**94.Binary Tree Inorder Traversal**

方法一：递归，这是最朴素简单的方法，时间复杂度O(n)，空间复杂度O(n)。

方法二：利用栈进行迭代。递归的时候隐式地维护了一个栈，而我们在迭代的时候需要显式地将这个栈模拟出来。思想：若节点还有左子树，就要先把左子树访问完；没有左子树可访问时，访问该节点，并尝试访问右子树。详细步骤见代码。时间复杂度O(n)，空间复杂度O(n)。

方法三：Morris遍历。Morris遍历算法整体步骤如下（假设当前遍历到的节点为x）：

1.如果x无左孩子，先将x的值加入答案数组，再访问x的右孩子，即x=x.right。

2.如果x有左孩子，则找到x左子树上最右的节点（即左子树中序遍历的最后一个节点，x在中序遍历中的前驱节点），我们记为predecessor。根据predecessor的右孩子是否为空，进行如下操作。

A.如果predecessor的右孩子为空，则将其右孩子指向x，然后访问x的左孩子，即x=x.left。

B.如果predecessor的右孩子不为空，则此时其右孩子指向x，说明我们已经遍历完x的左子树，我们将predecessor的右孩子置空，将x的值加入答案数组，然后访问x的右孩子，即 x=x.right。

3.重复上述操作，直至访问完整棵树。

其实整个过程我们就多做一步：假设当前遍历到的节点为x，将x的左子树中最右边的节点的右孩子指向x，这样在左子树遍历完成后我们通过这个指向走回了x，且能通过这个指向知晓我们已经遍历完成了左子树，而不用再通过栈来维护，省去了栈的空间复杂度。时间复杂度O(n)，空间复杂度O(1)。

