Flatten Binary Tree to Linked List

This problem asks us to flatten a binary tree to a linked list in preorder (root → left → right) order, modifying the tree in place (O(1) extra space).

Key Idea:

- We rearrange the tree such that:
  1. Each node's right child points to the next node in preorder.
  2. Each node's left child is null.
- Must avoid additional data structures (like arrays or stacks) for the follow-up.

Approach 1: Morris traversal style (O(1) space)

Steps:

1. Iterate through the tree starting from root.
2. For every node:
   - If the node has a left child:
     - Find the rightmost node in its left subtree (predecessor).
     - Connect this rightmost node's right to the node's right.
     - Move the left subtree to the right (node → right = node → left) and set node → left = NULL.
3. Move to the next node using node → right.

Complexity:

- Time: O(N) - each node is visited once, finding rightmost node per step still results in O(N) total.
- Space: O(1) extra - in-place manipulation.

Alternative:

- Recursive or stack-based preorder traversal (simpler to understand but O(N) space due to recursion/stack).