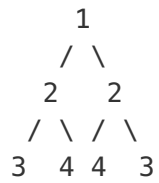
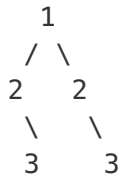
Symmetric Tree

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree is symmetric:



But the following is not:

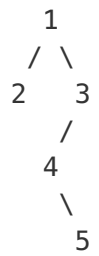


Note:
Bonus points if you could solve it both recursively and iteratively.

confused what `"{1,#,2,3}"` means? > read more on how binary tree is serialized on OJ.

OJ's Binary Tree Serialization:
The serialization of a binary tree follows a level order traversal, where '#' signifies a path terminator where no node exists below.

Here's an example:



The above binary tree is serialized as `"{1,2,3,#,#,4,#,#,5}"`.

Solution 1

Recursive--400ms:

```
public boolean isSymmetric(TreeNode root) {
    return root==null || isSymmetricHelp(root.left, root.right);
}

private boolean isSymmetricHelp(TreeNode left, TreeNode right){
    if(left==null || right==null)
        return left==right;
    if(left.val!=right.val)
        return false;
    return isSymmetricHelp(left.left, right.right) && isSymmetricHelp(left.right,
right.left);
}
```

Non-recursive(use Stack)--460ms:

```java
public boolean isSymmetric(TreeNode root) {
    if(root==null)  return true;

    Stack<TreeNode> stack = new Stack<TreeNode>();
    TreeNode left, right;
    if(root.left!=null){
        if(root.right==null) return false;
        stack.push(root.left);
        stack.push(root.right);
    }
    else if(root.right!=null){
        return false;
    }

    while(!stack.empty()){
        if(stack.size()%2!=0)   return false;
        right = stack.pop();
        left = stack.pop();
        if(right.val!=left.val) return false;

        if(left.left!=null){
            if(right.right==null)   return false;
            stack.push(left.left);
            stack.push(right.right);
        }
        else if(right.right!=null){
            return false;
        }

        if(left.right!=null){
            if(right.left==null)   return false;
            stack.push(left.right);
            stack.push(right.left);
        }
        else if(right.left!=null){
            return false;
        }
    }

    return true;
}
```

written by lvlolitte original link here

Solution 2

```
/**
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    bool isSymmetric(TreeNode *root) {
        TreeNode *left, *right;
        if (!root)
            return true;

        queue<TreeNode*> q1, q2;
        q1.push(root->left);
        q2.push(root->right);
        while (!q1.empty() && !q2.empty()){
            left = q1.front();
            q1.pop();
            right = q2.front();
            q2.pop();
            if (NULL == left && NULL == right)
                continue;
            if (NULL == left || NULL == right)
                return false;
            if (left->val != right->val)
                return false;
            q1.push(left->left);
            q1.push(left->right);
            q2.push(right->right);
            q2.push(right->left);
        }
        return true;
    }
};
```

written by JayfonLin original link here

Solution 3

```
 public boolean isSymmetric(TreeNode root) {
    if(root==null) return true;
    return isMirror(root.left,root.right);
}
public boolean isMirror(TreeNode p, TreeNode q) {
    if(p==null && q==null) return true;
    if(p==null || q==null) return false;
    return (p.val==q.val) && isMirror(p.left,q.right) && isMirror(p.right,q.left);
}
```

written by yangneu2015 original link here