

## 104. Maximum Depth of Binary Tree

Maximum Depth of Binary Tree

104. 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:

```
graph TD; 3((3)) --- 9((9)); 3 --- 20((20)); 20 --- 15((15)); 20 --- 7((7))
```

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

Output: `3`

13.4K 159 226 Online

```
C++  
1 class Solution {  
2 public:  
3     int maxDepth(TreeNode* root) {  
4         if (!root) {  
5             return 0;  
6         }  
7         return 1 + max(maxDepth(root->left), maxDepth(root->right));  
8     }  
9 };
```

Accepted Runtime: 0 ms

Case 1 Case 2

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

## 48. Validate Binary Search Tree

Validate Binary Search Tree

98. Validate Binary Search Tree

Given the `root` of a binary tree, determine if it is a *valid binary search tree (BST)*.

A **valid BST** is defined as follows:

- The left *subtree* of a node contains only nodes with keys **less than** the node's key.
- The right *subtree* of a node contains only nodes with keys **greater than** the node's key.
- Both the left and right subtrees must also be binary search trees.

Example 1:

```
graph TD; 2((2)) --- 1((1)); 2 --- 3((3))
```

Input: `root = [2,1,3]`

Output: `true`

17.4K 228 138 Online

```
C++  
1 class Solution {  
2 public:  
3     bool isValidBST(TreeNode* root) {  
4         return valid(root, LONG_MIN, LONG_MAX);  
5     }  
6 private:  
7     bool valid(TreeNode* node, long minimum, long maximum) {  
8         if (!node) return true;  
9         if (!(node->val > minimum && node->val < maximum)) return false;  
10        return valid(node->left, minimum, node->val) && valid(node->right, node->val, maximum);  
11    }  
12 };
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input: `root = [2,1,3]`

## 101. Symmetric Tree

101. Symmetric Tree

Easy Topics Companies

Given the `root` of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

**Example 1:**

Input: `root = [1,2,2,3,4,4,3]`  
Output: `true`

**Example 2:**

15.9K 197 126 Online

```
1 public:
2     bool isSymmetric(TreeNode* root) {
3         return isMirror(root->left, root->right);
4     }
5
6 private:
7     bool isMirror(TreeNode* n1, TreeNode* n2) {
8         if (n1 == nullptr && n2 == nullptr) {
9             return true;
10        }
11
12        if (n1 == nullptr || n2 == nullptr) {
13            return false;
14        }
15
16        return n1->val == n2->val && isMirror(n1->left, n2->right) && isMirror(n1->right, n2->left);
17    }
18
19 }
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

`root = [1,2,2,3,4,4,3]`

## 103. Binary Tree Zigzag Level Order Traversal

103. Binary Tree Zigzag Level Order Traversal

Medium Topics Companies

Given the `root` of a binary tree, return the *zigzag level order traversal* of its nodes' values. (i.e., from left to right, then right to left for the next level and alternate between).

**Example 1:**

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

11.3K 136 72 Online

```
1 v.push_back(ptr->val);
2
3 if (alternate) {
4     reverse(v.begin(), v.end());
5 }
6 res.push_back(v);
7 doLevelOrderTraversal(newQu, !alternate);
8
9 vector<vector<int>> zigzagLevelOrder(TreeNode* root) {
10     if (root == NULL) {
11         return res;
12     }
13     queue<TreeNode*> qu;
14     qu.push(root);
15     doLevelOrderTraversal(qu, false);
16     return res;
17 }
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

## 236. Lowest Common Ancestor of a Binary Tree

Lowest Common Ancestor of a

leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/description/

Problem List

Run

Submit

0

Premium

Description

Editorial

Solutions

Submissions

### 236. Lowest Common Ancestor of a Binary Tree

Medium Topics Companies

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the [definition of LCA on Wikipedia](#): "The lowest common ancestor is defined between two nodes  $p$  and  $q$  as the lowest node in  $T$  that has both  $p$  and  $q$  as descendants (where we allow **a node to be a descendant of itself**)."

**Example 1:**

```
graph TD
    3((3)) --> 5((5))
    3 --> 1((1))
    5 --> 6((6))
    5 --> 2((2))
    2 --> 7((7))
    2 --> 4((4))
    1 --> 8((8))
```

**Input:** root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1  
**Output:** 3  
**Explanation:** The LCA of nodes 5 and 1 is 3.

**Example 2:**

17.3K 116 238 Online

Code

C++ Auto

```
1 class Solution {
2 public:
3     TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
4         if (root == nullptr || root == p || root == q) {
5             return root;
6         }
7
8         TreeNode* left = lowestCommonAncestor(root->left, p, q);
9         TreeNode* right = lowestCommonAncestor(root->right, p, q);
10
11         if (left != nullptr && right != nullptr) {
12             return root;
13         }
14
15         return left != nullptr ? left : right;
16     }
17 };
```

Saved Ln 17, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

root = [3,5,1,6,2,0,8,null,null,7,4]

## 94. Binary Tree Inorder Traversal

Binary Tree Inorder Traversal - I

leetcode.com/problems/binary-tree-inorder-traversal/description/

Problem List

Run

Submit

0

Premium

Description

Editorial

Solutions

Submissions

### 94. Binary Tree Inorder Traversal

Easy Topics Companies

Given the `root` of a binary tree, return the *inorder traversal* of its nodes' values.

**Example 1:**

**Input:** root = [1,null,2,3]  
**Output:** [1,3,2]  
**Explanation:**

