1、简述下列术语，数据，数据元素，数据项，数据对象，数据类型，数据结构，逻辑结构、存储结构

（1）数据：数据是信息的载体，是描述客观事物属性的数、字符及所有能输入到计算机中并被计算机程序识别和处理的符号的集合。

（2）数据元素：数据元素是数据的基本单位，通常作为一个整体进行考虑和处理。

（3）数据项：数据项是构成数据元素的不可分割的最小单位。

（4）数据对象：数据对象是具有相同性质的数据元素的集合，是数据的一个子集。

（5）数据类型：数据类型是一个值的集合和定义在此集合上的一组操作的总称。包括原子类型。结构类型。抽象数据类型。

（6）数据结构：相互之间存在一种或多种特定关系的数据元素的集合；数据结构包括：逻辑结构、存储结构和数据的运算。

（7）逻辑结构：逻辑结构是指数据元素之间的逻辑关系。

（8）存储结构：存储结构是指数据结构在计算机中的表示（又称映像），也称物理结构。

2、简述说明数据的存储结构：（关注：顺序和链式存储的异同）

存储结构是指数据结构在计算机中的表示（又称映像），也称物理结构。

（1）顺序存储：逻辑上相邻的两个元素的物理位置也相邻。

优点：能够随机存取。每个元素占用最少的存储空间

缺点：插入删除需要移动大量的元素，不方便。只能使用相邻的一整块存储单元，因此可能产生较多的外部碎片。应用于频繁查询元素时使用。

（2）链式存储：逻辑上相邻的两个元素的物理位置不一定相邻，每个结点用一个指针来找到下一个结点的位置。

优点：插入和删除很方便；不会出现碎片现象，充分利用所有存储单元。

缺点：每个元素因存储指针而占用额外的存储空间。只能实现顺序存取，需要从第一个结点开始遍历；

应用于频繁的插入、删除、更新元素时。

（3）索引存储：在存储时，还附加建立索引表，索引表中的每一项称为索引项，索引项的一般形式是（关键字，地址）

优点：检索速度快

缺点：索引表占用存储空间，并且插入和删除一个数据时，对应的索引项也要插入和删除，会耗费较多的时间

（4）哈希存储：通过函数，根据数据的元素的关键字计算该元素的地址

优点：检索、增加和删除结点的操作比较快

缺点：可能会出现元素存储单元的冲突，解决冲突又需要增加时间和空间的开销。

3、算法的 5 个重要特性：

①有穷性。一个算法必须总在执行有穷步之后结束，且每一步都可在有穷时间内完成。（程序可以无穷，但算法要有穷性）

②确定性。算法中每条指令必须有确切的含义，对于相同的输入只能得出相同的输出。

③可行性。算法是可行的。算法描述的操作都可以通过已经实现的基本运算执行有限次来实现。

④输入。一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。

⑤输出。一个算法有一个或多个输出，这些输出是与输入有着某种特定关系的量。

4、"好"的算法的特性（真题考过）

①正确性。算法应能够正确地解决求解问题。

②可读性。算法应具有良好的可读性，算法描述清晰易懂，以帮助人们理解。

③健壮性。输入非法数据时，算法能适当地做出反应或进行处理，而不会产生莫名其妙的输出结果。

④效率与低存储量需求。效率是指算法执行的时间，存储量需求是指算法执行过程中所需要的最大存储空间，这两者都与问题的规模有关。

5、线性表、栈和队列三者异同及应用（真题考过两次）

（1）线性表、栈和队列都属于逻辑结构中的线性结构，都有线性结构的特点。在存储上，都可以用顺序存储或链式存储。

（2）线性结构的特点是在数据元素的非空有限集合中：除第一个元素外，每个元素有且仅有一个直接前驱。除最后一个元素外，每个元素有且仅有一个直接后继。

（3）栈和队列相同点

栈和队列是两种特殊的线性表，对插入、删除运算加以限制即受限的线性表。都可以通过顺序结构和链式结构实现。插入与删除的时间复杂度都是 O(1)，在空间复杂度两者相同。

