剑指36 二叉搜索树与双向链表

一开始思路没想对，不应该！

记住：二叉搜索树中序遍历时，节点访问顺序肯定是按值从小到大的。
题目的本质就是按大小顺序访问二叉搜索树的节点，访问的时候有记录下前一节点，就能同时进行构建双向链表。
写代码时候，不用脑内层层模拟递归的调用，只需要创建全局变量**pre**，每次访问的时候更新(不用思考到底到哪一层递归了)，那么pre就一定是按值大小顺序滚动前进的。

```
// 我的最终结果。正确答案
class Solution {
public:
    Node* pre;
    void DFS(Node* root) {
        if(!root) return;
        DFS(root->left);
        if(pre == NULL)
            pre = root;
        else {
            root->left = pre;
            pre->right = root;
            pre = root;
        }
        DFS(root->right);
    }

    Node* treeToDoublyList(Node* root) {
        if(!root) return NULL;
        pre = NULL;
        DFS(root);
        Node* head = pre;
        while(head->left) {
            head = head->left;
        }
        head->left = pre;
        pre->right = head;
        return head;
    }
};
```

```
// 一开始的结果，AC了，但是思路其实不太对
class Solution {
public:

    Node* DFS(Node* root, bool isleft) {
        if(!root) return NULL;

        Node* up = root;
        if(isleft) {
            while(up->right) up = up->right;
```

```
        }
        else {
            while(up->left) up = up->left;
        }

        Node* last = DFS(root->left, true);
        root->left = last;
        if(last) {
            last->right = root;
        }

        Node* next = DFS(root->right, false);
        root->right = next;
        if(next) {
            next->left = root;
        }

        return up;
    }

    Node* treeToDoublyList(Node* root) {
        if(!root) return NULL;
        Node* head = DFS(root, false);
        Node* tail = head;
        while(tail->right) tail = tail->right;
        tail->right = head;
        head->left = tail;
        return head;
    }
};
```