116. Populating Next Right Pointers in Each Node

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.
- You may assume that it is a perfect binary tree (ie, all leaves are at the same level, and every parent has two children).

Example:

Given the following perfect binary tree,

```
     1
   /  \
  2    3
 / \  / \
4  5  6  7
```

After calling your function, the tree should look like:

```
     1 -> NULL
   /  \
  2 -> 3 -> NULL
 / \  / \
4->5->6->7 -> NULL
```

```
public class L116 {
    public class TreeLinkNode {
        int val;
        TreeLinkNode left, right, next;
        TreeLinkNode(int x) {
            val = x;
        }
    }
//  这道题目就是层次遍历的题目
    public void connect(TreeLinkNode root) {
        if(root == null)
            return ;
        LinkedList<TreeLinkNode> queue = new LinkedList<>();
        queue.add(root);
        queue.add(null);//一层的最后需要添加一个空节点
        while (!queue.isEmpty()) {
            TreeLinkNode node = queue.pop();
            if(node != null) {//分为如果节点不为空，则判断是否为叶子节点，如果是则不添加子节点，如果不是则添加
                TreeLinkNode next = queue.peekFirst();
                    node.next = next;
    //这里一定要判断是否为叶子节点，要不然queue会一直添加其左右子节点。则永不为空
                    if(node.left != null && node.right != null) {
                        queue.add(node.left);
                        queue.add(node.right);
                    }
            }else {//如果节点为空，要判断是否为最后一层，是则返回，不是则加入空节点（下一层的最后一个节点）
                if(!queue.isEmpty()) {
                    queue.add(null);
                    continue;
                }else {
                    break;
                }
            }
        }
    }
}
```