Documentation for Populating Next Right Pointers in Each Node

Problem Description

- *You are given a perfect binary tree where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:*

```
cpp

struct Node {

  int val;

  Node *left;

  Node *right;

  Node *next;

}
```

Objective

- Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL. Initially, all next pointers are set to NULL.

Example 1:

Input: root = [1,2,3,4,5,6,7]

Output: [1,#,2,3,#,4,5,6,7,#]

Explanation:

Given the above perfect binary tree (Figure A), your function should populate each next pointer to point to its next right node, just like in Figure B. The serialized output is in level order as connected by the next pointers, with '#' signifying the end of each level.

Example 2:

Input: root = []

Output: []

Constraints:

- The number of nodes in the tree is in the range [0, 2^12 1].
- -1000 <= Node.val <= 1000

Follow-up:

- You may only use constant extra space.
- The recursive approach is fine. You may assume implicit stack space does not count as extra space for this problem.

Explanation

1. **Initialization:**

- If the root is None, return None.
- Initialize leftmost to the root node. This variable tracks the leftmost node of the current level.

2. **Level Order Traversal Using next Pointers:**

- Traverse the tree level by level using a while loop that continues as long as leftmost.left exists.
- For each level, use another while loop to traverse the nodes at the current level using the next pointers.

3. **Connecting Child Nodes:**

- For each node (head) at the current level, connect head.left.next to head.right.
- If head.next exists, connect head.right.next to head.next.left. This connects the right child of the current node to the left child of the next node at the same level.

4. **Move to the Next Level:**

- After processing all nodes at the current level, move to the leftmost node of the next level by setting leftmost to leftmost.left.

This approach ensures that each node's next pointer is correctly set using constant extra space, complying with the problem's follow-up requirement.