顺序表：

1、顺序表递增有序，插入元素 x，仍递增有序
2、用顺序表最后一个元素覆盖整个顺序表中最小元素，并返回该最小元素
3、将顺序表中的元素逆置
4、将（a1,a2,a3……am,b1,b2,……bn）转换成(b1,b2,……bn,a1,a2,a3,……am)
5、删除顺序表中所有值为 x 的元素(两种方法)
6、从顺序表中删除给定值在 s 到 t 之间（包含 s 和 t）的所有元素
7、从有序表中删除所有值重复的元素
8、两个递增有序表合并成一个递增有序表
9、求两个递增序列合并后的中位数(两种方法)
10、设计一个时间上尽可能高效的算法，找出数组中未出现的最小正整数
11、若一个整数序列中有过半相同元素，则称其为主元素，设计算法找出数组 A(a0,a1……an-1)的主元素。（其中 0<ai<n）若存在主元素则输出，否则返回-1

链表：

1、设计一个递归算法，删除不带头节点的单链表 L 中所有值为 x 的结点
2、删除带头节点单链表中所有值为 x 的结点
3、删除带头节点单链表中第一个值为 x 的结点
4、从尾到头反向输出单链表每个结点的值
5、试编写算法将单链表就地逆置
6、从链表中删除给定值在 s 到 t 之间（不包含 s 和 t）的所有元素
7、试编写在带头结点的单链表 L 中删除最小值点的高效算法（已知最小值唯一）
8、试编写在不带头结点的单链表 L 中删除最小值点的高效算法（已知最小值唯一）
9、给定一个单链表，按递增排序输出的单链表中各结点的数据元素，并释放节点所占空间
10、将一个带头节点的单链表 A 分解成两个带头节点的单链表 A 和 B，使 A 中含奇数位置元素，B 中含偶数位置元素，且相对位置不变
11、将一个单链表{a1,b1,a2,b2……an,bn}拆分成{ a1,a2……an }和{ bn,bn-1,……b1 }
12、删除递增链表中重复的元素
13、两个递增有序的单链表，设计算法成一个非递减有序的链表
14、两个递增有序的单链表，设计算法成一个非递增有序的链表
15、A, B 两个单链表递增有序，从 A，B 中找出公共元素产生单链表 C，要求不破坏 A，B 结点
16、A, B 两个单链表递增有序，从 A，B 中找出公共元素并存放于 A 链表中
17、两个序列分别为 A、B，将其存放到链表中，判断 B 是否是 A 的连续子序列
18、查找单链表中倒数第 k 个结点，若成功，则输出该节点的 data，并返回 1，否则返回 0
19、用单链表保存 m 个整数，并且|data|≤n，要求设计时间复杂度尽可能高效的算法，对于 data 绝对值相等的点，仅保留第一次出现的点
20、判断带头结点的循环双链表是否对称
21、有两个循环单链表，链表头指针分别为 h1,h2,试编写函数将 h2 链表接到 h1

之后，要求链接后仍保持循环链表形式
22、设有一个带头结点的循环单链表，其结点值为正整数，设计算法反复找出链表内最小值并不断输出，并将结点从链表中删除，直到链表为空，再删除表头结点
23、判断单链表是否有环
24、给定一个单链表 L(a1, a2, a3 ,..., an),将其重新排列为(a1, an, a2, an-1 ,...)

栈、队列、字符串

1、Q 是一个队列，S 是一个空栈，编写算法使得队列中元素逆置
2、判断单链表的全部 n 个字符是否中心对称
3、两个栈 s1,s2 都采用顺序存储，并共享一个存储区[0,...,maxsize-1]。采用栈顶相向，迎面增长的存储方式，设计 s1,s2 入栈和出栈的操作。
4、判断一个表达式中括号是否配对(假设只包含圆括号)
5、假设一个序列为 HSSHHHS，运用栈的知识，编写算法将 S 全部提到 H 之前，即为 SSSHHHH
6、利用一个栈实现以下递归函数的非递归计算
7、KMP 算法

