TREE

Node Traversal

```
In-order etc
```

Problem – Maximum Depth of Binary Tree

Easy

https://leetcode.com/problems/maximum-depth-of-binary-tree

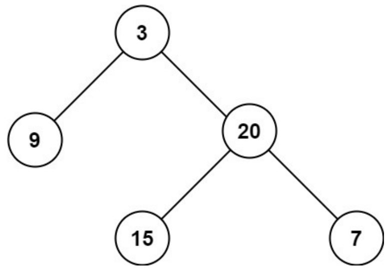
Given the **root** of a binary tree, return its maximum depth.

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

Solution – Maximum Depth of Binary Tree

Easy

https://leetcode.com/problems/maximum-depth-of-binary-tree

```
int maxDepth(TreeNode* root) {
    if (!root) return 0;
    int maxLeft = maxDepth(root->left);
    int maxRight = maxDepth(root->right);
    return std::max(maxLeft, maxRight) + 1;
}
```

Problem – Kth Smallest Element in a BST

Medium

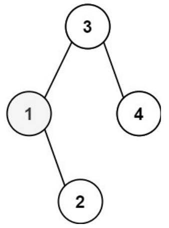
https://leetcode.com/problems/kth-smallest-element-in-a-bst

Given the **root** of a binary search tree, and an integer **k**, return the **k**th smallest value (1-indexed) of all the values of the nodes in the tree.

Example 1

Input: root = [3,1,4,null,2], k = 1

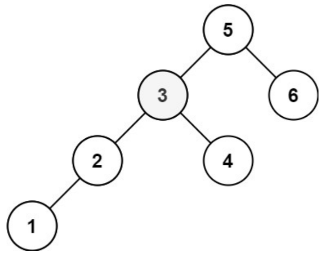
Output: 1



Example 2

Input: root = [5,3,6,2,4,null,null,1], k = 3

Output: 3



Solution – Maximum Depth of Binary Tree

Medium

https://leetcode.com/problems/maximum-depth-of-binary-tree

```
int kthSmallest(TreeNode* root, int k) {
    int count = 0;
    int output;
    traverse(root, count, output, k);
    return output;
}

// perform in-order traversal:  left, node, right
void traverse(TreeNode* node, int& count, int &output, int k) {
    if (!node) return;
    traverse(node->left, count, output, k);
    count++;
    if (count == k) {
        output = node->val;
        return;
    }
    traverse(node->right, count, output, k);
}
```