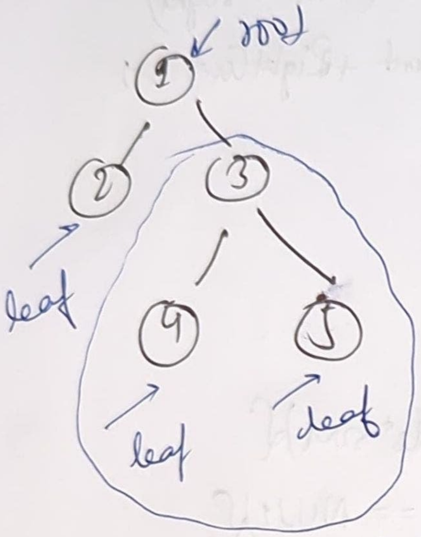
→ max depth

Height of a Binary Tree

↳ max dist from root to leaf
(dist measured in terms of node)



Max dist = 3

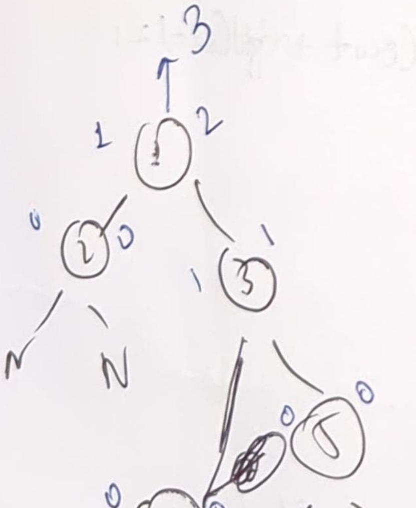
left Sub Tree height = 1

right Sub Tree height = 2

right Ht = 2

```
int height (root) {
    if (root == NULL) return 0;
    leftHt = height (root → left);
    rightHt = height (root → right);
    return max (leftHt, rightHt) + 1
```

TC: O(n)



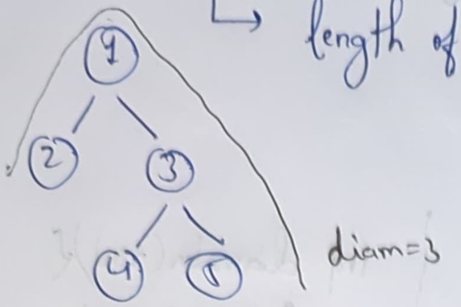
Code :

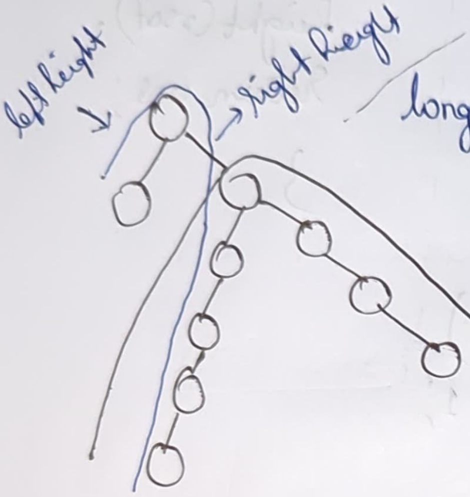
```
//height of a tree
int height (Node* root) {
    if (root == NULL) {
        return 0; }
    int leftHt = height (root → left);
    int rightHt = height (root → right);
    return max (leftHT, rightHT) + 1
}
```

Diameter of a Binary Tree

↳ length of longest path B/w^n any 2 nodes ↳ leaves

(in term of edges)





longest path needn't be necessary that it will include root node, it can be traced by right or left subtree also

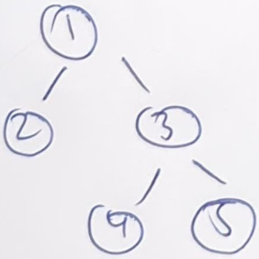
↳ longest path
↳ right Diameter (say)

3 Cases for any node

↳① diameter traced through root node
  ↳ length = left height + right height

② right Diameter can be longest value

③ left " " " "



Case 1 value : 3
Case 2 " : 0
Case 3 " : 2

Pseudo Code:

```
int diam(root) {
    if (root == NULL) return 0;
    leftDiam = diam(root→left)
    rightDiam = diam(root→right)
    currDiam = height(root→left) +
                    height(root→right)
    return max(leftDiam, rightDiam,
                    currDiam)
```

