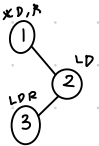


Binary Tree Inorder Traversal

: Given the root of a binary tree, return the inorder traversal of its nodes' values.

Depth-first Search (DFS)

→ Preorder : $\langle \text{root} \rangle \langle \text{left} \rangle \langle \text{right} \rangle$
→ **Inorder : $\langle \text{left} \rangle \langle \text{root} \rangle \langle \text{right} \rangle$**
→ Postorder : $\langle \text{left} \rangle \langle \text{right} \rangle \langle \text{root} \rangle$



Input: [1, null, 2, 3]

Output: [1, 3, 2]

Algorithm:

What's base case?

: leaf node

if not node:

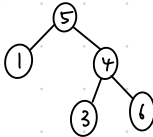
return None

left → root → right

traverse(node.left, out)

out.append(node.val)

traverse(node.right, out)



traverse(5) → traverse(1), stack(out.append(5), traverse(4))

→ traverse(1) → traverse(None), stack(out.append(1), traverse(None))

→ return None

→ out.append(1)

→ return None

→ out.append(5)

traverse(4) → traverse(3), stack(out.append(4), traverse(6))

→ traverse(3) → traverse(None), stack(out.append(3), traverse(None))

→ out.append(3)

→ traverse(None) → return None

→ out.append(4)