Trees

Suavi Demir - Interview Kickstart
October 18, 2020

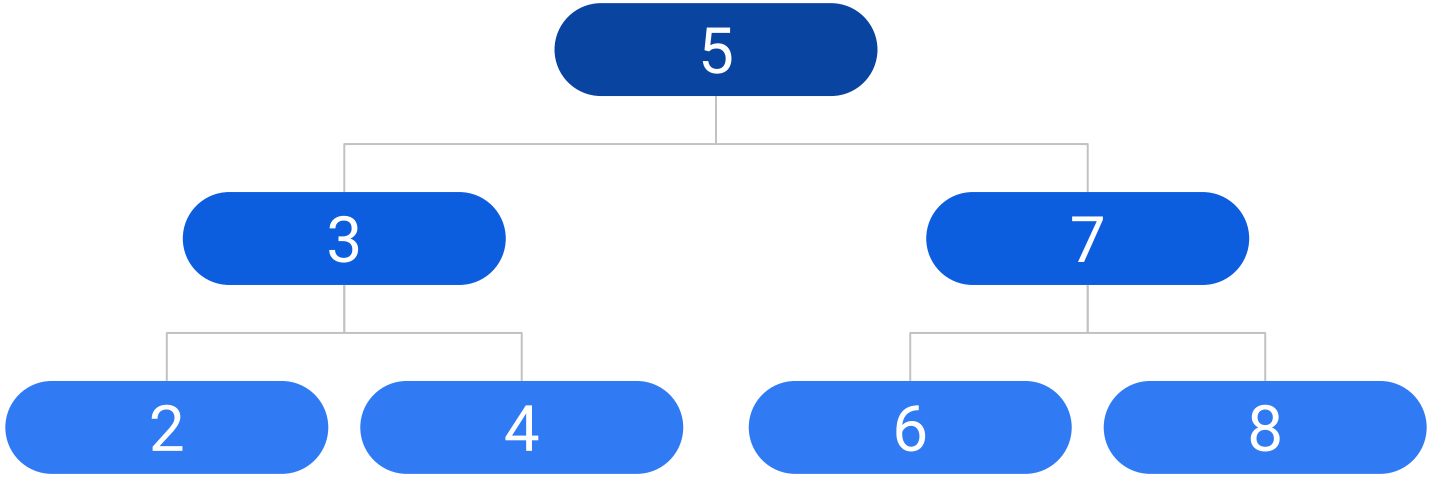
Agenda

- DFS - Depth First Traversal
    - Pre order
    - In-order
    - Post order
    - Using Recursion or Stack
- BFS - Breadth First Traversal
    - Using Queue
- BST - Binary Search Tree
- Tree Construction

Tree Types

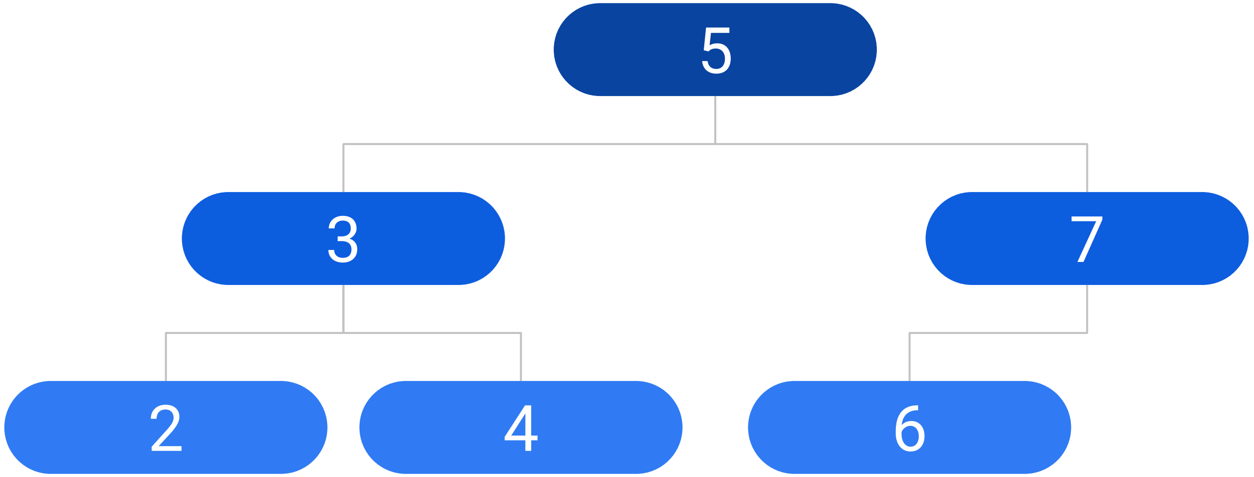
- BST - Binary Search Tree
- Binary Tree
  - each node has left and right child nodes (can be null)
- N-Ary Tree
  - each node has N number of children
  - no nulls, can be empty list

BST / Binary Search Tree



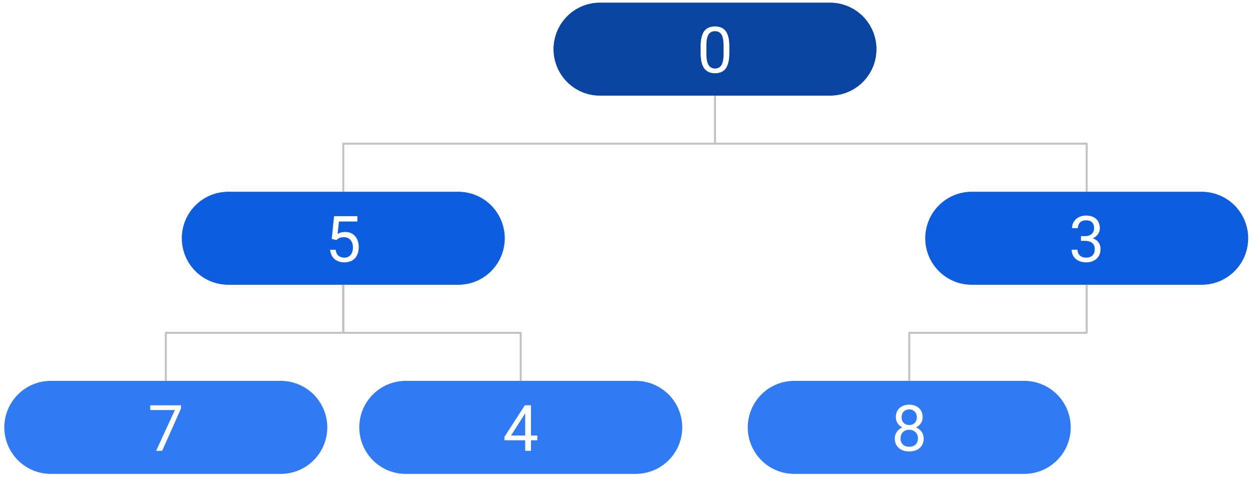
Sorted / Full BST / Balanced (height=log n)