{ LC: 543

TC: O(n*n)
O(n^2)

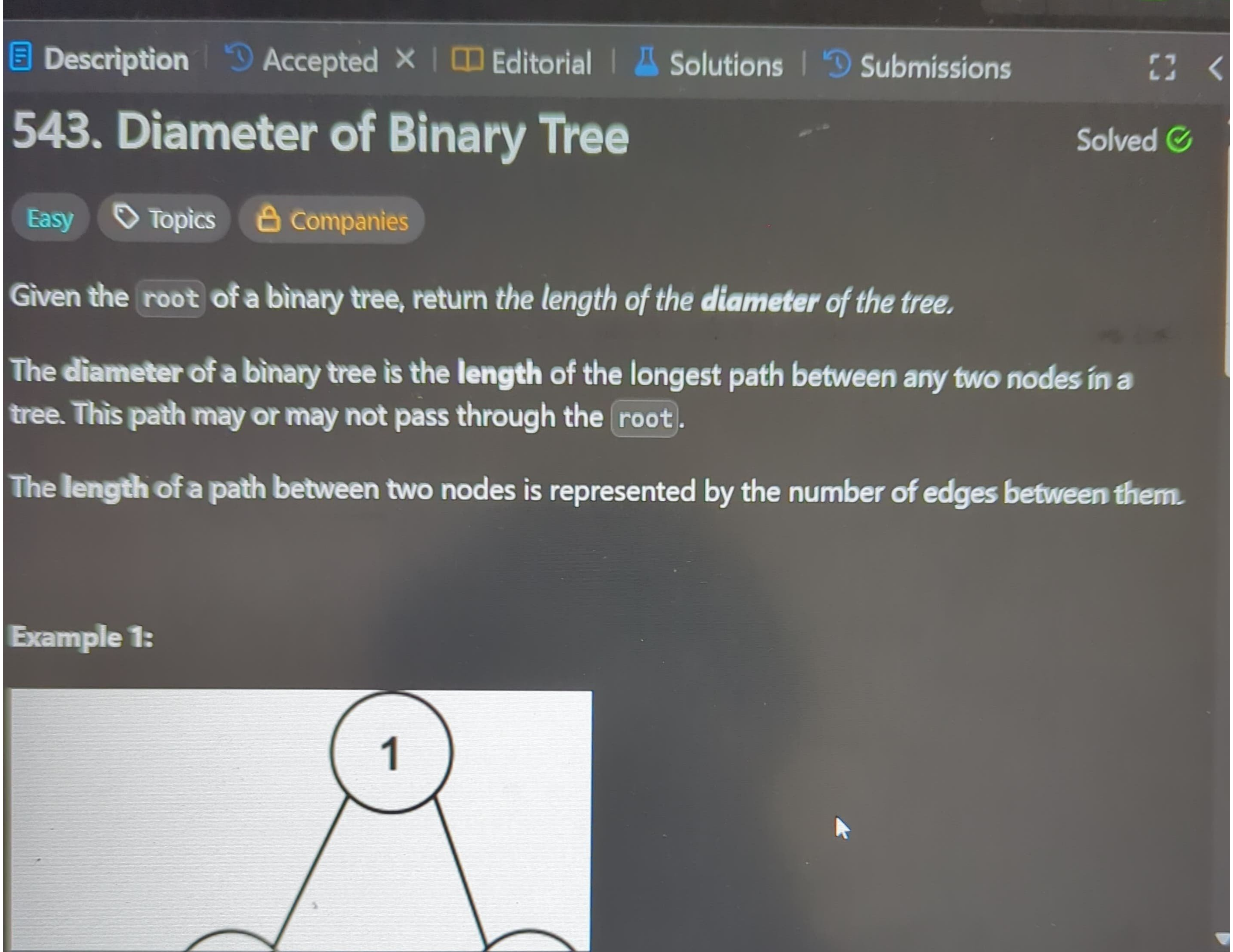
Optimal Approach: Go to each single node & Calulate Curr diameter & compare them

$\longrightarrow$ O(n)

Pseudo Code:

```
int ans = 0
int height (root) {
    if (root == NULL).
        return 0;
    leftHt = height(root → left)
    rightHt = height (root → right)
*  → ans = max (leftHt + rightHt, ans)
    return max (leftHt, rightHt) + 1;
}
```

```
int diameter(root){
    height(root).
    return ans;
}
```



