# 题目

请完成一个函数，输入一个二叉树，该函数输出它的镜像。

**例如输入：**

     4

   /   \

  2     7

 / \   / \

1   3 6   9

**镜像输出：**

     4

   /   \

  7     2

 / \   / \

9   6 3   1

**示例 1：**

输入：root = [4,2,7,1,3,6,9]

输出：[4,7,2,9,6,3,1]

**限制：**

0 <= 节点个数 <= 1000

**注意：**本题与主站 226 题相同：

https://leetcode-cn.com/problems/invert-binary-tree/

# 分析

## 方法一：递归法

**思路：**

**代码：**

/\*\*

\* 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\* mirrorTree(TreeNode\* root) {

if(nullptr == root)

return NULL;

TreeNode \*left = mirrorTree(root->left);

TreeNode \*right = mirrorTree(root->right);

root->left = right;

root->right = left;

return root;

}

};

## 方法二：迭代法

### 思路一：栈模拟

**思路：**

模型：栈模拟二叉树的先序遍历

循环结束条件：栈为空

实现操作：交换栈顶结点的左右子树

**代码**

class Solution {

public:

TreeNode\* mirrorTree(TreeNode\* root) {

stack<TreeNode\*> s;

s.push(root);

while (!s.empty()) {

TreeNode\* node = s.top();

s.pop();

if (node == NULL) {

continue;

}

swap(node->left, node->right);

s.push(node->left);

s.push(node->right);

}

return root;

}

};

### 思路二：队列模拟

**思路：**

模型：使用队列模拟二叉树的层次遍历

循环结束条件：队列为空

实现操作：交换队首结点的左右子树

**代码：**

class Solution {

public:

TreeNode\* mirrorTree(TreeNode\* root) {

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

TreeNode\* node = q.front();

q.pop();

if (node == NULL) {

continue;

}

swap(node->left, node->right);

q.push(node->left);

q.push(node->right);

}

return root;

}

};