BST / Binary Search Tree



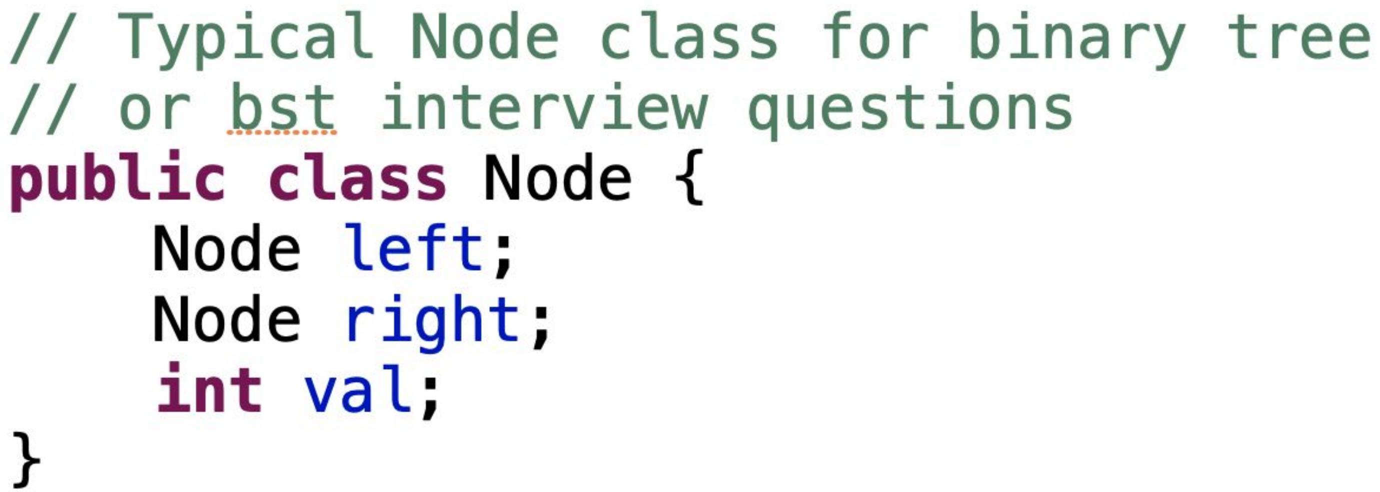
Sorted / Complete BST

Binary Tree



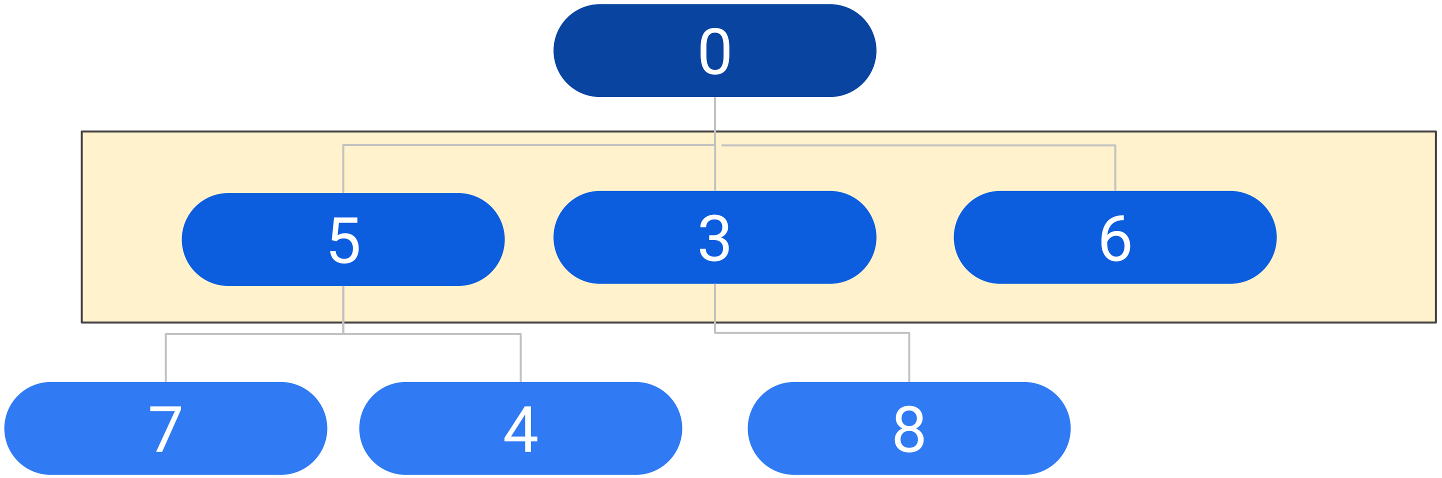
Each node has left and right child,
but they are not necessarily sorted

Binary Tree Node



```java
// Typical Node class for binary tree
// or bst interview questions
public class Node {
    Node left;
    Node right;
    int val;
}
```

N-Ary Tree

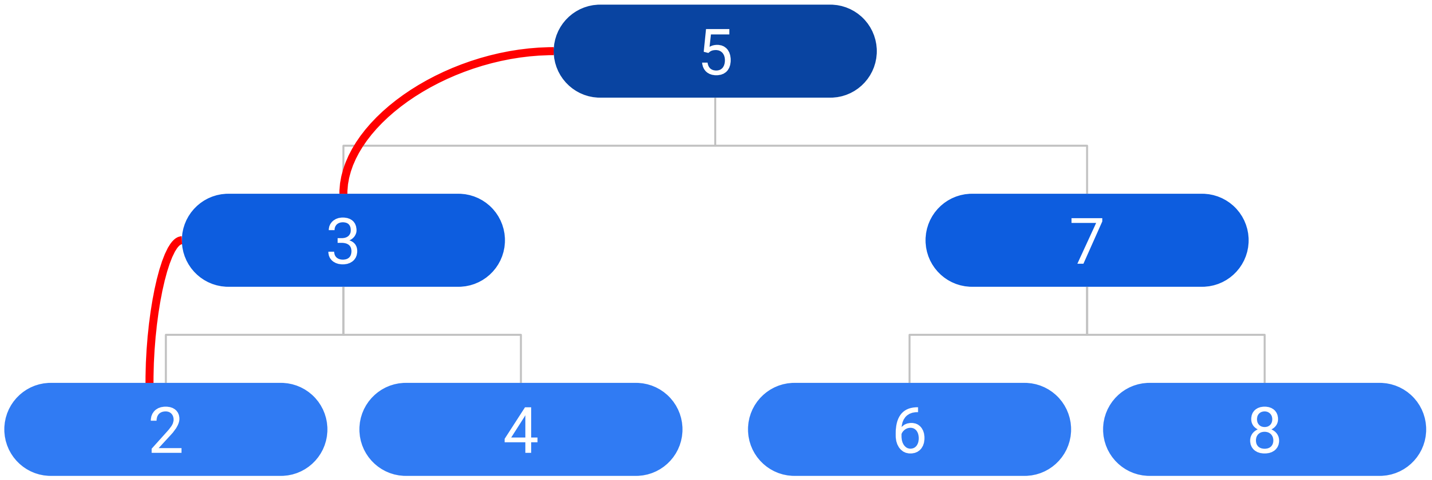


Each node has left and right child,
but they are not necessarily sorted

N-Ary Tree Node

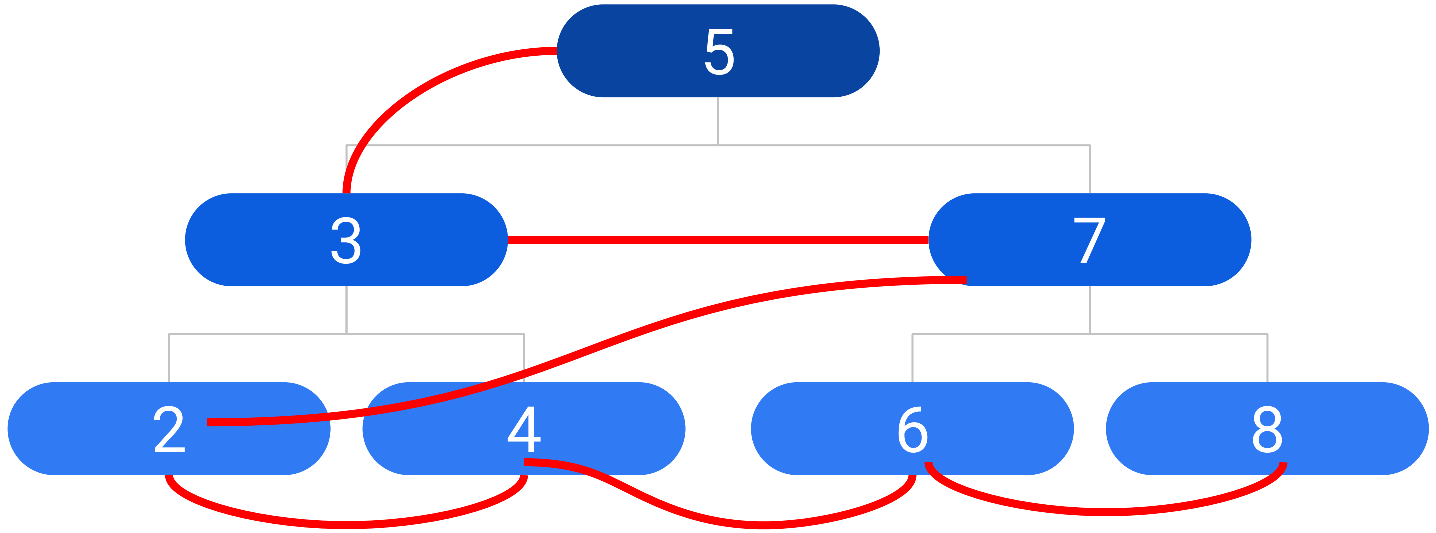
```
// N-Ary tree node would look like this:
public class Node {
    List<Node> children;
    int val;
}
```

