Problem 366: Find Leaves of Binary Tree

Problem Information

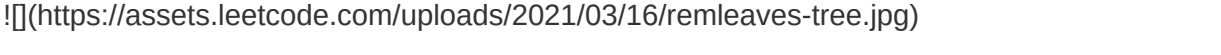
Difficulty: Medium
Acceptance Rate: 81.19%
Paid Only: Yes
Tags: Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, collect a tree's nodes as if you were doing this:

* Collect all the leaf nodes. * Remove all the leaf nodes. * Repeat until the tree is empty.

Example 1:



Input: root = [1,2,3,4,5] **Output:** [[4,5,3],[2],[1]] Explanation: [[3,5,4],[2],[1]] and [[3,4,5],[2],[1]] are also considered correct answers since per each level it does not matter the order on which elements are returned.

Example 2:

Input: root = [1] **Output:** [[1]]

Constraints:

* The number of nodes in the tree is in the range `[1, 100]`. * `-100 <= Node.val <= 100`

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:
vector<vector<int>> findLeaves(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 List<List<Integer>> findLeaves(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 findLeaves(self, root: Optional[TreeNode]) -> List[List[int]]:
```