

13
.9

Binary Tree Maximum Path Sum

HARD

■ What You Will Learn

- ✓ Why this problem is HARD — path can start and end anywhere
- ✓ The split-path vs single-branch distinction
- ✓ Postorder DFS with global max update
- ✓ How to handle negative values correctly

■ Intuition First

Find the path in a binary tree with the maximum sum of node values. The path can start and end at any node — it doesn't need to pass through the root, and it doesn't need to reach a leaf. But it must be a connected path (no branching).

The trick: at each node, you face a choice. You can SPLIT (use both left and right branches — this is the path's 'peak' node), or you can EXTEND upward (pass only one branch to your parent). You update the global max with the split value, but return only the better single branch to your parent.

■ The Key Distinction

- At each node, compute: $\text{gainLeft} = \max(0, \text{maxGain}(\text{left}))$ [ignore negative branches]
- Compute: $\text{gainRight} = \max(0, \text{maxGain}(\text{right}))$
- Path through this node = $\text{node.val} + \text{gainLeft} + \text{gainRight} \rightarrow$ update global max
- But return to parent: $\text{node.val} + \max(\text{gainLeft}, \text{gainRight})$ [can only go ONE way up]

■ Visual

Max path = $15 \rightarrow 20 \rightarrow 7 = 42$

```

-10
 / \
9  20
 / \
15 7

```

At node 20: $\text{gainLeft}=15$, $\text{gainRight}=7$. $\text{path_through_20} = 20+15+7=42 \rightarrow$ global max=42

Return to -10: $20 + \max(15,7) = 35$ (single branch upward)

At -10: $\text{gain_left}=9$, $\text{gain_right}=35$. $\text{path} = -10+9+35=34 < 42$. max stays 42 ✓

■ Dry Run

Step 1	maxGain(9): left=0,right=0. path=9. maxSum=9. return 9
Step 2	maxGain(15): left=0,right=0. path=15. maxSum=15. return 15
Step 3	maxGain(7): left=0,right=0. path=7. maxSum=15. return 7
Step 4	maxGain(20): gainL=15,gainR=7. path=20+15+7=42. maxSum=42. return 20+15=35
Step 5	maxGain(-10): gainL=max(0,9)=9, gainR=max(0,35)=35. path=-10+9+35=34<42. return -10+35=25
Step 6	Final answer = 42 ✓

■ Java Code

```
int maxSum = Integer.MIN_VALUE; // Track global max

public int maxPathSum(TreeNode root) {
    maxGain(root);
    return maxSum;
}

private int maxGain(TreeNode node) {
    if (node == null) return 0;
    // Only take positive contributions (ignore negative branches)
    int gainLeft = Math.max(0, maxGain(node.left));
    int gainRight = Math.max(0, maxGain(node.right));
    // Price of the path that includes this node as the 'peak'
    int pathThroughNode = node.val + gainLeft + gainRight;
    // Update global max (split path)
    maxSum = Math.max(maxSum, pathThroughNode);
    // Return max gain if we continue the path upward (one direction only)
    return node.val + Math.max(gainLeft, gainRight);
}
```

■ ■ Complexity

Case	Time	Space	What it means
All Cases	O(n)	O(h)	Visit every node once. Recursion depth = height

■ ■ Common Mistakes

- Passing gainLeft + gainRight back to parent — can't branch the path upward!
- Not using Math.max(0, ...) — negative subtrees should be excluded, not included

- Initializing `maxSum = 0` instead of `Integer.MIN_VALUE` — fails for all-negative trees

■ The Two Different Returns

- What you UPDATE the global max with: `node.val + gainLeft + gainRight` (can split here)
- What you RETURN to parent: `node.val + max(gainLeft, gainRight)` (must go one way only)
- These are two different values — the bug happens when you confuse them
- This pattern is the template for ALL 'maximum path' tree problems

■ Visual Thinking

Picture a mountain range (the tree). You want to find the highest ridge walk.

At any peak (node), you can include the left slope, right slope, both, or just yourself.

To continue walking, you can only go ONE way from this peak — you can't fork.

The 'split' gives you the best ridge at this peak. The 'return' lets you keep walking.

Animation: at each node, show two numbers: `path_through` (for global max) and `return value` (for parent).

■ Practice Problems

[Hard] Max path sum in tree [-3] — answer is -3

[Hard] Max path sum where path must start and end at leaves

[Hard] Count paths with sum equal to target (path sum III)

[Hard] Diameter of binary tree uses identical structure — compare both

■ Quick Revision Box

- At each node: `gainLeft = max(0, dfs(left))`, `gainRight = max(0, dfs(right))`
- Update global max with: `node.val + gainLeft + gainRight` (split path)
- Return to parent: `node.val + max(gainLeft, gainRight)` (single branch only)
- Init `maxSum = Integer.MIN_VALUE` (handle all-negative trees)
- $O(n)$ time, $O(h)$ space

■ Template for ALL maximum path tree problems