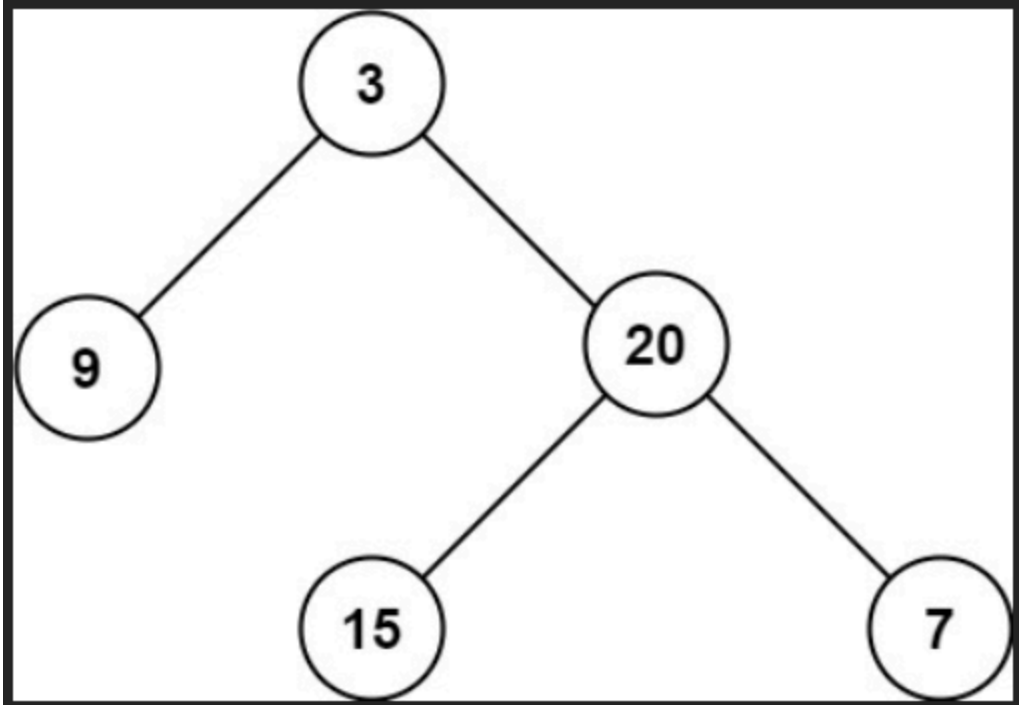
104. Maximum Depth of Binary Tree

Easy | Topics | Companies

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`

Python:

```
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def maxDepth(self, root: TreeNode) -> int:
        # Base case: if the node is None, depth = 0
        if not root:
            return 0

        # Recursively find the depth of left and right subtrees
        left_depth = self.maxDepth(root.left)
        right_depth = self.maxDepth(root.right)

        # Depth of current tree = max(left, right) + 1
        return max(left_depth, right_depth) + 1


# ------------------
# Example Usage:
```

```
# Input: root = [3,9,20,null,null,15,7]
# Let's manually build the tree:

root = TreeNode(3)
root.left = TreeNode(9)
root.right = TreeNode(20)
root.right.left = TreeNode(15)
root.right.right = TreeNode(7)

sol = Solution()
print(sol.maxDepth(root))  # Output: 3
```

JavaScript:

```
// Definition for a binary tree node.
function TreeNode(val, left = null, right = null) {
  this.val = val;
  this.left = left;
  this.right = right;
}

/**
 * @param {TreeNode} root
 * @return {number}
 */
var maxDepth = function(root) {
    // Base case: if root is null, depth = 0
    if (root === null) return 0;

    // Recursively find the depth of left and right subtrees
    let leftDepth = maxDepth(root.left);
    let rightDepth = maxDepth(root.right);

    // Return the maximum of the two depths + 1 (for current node)
    return Math.max(leftDepth, rightDepth) + 1;
};

// Example 1:
let root1 = new TreeNode(3);
root1.left = new TreeNode(9);
root1.right = new TreeNode(20, new TreeNode(15), new TreeNode(7));
console.log(maxDepth(root1)); // Output: 3

// Example 2:
let root2 = new TreeNode(1);
```

```
root2.right = new TreeNode(2);
console.log(maxDepth(root2)); // Output: 2
```

Java:

```
// Definition for a binary tree node.
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
    TreeNode() {}
    TreeNode(int val) { this.val = val; }
    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val = val;
        this.left = left;
        this.right = right;
    }
}
class Solution {
    public int maxDepth(TreeNode root) {
        if (root == null) {
            return 0; // base case: empty tree has depth 0
        }
        int leftDepth = maxDepth(root.left);   // get left subtree depth
        int rightDepth = maxDepth(root.right); // get right subtree depth
        return Math.max(leftDepth, rightDepth) + 1; // max depth + current node
    }
}
```