面试题27：二叉树的镜像

题目描述

操作给定的二叉树，将其变换为源二叉树的镜像。

输入描述

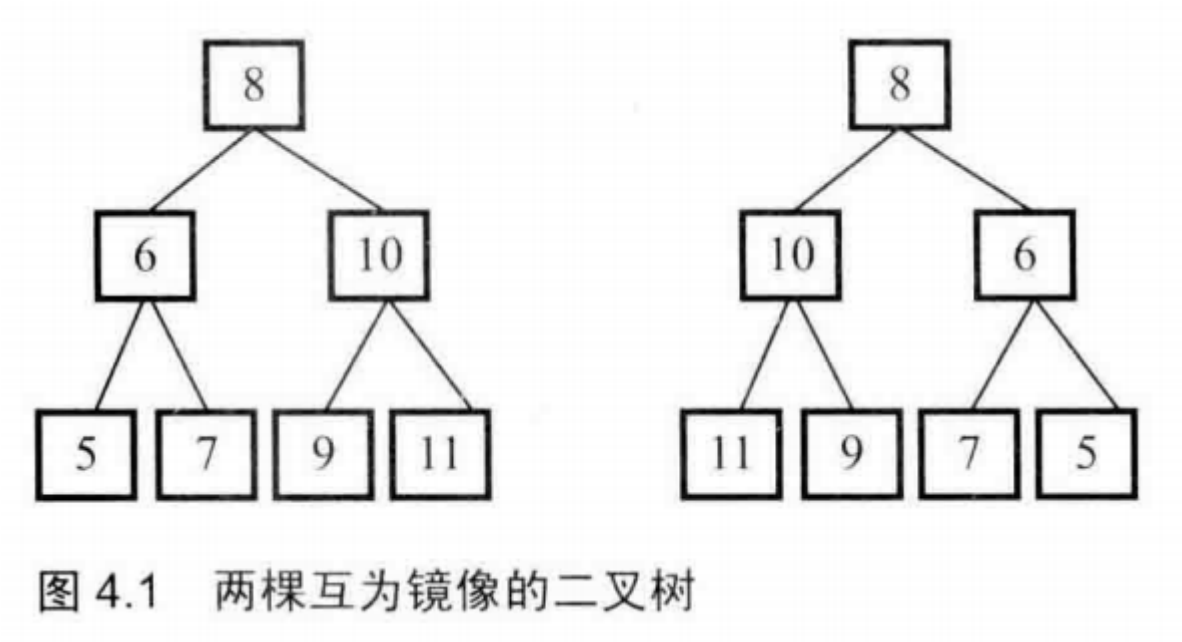

图 4.1　两棵互为镜像的二叉树

解题思路

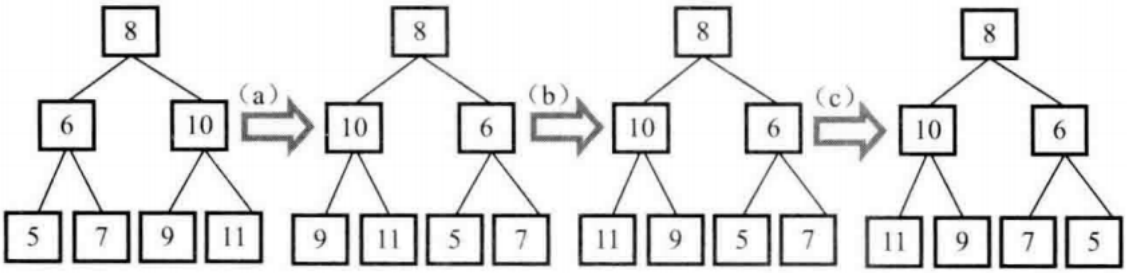

图 4.2　求二叉树镜像的过程

注：（a）交换根节点的左、右子树；（b）交换值为 10 的节点的左、右子节点；（c）交换值为 6 的节点的左、右子节点。

总结上面的过程，我们得出求一棵树的镜像的过程：先前序遍历这棵树的每个节点，如果遍历到的节点有子节点，就交换它的两个子节点。当交换完所有非叶节点的左、右子节点之后，就得到了树的镜像。

C++

```
/*
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
    TreeNode(int x) :
            val(x), left(NULL), right(NULL) {
    }
};*/
class Solution {
public:
    void Mirror(TreeNode *pRoot) {
        //判断根节点是否为空
        if(pRoot == nullptr)
```

```
            return;
        //判断根节点左右子树是否为空
        if(pRoot -> left == nullptr && pRoot -> right == nullptr)
            return;
        //左右子树镜像（Temp是临时）
        TreeNode *pTemp = pRoot -> left;
        pRoot -> left = pRoot -> right;
        pRoot -> right = pTemp;
        //判断左子树
        if(pRoot -> left)
            Mirror(pRoot -> left);
        //判断右子树
        if(pRoot -> right)
            Mirror(pRoot -> right);
    }
};
```

Java

```
/**
public class TreeNode {
    int val = 0;
    TreeNode left = null;
    TreeNode right = null;

    public TreeNode(int val) {

```

```
        this.val = val;

    }

}
*/
public class Solution {
    public void Mirror(TreeNode root) {
        //判断根节点是否为空
        if(root == null)
            return;
        //判断根节点左右子树是否为空
        if(root.left == null && root.right ==
null)
            return;
        //左右子树镜像（Temp是临时）
        TreeNode temp = root.left;
        root.left = root.right;
        root.right = temp;
        //判断左子树
        if(root.left != null)
            Mirror(root.left);
        //判断右子树
        if(root.right != null)
            Mirror(root.right);
    }
}
```