# **Invert Binary Tree**

**Recursive DFS**

public TreeNode invertTree(TreeNode root) {

if (root == null) return null;

// swap left and right children of root node

final TreeNode left = root.left,

right = root.right;

root.left = invertTree(right);

root.right = invertTree(left);

return root;

}

}

**Iterative DFS**

public TreeNode invertTree(TreeNode root) {

if (root == null) {

return null;

}

final Deque<TreeNode> stack = new LinkedList<>();

stack.push(root);

while(!stack.isEmpty()) {

final TreeNode node = stack.pop();

final TreeNode left = node.left;

node.left = node.right;

node.right = left;

if(node.left != null) {

stack.push(node.left);

}

if(node.right != null) {

stack.push(node.right);

}

}

return root;

}

**Iterative BFS**

public TreeNode invertTree(TreeNode root) {

if (root == null) {

return null;

}

final Queue<TreeNode> queue = new LinkedList<>();

queue.offer(root);

while(!queue.isEmpty()) {

final TreeNode node = queue.poll();

final TreeNode left = node.left;

node.left = node.right;

node.right = left;

if(node.left != null) {

queue.offer(node.left);

}

if(node.right != null) {

queue.offer(node.right);

}

}

return root;

}