三种非递归遍历算法（栈）

中序

根据中序遍历的顺序，对于任一结点，优先访问其左孩子，而左孩子结点又可以看做一根结点，然后继续访问其左孩子结点，直到遇到左孩子结点为空的结点才进行访问，然后按相同的规则访问其右子树。

其处理过程如下：

对于任一结点Node：

1）遍历左子树，将node入栈，直至左子树为空

2）若其左孩子为空，则取栈顶元素并进行出栈操作，访问该栈顶结点，将该节点的值压入结果数组中，然后将当前的Node置为栈顶结点的右孩子；

3）直到Node为NULL并且栈为空则遍历结束。

前序

根据前序遍历访问的顺序，优先访问根结点，然后再分别访问左孩子和右孩子。即对于任一结点，其可看做是根结点，因此可以直接访问，访问完之后，若其左孩子不为空，按相同规则访问它的左子树；当访问其左子树时，再访问它的右子树。

其处理过程如下：

对于任一结点Node：

1）访问结点Node，将该节点的值压入结果数组中，并将结点Node入栈;

2）遍历左子树，直至左孩子为空，取栈顶结点，并将栈顶结点的右孩子置为当前的结点Node，循环至1);

3）直到Node为NULL并且栈为空，则遍历结束。

后序

-在后序遍历中，要保证左孩子和右孩子都已被访问并且左孩子在右孩子前访问才能访问根结点。

其处理过程如下：

对于任一结点Node：

1）将其入栈，遍历左子树，直到搜索到没有左孩子的结点，此时该结点出现在栈顶，但是此时不能将其出栈并访问， 因此其右孩子还未被访问

2）所以接下来按照相同的规则对其右子树进行相同的处理，判断其右子树是否为空或者右子树是否是最近访问的节点，否则将Node置为其右结点

3）当其没有右子树时，或者节点的右子树节点是最近访问过的节点（表示右子树已经访问过了），则将节点的值压入在结果数组中