常用数据结构及相关算法实现复盘

顺序栈 解决 回文判断问题

链栈 解决 括号匹配问题

循环双链表 解决 线性表对称问题

--🡪 返回值 bool 思路：假设问题成立，一旦出现不成立情况，立即判断为false，退出循环。

顺序队列 解决 迷宫路径问题 思路：广度优先遍历、最短路径

循环链表：初始化空表时，头结点指针域指向自身 🡪 统一所有插入操作

插入条件改为：j == i 🡪 位置查找循环结束时，如果j==i，插入位置要么在L之前，要么刚好是L的位置，而此时L相当于非循环链表的NULL的位置，因此可以插入

删除条件改为： p ！= L 🡪 删除找要删除的位置而不是前驱 –> 统一操作

二叉树的非递归遍历：

后序遍历：

思路：左子树、右子树、根

因此算法分两步：

1. 找到最左下结点，其左孩子为NULL，此时可以认为最左下结点的左子树已被访问 🡪 统一最左下结点和其他根结点的情况
2. 判断栈顶结点的右子树是否已被访问，因为对于任意一棵树，后序遍历最后访问的是根结点，因此只需判断当前结点（栈顶结点）的右孩子是否已经被访问（同理，如果右孩子为空，则可以认为右孩子已被访问 🡪 统一有或没有右子树的情况），1)如果已访问，则访问栈顶结点 –> 出栈，此时栈顶结点的左孩子（刚刚出栈的结点）已被访问，因此继续判断当前栈顶结点的右孩子是否已被访问（或者为空）；2)如果未访问，那么对右子树执行算法步骤1。

算法执行以上两步，直至栈空（算法是访问完根结点就将其出栈，而栈存放的是某棵树的根结点，因此当最后一个根结点访问完，然后将其出栈，那么树就遍历完了） 🡪 后序遍历完成

中序遍历：

思路：左子树 🡪 根 🡪 右子树

辅助数据结构：栈

算法分两步：

1. 从根结点开始找到最左下结点（边找边进栈），此时最左下结点的左孩子为NULL，即可认为最左下结点的左子树已被访问过，此时最左下结点可以看做是有一棵空左子树的根结点。
2. 如果栈不空，出栈（算法最开始的时候访问的是第一步找到的最左下结点），访问结点，此时可认为以栈顶结点作为根的子树的根结点已被访问，然后对出栈结点的右孩子进行第一步的操作。当某棵子树的最右下结点为空时，那么这棵子树遍历完（因为左、根、右），此时如果这棵子树是其根结点的左子树，那么栈顶结点就是它的根结点，继续访问它的根结点。

算法执行以上两步，直到栈为空并且指向当前结点的指针为空。