给定二叉树根结点 `root` ，此外树的每个结点的值要么是 0，要么是 1。

返回移除了所有不包含 1 的子树的原二叉树。

(节点 X 的子树为 X 本身，以及所有 X 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：

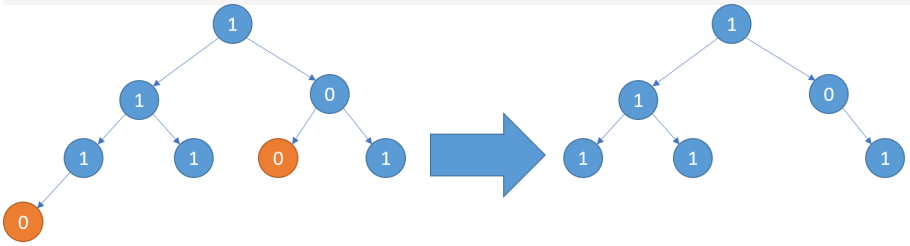
输入： `[1,0,1,0,0,0,1]`

输出： `[1,null,1,null,1]`

示例3：

输入： `[1,1,0,1,1,0,1,0]`

输出： `[1,1,0,1,1,null,1]`



说明:

- 给定的二叉树最多有 `100` 个节点。

- 每个节点的值只会为 `0` 或 `1` 。

分析，该题的判定条件为，该树左右子树中包含1吗？包含，不剪，不包含，剪去。那么进行深度遍历，动态的剪纸，并将该子树中包含1的情况返回为上一个节点即可。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
bool dfs(TreeNode* root){
    bool left=true;
    bool right=true;
    if(root->left!=NULL){
        left=dfs(root->left);
        if(left){
            root->left=NULL;
        }
    }
    if(root->right!=NULL){
        right=dfs(root->right);
        if(right){
            root->right=NULL;
        }
    }
    if(root->val==1){
        return false;
    }else{
        return left&&right;
    }
```

```
}
class Solution {
public:
    TreeNode* pruneTree(TreeNode* root) {
        if(root==NULL){
            return root;
        }
        dfs(root);
        return root;
    }
};
```