InOrder Morris Traversal

```
def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        if root is None:
            return root
        ans = []
        curr = root
        while curr!=None:
            leftNode = curr.left
            if leftNode is None:
                ans.append(curr.val)
                curr = curr.right
            else:
                rightMost = self.getRightMostNode(leftNode,curr)

                if rightMost.right is None:
                    rightMost.right = curr
                    curr = curr.left
                else:
                    rightMost.right= None
                    ans.append(curr.val)
                    curr = curr.right
        return ans

    def getRightMostNode(self,leftNode,curr):
        while leftNode.right!=None and leftNode.right!=curr:
            leftNode = leftNode.right
        return leftNode
```

Time Complexity : O(n) If we take a closer look, we can notice that every edge of the tree is traversed at most three times. And in the worst case, the same number of extra edges (as input tree) are created and removed.