Problem 814: Binary Tree Pruning

Problem Information

Difficulty: Medium
Acceptance Rate: 72.40%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, return _the same tree where every subtree (of the given tree) not containing a_`1` _has been removed_.

A subtree of a node `node` is `node` plus every node that is a descendant of `node`.

Example 1:

![](https://s3-lc-upload.s3.amazonaws.com/uploads/2018/04/06/1028_2.png)

Input: root = [1,null,0,0,1] **Output:** [1,null,0,null,1] **Explanation:** Only the red nodes satisfy the property "every subtree not containing a 1". The diagram on the right represents the answer.

Example 2:

![](https://s3-lc-upload.s3.amazonaws.com/uploads/2018/04/06/1028_1.png)

Input: root = [1,0,1,0,0,0,1] **Output:** [1,null,1,null,1]

Example 3:

![](https://s3-lc-upload.s3.amazonaws.com/uploads/2018/04/05/1028.png)

Input: root = [1,1,0,1,1,0,1,0] **Output:** [1,1,0,1,1,null,1]

Constraints:

* The number of nodes in the tree is in the range `[1, 200]`. * `Node.val` is either `0` or `1`.

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
* };
*/
class Solution {
public:
TreeNode* pruneTree(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode pruneTree(TreeNode root) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def pruneTree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
```