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

**Example 1:**



**Input:** root = [1,null,2,3]

**Output:** [1,3,2]

**Example 2:**

**Input:** root = []

**Output:** []

**Example 3:**

**Input:** root = [1]

**Output:** [1]

**Example 4:**



**Input:** root = [1,2]

**Output:** [2,1]

**Example 5:**



**Input:** root = [1,null,2]

**Output:** [1,2]

**----------------------------------------------------------------------------------**

**### 1. Binary Tree Inorder Traversal**

Source: [LeetCode]( https://leetcode.com/problems/binary-tree-inorder-traversal/

**#### Solution**

We can make our In Order Traversal container as a List of Integers. Then, we’ll create a helper method which will traverse the tree in order and add the values to the list. Our InOrderWalk helper method takes in a List of Integers and a TreeNode.

Our recursive method’s break case is if the current TreeNode is null, stop running recursion. Otherwise, check left child, add the value of current node, then check right child. Our List which goes In Order will be returned.

```public List<Integer> inorderTraversal(TreeNode root) {

List<Integer> l = new ArrayList<Integer>();

inOrderWalk(l, root);

return l;

}

public void inOrderWalk(List<Integer> l, TreeNode curr)

{

if(curr == null) {

return;

}

inOrderWalk(l, curr.left);

l.add(curr.val);

inOrderWalk(l, curr.right);

}

}```

**#### Driver For Solution**

[1,null,2,3] : [1,3,2]

[1,null,2] : [1,2]

<!-- Don't remove -->

Go to [Top](#top)