（4）栈和队列不同点

①栈是限定仅能在表尾（称为栈顶）一端进行插入、删除操作的线性表。因而栈的特性是后进先出（LIFO）。栈常见的应用有括号匹配，表达式求值和递归。

②队列是限定仅能在表头（称为队头）进行删除，表尾（称为队尾）进行插入的线性表。因而队列的特性是先进先出（FIFO）。

队列常见的应用有层次遍历，解决主机与外部设备之间速度不匹配的问题，解决由多用户引起的资源竞争问题。

6、顺序表和链表的比较

	顺序表	链表
存储方式	采用顺序存储时，逻辑上相邻的元素，对应的物理存储位置也相邻。	采用链式存储时，逻辑上相邻的元素，物理存储位置不一定相邻，对应的逻辑关系是通过指针链接来表示的。
存取（读写）方式	可随机存取，也可顺序存取	顺序存取
插入、删除	插入删除操作麻烦，需移动大量元素 时间复杂度为$O(n)$，时间开销主要来自移动元素。	插入删除操作只需要修改指针 时间复杂度为$O(n)$，时间开销主要来自查找目标元素。
查找	按位查找：$O(1)$ 按值查找：无序时为$O(n)$ 有序时可用折半查找，为$O(\log_2 n)$	按位查找：$O(n)$ 按值查找：$O(n)$
存储空间	预先分配，会导致空间闲置或溢出现象	动态分配，不会出现空间闲置或溢出现象
适用情况	①表长变化不大，且能事先确定变化的范围　②很少进行插入或删除操作，经常按元素位置序号访问数据元素。	①表的容量会动态变化 ②频繁进行插入或删除操作。

7、线性表有两种存储结构：一是顺序表，二是链表，试问:

（1）如果有多个线性表同时共存，并且在处理过程中各表的长度会动态地发生变化，线性表的总数也会自动地改变。在此情况下,应选用哪种存储结构?为什么?

（2）若线性表的总数基本稳定且很少进行插人和删除操作，但要求以最快的速度存取线性表中的元素，那么应采用哪种存取结构?为什么?

（1）应选用链式存储结构。

由于链式存储结构可以用任意的存储空间来存储线性表中的各数据元素，且其存储空间可以是连续的,也可以不连续；此外，这种存储结构对元素进行插人和删除操作时都无须移动元素，修改指针即可，所以很适用于线性表容量变化的情况，这种动态存储结构适合于多个线性表同时共存。

（2）应选用顺序存储结构。

若表的总数基本稳定，且很少进行插入和删除，则顺序表可以充分发挥其存取速度快、存储利用率高的优点。

8、头指针、头结点、首元结点区别；引入头结点的优点（真题考过）

（1）不管带不带头结点，头指针都始终指向链表的第一个结点，而头结点是带头结点的链表中的第一个结点，结点内通常不存储信息。首元素结点是指链表中存储线性表中第一个元素结点。

（2）引入头结点的优点：

①便于首元结点的处理：

由于第 1 个数据结点的位置被存放在头结点的指针域中，所以链表的第一个位置上的操作和其他位置上的操作一致，无须进行特殊处理。即方便运算的实现。

②便于空表和非空表的统一处理

无论链表是否为空，其头指针都指向头结点的非空指针（空表中头结点的指针域为空），因此空表和非空表的处理也就得到了统一。

9、为什么在单循环链表中设置尾指针比设置头指针更好?

尾指针 rear 是指向终端结点的指针，用它来表示单循环链表可以使得查找链表的开始结点和终端结点都很方便。查找时间复杂度为 O（1）。

若用头指针来表示该链表，则查找开始结点为 O（1），终端结点为 O（n）。

10、队列假溢出+处理

（1)在队列的顺序存储结构中，设队头指针为front，队尾指针 rear，队的容量(存储空间的大小)为MaxSize。当有元素加入队列时，若 rear = MaxSize(初始时 rear=0)则发生队列的上溢现象，不能再向队列中加入元素。所谓队列假溢出现象是指队列中还有剩余空间但元素却不能进人队列，这种现象是由于队列的设计不合理所致。

