给定一个二叉树

```
struct TreeLinkNode {
  TreeLinkNode *left;
  TreeLinkNode *right;
  TreeLinkNode *next;
}
```

填充它的每个 next 指针，让这个指针指向其下一个右侧节点。如果找不到下一个右侧节点，则将 next 指针设置为 `NULL`。

初始状态下，所有 next 指针都被设置为 `NULL`。

说明:

- 你只能使用额外常数空间。
- 使用递归解题也符合要求，本题中递归程序占用的栈空间不算做额外的空间复杂度。
- 你可以假设它是一个完美二叉树（即所有叶子节点都在同一层，每个父节点都有两个子节点）。

示例:

给定完美二叉树，

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

调用你的函数后，该完美二叉树变为：

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

思路，进行深度遍历，然后进行前树遍历，把最早的左边的节点统计起来，然后动态指向其同层的右边节点，然后再换成右边几点，继续统计

代码

```
/**
 * Definition for binary tree with next pointer.
```

```
* struct TreeLinkNode {
*  int val;
*  TreeLinkNode *left, *right, *next;
*  TreeLinkNode(int x) : val(x), left(NULL), right(NULL), next(NULL) {}
* };
*/
class Solution {
public:
    void dfs(TreeLinkNode *root,map <int,TreeLinkNode *> &left,int floor){
        if(root==NULL){
            return;
        }
        if(left[floor]==NULL){
            left[floor]=root;
        }else{
            left[floor]->next=root;
            left[floor]=root;
        }
```

```
        if(root->left!=NULL){
            dfs(root->left,left,floor+1);
        }
        if(root->right!=NULL){
            dfs(root->right,left,floor+1);
        }
    }
    void connect(TreeLinkNode *root) {
        map <int,TreeLinkNode *> left;
        dfs(root,left,0);
    }
};
```