116. Populating Next Right Pointers in Each Node

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:

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

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:

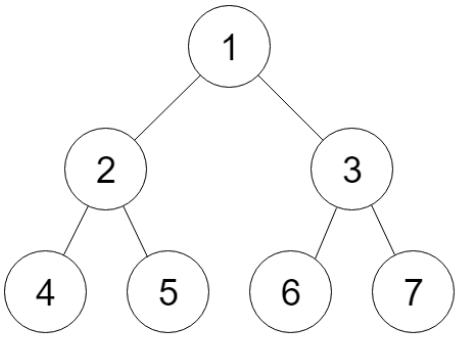

Figure A

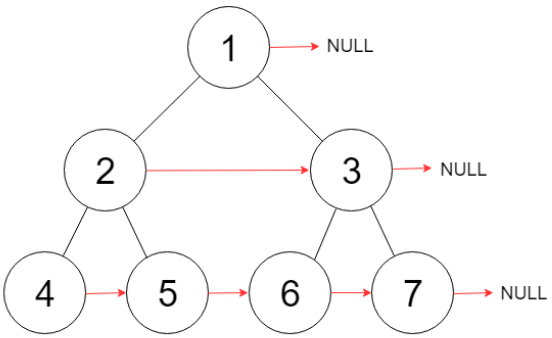

Figure B

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.

```
"""
# Definition for a Node.
class Node(object):
    def __init__(self, val=0, left=None, right=None,
next=None):
        self.val = val
        self.left = left
        self.right = right
        self.next = next
"""

class Solution(object):
    def connect(self, root):
        """
        :type root: Node
        :rtype: Node
        """
        if not root:
            return root
        if root.left:
            left, right = root.left, root.right
            self.connect(left)
            self.connect(right)
            while left:
                left.next = right
                left, right = left.right, right.left

        return root
```