（2）解决队列上溢的方法有以下几种：

①建立一个足够大的存储空间，但会降低空间的使用效率。

②当出现假溢出时可采用以下几种方法：

a、采用平移元素的方法：每当队列中加入一个元素时，队列中已有的元素向队头移动一个位置(当然要有空闲的空间可供移动)。

b、每当删除一个队头元素时，则依次移动队中的元素，始终使 front 指针指向队列中的第一个位置。

c、采用循环队列方式：把队列看成一个首尾相接的环形队列，在环形队列上进行插人或删除运算时仍然遵循“先进先出”的原则。

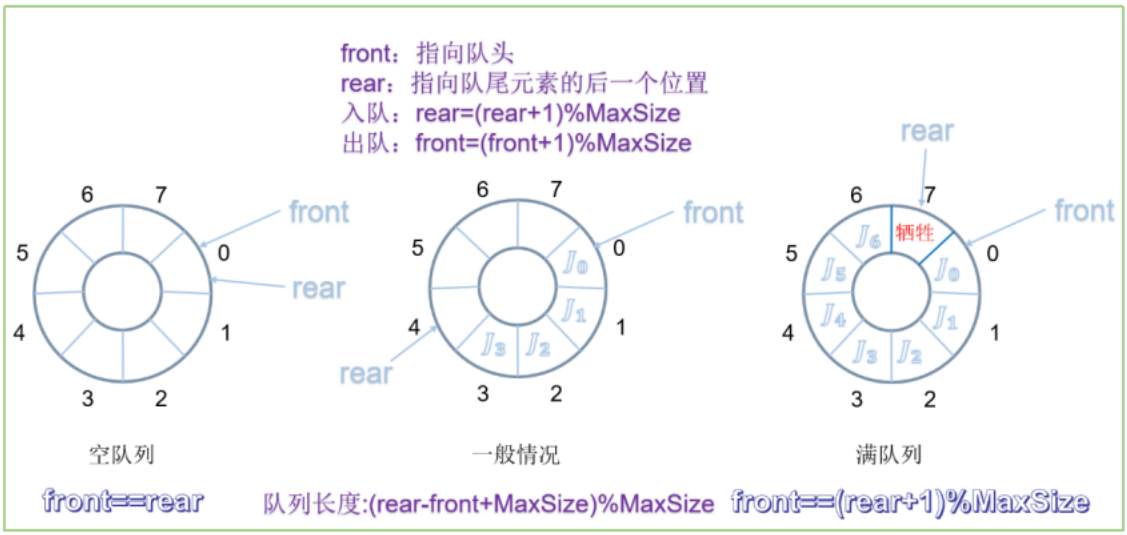
11、如何区分循环队列是队空还是队满？

牺牲一个单元来区分队空和队满，入队时少用一个队列单元，这是一种较为普遍的做法，约定以“队头指针在队尾指针的下一位置作为队满的标志”，如图所示。

队满条件：(Q.rear+1)%MaxSize==Q.front。

队空条件仍：Q.front==Q.rear。

队列中元素的个数：(Q.rear-Q.front+MaxSize)% MaxSize。



12、栈在后缀表达式求值的算法思想：

依次扫描表达式的每一项

①如果是操作数，则压入栈

②如果是运算符，则弹出两个栈顶元素，执行相应运算，运算结果压回栈顶，

③表达式的所有项都扫描完后，栈顶存放的元素就是最终的结果。

13、栈在递归中是怎样运用的？

①若在一个函数、一个过程或者一个数据结构的定义中直接或者间接的调用了它自身，则这个函数、这个过程、这个数据结构称为是递定义的，简称为递归。

②递归所用到的是系统管理栈

每进入一层递归，就将递归调用所需信息压入栈顶。

每退出一层递归，就从栈顶弹出相应信息。

③优点：结构清晰，程序易读

缺点：大量的递归调用会建立函数的副本，耗费大量的时间和内存。递归调用过程包含重复地计算，效率不高。当递归次数过深的时候，容易出现堆栈溢出的现象。

14、树与二叉树的区别

①二叉树的结点至多有两个子树，树则不然；

②二叉树的一个结点的子树有左右之分，而树的子树没有次序。

15、度为 2 的树与二叉树与的区别

①度为 2 的树至少有 3 个结点，而二叉树可以为空。

②度为 2 的树不分左子树右子树，是无序的。而二叉树区分左子树和右子树且次序不可颠倒。

16、度为 2 的有序树与二叉树与的区别

①度为 2 的树至少有 3 个结点，而二叉树可以为空。

②度为 2 的有序树的孩子的左右次序是相对于另一孩子而言的，若某个结点只有一个孩子，则这个孩子就无须区分其左右次序，而二叉树无论其孩子数是否为 2，均需确定其左右次序，即二叉树的结点次序不是相对于另一结点而言，而是确定的。

