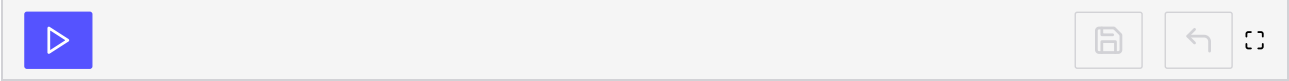
Java

main.java

BinaryTree.java

TreeNode.java

```
class MaxTreePathSum {

    public int maxSum = Integer.MIN_VALUE;

    public int maxContrib(TreeNode<Integer> root) {
        if (root == null)
            return 0;

        // sum of the left and right subtree
        int maxLeft = this.maxContrib(root.left);
        int maxRight = this.maxContrib(root.right);

        //max sum on the left and right sub-trees of root
        if (maxLeft > 0)
            leftSubtree = maxLeft;

        if (maxRight > 0)
            rightSubtree = maxRight;

        // the value to start a new path where `node` is a highest node
        int valueNewpath = root.data + leftSubtree + rightSubtree;

        // update maxSum if it's better to start a new path
        this.maxSum = Math.max(this.maxSum, valueNewpath);
```



Binary Tree Maximum Path Sum

Time complexity

The time complexity of this solution is $O(n)$, where n represents the number of nodes because each node is visited two times at most.

Space complexity

The space complexity of this solution is $O(h)$, where h is the height of the tree. This is because our recursive algorithm uses space on the call stack, which can grow to the height (h) of the binary tree. The complexity will be $O(logn)$ for a balanced tree and $O(n)$ for a degenerate tree.

Back

Binary Tree Maximum ...

Next

Maximum Depth of Bi...

Mark as Completed