

WORKSHEET-6**Student Name:** Amit Kumar**UID:** 22BCS16327**Branch:** CSE**Section/Group:** NTPP-603-B**Semester:** 6th**Date of Performance:** 27/2/25**Subject Name:** AP-2**Subject Code:** 22CSP-351

Aim(i): 104. 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.

Source Code:

```
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if (root == nullptr) {
            return 0;
        }
        int leftdepth = maxDepth(root->left);
        int rightdepth = maxDepth(root->right);

        return 1 + max(leftdepth, rightdepth);
    }
};
```

OUTPUT:

• Case 1

• Case 2

Input

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

Output

3

Expected

3

Accepted 86 / 86 testcases passed

ILB2SrjEs1 submitted at Feb 27, 2025 09:38

Editorial

Solution

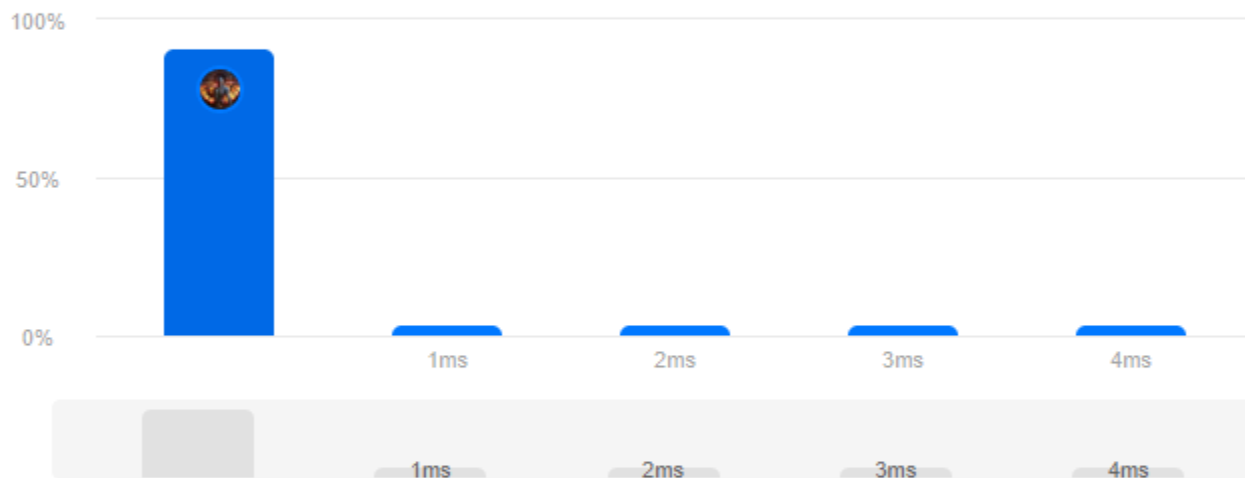
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

21.97 MB | Beats 47.88%



Code | C++

```
class Solution {  
  
bool isPossible(TreeNode* root, long long l, long long r){  
    if(root == nullptr) return true;  
    if(root->val < l and root->val > r)  
        return isPossible(root->left, l, root->val) and  
               isPossible(root->right, root->val, r);  
}
```

LEARNING OUTCOME:

1. We learnt about Depth og trees.
2. We learnt how to use null pointer.
3. We learnt how to use if else statement.

Aim(ii): 98. 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.

You are given an API `bool isBadVersion(version)` which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Source Code:

```
class Solution {

bool isPossible(TreeNode* root, long long l, long long r){
    if(root == nullptr) return true;
    if(root->val < r and root->val > l)
        return isPossible(root->left, l, root->val) and
                isPossible(root->right, root->val, r);
    else return false;
}

public:
    bool isValidBST(TreeNode* root) {
        long long int min = -1000000000000, max = 1000000000000;
        return isPossible(root, min, max);
    }
};
```

OUTPUT:

☒ Testcase | [Test Result](#)

• Case 1

• Case 2

Input

root =
[2,1,3]

