一、题目说明

题目114. Flatten Binary Tree to Linked List，将一个二叉树“原地”压缩为“链表”形态的二叉树。难度为Medium！

二、我的解答

这个题目如果允许使用栈的话Easy，先序遍历二叉树，右子树入栈，左子树入栈。当栈不空的时候，将栈顶元素放到右子树即可。

```
class Solution{
    public:
        void flatten(TreeNode* root){

            //先根遍历
            if(root==NULL) return;
            if(root->left==NULL && root->right==NULL) return;
            TreeNode *p,* cur = root;

            stack<TreeNode*> st;
            if(root->right !=NULL){
                st.push(root->right);
            }
            if(root->left !=NULL){
                st.push(root->left);
            }

            while(! st.empty()){
                p = st.top();
                st.pop();
                cur->left = NULL;
                cur->right = p;
                cur = cur->right;

                if(cur->right !=NULL){
                    st.push(cur->right);
                }
                if(cur->left !=NULL){
                    st.push(cur->left);
                }
            }

            return;
        }
};
```

性能：

```
Runtime: 12 ms, faster than 27.18% of C++ online submissions for Flatten Binary
Tree to Linked List.
Memory Usage: 11.6 MB, less than 8.33% of C++ online submissions for Flatten
Binary Tree to Linked List.
```

三、优化措施

此处的“原地”，理解起来不能使用栈的。在这种情况下，将**右子树**作为**左子树的最右下节点的右子树**，**左子树变为右子树**即可。

```
class Solution{
    public:
        void flatten(TreeNode* root){
            if(root==NULL) return;
            if(root->left !=NULL){
                TreeNode* pre = root->left;
                while(pre->right !=NULL){
                    pre = pre->right;
                }
                pre->right = root->right;
                root->right = root->left;
                root->left = NULL;
            }
            flatten(root->right);
            return;
        }
};
```

性能如下：

```
Runtime: 4 ms, faster than 95.35% of C++ online submissions for Flatten Binary
Tree to Linked List.
Memory Usage: 11.3 MB, less than 8.33% of C++ online submissions for Flatten
Binary Tree to Linked List.
```

这个性能还一般，用morris方法（线索化二叉树），空间复杂度可以到O(1)。