Binary Tree Pruning

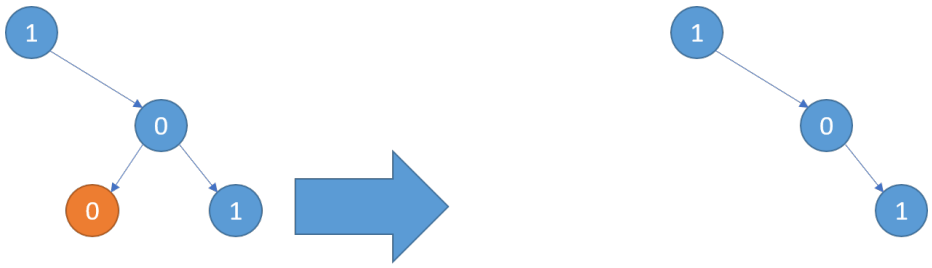
We are given the head node `root` of a binary tree, where additionally every node's value is either a 0 or a 1.

Return the same tree where every subtree (of the given tree) not containing a 1 has been removed.

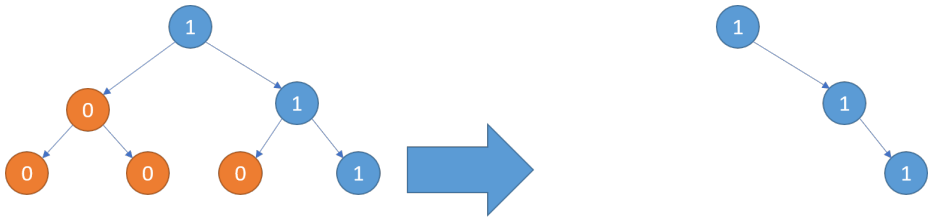
(Recall that the subtree of a node X is X, plus every node that is a descendant of X.)

Example 1:
Input: `[1,null,0,0,1]`
Output: `[1,null,0,null,1]`

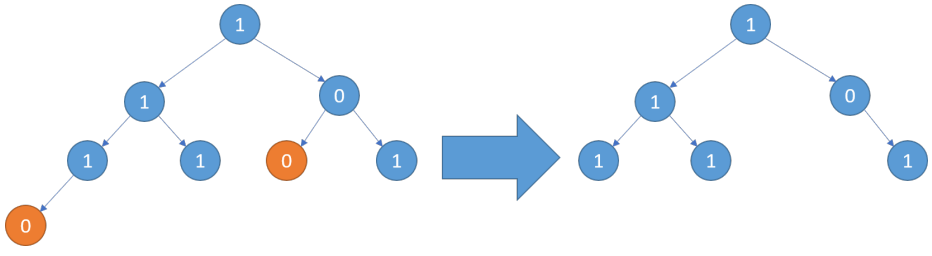
Explanation:
Only the red nodes satisfy the property "every subtree not containing a 1".
The diagram on the right represents the answer.



Example 2:
Input: `[1,0,1,0,0,0,1]`
Output: `[1,null,1,null,1]`



Example 3:
Input: `[1,1,0,1,1,0,1,0]`
Output: `[1,1,0,1,1,null,1]`



Note:

- The binary tree will have at most `100 nodes`.
- The value of each node will only be `0` or `1`.

Solution 1

```
class Solution:
    def pruneTree(self, root):
        """
        :type root: TreeNode
        :rtype: TreeNode
        """
        if not root: return root
        root.left, root.right = self.pruneTree(root.left), self.pruneTree(root.right
)
        return None if not root.val and not root.left and not root.right else root
```

written by Hemant_323 original link here

Solution 2

Recursive Solution, very self-explaining

```
if root == null: return null
root.left = pruneTree(root.left)
root.right = pruneTree(root.right)
if root.left == null and root.right == null and root.val == 0: return null
return root
```

C++

```
    TreeNode* pruneTree(TreeNode* root) {
        if (!root) return NULL;
        root->left = pruneTree(root->left);
        root->right = pruneTree(root->right);
        if (!root->left && !root->right && root->val == 0) return NULL;
        return root;
    }
```

Java:

```
    public TreeNode pruneTree(TreeNode root) {
        if (root == null) return null;
        root.left = pruneTree(root.left);
        root.right = pruneTree(root.right);
        if (root.left == null && root.right == null && root.val == 0) return null;
        return root
    }
```

Python

```
    def pruneTree(self, root):
        if not root: return None
        root.left = self.pruneTree(root.left)
        root.right = self.pruneTree(root.right)
        if not root.left and not root.right and not root.val: return None
        return root
```

If you like less lines:
2-lines C++

```
    TreeNode* pruneTree(TreeNode* root) {
        if (root) root->left = pruneTree(root->left), root->right = pruneTree(root-
>right);
        return (root && (root->left || root->right || root->val)) ? root : NULL;
    }
```

2-lines Python

```
    def pruneTree(self, root):
        if root: root.left, root.right = self.pruneTree(root.left), self.pruneTree(
root.right)
        if root and (root.left or root.right or root.val): return root
```

written by lee215 original link here

Solution 3

```
class Solution(object):
    def pruneTree(self, root):
        if not root: return None
        root.left = self.pruneTree(root.left)
        root.right = self.pruneTree(root.right)
        if root.val:
            return root
        else:
            return root if root.left or root.right else None
```

written by licaiuu original link here