17、简述满二叉树，完全二叉树，二叉排序树，平衡二叉树的特性（真题考过）

（1）满二叉树：

①一棵高度为 h，且含有 2^h-1 个结点的二叉树称为满二叉树，即树中的每层都含有最多的结点。

②除叶子结点之外的每个结点度数均为 2。

③满二叉树一定是完全二叉树。

（2）完全二叉树：

①高度为 h、有 n 个结点的二叉树，当且仅当其每个结点都与高度为 h 的满二叉树中编号为 1～n 的结点一一对应时，称为完全二叉树。

②只允许最后一层有空缺结点且空缺在右边；叶子结点只可能在层次最大的两层上出现。

③若有度为 1 的结点，则只可能有一个，且该结点只有左孩子而无右孩子

（3）二叉排序树：

①左子树上所有的关键字均小于根结点，右子树上所有关键字均大于根结点。左子树和右子树又分别是一棵二叉排序树。

②构造二叉排序树时，用两组相同的数据，若数据的排列顺序不同，构造出的二叉排序树也不一样。

③二叉排序树的叶子结点到根结点的路径不一定是有序的。

④对二叉排序树进行中序遍历，可得到一个递增的有序序列。

（4）平衡二叉树

树中每一个结点的左子树，右子树高度之差的绝对值小于等于 1。

18、树的存储结构有哪些?

①双亲表示法：采用一组连续的存储空间存储每个结点，同时在每个结点后面增设一个伪指针指向其双亲节点。

②孩子表示法：将每个结点的孩子结点用单链表链接起来形成一个线性结构。

③孩子兄弟表示法：以二叉链表作为树的存储结构，其左指针指向第一个孩子结点，右指针指向其相邻的兄弟结点。可方便实现树转换为二叉树的操作，易于查找结点的孩子等，但缺点是从当前结点查找其双亲结点比较麻烦。

19、哈夫曼树的定义（看知识点）

20、哈夫曼树的构造（看知识点）

21、哈夫曼树的特点（看知识点）

22、表示有 1000 个顶点、1000 条边的有向图的邻接矩阵有多少个矩阵元素？是否稀疏矩阵？

答：106，不一定是稀疏矩阵（稀疏矩阵的定义是非零个数远小于该矩阵元素个数，且分布无规律）

23、如何判断一个有向图有环？

利用拓扑排序或深度优先遍历

①如果图是有环的，那么不会存在入度为 0 的点，是无法进行拓扑排序的。

②使用深度优先遍历，按退出过程的先后顺序记录下的顶点是逆向拓扑有序序列。若在执行未退出前，出现顶点到的回边，则说明存在包含顶点和顶点的环。

24、有 n 个顶点的有向强连通图最多有多少条边？最少有多少条边

强连通图的定义：图中任意两个顶点v_i和v_j之间都连通，即从v_i到v_j和从v_j到v_i都有路径，则称图为强连通图。在保证图是连通的情况下，n 个顶点的有向图中，边最多的情况就是每两个顶点之间都有两条边，最少的情况是图中所有顶点通过边串成一个环。

即最多有 n（n-1）条边，最少有 n 条边。

25、邻接矩阵与邻接表的对比

	邻接矩阵	邻接表
	表示唯一	表示不唯一
空间复杂度	有向图：$O(V^2)$ 无向图：$O(V^2)$	有向图：O(|V|+2|E|) 无向图：O(|V|+|E|)
求顶点 v_i 的度	无向图：第 i 行（列）上非 0 元素个数 有向图：出度：第 i 行上非 0 元素个数 入度：第 i 列上非 0 元素个数 总度数=出度+入度	无向图：第 i 个边表的结点数 有向图：出度：第 i 个单链表中边表的结点数 入度：所有编号为 i 的边表的结点数 （需遍历整个表） 总度数=出度+入度
是否容易看任意两顶点 i 和 j 之间有边	容易 无向图/有向图：对于任意顶点 i 和 j，邻接矩阵中 arcs[i][j]或 arcs[j][i]为 1 表示有边相连，否则无边相连	不容易（需要在相应结点对应的边表中查找另一结点，效率较低。） 无向图/有向图：对于任意顶点 i 和 j，若从顶点表结点 i（或 j）出发找到编号为 j（或 i）的边表结点，表示有边相连；否则无边相连。
是否容易找顶点的邻边	不容易，需扫描一行，花费的时间为 O(n)。	容易
求顶点的总边数	需检测整个矩阵 $O(n^2)$ 无向图：边数=矩阵 1 的个数/ 2 有向图：边数=矩阵 1 的个数	计算边表结点个数之和 无向图：边数=边表结点个数/ 2 有向图：边数=边表结点个数
用途	适合于稠密图	适合于稀疏图

