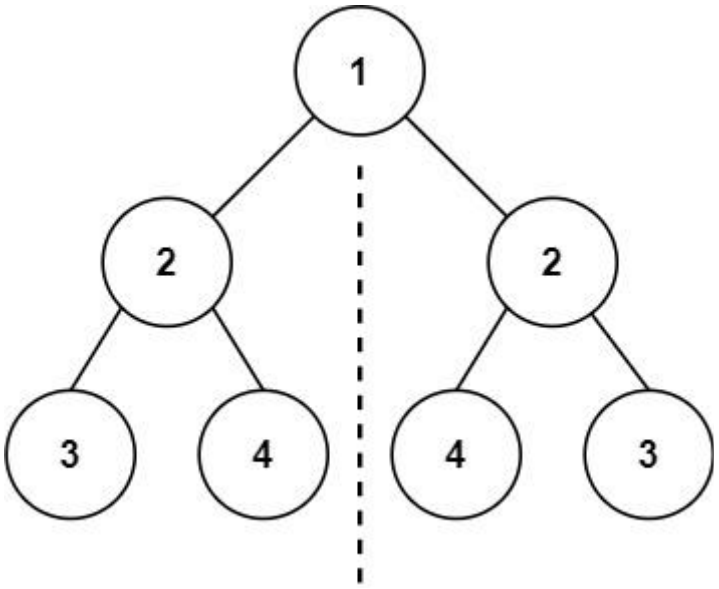
Symmetric Tree

Question: https://leetcode.com/problems/symmetric-tree/

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
```

My approach:
It is clearly visible that we are performing a **BFS** operation and comparing left with right.
Specifically **left.left** with **right.right.**
So I created a manipulated BFS solution to solve the problem.

BFS Explanation: https://www.geeksforgeeks.org/level-order-tree-traversal/

My Code: https://gist.github.com/vermaayush680/12dad7ad55dffec753d724575f167479

```
if not root:
    return True
```

```
queue=[root.left,root.right]

while(len(queue)>0):
    left=queue.pop(0)
    right=queue.pop(0)

    if not left and not right:
        continue
    elif left and right and left.val==right.val:
        pass
    else:
        return False

    queue.append(left.left)
    queue.append(right.right)
    queue.append(left.right)
    queue.append(right.left)
return True
```

Another approach I could think of was to traverse any 1 side, either left or right and save its mirror image(left->right and right->left)
Then check if this mirror image is equivalent to the other side or not.
Will try this approach later.