DFS / Depth First Traversal Overview



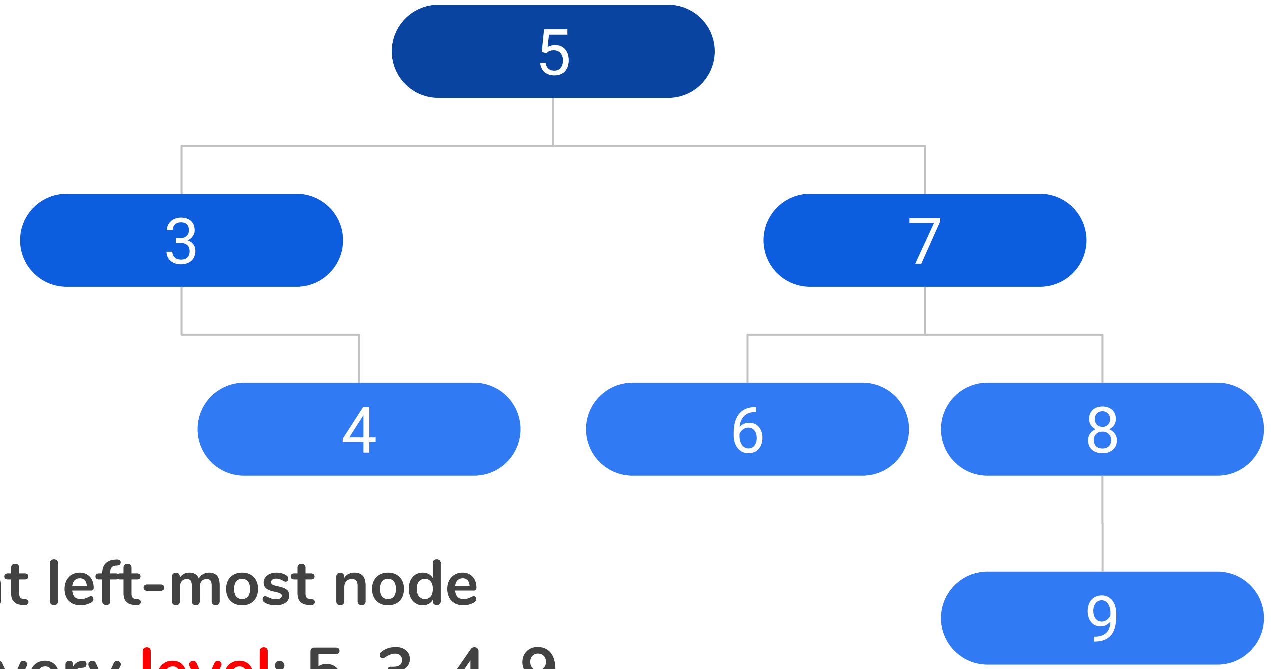
Space complexity is the height which is log n

