LeetCode 543 — Diameter of Binary Tree

🔗 https://leetcode.com/problems/diameter-of-binary-tree/

1. Problem Title & Link

Title: LeetCode 543: Diameter of Binary Tree
Link: https://leetcode.com/problems/diameter-of-binary-tree/

2. Problem Statement (Short Summary)

Given the root of a binary tree, return the **length of the diameter** of the tree.
Length = number of edges on the longest path between any two nodes in the tree.

Note: Some descriptions count nodes; LeetCode expects **edges** (so a single node → diameter = 0).

3. Examples (Input → Output)

Example 1

Input:

```
   1
  / \
 2   3
/ \
4   5
```

Output: 3
Explanation: Longest path is 4 → 2 → 1 → 3 (3 edges).

Example 2

Input: root = [1,2]
Output: 1
Explanation: Path 2 → 1 (1 edge).

Example 3

Input: root = [1]
Output: 0

4. Constraints

- 0 <= number of nodes <= 10^4
- Node values irrelevant for diameter
- Tree can be skewed (depth up to n)
- Must be computed in O(n) time for optimal solution

5. Core Concept (Pattern / Topic)

⭐ **Tree DFS (Postorder) — Height + Global State**

Also: Divide & Conquer, recursion on trees.

6. Thought Process (Step-by-Step Explanation)

Intuition

For each node, longest path that goes through that node = (max depth of left subtree) + (max depth of right subtree).

Diameter is the maximum of these values across all nodes.

Steps

1. Use DFS that returns the **depth** (max nodes from current node down to a leaf) — base returns 0 for null.
2. For a node: left = depth(node.left), right = depth(node.right).
3. Candidate diameter through node = left + right (this counts edges because depth returns number of nodes on path from child to leaf; when child depth is 1 for leaf, sum behaves as edge count).
4. Keep global max; return 1 + max(left, right) as depth for parent.

This is O(n): each node visited once.

7. Visual / Intuition Diagram (ASCII)

```
   1
  / \
 2   3
/ \
4  5
```

At node 2:

left depth = 1 (node 4)

right depth = 1 (node 5)

candidate diameter = 1 + 1 = 2 (edges: 4-2-5)

At node 1:

left depth = 1 + max(1,1) = 2 (through 2)

right depth = 1 (node 3)

candidate diameter = 2 + 1 = 3 (edges: 4-2-1-3)

8. Pseudocode (Language Independent)

```
global_max = 0
```

```
function depth(node):
    if node == null:
        return 0
    left = depth(node.left)
    right = depth(node.right)
    global_max = max(global_max, left + right)
    return 1 + max(left, right)

call depth(root)
return global_max
```

9. Code Implementation

✅ **Python**

```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 diameterOfBinaryTree(self, root):
        self.ans = 0

        def depth(node):
            if not node:
                return 0
            left = depth(node.left)
            right = depth(node.right)
            # update diameter (number of edges)
            self.ans = max(self.ans, left + right)
            # return height in nodes for parent calculations
            return 1 + max(left, right)

        depth(root)
        return self.ans
```

✅ **Java**

```java
// Definition for a binary tree node.
// public class TreeNode {
//     int val;
```

```
//     TreeNode left;
//     TreeNode right;
//     TreeNode(int x) { val = x; }
// }

class Solution {
    private int ans = 0;

    public int diameterOfBinaryTree(TreeNode root) {
        depth(root);
        return ans;
    }

    private int depth(TreeNode node) {
        if (node == null) return 0;
        int left = depth(node.left);
        int right = depth(node.right);
        ans = Math.max(ans, left + right);
        return 1 + Math.max(left, right);
    }
}
```

10. Time & Space Complexity

- **Time:** O(n) — visit each node once.
- **Space:** O(h) recursion stack where h = tree height (O(n) worst, O(log n) if balanced).

11. Common Mistakes / Edge Cases

- ❌ **Counting nodes vs edges confusion** — be explicit: LeetCode expects **edges**. The left + right update yields edges when depth returns number of nodes in path from child to deepest leaf (with null → 0).
- ❌ Forgetting to update a global maximum inside DFS.
- ❌ Using BFS level counts incorrectly (trickier to compute diameter directly).
- Edge cases: empty tree → 0, single node → 0, skewed tree (diameter = n-1).

12. Detailed Dry Run (Step-by-Step Table)

Tree:

```
  1
 /
2
/
```

3

This is a left skewed chain of 3 nodes.

Node	left depth	right depth	candidate (left+right)	depth returned
3	0	0	0	1
2	1	0	1	2
1	2	0	2	3

Final ans = 2 (edges), which equals n-1 = 2.

13. Common Use Cases (Real-Life / Interview)

- Longest communication path in a hierarchical network
- Longest route through a tree-like road map
- Used in algorithmic questions combining tree DP and path-sum logic

14. Common Traps (Important!)

- Mixing up definitions of depth/height (nodes vs edges) — be consistent.
- Not handling null base correctly.
- Trying to compute diameter by separate left/right scans repeatedly → $O(n^2)$.

15. Builds To (Related LeetCode Problems)

- LC 124 — Binary Tree Maximum Path Sum (similar idea but sums values)
- LC 543 (this) → foundation for LC 124
- LC 865 — Smallest Subtree with all the Deepest Nodes
- LC 112 — Path Sum variants

16. Alternate Approaches + Comparison

Approach	Time	Space	Notes
Single DFS (height + global update)	O(n)	O(h)	Best and simplest
Two-pass (compute all pairs)	$O(n^2)$	O(n)	Inefficient on skewed trees
Preprocessing + LCA-style methods	O(n log n) preprocess, O(log n) query	Higher	Useful for repeated diameter queries on dynamic trees

17. Why This Solution Works (Short Intuition)

At any node the longest path passing through it equals the longest downward path in its left subtree plus the longest downward path in its right subtree. The global maximum of these sums across all nodes is the tree diameter.

18. Variations / Follow-Up Questions

- Return path nodes (not just length).
- Diameter in terms of **node count** instead of edges (just add 1 to edges when non-empty).
- Weighted tree diameter (edge weights) — use sums instead of counts.
- Recompute diameter in a dynamic tree with insert/delete (advanced).