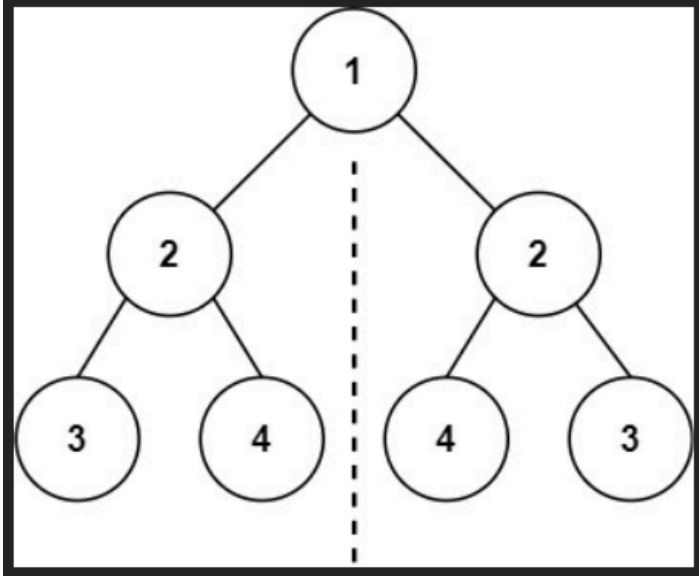
101. Symmetric Tree

Solved

Easy | Topics | Companies

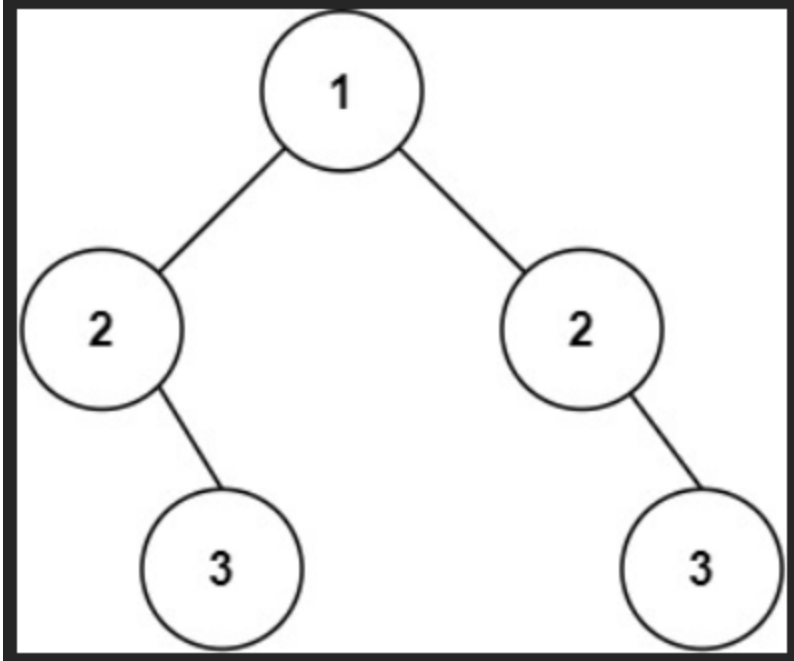
Given the `root` of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

Example 1:




```
Input: root = [1,2,2,3,4,4,3]
Output: true
```

Example 2:




```
Input: root = [1,2,2,null,3,null,3]
Output: false
```

Constraints:

- The number of nodes in the tree is in the range `[1, 1000]`.
- `-100 <= Node.val <= 100`

Follow up: Could you solve it both recursively and iteratively?

Python:

```
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

```
class Solution:
    # Recursive solution
    def isSymmetric(self, root: TreeNode) -> bool:
        if not root:
            return True

        def isMirror(t1: TreeNode, t2: TreeNode) -> bool:
            if not t1 and not t2:
                return True
            if not t1 or not t2:
                return False
            return (t1.val == t2.val and
                    isMirror(t1.left, t2.right) and
                    isMirror(t1.right, t2.left))

        return isMirror(root.left, root.right)

    # Iterative solution (using queue)
    def isSymmetricIterative(self, root: TreeNode) -> bool:
        if not root:
            return True

        queue = [(root.left, root.right)]

        while queue:
            t1, t2 = queue.pop(0)

            if not t1 and not t2:
                continue
            if not t1 or not t2:
                return False
            if t1.val != t2.val:
                return False

            # Push children in mirrored order
            queue.append((t1.left, t2.right))
            queue.append((t1.right, t2.left))

        return True
```

JavaScript:

```
var isSymmetric = function(root) {
  if (!root) return true;
```

```
  let queue = [];
  queue.push(root.left, root.right);

  while (queue.length > 0) {
    let t1 = queue.shift();
    let t2 = queue.shift();

    if (!t1 && !t2) continue;
    if (!t1 || !t2) return false;
    if (t1.val !== t2.val) return false;

    // Enqueue children in mirrored order
    queue.push(t1.left, t2.right);
    queue.push(t1.right, t2.left);
  }

  return true;
};
```

Java:

```
/**
 * Definition for a binary tree node. */
public class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode() {}
    TreeNode(int val) { this.val = val; }
    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val = val;
        this.left = left;
        this.right = right;
    }
}

class Solution {
     // Main function
    public boolean isSymmetric(TreeNode root) {
        if (root == null) return true;
        return isMirror(root.left, root.right);
    }

    // Recursive helper function
    private boolean isMirror(TreeNode t1, TreeNode t2) {
```

```
        if (t1 == null && t2 == null) return true;
        if (t1 == null || t2 == null) return false;
        return (t1.val == t2.val)
            && isMirror(t1.left, t2.right)
            && isMirror(t1.right, t2.left);
    }
}
```