LeetCode 第 124 号问题：二叉树中的最大路径和

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 124 号问题：二叉树中的最大路径和。题目难度为 Hard，目前通过率为 39.9% 。

题目描述

给定一个非空二叉树，返回其最大路径和。

本题中，路径被定义为一条从树中任意节点出发，达到任意节点的序列。该路径至少包含一个节点，且不一定经过根节点。

示例 1:

```
输入: [1,2,3]

       1
      / \
     2   3

输出: 6
```

示例 2:

```
输入: [-10,9,20,null,null,15,7]

   -10
   / \
  9  20
    /  \
   15   7

输出: 42
```

题目解析

二叉树问题，题目要求出一个二叉树的最大路径和，路径和就是把一条路径上面节点的值加起来，这一题的难点在于路径的方向不固定，只要是任意两点间的通路都算是有效路径，如果不提前列出合理的规划，这道题将无从下手。一般来说，解决树的问题都需要用到递归，**树上的搜索，本质上也是深度优先搜索**，但是这里会有两种考虑方式，一个是**自底向上的分治**，也就是进入递归，一开始不做任何节点上面的计算或者是处理，直接进入到下一层递归，直到到了最底层，然后再开始计算并返回答案，然后上层树节点的递归函数就会收到下层返回的结果，这样做的好处是，一个节点可以获知其子树的局部答案；另外一个是**自顶向下的遍历搜索**，这个和之前的思路完全相反，也就是先处理当前节点的内容，处理完后去到下一层节点，这种方法一般没有返回值，但是一般会有一个全局或者是引用变量，用来记录遍历过程中的内容。

我们再回过头来看这道题，在递归遍历的过程中，对于当前节点，其在路径中可以是路径尾，路径头（假设路径是从上到下的，其实在这道题中，没有头尾的概念），也可以是路径中的一个节点。那怎么判断呢？这时我们得需要当前节点左右子树的信息，所以我们可以考虑使用之前提到的 **自底向上** 的分治，有了当前节点，左右子树到当前节点的最

大路径，我们可以看看这里会有几种情况，我用 **root** 表示当前节点，**left** 表示左子树到 root 的最大和的路径，**right** 表示右子树到 root 的最大和的路径：

- root 和左右路径形成路径（left - root - right）
- root 和左路径形成路径（left - root）
- root 和右路径形成路径（root - right）
- root 自成路径（root）

你可以看到这四种情况都会把当前节点考虑在内，我们可以更新这里的最大值。但是需要注意的是，我们返回的时候，第一种情况是不能返回的，因为对于上一层节点来说，其无法形成有效的路径，因此我们只需要将 2，3，4 中的最大值返回即可，当然，更新全局答案的时候，这 4 种情况都需要考虑在内的。

代码实现

```
private int maximum = Integer.MIN_VALUE;

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

    helper(root);

    return maximum;
}

private int helper(TreeNode root) {
    if (root == null) {
        return 0;
    }
    // 如果左右子树返回的最大路径值小于 0
    // 直接将值设为 0，也就是不考虑对应的路径
    int leftMax = Math.max(0, helper(root.left));
    int rightMax = Math.max(0, helper(root.right));

    maximum = Math.max(root.val + leftMax + rightMax, maximum);

    return Math.max(leftMax + root.val, rightMax + root.val);
}
```

动画描述