Symmetric Tree

Key Idea:

A tree is symmetric if:

- The left subtree is a mirror of the right subtree.
- Conditions for mirror:
  1. Node values must match.
  2. The left child of one subtree equals the right child of the other (and vice versa).

We solve this by comparing two subtrees at a time:

- compare left → left with right → right.
- compare left → right with right → left.

Recursive Solution:

Algorithm:

```
1. Define a helper function isMirror ($t_1$, $t_2$):
    · If both are null → symmetric.
    · If only one is null → not symmetric.
    · Check $t_1$ → val == $t_2$ → val and recursively:
        · isMirror ($t_1$ → left, $t_2$ → right)
        · isMirror ($t_1$ → right, $t_2$ → left)
2. Call isMirror (root → left, root → right).
```

Iterative Solution:

Idea:

```
Use a queue to store node pairs to compare:
    · Push (root → left, root → right).
    · While queue not empty:
        · Pop pair.
        · If both null → continue.
        - Push children in mirrored order:
            · (left → left, right → right)
            · (left → right, right → left)
```

Complexity:

- Time: O(N) - each node visited once.
- Space: O(H) recursion stack (worst O(N)) or O(N) for queue.