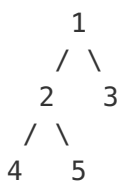
Find Leaves of Binary Tree

Given a binary tree, find all leaves and then remove those leaves. Then repeat the previous steps until the tree is empty.

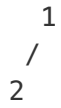
Example:
Given binary tree



Returns `[4, 5, 3], [2], [1]`.

Explanation:

1. Remove the leaves `[4, 5, 3]` from the tree



2. Remove the leaf `[2]` from the tree

3. Remove the leaf `[1]` from the tree

Returns `[4, 5, 3], [2], [1]`.

Credits:
Special thanks to @elmirap for adding this problem and creating all test cases.

Solution 1

```
class Solution {
private:
    int dfs(TreeNode* root, vector<vector<int>>& res){
        if(!root) return 0;
        int level = max(dfs(root->left, res), dfs(root->right, res)) + 1;
        if(level > (int)res.size()) res.push_back(vector<int>());
        res[level - 1].push_back(root->val);
        return level;
    }
public:
    vector<vector<int>> findLeaves(TreeNode* root) {
        vector<vector<int>> res;
        dfs(root, res);
        return res;
    }
};
```

written by sxycwzwzq original link here

Solution 2

```
public class Solution {
    private List<List<Integer>> res = new ArrayList<>();
    public List<List<Integer>> findLeaves(TreeNode root) {
        helper(root);
        return res;
    }
    private int helper(TreeNode node){
        if(null==node){
            return -1;
        }
        int level = 1 + Math.max(helper(node.left), helper(node.right));
        if(res.size()<level+1){
            res.add(new ArrayList<Integer>());
        }
        res.get(level).add(node.val);
        return level;
    }
}
```

written by sky-xu original link here

Solution 3

The essential of problem is not to find the leaves, but group leaves of same level together and also to cut the tree. This is the exact role backtracking plays. The helper function returns the level which is the distance from its furthest subtree leaf to root, which helps to identify which group the root belongs to

```
    public class Solution {
        public List<List<Integer>> findLeaves(TreeNode root) {
            List<List<Integer>> list = new ArrayList<>();
            findLeavesHelper(list, root);
            return list;
        }

    // return the level of root
        private int findLeavesHelper(List<List<Integer>> list, TreeNode root) {
            if (root == null) {
                return -1;
            }
            int leftLevel = findLeavesHelper(list, root.left);
            int rightLevel = findLeavesHelper(list, root.right);
            int level = Math.max(leftLevel, rightLevel) + 1;
            if (list.size() == level) {
                list.add(new ArrayList<>());
            }
            list.get(level).add(root.val);
            root.left = root.right = null;
            return level;
        }
    }
```

written by xuyirui original link here