# 题目

给定一个二叉树，其中所有的右节点要么是具有兄弟节点（拥有相同父节点的左节点）的叶节点，要么为空，将此二叉树上下翻转并将它变成一棵树， 原来的右节点将转换成左叶节点。返回新的根。

例子:

输入: [1,2,3,4,5]

1

/ \

2 3

/ \

4 5

输出: 返回二叉树的根 [4,5,2,#,#,3,1]

4

/ \

5 2

/ \

3 1

说明:

对 [4,5,2,#,#,3,1] 感到困惑? 下面详细介绍请查看 二叉树是如何被序列化的。

二叉树的序列化遵循层次遍历规则，当没有节点存在时，'#' 表示路径终止符。

这里有一个例子:

1

/ \

2 3

/

4

\

5

上面的二叉树则被序列化为 [1,2,3,#,#,4,#,#,5].

# 分析

## 方法一：递归

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode(int x) : val(x), left(NULL), right(NULL) {}

\* };

\*/

class Solution {

public:

TreeNode\* upsideDownBinaryTree(TreeNode\* root) {

if (root == NULL || root->left == NULL) return root;

auto l = root->left;

auto r = root->right;

root->left = NULL;

root->right = NULL;

auto res = upsideDownBinaryTree(l);

l->left = r;

l->right = root;

return res;

}

};

解题思路

解法1：递归

后序遍历的递归。保存当前root的左子树和右子树。递归翻转左子树。然后赋值原root的右子树到原来root左子树的左子树，令root为原左子树的右子树，置空root的左右孩子，返回递归翻转的左子树的新根flippedLeft。

解法2：迭代

类似于链表的翻转，只不过要两个临时变量tmp（保存上一次访问的右子树），pre（保存上一次访问的左子树）。最后返回pre代表的最左的左子树根。

代码

class Solution1 {

public:

TreeNode\* upsideDownBinaryTree(TreeNode\* root) {

if (!root || !root->left) {

return root;

}

TreeNode\* left = root->left;

TreeNode\* right = root->right;

TreeNode\* flippedLeft = upsideDownBinaryTree(left);

left->left = right; // turn original right into left

left->right = root; // turn original root into right

root->left = nullptr;

root->right = nullptr;

return flippedLeft;

}

};

class Solution {

public:

TreeNode\* upsideDownBinaryTree(TreeNode\* root) {

TreeNode\* cur = root;

TreeNode\* pre = nullptr;

TreeNode\* tmp = nullptr;

while (cur) {

auto next = cur->left;

cur->left = tmp; // copy original right to left

tmp = cur->right; // save next level original right

cur->right = pre; // copy original root to right

pre = cur; // save cur root, i.e: next level left

cur = next;

}

return pre;

}

};