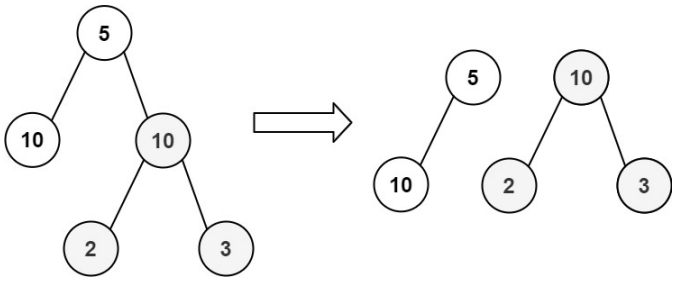
663. Equal Tree Partition

Medium 445 35 Add to List Share

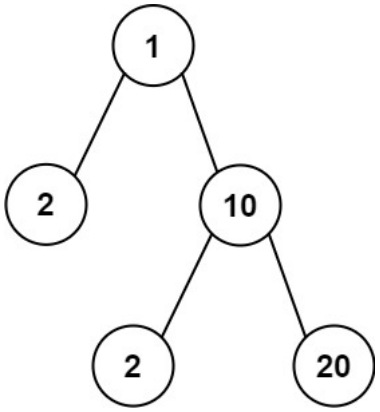
Given the `root` of a binary tree, return `true` *if you can partition the tree into two trees with equal sums of values after removing exactly one edge on the original tree.*

Example 1:




```
Input: root = [5,10,10,null,null,2,3]
Output: true
```

Example 2:




```
Input: root = [1,2,10,null,null,2,20]
Output: false
Explanation: You cannot split the tree into two trees with
equal sums after removing exactly one edge on the tree.
```

Constraints:

- The number of nodes in the tree is in the range `[1, 10^4]`.
- `-10^5 <= Node.val <= 10^5`

Accepted 28,867 Submissions 69,714

Seen this question in a real interview before? Yes No

Companies

Related Topics

Similar Questions

C++ Autocomplete

```cpp
/**
 * 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) {}
 * };
 */

//m-1
// tried solving it through maximum produt of splitted tree but this one can be done through
// m-2 also ...

class Solution {
public:
    int sumOfTree(TreeNode* root)
    {
        if(!root)
            return 0;

        if(!root->left and !root->right)
            return root->val;

        return (root->val + sumOfTree(root->left) + sumOfTree(root->right));
    }

    int equalSumTree(TreeNode* root,int &sum,bool &ans)
    {
        if(!root)
            return 0;

        if(!root->left and !root->right)
            return root->val;

        int left = equalSumTree(root->left,sum,ans);
        if(ans)
            return -1;
        int right = equalSumTree(root->right,sum,ans);
        if(ans)
            return -1;

        if(root->left)
```

NEW

Testcase Run Code Result Debugger

Accepted Runtime: 2 ms

Your input `[0,null,0]`

Output `true` Diff

Expected `true`