BFS Breadth First Traversal



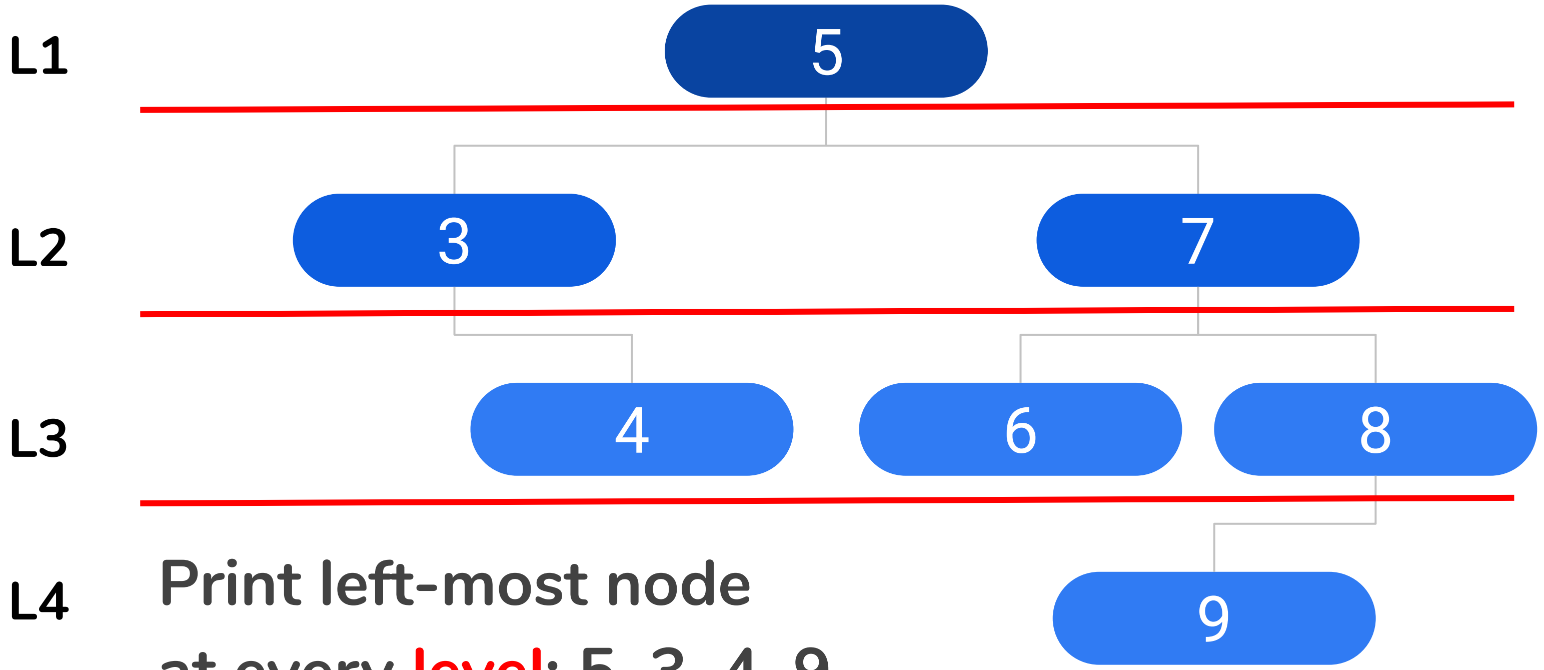
Space complexity is max level size

Example Question: BFS



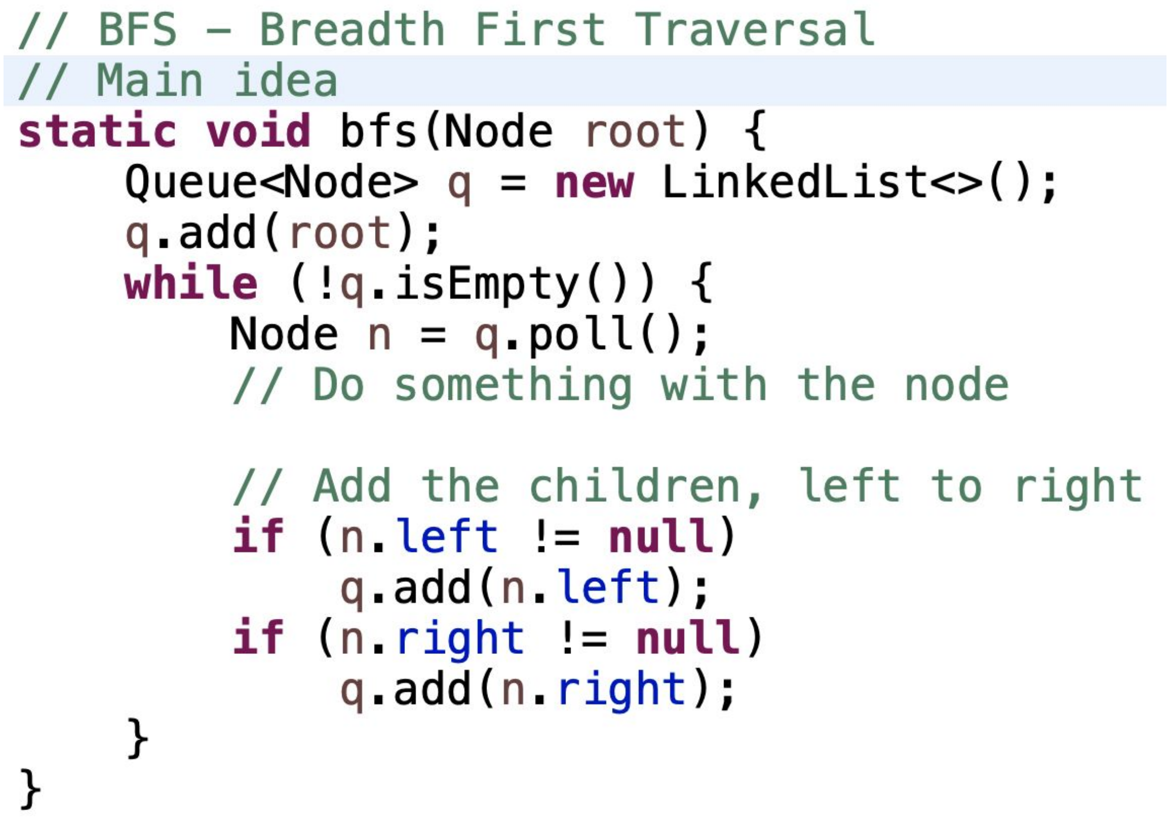
Print left-most node
at every level: 5, 3, 4, 9

Example Question: BFS



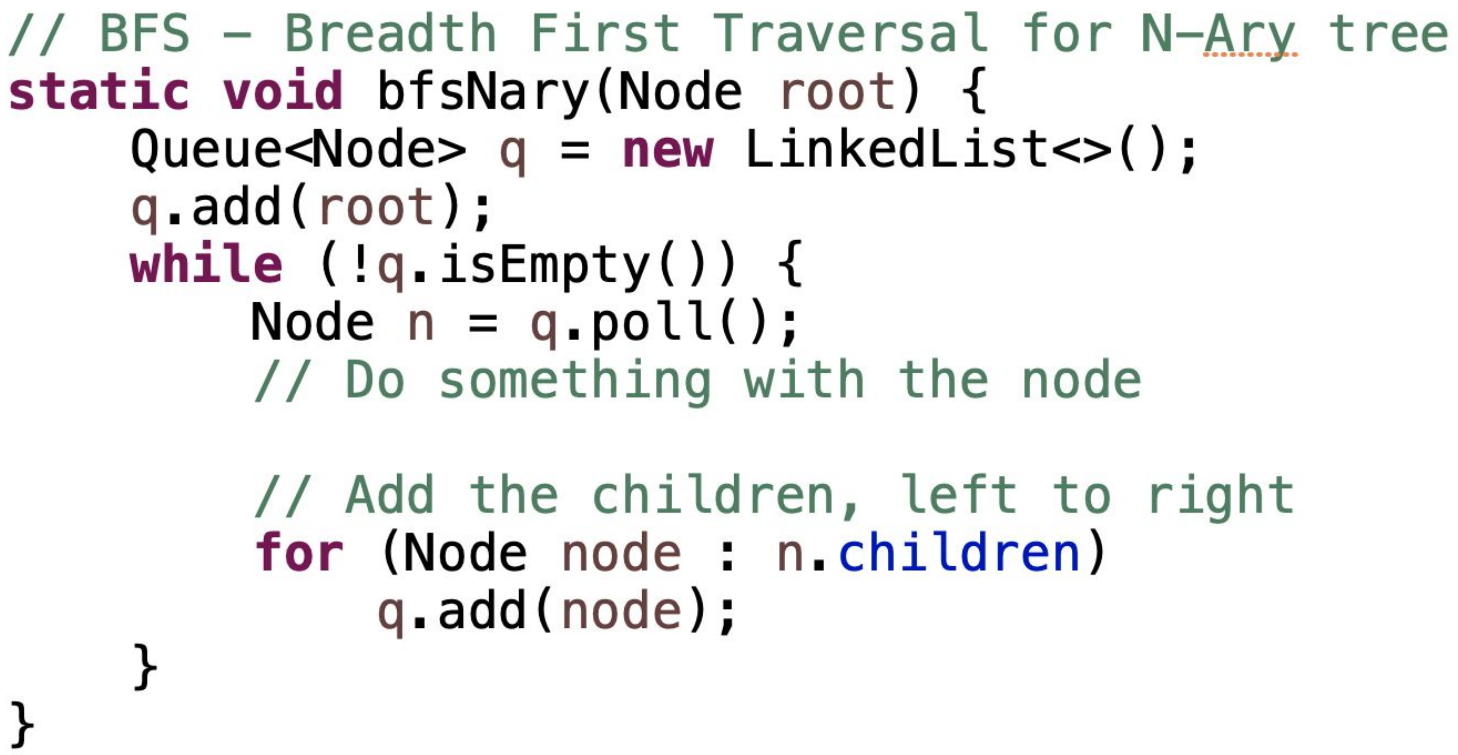
Print left-most node
at every level: 5, 3, 4, 9

