Symmetric / Mirror Binary Tree LeetCode

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

Example:

```
         5
       /   \
      2     2
    /  \   /  \
   4    3 3    4
```

Output: The Tree is Symmetric/Mirror Binary Tree

Approach 1: Function to check if a binary tree is symmetric using recursive approach

- The recursive approach checks symmetry by comparing the left subtree of the root with the right subtree.
- It starts by comparing the values of the left and right nodes of the root.
- Then, it recursively checks if the left subtree's left child is symmetric with the right subtree's right child and if the left subtree's right child is symmetric with the right subtree's left child.
- If all these conditions are met for each corresponding pair of nodes, the tree is symmetric.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is checked once.**

Space Complexity:

- **The space complexity is O(H), where H is the height of the tree, due to the recursive call stack.**

Approach 2: Function to check if a binary tree is symmetric using iterative approach

- The iterative approach checks symmetry using a level-order traversal with a queue.
- It compares corresponding nodes in the left and right subtrees level by level.
- For each pair of nodes, it checks if their values are equal.

- If all pairs of nodes are symmetric for each level, the tree is symmetric.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once during traversal.**

Space Complexity:

- **The space complexity is O(N) in the worst case, where all nodes are enqueued, as it uses a queue for traversal.**

Conclusion:

Both the recursive and iterative approaches effectively check if a binary tree is symmetric. They share the same time complexity of O(N), ensuring efficient traversal of all nodes. However, their space complexities differ.

- The recursive approach has a space complexity of O(H).
- The iterative approach has a space complexity of O(N) due to the queue.

In terms of memory efficiency, the recursive approach is preferable when memory is a concern, as it has a lower space complexity. However, the iterative approach provides a straightforward solution and is suitable for relatively small trees.