Maximum Depth of Binary Tree

Problem Statement

Given the root of a binary tree, return its maximum depth. A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Example 1

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

Output: 3

Example 2

Input: root = [1,null,2]

Output: 2

Constraints

- The number of nodes in the tree is in the range [0, 10^4].
- -100 <= Node.val <= 100

Approach

To find the maximum depth of a binary tree, we can use a recursive approach. The idea is to traverse the tree from the root node to the leaf nodes, calculating the depth at each node. The depth of a node is 1 plus the maximum depth of its left and right subtrees.

Explanation

1. **TreeNode Class:**

- The TreeNode class defines a node in the binary tree.
- Each node has a value (val), a left child (left), and a right child (right).

2. **Solution Class:**

- The Solution class contains the method maxDepth which calculates the maximum depth of the binary tree.

3. **maxDepth Method:**

- The method maxDepth takes the root node of the binary tree as an input.
- If the root is None (i.e., the tree is empty), it returns 0.
- Otherwise, it recursively calculates the depth of the left subtree (left_depth) and the depth of the right subtree (right_depth).
- The maximum depth of the current node is the maximum of the depths of its left and right subtrees, plus 1 (for the current node).
- This process continues until all nodes are visited, and the depth of the deepest path is returned.

Complexity Analysis

- **Time Complexity:** O(n), where n is the number of nodes in the tree. Each node is visited exactly once.
- **Space Complexity:** O(h), where h is the height of the tree. This space is used for the recursion stack. In the worst case (unbalanced tree), the space complexity can be O(n).

This solution efficiently finds the maximum depth of a binary tree using a simple and intuitive recursive approach.