Output

true

Expected

true

← All Submissions

Accepted 86 / 86 testcases passed

ILB2SrjEs1 submitted at Feb 27, 2025 09:38

Editorial

Solution

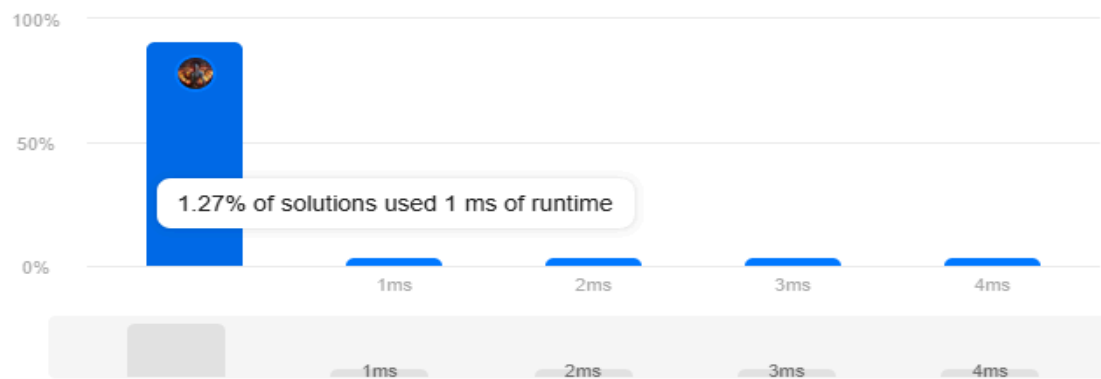
Runtime

0 ms | Beats 100.00%

[Analyze Complexity](#)

Memory

21.97 MB | Beats 47.88%



Code | C++

```
class Solution {  
  
    bool isPossible(TreeNode* root, long long l, long long r){  
        if(root == nullptr) return true;  
        if(root->val < l and root->val > r)  
            return isPossible(root->left, l, root->val) and  
                   isPossible(root->right, root->val, r);  
    }  
};
```

Learning Outcomes

1. We learnt how to use Long.
2. We learnt how to validate Binary search trees.

Aim(iii): 101. Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

.

Source Code:

```
class Solution {
public:
    bool isMirror(TreeNode* left, TreeNode* right) {
        if (!left && !right) return true;
        if (!left || !right) return false;
        return (left->val == right->val) &&
isMirror(left->left, right->right) && isMirror(left->right,
right->left);
    }

    bool isSymmetric(TreeNode* root) {
        if (!root) return true;
        return isMirror(root->left, root->right);
    }

};

};
```

OUTPUT:

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

Output

```
true
```

Expected

```
true
```

Description | Editorial | Solutions | **Accepted** X | Submissions

← All Submissions

Accepted 200 / 200 testcases passed

🔥 ILB2SrjEs1 submitted at Feb 27, 2025 09:57

Editorial Solution

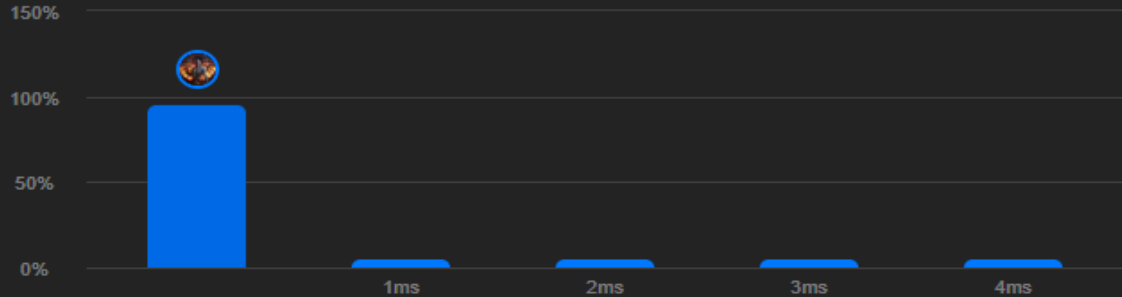
Runtime ⓘ

0 ms | Beats 100.00% 🏆

🔮 Analyze Complexity

Memory ⓘ

18.48 MB | Beats 58.70% 🏆



Code | C++

```
class Solution {  
public:  
    bool isMirror(TreeNode* left, TreeNode* right) {  
        if (!left && !right) return true;  
        if (!left || !right) return false;  
        return (left->val == right->val) && isMirror(left->left, right->right) && isM:  
    }  
}
```


Learning Outcomes

1. We learnt about Symmetric trees.
2. We learnt usage of Boolean.