117. Populating Next Right Pointers in Each Node II

Difficulty: Medium

Related Topics: Tree, Depth-first Search

Similar Questions: Populating Next Right Pointers in Each Node

Problem

Given a binary tree

```
struct TreeLinkNode {
  TreeLinkNode *left;
  TreeLinkNode *right;
  TreeLinkNode *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`.

Note:

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

Example:

Given the following binary tree,

```
     1
   /  \
  2    3
 / \    \
4   5    7
```

After calling your function, the tree should look like:

```
     1 -> NULL
   /  \
  2 -> 3 -> NULL
 / \    \
4-> 5 -> 7 -> NULL
```

Solution 1

```
/**
 * Definition for binary tree with next pointer.
 * function TreeLinkNode(val) {
 *     this.val = val;
 *     this.left = this.right = this.next = null;
 * }
 */

/**
 * @param {TreeLinkNode} root
 * @return {void} Do not return anything, modify tree in-place instead.
 */
var connect = function(root) {
  var stack = [];
  var tmp = null;
  var node = null;
  var next = null;
  var level = 0;

  if (root) stack.push([root, 0]);

  while (stack.length) {
    tmp = stack.shift();
    node = tmp[0];
    level = tmp[1];

    next = stack[0] && stack[0][1] === level ? stack[0][0] : null;

    node.next = next;

    if (node.left) stack.push([node.left, level + 1]);
    if (node.right) stack.push([node.right, level + 1]);
  }
};
```

Explain:

nope.

Complexity:

- Time complexity : O(n).
- Space complexity : O(n).

Solution 2

```
/**
 * Definition for binary tree with next pointer.
 * function TreeLinkNode(val) {
 *     this.val = val;
 *     this.left = this.right = this.next = null;
 * }
 */

/**
 * @param {TreeLinkNode} root
 * @return {void} Do not return anything, modify tree in-place instead.
 */
var connect = function(root) {
  var now = root;
  var cur = null;
  var tmp = null;
  var last = null;
  while (now) {
    tmp = new TreeLinkNode(0);
    last = tmp;
    cur = now;
    while (cur) {
      if (cur.left) { last.next = cur.left; last = last.next; }
      if (cur.right) { last.next = cur.right; last = last.next; }
      cur = cur.next;
    }
    now = tmp.next;
  }
};
```

Explain:

nope.

Complexity:

- Time complexity : O(n).
- Space complexity : O(1).