```
graph TD
    1((1)) --> 2((2))
    2 --> 3((3))
```

14K 190 179 Online

Code

C++ Auto

```
1 class Solution {
2 public:
3     vector<int> inorderTraversal(TreeNode* root) {
4         vector<int> res;
5         inorder(root, res);
6         return res;
7     }
8
9 private:
10    void inorder(TreeNode* node, vector<int>& res) {
11        if (!node) {
12            return;
13        }
14        inorder(node->left, res);
15        res.push_back(node->val);
16        inorder(node->right, res);
17    }
18 };
```

Saved Ln 18, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input

root = [1,null,2,3]

## 102. Binary Tree Level Order Traversal

Binary Tree Inorder Traversal - L

leetcode.com/problems/binary-tree-inorder-traversal/description/

Problem List

Run

Submit

100

Premium

Description

Editorial

Solutions

Submissions

### 94. Binary Tree Inorder Traversal

Easy Topics Companies

Given the `root` of a binary tree, return *the inorder traversal of its nodes' values*.

**Example 1:**

Input: `root = [1,null,2,3]`

Output: `[1,3,2]`

Explanation:

```
graph TD; 1((1)) --> 2((2)); 2 --> 3((3))
```

14K

190

179 Online

C++

Auto

Ln 18, Col 3

```
1 class Solution {
2 public:
3     vector<int> inorderTraversal(TreeNode* root) {
4         vector<int> res;
5         inorder(root, res);
6         return res;
7     }
8
9 private:
10    void inorder(TreeNode* node, vector<int>& res) {
11        if (!node) {
12            return;
13        }
14        inorder(node->left, res);
15        res.push_back(node->val);
16        inorder(node->right, res);
17    }
18 }
```

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4

Input

```
root =
[1,null,2,3]
```

### 230. Kth Smallest Element in a BST

230. Kth Smallest Element in a BST

Medium

Topics Companies Hint

Given the `root` of a binary search tree, and an integer `k`, return the  $k^{\text{th}}$  smallest value (1-indexed) of all the values of the nodes in the tree.

**Example 1:**

```
graph TD; 3((3)) --> 1((1)); 3 --> 4((4)); 1 --> 2((2))
```

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

Code

```
C++
1  }
2  if(curr->left==NULL){
3      value=curr->val;
4      count++;
5      curr=curr->right;
6  }else{
7      TreeNode* leftnode=curr->left;
8      while(leftnode->right!=NULL){
9          leftnode=leftnode->right;
10     }
11     leftnode->right=curr;
12     TreeNode* temp=curr;
13     curr=curr->left;
14     temp->left=NULL;
15 }
16 }return value;
17 }
```

Saved

Ln 40, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
root =
[3,1,4,null,2]
```

## 116. Populating Next Right Pointers in Each Node

Populating Next Right Pointers

leetcode.com/problems/populating-next-right-pointers-in-each-node/description/

Problem List

Run

Submit

🔍

📄

Description

Editorial

Solutions

Submissions

### 116. Populating Next Right Pointers in Each Node

Medium Topics Companies

You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

```
struct Node {
    int val;
    Node *left;
    Node *right;
    Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.

Initially, all next pointers are set to NULL.

**Example 1:**

10K 73 54 Online

Code

C++ Auto

```
28     int n = q.size();
29
30     for(int i=0;i<n;i++){
31         Node* t = q.front();
32         q.pop();
33
34         if(i!=n-1){
35             t->next=q.front();
36         }
37
38         if(t->left) q.push(t->left);
39         if(t->right) q.push(t->right);
40     }
41     return root;
42 }
43
44
45 }
```

Saved Ln 45, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

root = [1,2,3,4,5,6,7]

## 404. Sum of Left Leaves

Sum of Left Leaves - LeetCode

leetcode.com/problems/sum-of-left-leaves/description/

Problem List

Run

Submit

🔍

📄

Description

Editorial

Solutions

Submissions

### 404. Sum of Left Leaves

Easy Topics Companies

Given the **root** of a binary tree, return the **sum of all left leaves**.

A **leaf** is a node with no children. A **left leaf** is a leaf that is the left child of another node.

**Example 1:**

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

5.6K 92 23 Online

Code

C++ Auto

```
13     auto [node, isLeft] = q.front();
14     q.pop();
15
16     if (isLeft && !node->left && !node->right) {
17         totalSum += node->val;
18     }
19
20     if (node->left) {
21         q.push({node->left, true});
22     }
23     if (node->right) {
24         q.push({node->right, false});
25     }
26 }
27
28 return totalSum;
29
30 }
```

Saved Ln 30, Col 3

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

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