Binary Tree Level Order Traversal Documentation

Problem Description

Given the root of a binary tree, return the level order traversal of its nodes' values (i.e., from left to right, level by level).

Example 1

Input: root = [3, 9, 20, null, null, 15, 7]

Output: [[3], [9, 20], [15, 7]]

Example 2

Input: root = [1]

Output: [[1]]

Example 3

Input: root = []

Output: []

Constraints

- The number of nodes in the tree is in the range [0, 2000].
- -1000 <= Node.val <= 1000

Approach

The solution involves performing a level-order traversal on the binary tree, which is essentially a breadth-first traversal. We use a queue to facilitate this traversal. Each node is processed level by level, and their values are added to the result list in the order they are encountered.

Detailed Steps

1. **Initialization:**

- If the root is None, return an empty list.
- Initialize an empty list result to store the final level order traversal.
- Initialize a queue and add the root node to it.

2. **Breadth-First Traversal:**

- *While the queue is not empty:*
  - Determine the number of nodes at the current level (level_size).
  - Initialize an empty list level_nodes to store the values of the nodes at the current level.
  - *Process each node at the current level:*
    - ✓ Dequeue a node from the queue.
    - ✓ Add its value to level_nodes.
    - ✓ If the node has a left child, enqueue the left child.
    - ✓ If the node has a right child, enqueue the right child.
- After processing all nodes at the current level, add level_nodes to result.

3. **Return the Result:**

- After all levels have been processed, return result.

Explanation of the Code

1. **Class Definitions:**
    - *TreeNode:* A class representing a node in the binary tree. It has three attributes: val (the value of the node), left (the left child), and right (the right child).
    - *Solution:* A class containing the method levelOrder.

2. **Method levelOrder:**

- The method takes the root of the binary tree as input and returns a list of lists containing the values of the nodes at each level.
- If the root is None, it immediately returns an empty list.
- It initializes a queue using deque and adds the root node to it.
- It processes nodes level by level until the queue is empty. For each level, it processes all nodes, collects their values, and adds their children to the queue for the next level.
- It appends the list of values of each level to the result list.
- Finally, it returns the result list containing the level order traversal of the binary tree.

Example Execution

Example 1:

Input: root = [3, 9, 20, null, null, 15, 7]

Execution:

Level 0: [3]

Level 1: [9, 20]

Level 2: [15, 7]

Output: [[3], [9, 20], [15, 7]]

Example 2:

Input: root = [1]

Execution:

Level 0: [1]

Output: [[1]]

Example 3:

Input: root = []

Execution:

No nodes to process.

Output: []

This documentation provides a detailed explanation of the problem, the approach to solving it, the implementation in Python, and examples to illustrate the solution.