1. 题目
2. 题干

给定一个二叉树，检查它是否是镜像对称的。

1. 示例

二叉树 [1,2,2,3,4,4,3] 是对称的。

1

/ \

2 2

/ \ / \

3 4 4 3

1. 题解
2. 思路
3. 递归思路

二叉树的题目一般都可以用递归和迭代两种思路来做。这个题目要求判断一个树的左子树是否和右子树镜像对称，那么镜像对称要如何判断呢？应该满足以下的条件：

1. 两棵树的根节点值相等
2. 每棵树的左子树都和另一棵树的右子树镜像对称

这样，就形成了递归的条件，之前我们做的递归题目都是通过一个节点指针进行递归，这种需要比较的题目就可以尝试使用两个节点指针进行递归，比较两个指针所指的节点是否镜像对称。

2.迭代思路

通常，当我们要将递归思路转换为迭代时，通过一个队列或者栈来过渡。初始时，将根节点入队两次，在每次迭代中我们取出队列的两个节点，比较两个节点是否相等。然后分别插入第一个节点的左子树，第二个节点的右子树；然后插入第一个节点的右子树，第二个节点的左子树。反正就是比较不同节点的左右子树，只要有不相等的，就返回false。

1. 代码

Java：

1. 递归思路：

/\*\*

 \* Definition for a binary tree node.

 \* public class TreeNode {

 \*     int val;

 \*     TreeNode left;

 \*     TreeNode right;

 \*     TreeNode(int x) { val = x; }

 \* }

 \*/

class Solution {

    public boolean isSymmetric(TreeNode root) {

        //镜像对称，左子树和右子树镜像对称，使用两个指针分别指向左子树，右子树

        return dfs(root, root);

    }

    public boolean dfs(TreeNode left, TreeNode right){

        //都为null

        if(left == null && right == null){

            return true;

        }

        //单个为null，说明不相等

        if(left == null || right == null){

            return false;

        }

        //都不为null，需要比较两个节点的值是否相等，以及子树的对称情况

        return (left.val == right.val) && dfs(left.left, right.right) && dfs(left.right, right.left);

    }

}

2.迭代思路

Java：

class Solution {

public boolean isSymmetric(TreeNode root) {

return check(root, root);

}

public boolean check(TreeNode u, TreeNode v) {

Queue<TreeNode> q = new LinkedList<TreeNode>();

q.offer(u);

q.offer(v);

while (!q.isEmpty()) {

u = q.poll();

v = q.poll();

if (u == null && v == null) {

continue;

}

if ((u == null || v == null) || (u.val != v.val)) {

return false;

}

q.offer(u.left);

q.offer(v.right);

q.offer(u.right);

q.offer(v.left);

}

return true;

}

}