Problem 117: Populating Next Right Pointers in Each Node II

Problem Information

Difficulty: Medium
Acceptance Rate: 56.61%
Paid Only: No
Tags: Linked List, Tree, Depth-First Search, Breadth-First Search, Binary Tree

Problem Description

Given a binary tree

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:

![](https://assets.leetcode.com/uploads/2019/02/15/117_sample.png)

Input: root = [1,2,3,4,5,null,7] **Output:** [1,#,2,3,#,4,5,7,#] **Explanation:** Given the above 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, 6000]`. * `-100 <= Node.val <= 100`

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.

Code Snippets

C++:

```
/*
// Definition for a Node.
class Node {
public:
int val;
Node* left;
Node* right;
Node* next;

Node() : val(0), left(NULL), right(NULL), next(NULL) {}

Node(int _val) : val(_val), left(NULL), right(NULL), next(NULL) {}

Node(int _val, Node* _left, Node* _right, Node* _next)
: val(_val), left(_left), right(_right), next(_next) {}
};
*/

class Solution {
public:
Node* connect(Node* root) {

}
};
```

Java:

```
/*
// Definition for a Node.
class Node {
```

```
public int val;
public Node left;
public Node right;
public Node next;

public Node() {}

public Node(int _val) {
val = _val;
}

public Node(int _val, Node _left, Node _right, Node _next) {
val = _val;
left = _left;
right = _right;
next = _next;
}
};
*/

class Solution {
public Node connect(Node root) {

}
}
```

Python3:

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

class Solution:
def connect(self, root: 'Node') -> 'Node':
```