26、请简述深度优先遍历、广度优先遍历的基本思想。

（1）深度优先遍历：

①首先访问出发点 v，将其标记为已访问过；选取与 v 邻接的未被访问的任意一个顶点 w，并访访问它；再选取与 w 邻接的未被访问的任一顶点并访问，以此重复进行。

②当一个顶点所有的邻接顶点都被访问过时，则依次退回到最近被访问过的顶点，若该顶点还有其他邻接顶点未被访问，则从这些未被访间的顶点中取一个并重复上述访问过程，直至图中所有顶点都被访问过为止。

（2）广度优先遍历

首先访问起始顶点 v，然后选取与 v 邻接的全部顶点 w_1,.....w_n，进行访问，再依次访问与 w_1,.....w_n 邻接的全部顶点（已访问过的除外），以此类推，重复上述过程，直到所有顶点都被访问过为止。

27、静态查找表与动态查找表的区别

（1）对查找表经常进行的操作一般有 4 种：

①查询某个特定的数据元素是否在查找表中；

②检索满足条件的某个特定的数据元素的各种属性；

③在查找表中插入一个数据元素；

④从查找表中删除某个数据元素。

（2）若一个查找表的操作只涉及上述操作①和②，则无须动态地修改查找表，此类查找表称为静态查找表。与此对应，需要动态地插入或删除的查找表称为动态查找表。

适合静态查找表的查找方法有顺序查找、折半查找、散列查找等；

适合动态查找表的查找方法有二叉排序树的查找、散列查找等。二叉平衡树和 B 树都是二叉排序树的改进。

28、顺序表顺序查找、折半查找、分块查找的 3 种查找方法比较：

①就平均查找长度而言，折半查找的平均查找长度取小，分块查找次之，顺序查找最大。

②就表的结构而言，顺序查找对有序表、无序表均适用，折半查找仅适用于有序表，而分块查找要求表中元素至少是分块有序的。

③就表的存储结构而言，顺序查找和分块查找可以适用于顺序表和链表两种存储结构，而折半查找只适用于以顺序表作为存储结构的表。

④分块查找综合了顺序查找和折半查找的优点，既能较快速地查找，又能适应动态变化的要求。

29、对长度为 2^k-1 的有序表进行折半查找，在成功查找时，最少需要多少次关键字比较?最多需要多少次关键字比较?查找失败时的平均比较次数是多少?

（1）当查找的记录正好处于中间位置时，关键字比较次数最少，所以折半查找在成功查找时最少需要 1 次关键字比较。折半查找在成功查找时最多关键字比较次数为判定树的高度，即 $\log_2\left(2^k-1+1\right)=k$。

（2）折半查找在不成功查找时比较过程都落在外部节点中，将判定树看成是一棵满二叉树，所有外部节点的高度为 k+1，其关键字比较次数为 k，所以查找失败时的平均比较次数是 k 次。

（注：若不是满树，查找失败的最少比较次数为 k-1，最多比较次数为 k（王道 7.2 选择题第 12 题）

30、将二叉排序树 T 的前序序列中的关键字依次插入到一棵空的二叉排序树中，得到二叉排序树 T'，T'与 T 是否相同？为什么？

构造出来的二叉排序树与原来的相同。

因为二叉排序树是二叉树，它的前序序列的第一个元素一定是二叉排序树的根，而对应前序序列的根后面的所有元素分为两组：从根的后一元素开始，其值小于根的值的一组元素就是树的左子树的结点的前序序列，剩下的元素的值大于根的值，即为树的右子树的结点的前序序列。在把前序序列的元素依次插入初始为空的二叉排序树时，第一个元素就成为树的根，它后面第一组元素的值都小于根结点的值，可以递归建立根的左子树；第二组元素的值都大于根结点的值，可以递归地建立根的右子树。最后插入的结果就是一棵与原来二叉排序树相同的二叉排序树。

