

13
.7

Diameter of Binary Tree

EASY

■ What You Will Learn

- ✓ Diameter = longest path between ANY two nodes (may not pass through root)
- ✓ How height computation gives diameter as a side-product
- ✓ Why a single postorder DFS solves it in $O(n)$
- ✓ The global variable trick to track max diameter

■ Intuition First

The diameter of a binary tree is the length of the longest path between any two nodes. The path doesn't have to pass through the root!

Key insight: for any node, the longest path THROUGH that node = $\text{leftHeight} + \text{rightHeight}$. So if we compute height for every node (postorder), we can track the maximum of $(\text{leftH} + \text{rightH})$ seen at any node. That maximum is the diameter.

■ The Formula

- At every node: diameter through this node = $\text{leftHeight} + \text{rightHeight}$
- Diameter of tree = max of $(\text{leftH} + \text{rightH})$ across ALL nodes
- Track this max in a global/instance variable
- Return height (not diameter) from the helper function

■ Visual

Diameter = 3 (path: $4 \rightarrow 2 \rightarrow 5$ or $4 \rightarrow 2 \rightarrow 1 \rightarrow 3$ — both length 3)

```

1
 / \
2 3
 / \
4 5

```

At node 2: $\text{leftH}=1$, $\text{rightH}=1$. $\text{diameter_through_2} = 1+1 = 2$

At node 1: $\text{leftH}=2$, $\text{rightH}=1$. $\text{diameter_through_1} = 2+1 = 3$

max diameter = 3 ✓

■ Dry Run

Step 1	helper(4): L=0, R=0. diameter=max(0,0+0)=0. return 1
Step 2	helper(5): L=0, R=0. diameter=max(0,0+0)=0. return 1
Step 3	helper(2): L=1, R=1. diameter=max(0,1+1)=2. return 2
Step 4	helper(3): L=0, R=0. diameter=max(2,0+0)=2. return 1
Step 5	helper(1): L=2, R=1. diameter=max(2,2+1)=3. return 3. Answer=3 ✓

■ Java Code

```
int diameter = 0; // Global to track max

public int diameterOfBinaryTree(TreeNode root) {
    height(root);
    return diameter;
}

// Returns height, updates diameter as side effect
private int height(TreeNode node) {
    if (node == null) return 0;
    int leftH = height(node.left);
    int rightH = height(node.right);
    // Update diameter: path through this node
    diameter = Math.max(diameter, leftH + rightH);
    return 1 + Math.max(leftH, rightH); // Return height
}
```

■■ Complexity

Case	Time	Space	What it means
All Cases	$O(n)$	$O(h)$	One DFS pass. Height propagates up, diameter updated at each node

■■ Common Mistakes

- Thinking diameter always passes through root — it doesn't! Track max at EVERY node
- Returning diameter from helper instead of height — the helper MUST return height
- Off-by-one: diameter = number of edges, not nodes. leftH+rightH counts edges correctly

■ Generalizing This Pattern

- Any 'maximum path' tree problem uses this same structure
- Binary Tree Maximum Path Sum uses the same pattern with values instead of heights
- The pattern: compute something bottom-up, update a global max at each node, return partial result up

■ Visual Thinking

Imagine stretching a rubber band between two leaf nodes — the diameter is the tightest stretch.

The rubber band can arc through any node, not necessarily the root.

At each node, you ask: if I'm the 'top' of the arc, how long is my arc? = leftH + rightH.

Track the longest arc seen anywhere in the tree.

Animation: at each node, show leftH + rightH as an arc. Highlight the maximum arc in green.

■ Practice Problems

[Easy] Diameter of tree with only 2 nodes [1, 2]?

[Med] Diameter in terms of nodes instead of edges

[Med] Longest path between two leaves (must pass through an ancestor)

[Hard] Binary Tree Maximum Path Sum (same structure, uses values not heights)

■ Quick Revision Box

- Diameter = $\max(\text{leftH} + \text{rightH})$ across all nodes
- Helper returns HEIGHT, updates global diameter as side effect
- Postorder DFS — $O(n)$ time, $O(h)$ space
- Path doesn't need to pass through root
- $\text{leftH} + \text{rightH}$ = number of edges in path through this node