```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.left)
        queue = tmp
        res += 1
    return res
```

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	你的代码是正确的，答案是正确的。在你的代码中，你首先将根节点添加到队列中，然后进入一个循环，直到队列为空。在每次循环中，你都会将队列中的所有节点的左子节点和右子节点添加到队列中。然后，你增加深度计数器res。最后，你返回深度计数器res。 Your code is correct, the answer is correct. In your code, you first add the root node to the queue, then enter a loop until the queue is empty. In each loop, you add the left child node and right child node of all nodes in the queue to the queue. Then, you increase the depth counter res. Finally, you return the depth counter res.	876
Qwen-14B-Chat (SFT)	你的代码会返回二叉树的节点数量，而不是最大深度。这是因为你在每次循环中都增加了res的值，而不是只在找到新的最深节点时增加。你可以通过在找到新的最深节点时才增加res的值来修复这个问题。以下是修复后的代码：	949

Qwen-14B-Chat (SFT)

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        if len(queue) > res:
            res = len(queue)
    return res
```

Your code will return the number of nodes in the binary tree, not the maximum depth. This is because you increase the value of ”res” in every loop, instead of only increasing it when a new deepest node is found. You can fix this problem by increasing the value of ”res” only when a new deepest node is found. Here is the modified code:

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        if len(queue) > res:
            res = len(queue)
    return res
```