Documentation for Minimum Depth of Binary Tree Solution

Problem Statement

Given a binary tree, find its minimum depth. The minimum depth is defined as the number of nodes along the shortest path from the root node down to the nearest leaf node. Note that a leaf is a node with no children.

Example 1:

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

Output: 2

Example 2:

Input: root = [2, null, 3, null, 4, null, 5, null, 6]

Output: 5

Constraints:

- The number of nodes in the tree is in the range [0, 10^5].
- -1000 <= Node.val <= 1000

Solution

The solution uses a breadth-first search (BFS) approach to find the minimum depth of the binary tree. BFS is chosen because it explores all nodes at the present depth level before moving on to nodes at the next depth level, ensuring that the first leaf node encountered is at the minimum depth.

Algorithm

1. **Initialization:**

- If the root is None, return a depth of 0 as the tree is empty.
- Initialize a queue with the root node and its depth as 1.

2. **Breadth-First Search (BFS):**

- *While the queue is not empty, repeat the following steps:*
  - Dequeue the front node and its depth.
  - Check if the node is a leaf node (i.e., it has no left and right children). If it is, return the current depth as the minimum depth.
  - If the node has a left child, enqueue the left child with its depth incremented by 1.
  - If the node has a right child, enqueue the right child with its depth incremented by 1.

Explanation of the Code

1. **TreeNode Class:**

- Defines a binary tree node with a value (val), left child (left), and right child (right).

2. **Solution Class:**

- Contains the minDepth method which calculates the minimum depth of the binary tree.
- Checks if the root is None, returning 0 for an empty tree.
- Uses a queue to perform BFS. Each element in the queue is a tuple containing a node and its depth.
- *Iteratively processes nodes from the queue:*
  - If a leaf node is encountered, the current depth is returned.
  - If a node has children, they are added to the queue with an incremented depth.

Complexity Analysis

- **Time Complexity:** (O(n)), where (n) is the number of nodes in the tree. Each node is visited at most once.
- **Space Complexity:** (O(n)), where (n) is the number of nodes in the tree. In the worst case, the queue could contain all nodes at the last level.