BFS - Breadth First Traversal



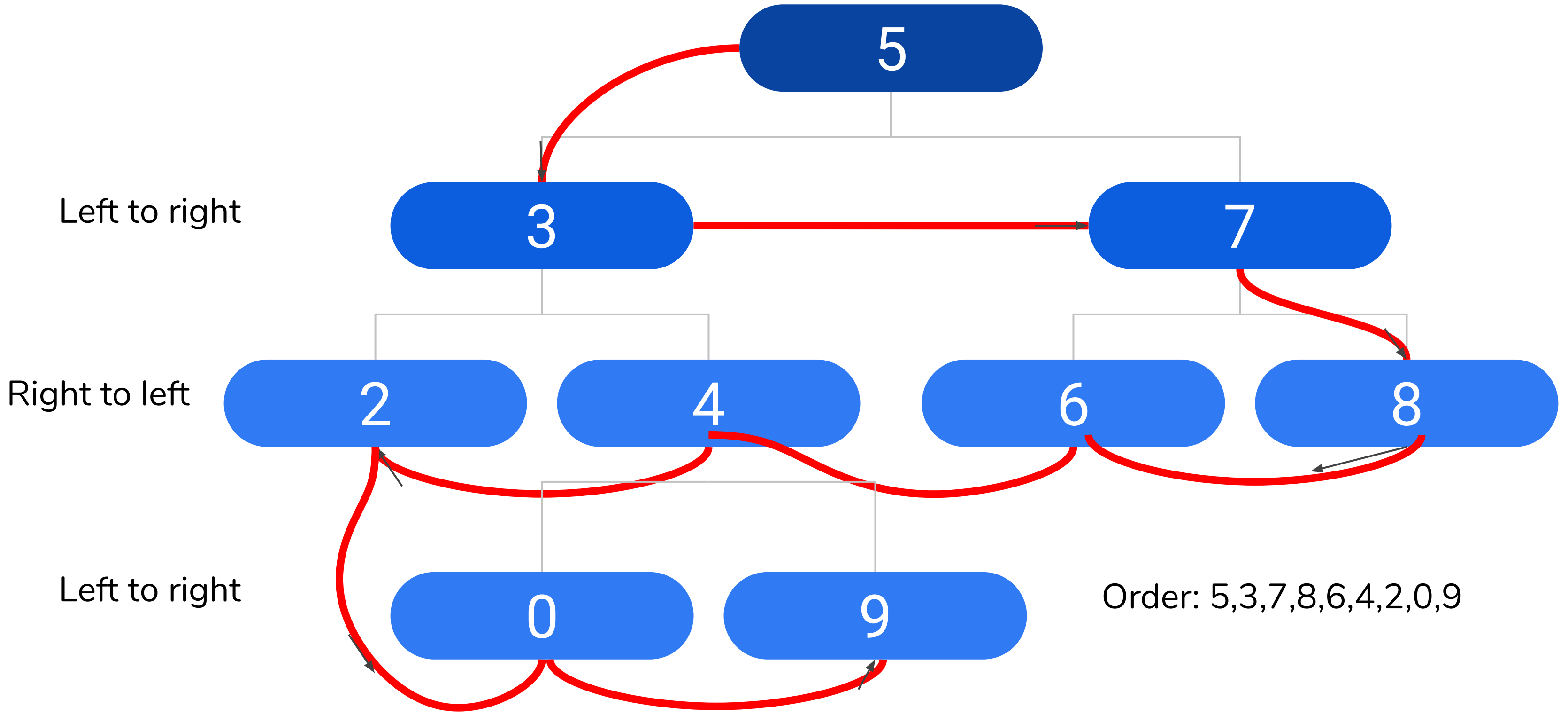
```
// BFS - Breadth First Traversal
// Main idea
static void bfs(Node root) {
    Queue<Node> q = new LinkedList<>();
    q.add(root);
    while (!q.isEmpty()) {
        Node n = q.poll();
        // Do something with the node

        // Add the children, left to right
        if (n.left != null)
            q.add(n.left);
        if (n.right != null)
            q.add(n.right);
    }
}
```

BFS - N-ary Tree



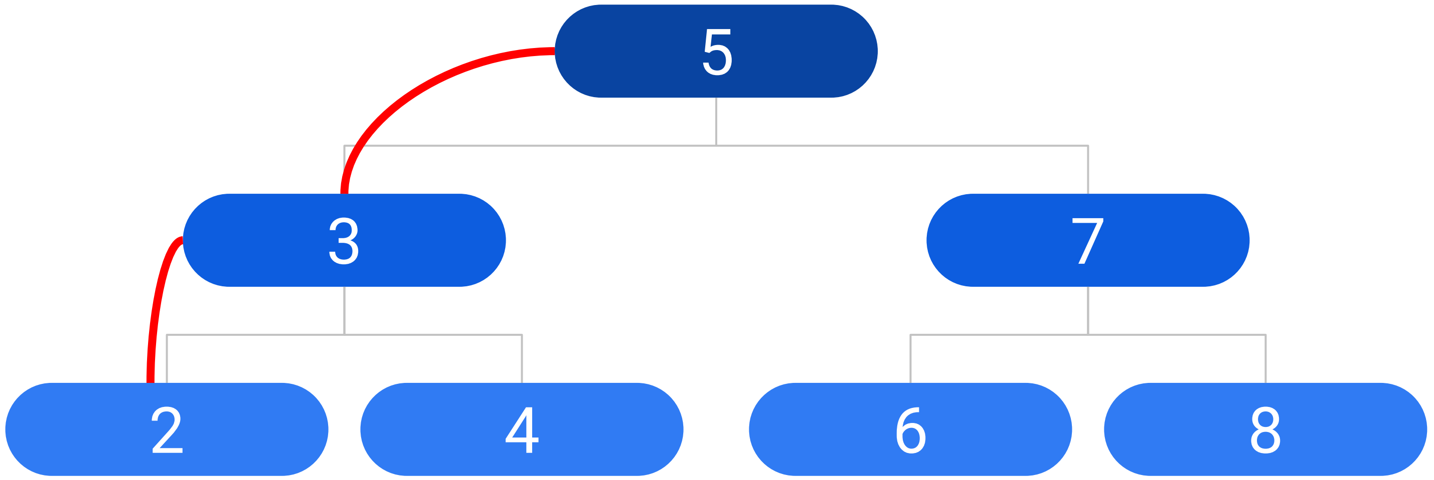
```java
// BFS - Breadth First Traversal for N-Ary tree
static void bfsNary(Node root) {
    Queue<Node> q = new LinkedList<>();
    q.add(root);
    while (!q.isEmpty()) {
        Node n = q.poll();
        // Do something with the node

        // Add the children, left to right
        for (Node node : n.children)
            q.add(node);
    }
}
```

