

13
8

Lowest Common Ancestor

MEDIUM

What You Will Learn

- ✓ What LCA means and why it matters
- ✓ Recursive DFS approach — elegant and $O(n)$
- ✓ The three cases when a node becomes the LCA
- ✓ LCA of BST (faster $O(\log n)$ approach using BST property)

Intuition First

The Lowest Common Ancestor (LCA) of two nodes p and q is the deepest node that has both p and q as descendants (a node is a descendant of itself).

Think of a family tree. LCA of two cousins is their grandparent. LCA of a parent and child is the parent itself.

The recursive insight: a node X is the LCA if: (1) p and q are in different subtrees of X , OR (2) X itself is p or q and the other is in its subtree.

The 3 Cases

- Case 1: $\text{node} == p$ or $\text{node} == q \rightarrow$ this is a potential LCA, return node
- Case 2: p is in left subtree, q is in right (or vice versa) $\rightarrow$ current node IS the LCA
- Case 3: both in same subtree $\rightarrow$ the LCA is deeper, return the found node from that subtree
- If one subtree returns null $\rightarrow$ answer is entirely in the other subtree

Visual

```

3
 / \
5 1
 / \ / \
6 2 0 8
 / \
7 4

```

$\text{LCA}(5, 1) = 3$ (5 is left, 1 is right $\rightarrow$ 3 is LCA)

$\text{LCA}(5, 4) = 5$ (4 is a descendant of 5 $\rightarrow$ 5 is its own LCA)

$\text{LCA}(6, 4) = 5$ (6 is left of 5, 4 is right of 5)

■ Dry Run — LCA(6, 4)

Step 1	<code>lca(3,6,4)</code> : go left to 5, go right to 1
Step 2	<code>lca(5,6,4)</code> : go left to 6, go right to 2
Step 3	<code>lca(6,6,4)</code> : <code>node==6==p</code> $\rightarrow$ return node 6 (found p)
Step 4	<code>lca(2,6,4)</code> : go left to 7 (null back), go right to 4
Step 5	<code>lca(4,6,4)</code> : <code>node==4==q</code> $\rightarrow$ return node 4 (found q)
Step 6	Back at <code>lca(2)</code> : <code>left=null</code> , <code>right=4</code> . Return 4 upward
Step 7	Back at <code>lca(5)</code> : <code>left=6(non-null)</code> , <code>right=4(non-null)</code> $\rightarrow$ BOTH sides found $\rightarrow$ return node 5 = LCA ✓

■ Java Code — General Binary Tree

```
public static TreeNode lowestCommonAncestor(TreeNode root,
TreeNode p, TreeNode q) {
    // Base case: empty or found p or q
    if (root == null || root == p || root == q) return root;
    // Search in left and right subtrees
    TreeNode left = lowestCommonAncestor(root.left, p, q);
    TreeNode right = lowestCommonAncestor(root.right, p, q);
    // If both sides found something  $\rightarrow$  current node is LCA
    if (left != null && right != null) return root;
    // Otherwise LCA is in whichever subtree returned non-null
    return left != null ? left : right;
}
```

■ Java Code — BST Variant (Faster!)

For BST: use the sorted property to navigate directly — no need to search both sides.

```
public static TreeNode lcaBST(TreeNode root, TreeNode p, TreeNode q) {
    while (root != null) {
        if (p.val < root.val && q.val < root.val) {
            root = root.left; // Both in left subtree
        } else if (p.val > root.val && q.val > root.val) {
            root = root.right; // Both in right subtree
        } else {
            return root; // Split point = LCA
        }
    }
}
```

```
return null;
}
```

■ Complexity

Case	Time	Space	What it means
General BT	$O(n)$	$O(h)$	May search entire tree. Stack depth = height
BST	$O(h)$	$O(1)$	Follow BST property — at most $O(\log n)$ for balanced

■ Common Mistakes

- Forgetting the case where one of p, q is an ancestor of the other
- The base case $root == p$ returns $root$ immediately — don't search further (p IS the LCA if q is below it)
- Not returning the correct side when only one side finds a match

■ Visual Thinking

Imagine two hikers (p and q) starting at the bottom of two hills (leaf nodes).

They each walk upward toward the peak (root). The first point where their paths MERGE is the LCA.

Recursive DFS: send two scouts (left call, right call). If both come back with a find → you're the LCA.

Animation: highlight paths from p to root and q to root. The first shared node glows as LCA.

■ Practice Problems

[Med] LCA of BST given $p=2$, $q=8$ in tree $[6,2,8,0,4,7,9,null,null,3,5]$

[Med] Distance between two nodes in a tree (uses LCA)

[Hard] LCA with parent pointers (linked list intersection problem)

[Hard] LCA of multiple nodes (generalize to k nodes)

■ Quick Revision Box

- LCA = deepest node that is ancestor of both p and q
- Base case: null or found p/q → return node

- Both sides non-null $\rightarrow$ current is LCA
- One side null $\rightarrow$ LCA is in the other subtree
- $O(n)$ general tree; $O(h)$ for BST using value comparison