树

1、计算二叉树中所有结点个数
2、计算二叉树中所有叶子节点的个数
3、计算二叉树中所有双分支的节点个数
4、计算二叉树的深度
5、(a-(b+c))*(d/e)存储在二叉树，遍历求值
6、找出二叉树中最大值的点
7、判断两个二叉树是否相似（指都为空或者都只有一个根节点，或者左右子树都相似）
8、把二叉树所有节点左右子树交换
9、查找二叉树中 data 域等于 key 的结点是否存在，若存在，将 q 指向它，否则 q 为空
10、输出先序遍历第 k 个结点的值
11、求二叉树中值为 x 的层号
12、树中元素为 x 的结点，删除以它为根的子树
13、利用结点的右孩子指针将一个二叉树的叶子节点从左向右连接成一个单链表（head 指向第一个，tail 指向最后一个）
14、输出根节点到每个叶子结点的路径
15、已知满二叉树先序序列存在于数组中，设计算法将其变成后序序列
16、先序与中序遍历分别存在两个一维数组 A，B 中，试着建立二叉链表
17、二叉树以顺序方式存在于数组 A 的中，设计算法以二叉链表表示
18、增加一个指向双亲节点的 parent 指针，输出所有节点到根节点的路径
19、先序非递归遍历二叉树
20、中序非递归遍历二叉树
21、后序非递归遍历二叉树

22、在二叉树中查找值为 x 的结点，打印出值为 x 的所有祖先
23、找到 p 和 q 最近公共祖先结点 r
24、层次遍历
25、试给出自下而上从右到左的层次遍历
26、求解二叉树的宽度
27、用层次遍历求解二叉树的高度
28、判断二叉树是否为完全二叉树
29、计算二叉树的带权路径长度（叶子节点）
30、将给定的二叉树转化为等价的中缀表达式
31、建立中序线索二叉树
32、中序遍历线索二叉树
33、先序建立二叉搜索树并先序遍历
34、寻找中序线索二叉树的前驱结点
35、用孩子兄弟表示法求树所有叶子结点个数
36、用孩子兄弟表示法求树的高度
37、已知一棵树的层次序列和每个节点的度，编写算法构造此树的孩子兄弟链表。

查找

1、顺序表递增有序，设计算法在最少的时间内查找值为 x 的元素。若找到，则将其于后继元素位置交换，否则按照递增顺序插入顺序表
2、找到单链表中值为 key 的元素，将其与前一个结点位置互换。
3、在顺序表中二分查找值为 key 的元素
4、判断给定二叉树是否是二叉(搜索)排序树
5、寻找二叉排序树中最大值和最小值
6、设求出指定结点在给定二叉排序树的层次
7、输出二叉搜索树中所有值大于 key 的结点
8、判断一个二叉树是否为平衡二叉树

图

头插法建立图(邻接表结构体)
1、已知无向连通图 G 由顶点集 V 和边集 E 组成，|E|>0,当 G 中度为奇数的顶点个数为不大于 2 的偶数时，G 存在包含所有边且长度为|E|的路径（称为 EL 路径）。设图 G 采用邻接矩阵存储，设计算法判断图中是否存在 EL 路径，若存在返回 1，否则返回 0。
2、图的广度优先遍历（BFS）
3、利用 BFS 求无向图的最短路径
4、图的深度优先遍历（DFS)
5、图采用邻接表存储，设计算法判断 i 和 j 结点之间是否有路径（以下全是邻接表存储）
6、设计算法判断无向图是否是一棵树
7、设计算法，求无向连通图距顶点 v 最远的一个结点（即路径长度最大）
8、写出深度优先遍历的非递归算法
9、输出无向图点 u 到点 v 的所有路径

10、求无向图的连通分量个数
11、邻接表转化成邻接矩阵
12、判断无向图是否存在环 (并查集)
13、邻接矩阵转化成邻接表：

排序

1、直接插入排序
2、折半插入排序
3、希尔排序
4、冒泡排序
5、快速排序
6、选择排序
7、堆排序
8、归并排序