Print Binary Tree in Zigzag Order



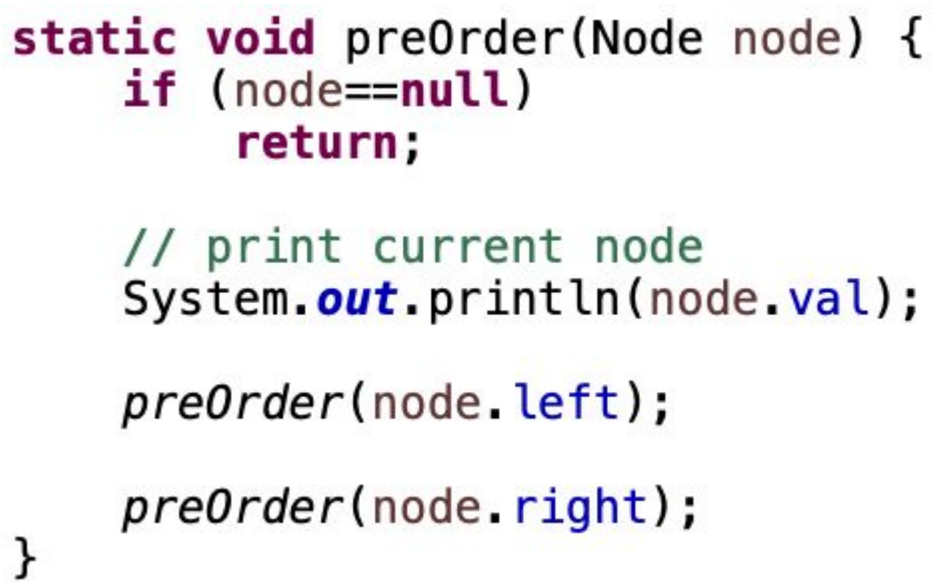
Order: 5,3,7,8,6,4,2,0,9

DFS / Depth First Traversal Overview



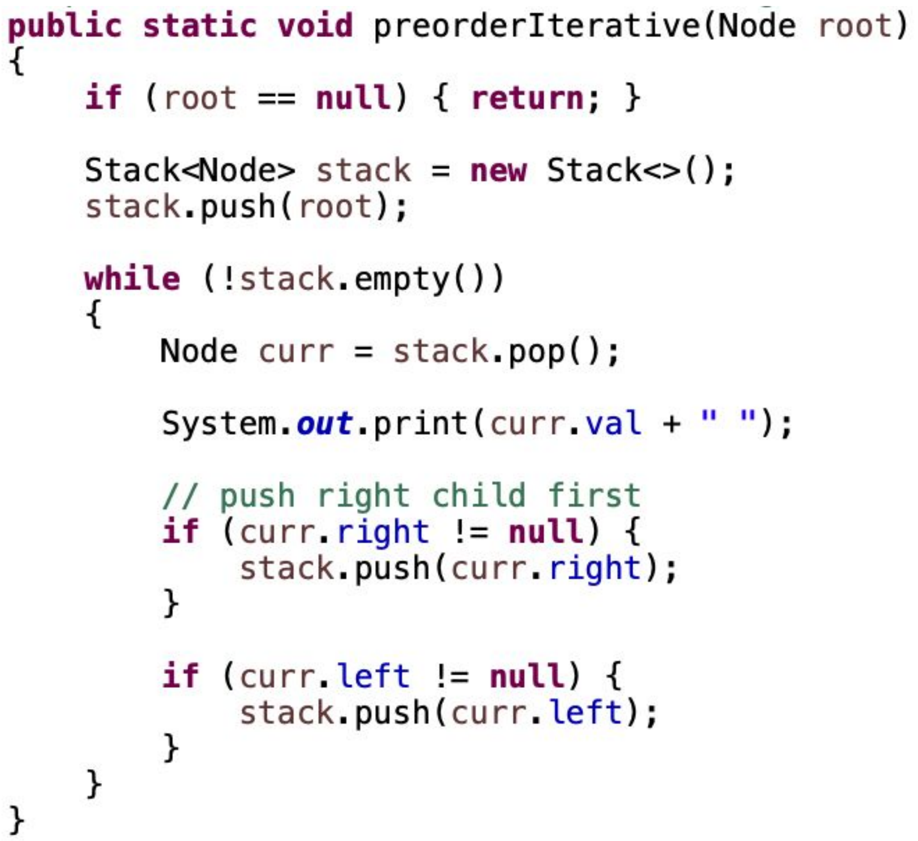
Pre-order traversal: 5, 3, 2, 4, 7, 6, 8

Pre-Order Traversal



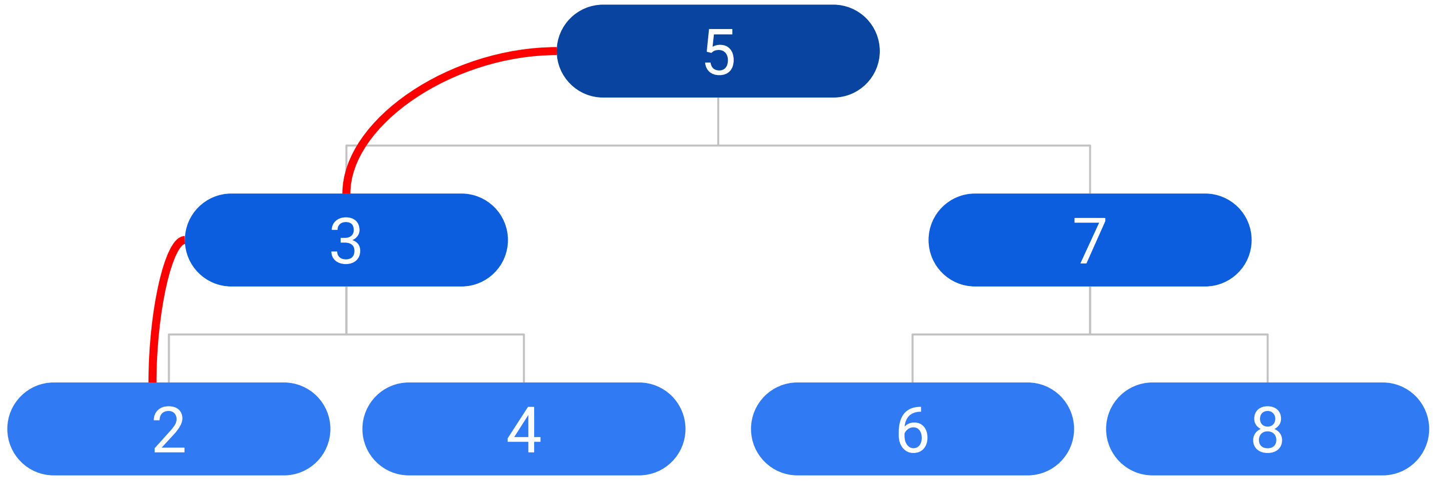
```
static void preOrder(Node node) {
    if (node==null)
        return;

    // print current node
    System.out.println(node.val);

    preOrder(node.left);

    preOrder(node.right);
}
```

Pre-Order, Iterative



```java
public static void preorderIterative(Node root)
{
    if (root == null) { return; }

    Stack<Node> stack = new Stack<>();
    stack.push(root);

    while (!stack.empty())
    {
        Node curr = stack.pop();

        System.out.print(curr.val + " ");

        // push right child first
        if (curr.right != null) {
            stack.push(curr.right);
        }

        if (curr.left != null) {
            stack.push(curr.left);
        }
    }
}
```

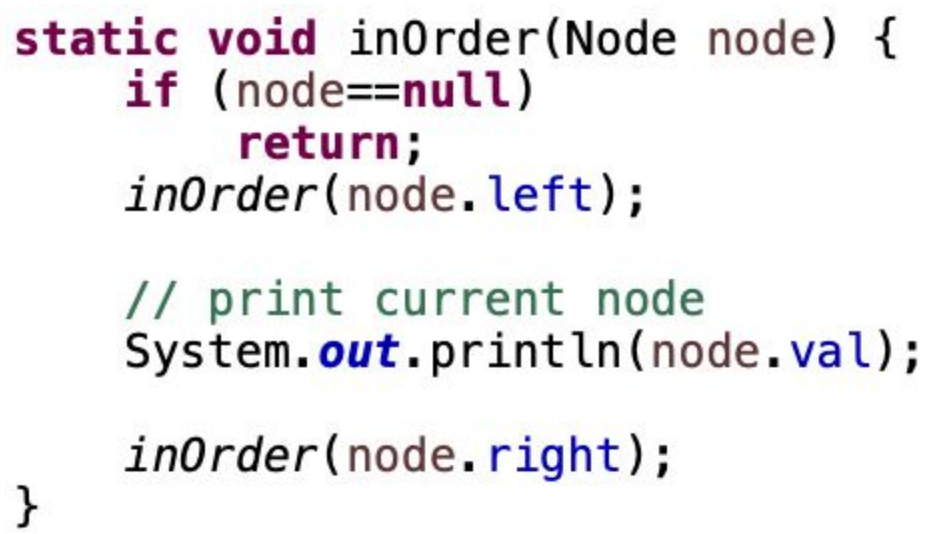
DFS / Depth First Traversal Overview



In-order traversal: 2, 3, 4, 5, 6, 7, 8

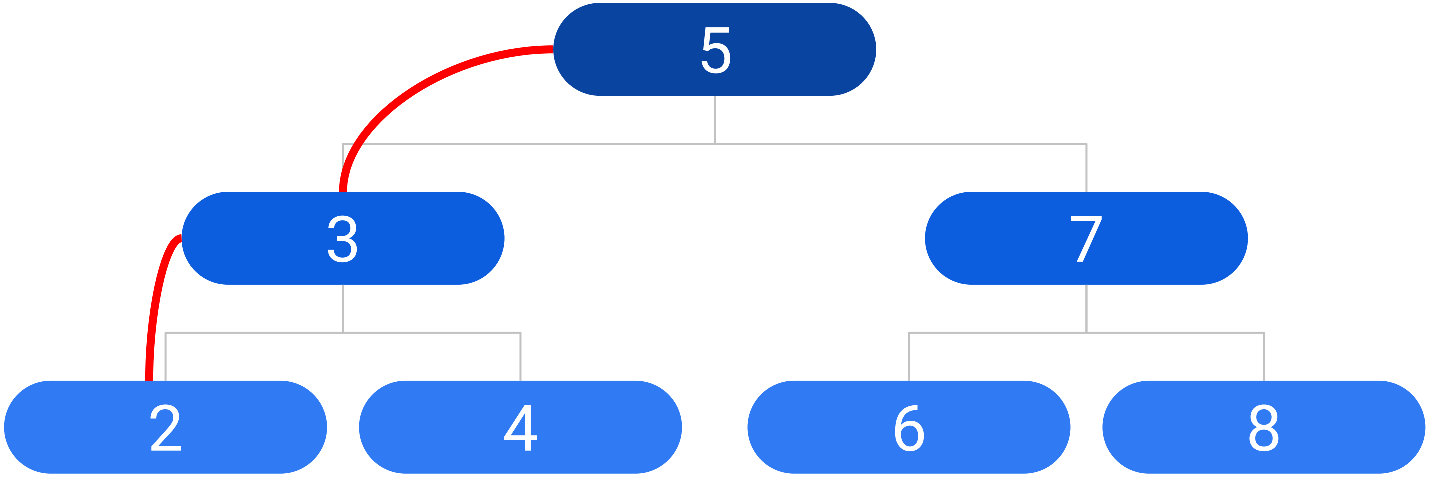
(Sorted order for BST)