543. Diameter of Binary Tree

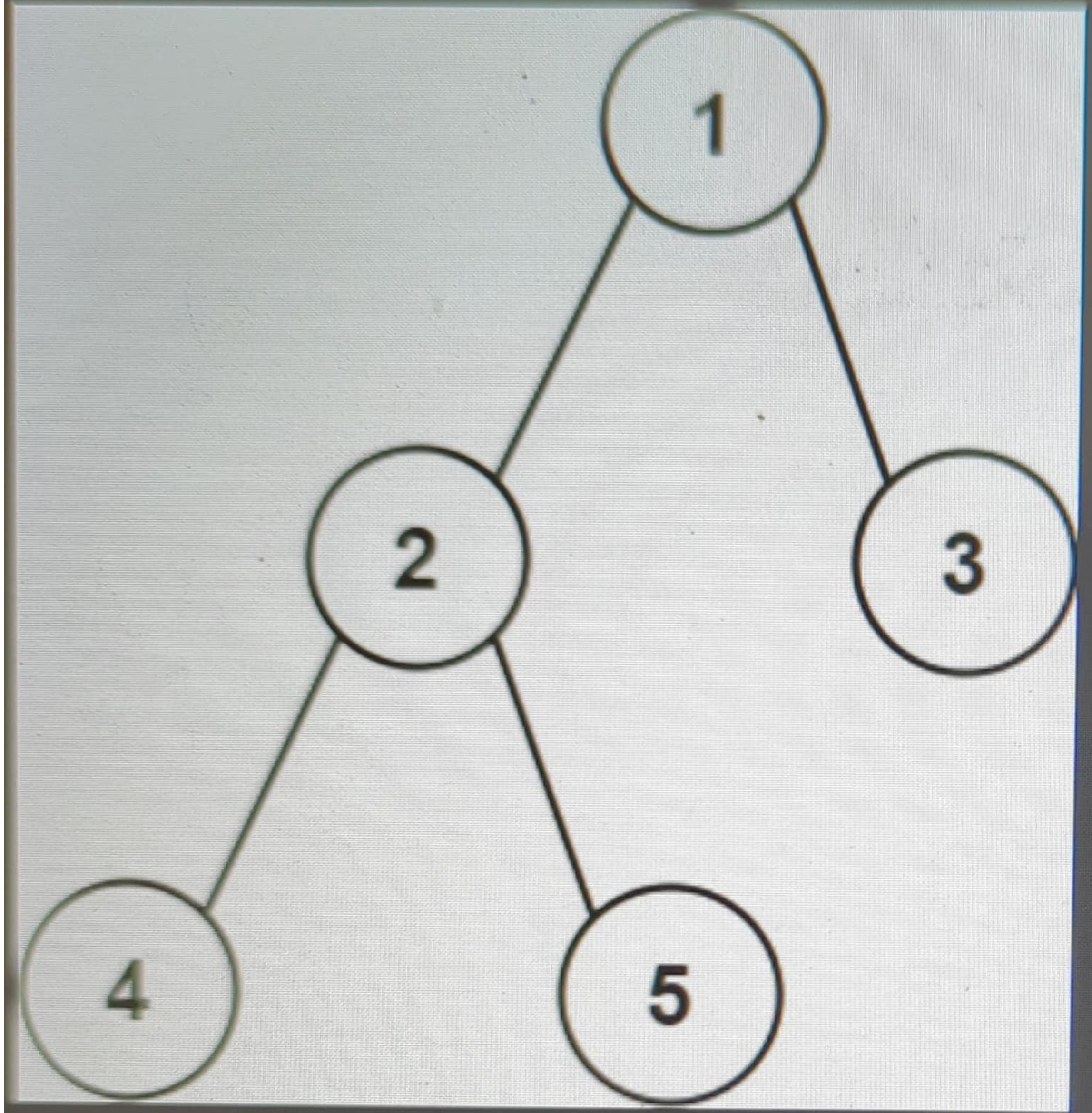
Given the `root` of a binary tree, return *the length of the* ***diameter*** *of the tree.*

The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the `root`.

