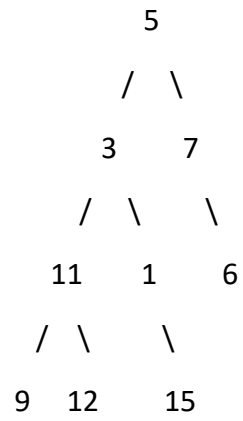
Zig Zag Level Order Traversal of Binary Tree LeetCode

Given the root of a binary tree, return *the zigzag level order traversal of its nodes' values*. (i.e., from left to right, then right to left for the next level and alternate between).

Example:



Output: The Level Order Zig Zag Traversal:

Level 0: 5

Level 1: 7 3

Level 2: 11 1 6

Level 3: 15 12 9

Approach 1: Function for recursive zigzag traversal

- The recursive approach performs zigzag traversal by recursively exploring the left and right subtrees.
- At each level, it determines whether to add nodes from left to right or right to left based on the level's parity (even or odd).
- The traversal is performed level by level, and the result is stored as a vector of vectors.

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(H), where H is the height of the tree, due to the recursive call stack.**

Approach 2: Function for iterative zigzag traversal

- The iterative approach performs zigzag traversal using a queue for level-order traversal.
- It maintains a flag to indicate the direction of traversal (left to right or right to left) for each level.
- At each level, it processes nodes in the queue according to the direction, enqueues their children, and updates the direction for the next level.
- The result is stored as a vector of vectors.

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 perform a zigzag traversal of a binary tree. 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) in the worst case, making it suitable for relatively small trees.

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.