Problem 545: Boundary of Binary Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 47.65%
Paid Only: Yes
Tags: Tree, Depth-First Search, Binary Tree

Problem Description

The **boundary** of a binary tree is the concatenation of the **root** , the **left boundary** , the **leaves** ordered from left-to-right, and the **reverse order** of the **right boundary**.

The **left boundary** is the set of nodes defined by the following:

* The root node's left child is in the left boundary. If the root does not have a left child, then the left boundary is **empty**. * If a node is in the left boundary and has a left child, then the left child is in the left boundary. * If a node is in the left boundary, has **no** left child, but has a right child, then the right child is in the left boundary. * The leftmost leaf is **not** in the left boundary.

The **right boundary** is similar to the **left boundary** , except it is the right side of the root's right subtree. Again, the leaf is **not** part of the **right boundary** , and the **right boundary** is empty if the root does not have a right child.

The **leaves** are nodes that do not have any children. For this problem, the root is **not** a leaf.

Given the `root` of a binary tree, return _the values of its**boundary**_.

Example 1:

`![](https://assets.leetcode.com/uploads/2020/11/11/boundary1.jpg)`

Input: root = [1,null,2,3,4] **Output:** [1,3,4,2] **Explanation:** - The left boundary is empty because the root does not have a left child. - The right boundary follows the path starting from the root's right child 2 -> 4. 4 is a leaf, so the right boundary is [2]. - The leaves from left to right are [3,4]. Concatenating everything results in [1] + [] + [3,4] + [2] = [1,3,4,2].

Example 2:

`![](https://assets.leetcode.com/uploads/2020/11/11/boundary2.jpg)`

Input: root = [1,2,3,4,5,6,null,null,null,7,8,9,10] **Output:** [1,2,4,7,8,9,10,6,3] **Explanation:** - The left boundary follows the path starting from the root's left child 2 -> 4. 4 is a leaf, so the left boundary is [2]. - The right boundary follows the path starting from the root's right child 3 -> 6 -> 10. 10 is a leaf, so the right boundary is [3,6], and in reverse order is [6,3]. - The leaves from left to right are [4,7,8,9,10]. Concatenating everything results in [1] + [2] + [4,7,8,9,10] + [6,3] = [1,2,4,7,8,9,10,6,3].

Constraints:

* The number of nodes in the tree is in the range `[1, 104]`. * `-1000 <= Node.val <= 1000`

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