Problem 114: Flatten Binary Tree to Linked List

Problem Information

Difficulty: Medium
Acceptance Rate: 69.68%
Paid Only: No
Tags: Linked List, Stack, Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, flatten the tree into a "linked list":

* The "linked list" should use the same `TreeNode` class where the `right` child pointer points to the next node in the list and the `left` child pointer is always `null`. * The "linked list" should be in the same order as a [**pre-order****traversal**](https://en.wikipedia.org/wiki/Tree_traversal#Pre-order,_NLR) of the binary tree.

Example 1:

![](https://assets.leetcode.com/uploads/2021/01/14/flaten.jpg)

Input: root = [1,2,5,3,4,null,6] **Output:** [1,null,2,null,3,null,4,null,5,null,6]

Example 2:

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

Example 3:

Input: root = [0] **Output:** [0]

Constraints:

* The number of nodes in the tree is in the range `[0, 2000]`. * `-100 <= Node.val <= 100`

Follow up: Can you flatten the tree in-place (with `O(1)` extra space)?

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:
void flatten(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 void flatten(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 flatten(self, root: Optional[TreeNode]) -> None:
"""
Do not return anything, modify root in-place instead.
"""
```