#LeetCode 124

https://leetcode.com/problems/binary-tree-maximum-path-sum/description/

Yifeng Zeng

#题目描述

124. Binary Tree Maximum Path Sum

#思路报告

The problem explicitly says:

> "The path must contain at least one node and does not need to go through the root."

So this is a child-to-parent-to-child path sum. So we can split this path sum into three parts. A parent-to-child path sum from left, the root value itself, and a parent-to-child path sum from right. So we traverse the whole tree, for each root (parent), we need to compare this path's sum to the global maximum. Since we need the left side and right side of the parent-to-child path sum, we can use bottom up method.

When writing the code, we can discuss the corner cases: if left/right side path sum is less than 0, we need carefully compare all kinds of the situation. Another trick I learned in the class is to use int[] max; instead of the global variable.

代码如下:

```
public int maxPathSum(TreeNode root) {
    if (root == null) {
        return 0;
    }

    int[] max = new int[1];
    max[0] = root.val;
    maxPathSum(root, max);
    return max[0];
}
```

```
private int maxPathSum(TreeNode root, int[] max) {
    if (root == null) {
        return 0;
    }

    int left = maxPathSum(root.left, max);
    int right = maxPathSum(root.right, max);

    int ret = Math.max(root.val, Math.max(left + root.val, right + root.val));
    max[0] = Math.max(max[0], Math.max(ret, left + root.val + right));
    return ret;
}
```

#套路总结

- If we need information from left/right child and do not need pass any parent information down to children, we can just use bottom up method.
- Use int[] max = new int[1]; instead of a global variable.