Write a program for Lowest Common Ancestors.

Mr. Akash Yadav
Assistant Professor
Artificial Intelligence & Data Science

Lesson Plan

Subject/Course	**Competitive Coding**
Lesson Title	**Lowest Common Ancestor (LCA) in a Binary Tree**

Lesson Objectives
Understand the concept of the Lowest Common Ancestor (LCA) in a binary tree.
Learn how to trace paths of two nodes and identify their shared ancestor.
Implement recursive logic to efficiently find the LCA.
Analyze the time and space complexity of the LCA approach.

Problem Statement:

Given a binary tree and two node values, find their **Lowest Common Ancestor (LCA)**.

If one node is the ancestor of the other, return that node itself.

Concept

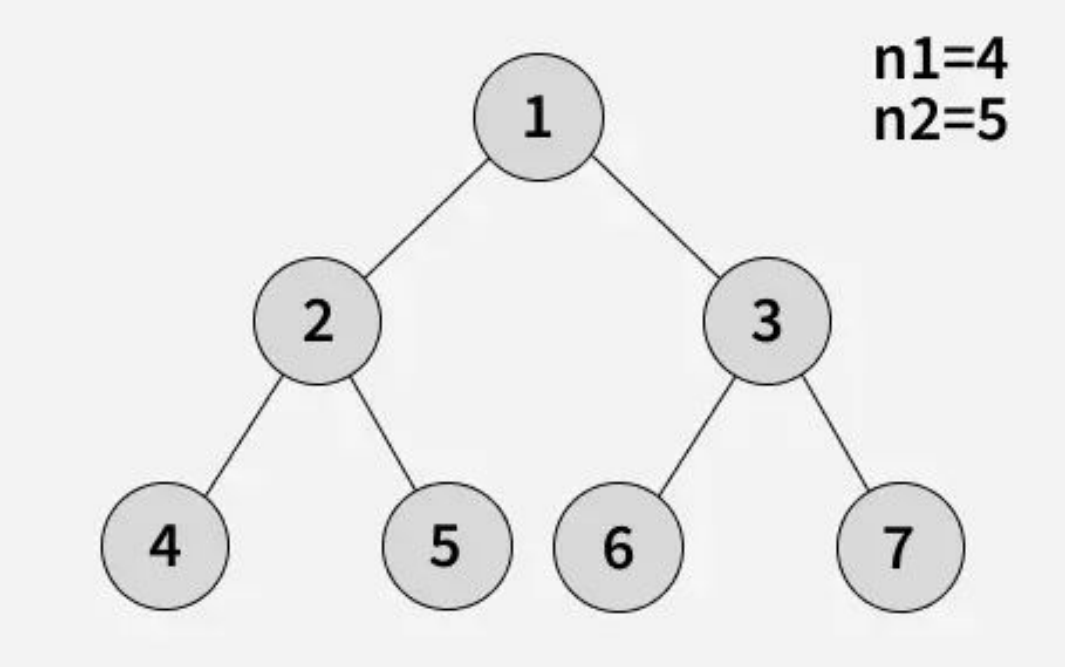
- The **Lowest Common Ancestor** (LCA) of two nodes n1 and n2 in a binary tree is the lowest (deepest) node that has both n1 and n2 as descendants.

- In simpler terms, it's the common parent of both nodes that appears lowest in the tree.
- We can find the LCA efficiently using **recursion**.

Algorithm/Logic

1. If the root is NULL, return NULL.
2. If the root matches either of the two target nodes, return the root.
3. Recursively search for the two nodes in the **left** and **right** subtrees.
4. If both sides return non-NULL, the current node is the **LCA**.
5. Otherwise, return the non-NULL side.

Visualization



Output: 2
Explanation: As shown below, LCA of 4 and 5 is 2.

Code Implementation

```
public class LowestCommonAncestor {

    static class Node {
        int data;
        Node left, right;
        Node(int val) { data = val; }
    }

    static Node findLCA(Node root, int n1, int n2) {
        if (root == null) return null;
        if (root.data == n1 || root.data == n2)
            return root;

        Node leftLCA = findLCA(root.left, n1, n2);
        Node rightLCA = findLCA(root.right, n1, n2);

        if (leftLCA != null && rightLCA != null)
            return root;
        return (leftLCA != null) ? leftLCA : rightLCA;
    }
```

```
    public static void main(String[] args) {
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.left.right = new Node(5);
        root.right.left = new Node(6);
        root.right.right = new Node(7);

        int n1 = 4, n2 = 5;
        Node lca = findLCA(root, n1, n2);
        System.out.println("LCA of " + n1 + " and " + n2 + " is: " + lca.data);
    }
}
```

Output :

```
LCA of 4 and 5 is: 2
```

Time & Space Complexity

Operation	Time Complexity	Space Complexity	Explanation
Find LCA (Recursive)	O(n)	O(h)	Traverses all nodes; recursion depth = tree height (h).

Summary

- The **Lowest Common Ancestor (LCA)** identifies the **shared parent** of two given nodes.
- Solved efficiently using **recursion** with **O(n)** time complexity.
- Widely used in **network routing**, **taxonomy classification**, and **organization hierarchies**.

Practice Questions:

- **Lowest Common Ancestor of a Binary Tree — LeetCode #236**

🔗 *https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/*

Concept:

Use recursion to find the deepest common parent node between two given nodes in a binary tree.

Why Practice:

- Reinforces **recursive tree traversal** logic.
- Builds foundation for **graph ancestry and path problems**.

Thanks