# 题目

给定一个二叉树根节点 root ，树的每个节点的值要么是0，要么是1。请剪除该二叉树中所有节点的值为0的子树。

节点node的子树为 node本身，以及所有node 的后代。

示例 1:

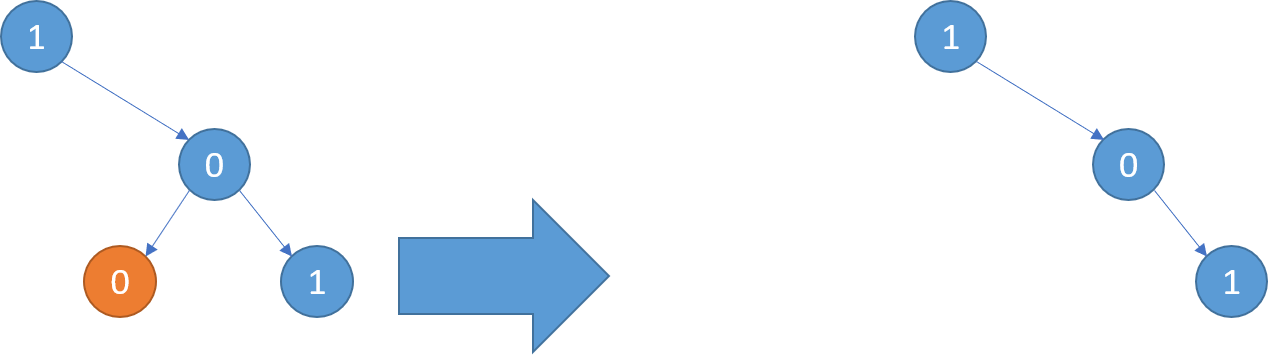
输入: [1,null,0,0,1]

输出: [1,null,0,null,1]

解释:

只有红色节点满足条件“所有不包含 1 的子树”。

右图为返回的答案。

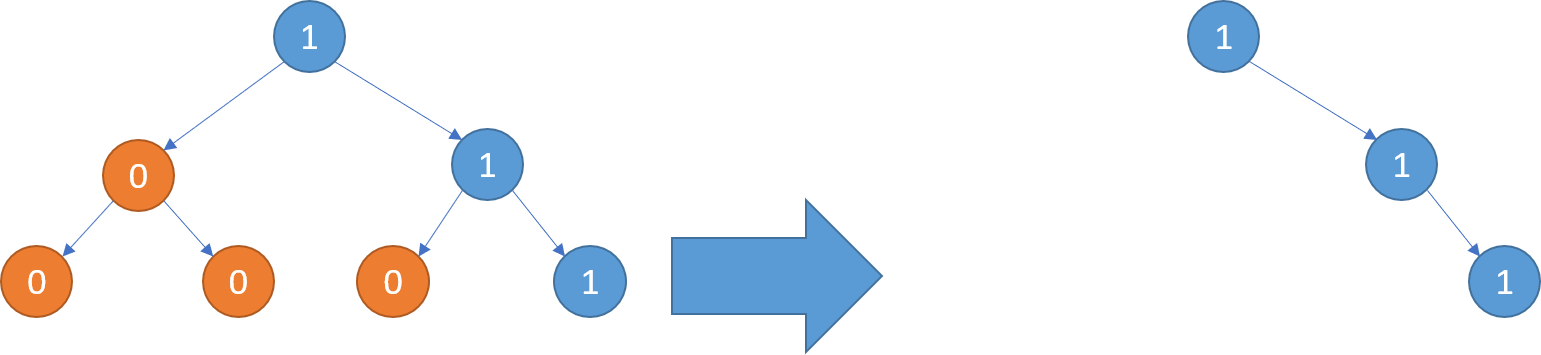


示例 2:

输入: [1,0,1,0,0,0,1]

输出: [1,null,1,null,1]

解释:

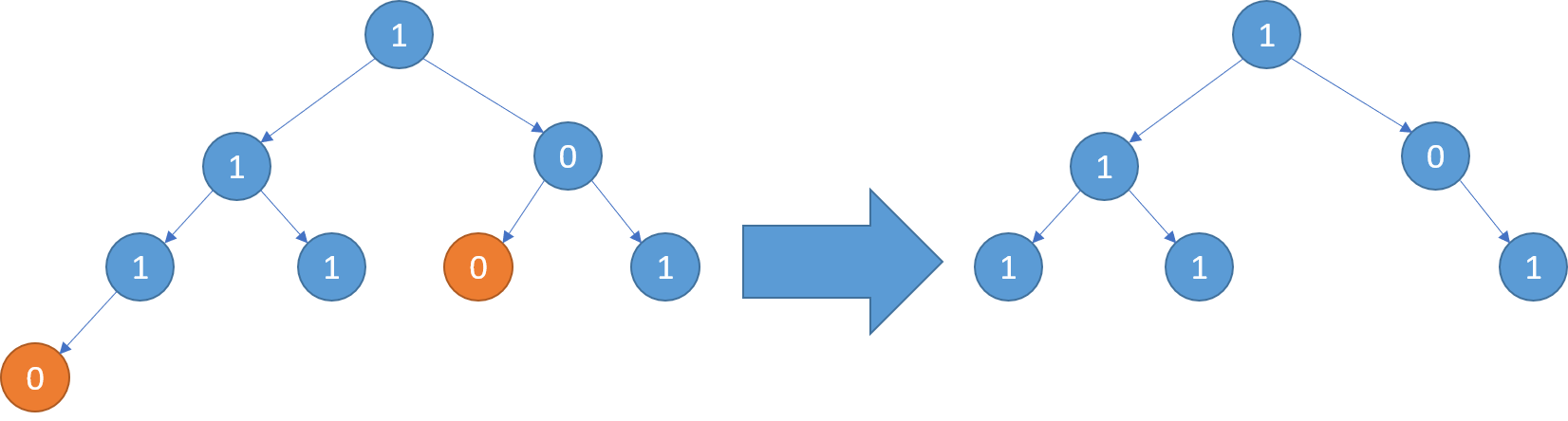


示例 3:

输入: [1,1,0,1,1,0,1,0]

输出: [1,1,0,1,1,null,1]

解释:



提示:

二叉树的节点个数的范围是[1,200]

二叉树节点的值只会是0或1

注意：本题与主站814题相同：

https://leetcode-cn.com/problems/binary-tree-pruning/

# 分析

**思路：**

若把全为0的二叉树称为零树，那么判断一棵树为零树的的规则是，左右子树都为零树或者空指针且根节点的值为0。因为二叉树结构的递归性质，所以可以用同样的规则判断左右子树是否为零树。在使用递归函数时，让根节点的左右指针指向左右子树递归函数的返回值，当该二叉树判断为零树（左右指针指向空指针且其根节点的值为0），则返回空指针，反之则返回根节点。

整个递归函数的过程都是先处理左右孩子节点再处理当前节点，这是后序遍历的一种变型。

**代码：**

class Solution {

public:

TreeNode\* pruneTree(TreeNode\* root) {

if (root == nullptr) {

return nullptr;

}

TreeNode\* left = pruneTree(root->left);

TreeNode\* right = pruneTree(root->right);

if (root->val == 0 && left == nullptr && right == nullptr) {

return nullptr;

}

root->left = left;

root->right = right;

return root;

}

};