In-Order Traversal



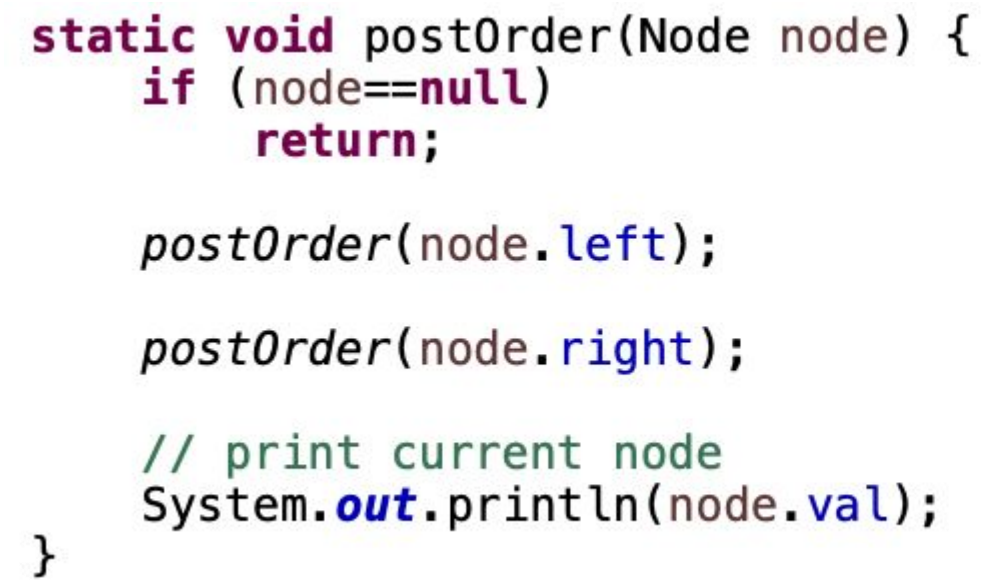
```
static void inOrder(Node node) {
    if (node==null)
        return;
    inOrder(node.left);

    // print current node
    System.out.println(node.val);

    inOrder(node.right);
}
```

DFS Depth First Search Overview



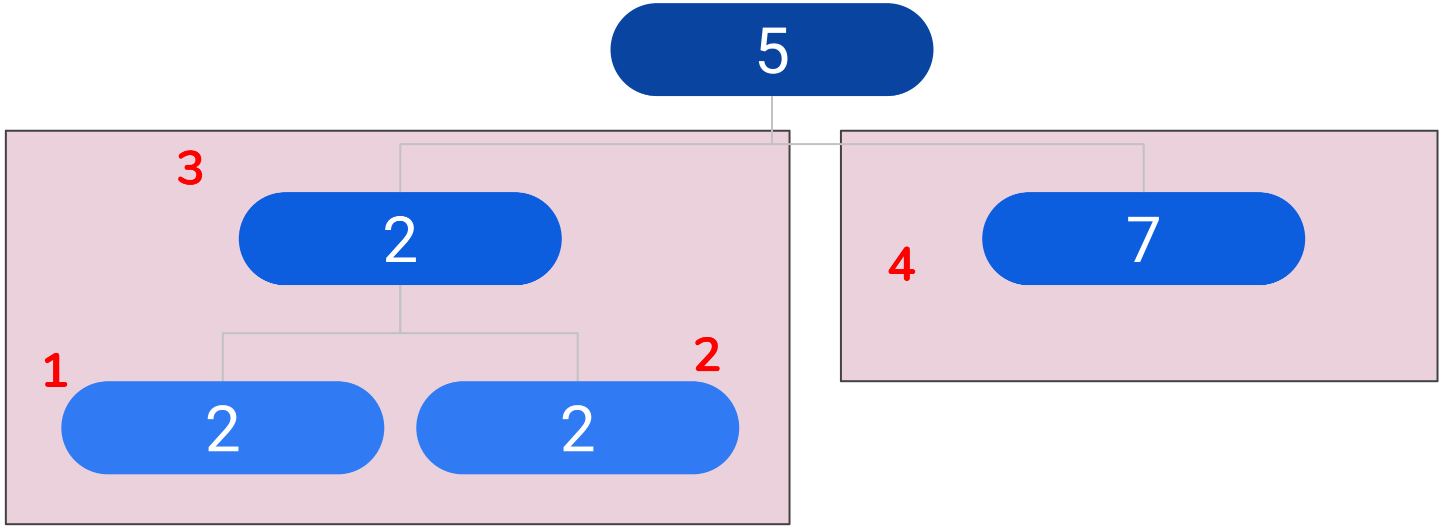
Post-order traversal: 2, 4, 3, 6, 8, 7, 5

Post-Order Traversal



```java
static void postOrder(Node node) {
    if (node==null)
        return;

    postOrder(node.left);

    postOrder(node.right);

    // print current node
    System.out.println(node.val);
}
```

