Binary Tree Level Order Traversal II

Problem Description

Given the root of a binary tree, return the bottom-up level order traversal of its nodes' values. This means the traversal should be performed from left to right, level by level, from the leaf nodes up to the root.

Example 1

Input:

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

Output:

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

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

Solution

The solution involves performing a level order traversal of the binary tree, but instead of returning the result from the root to the leaves, we need to return it from the leaves to the root. This can be achieved by performing a normal level order traversal and then reversing the order of the levels before returning the result.

Explanation

1. **Initialization:**

- If the root is None, return an empty list since there are no nodes to traverse.
- Initialize an empty list result to store the final bottom-up level order traversal.
- Use a deque queue to perform a breadth-first search (BFS), starting with the root node.

2. **BFS Traversal:**

- While there are nodes in the queue, process each level one by one.
- For each level, determine the number of nodes (level_size) and initialize a list level_nodes to store the values of the nodes at the current level.
- Dequeue nodes one by one, add their values to level_nodes, and enqueue their left and right children if they exist.

3. **Store and Reverse Levels:**

- Append the level_nodes list to result after processing each level.
- After processing all levels, reverse the result list to obtain the bottom-up level order traversal.

This approach ensures that the nodes are traversed level by level, and reversing the list at the end gives the required bottom-up order.