1. 题目
2. 题干

翻转一颗二叉树。

1. 示例

示例：

输入：

4

/ \

2 7

/ \ / \

1 3 6 9

输出：

4

/ \

7 2

/ \ / \

9 6 3 1

1. 题解
2. 思路

关于二叉树的问题，第一种思路当然是递归，即深度优先搜索。递归的终止条件是当前的节点为null则返回null，非空的话当前节点的左节点就是之前右节点的递归结果，当前节点的右节点就是之前左节点的递归结果。

第二种思路是广度优先搜索，将每一层中的数都取出来，再一个一个地迭代，交换左右节点。由于广度优先搜索是按照顺序处理的，是先进先出的顺序，这就需要队列了，首先将根节点放入队列中，然后将队列中的每个节点的子节点都放入队列中，一个一个地交换左右子树。队列会按照每一层节点的顺序一个个地处理，这种方法用的是迭代的思想。

1. 代码实现

Java：

1. 递归

/\*\*

 \* Definition for a binary tree node.

 \* public class TreeNode {

 \*     int val;

 \*     TreeNode left;

 \*     TreeNode right;

 \*     TreeNode(int x) { val = x; }

 \* }

 \*/

class Solution {

    public TreeNode invertTree(TreeNode root) {

        return dfs(root);

    }

    public TreeNode dfs(TreeNode root){

        if(root == null){

            return null;

        }

        TreeNode temp = root.left;

        root.left = dfs(root.right);

        root.right = dfs(temp);

        return root;

    }

}

1. 迭代

public TreeNode inverTree(TreeNode root){  
 if(root == null){  
 return null;  
 }  
 LinkedList<TreeNode> queue = new LinkedList<TreeNode>();  
 queue.add(root);  
 while(!queue.isEmpty()){  
 TreeNode tmp = queue.poll();  
 TreeNode left = tmp.left;  
 tmp.left = tmp.right;  
 tmp.right = left;  
 if(tmp.left != null){  
 queue.add(tmp.left);  
 }  
 if(tmp.right != null){  
 queue.add(tmp.right);  
 }  
 }  
 return root;  
}