Example Question - Uni-Value Trees

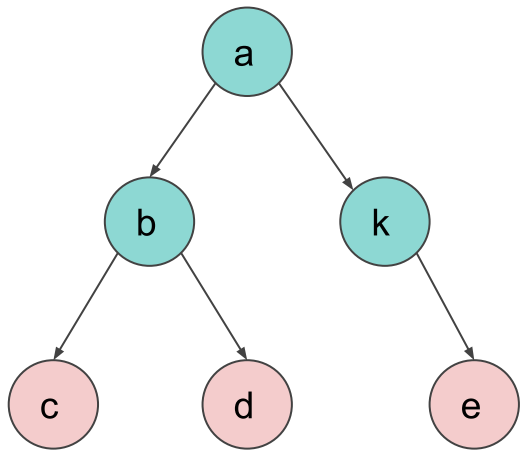


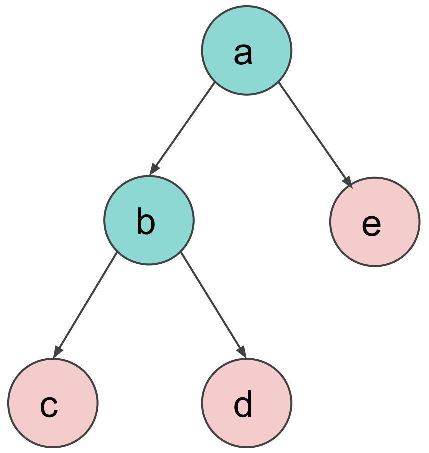
Q: Count Uni-Value Trees, Answer: 4

Example Question: DFS

Given two trees, compare their leaf nodes left to right

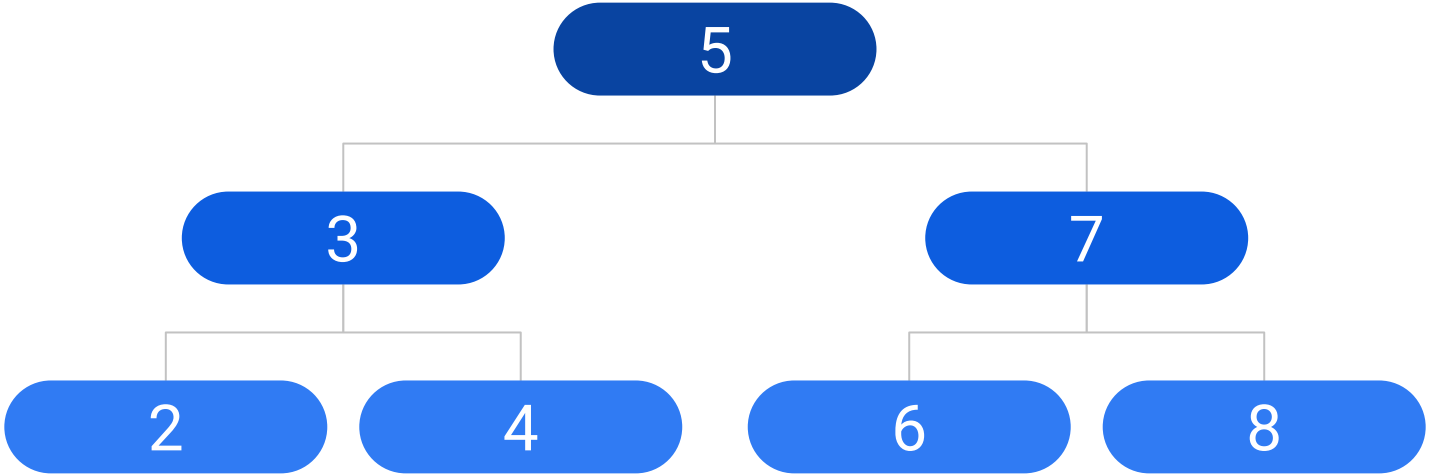
Two Trees - Compare Leaf Nodes





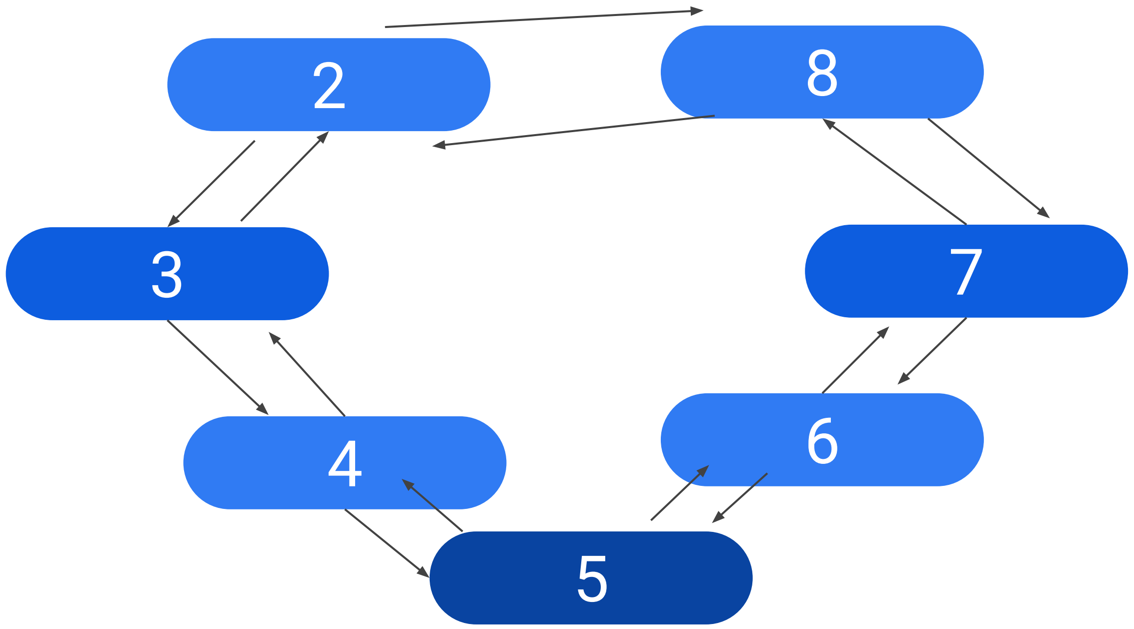
cde == cde

Example Question: BST to LinkedList

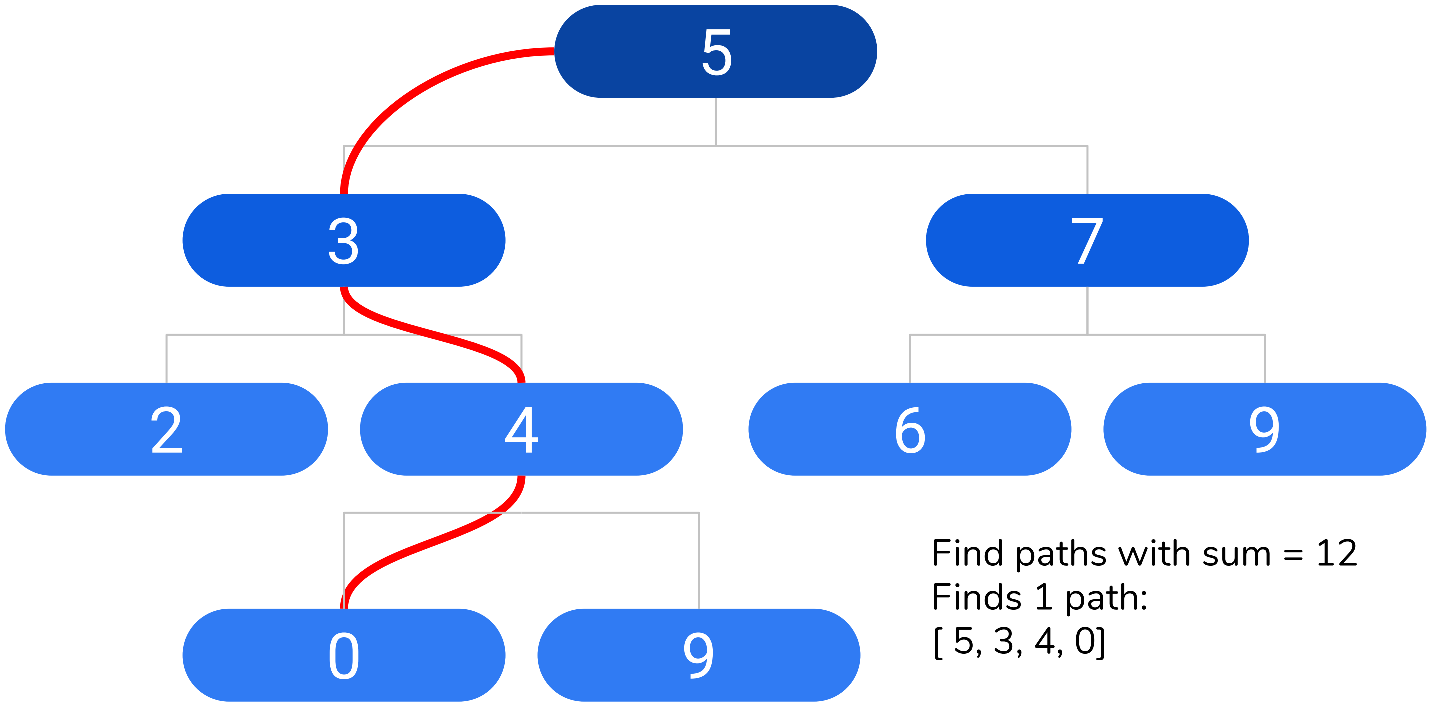


Using Node.left as previous, and Node.right as next, convert BST to doubly linked list.

Example Question: BST to LinkedList



Path Sum = N



Find paths with sum = 12
Finds 1 path:
[5, 3, 4, 0]