Documentation: Flatten Binary Tree to Linked List

Problem Description

Given the root of a binary tree, the task is to flatten the tree into a "linked list." The transformation must be performed in such a way that:

1. The "linked list" uses the same TreeNode class.
2. The right child pointer of each node points to the next node in the list.
3. The left child pointer of each node is always null.
4. The order of nodes in the "linked list" is the same as a pre-order traversal of the binary tree.

Example 1:

- **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

The task should be performed in-place, with an O(1) extra space complexity.

Solution Approach

The solution involves using a stack to facilitate the pre-order traversal of the binary tree. The key steps are:

1. **Initialization:** Start by checking if the root is null. If it is, return immediately since there's nothing to flatten.

2. **Stack Usage:** Use a stack to simulate the traversal. Initially, push the root node onto the stack.

3. **Traversal:**

- Pop a node from the stack and process it.
- If there's a previously processed node, update its right pointer to point to the current node and set its left pointer to null.
- Push the right child of the current node onto the stack (if it exists), followed by the left child (if it exists). This ensures that nodes are processed in pre-order (root, left, right).

4. **Update Pointers:** Continue this process until the stack is empty. The previously processed node's right pointer is updated to point to the current node, effectively flattening the tree into a linked list.

5. **In-Place Modification:** The algorithm modifies the tree in-place without using extra space beyond the implicit stack space used during recursion.

This approach ensures that the tree is flattened in a pre-order traversal manner, maintaining the constraints and requirements specified.