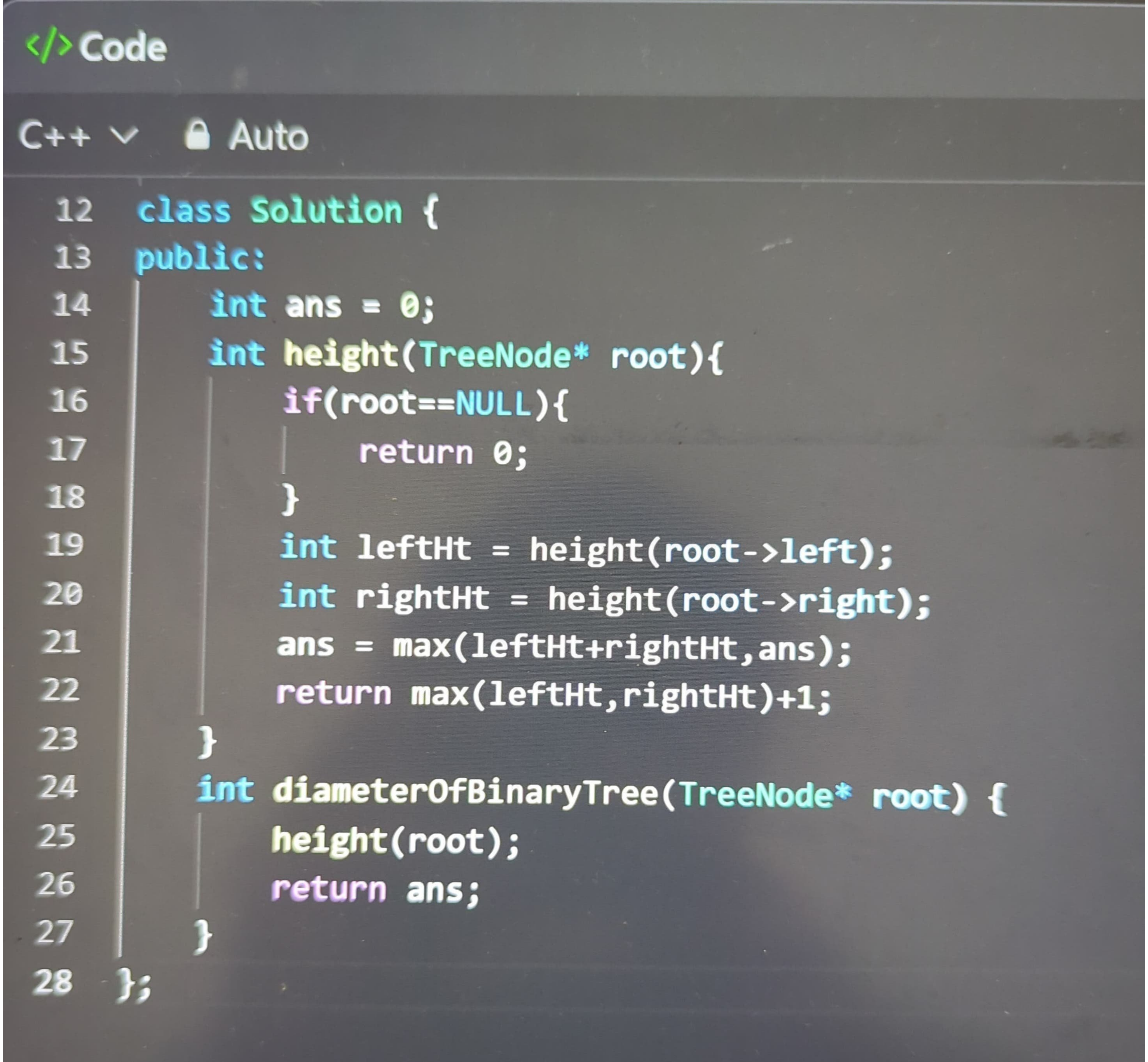
The **length** of a path between two nodes is represented by the number of edges between them.

Example 1:

Example 1:



Input: root = [1,2,3,4,5]
Output: 3
Explanation: 3 is the length of the path [4,2,1,3] or [5,2,1,3].



```cpp
class Solution {
public:
    int ans = 0;
    int height(TreeNode* root){
        if(root==NULL){
            return 0;
        }
        int leftHt = height(root->left);
        int rightHt = height(root->right);
        ans = max(leftHt+rightHt,ans);
        return max(leftHt,rightHt)+1;
    }
    int diameterOfBinaryTree(TreeNode* root) {
        height(root);
        return ans;
    }
};
```