31、设二叉排序树中的关键字互不相同，则其中的最小元素必无左子女，最大元素必无右子女。此命题是否正确？最小元素和最大元素一定是叶子结点吗？一个新元素总是作为叶子结点插入二叉排序树吗？

（1）最小元素必无左子女，最大元素必无右子女，此命题正确。

根据二叉排序树的定义，二叉排序树的根结点的关键字值一定大于左子树（若非空）上所有结点的关键字值，同时一定小于右子树（若非空）上所有结点的关键字值，而根结点的左、右子树也是二叉排序树。这是一个递归的定义。因此寻找最小元素的过程可以是一个递归的过程，逐层查找树（或子树）根结点的左子树，直至根结点的左子树空为止，这个子树的根结点就是最小元素所在结点，它无左子女；同样，寻找最大元素的过程也可以是一个递归的过程，逐层查找树（或子树）根结点的右子树，直至根结点的右子树空为止，这个子树的根结点就是最大元素所在结点，它无右子女。

（2）最小元素和最大元素不一定是叶子结点。

具有最小关键字值的结点可以有右子树，具有最大关键字值的结点可以有左子树。

（3）一个新元素总是作为叶子结点插入二叉排序树的。

在插入新元素时，插入算法总是先调用查找算法，从根开始查找是否具有相等关键字值的结点，如果查找成功，则不插入；如果查找失败，则新结点作为叶子结点插入到刚刚查找失败的位置。

32、若分别对大小均为 n 的有序顺序表和无序顺序表进行顺序查找，试在下列 3 种情况下分别讨论两者在等概率时的平均查找长度是否相同（在表的尾部有一个查找结束标记关键字）。

1）查找不成功，即表中没有关键字等于给定值 Key。

2）查找成功，且表中只有一个关键字等于给定值 Key。

3）查找成功，且表中有若干关键字等于给定值 Key，要求找出所有这些记录。

分析：1）查找长度不同。原因：扫描有序表，只要发现第一个比关键字大的值，即可结束扫描，无须扫描整个表，因此平均查找长度与无序表不同。

2）查找成功时，对于两种表都是逐个扫描关键字，直到找到与给定的 Key 相同的关键字为止，因此平均查找长度均为（n+1）/2。

3）有序表中相同关键字在表中的位置一定是相邻的，而无序表中相同关键字是随机散落在表中的。因此，对于有序表，找到第一个与 Key 相同的元素后，只需继续扫描，找到最后一个与 Key 不相同的元素即可；而对于无序表，则需要扫描整个表。对于无序表的平均查找长度为 n+1，而对于有序表则小于 n+1。

答：

1）不同，有序顺序表为（n+1）/2，无序顺序表为 n+1。

2）相同，有序顺序表为（n＋1）/2，无序顺序表也为（n+1）/2。

3）不同，有序顺序表小于 n+1，无序顺序表为 n+1。

33、从 1000 个人中选 100 个人，适合用哪个排序？为什么（真题考过）

堆排序适合于数据量大的场合，且所需的辅助空间较小，堆排序会将所有的数据建成一个堆，最大或最小的数据在堆顶，然后将堆顶数据和序列的最后一个数据交换。接下来再次重建堆，交换数据，依次下去，就可以排序所有的数据。

34、在堆排序、快速排序、归并排序中：

1）若只从存储空间考虑，则应首选哪种排序算法，其次选取哪种排序算法，最后选取哪种排序算法？

2）若只从排序结果的稳定性考虑，应选取哪种排序算法？

3）若只从平均情况下排序最快考虑，应选取哪种排序算法？

4）若只从最坏情况下排序最快并且要节省内存考虑，应选取哪种排序算法？

1）若只从存储空间考虑，则应首选堆排序（O(1)），其次选取快速排序（$O(\log_2 n)$），最后选取归并排序（O(n)）。

2）若只从排序结果的稳定性来考虑，则应该选取归并排序。

3）若只从平均情况下排序最快考虑，则应选取快速排序。

35、排序算法时间性能对比

①时间复杂度为$O(n\log_2 n)$的方法有：快速排序、堆排序、2 路归并排序；其中以快速排序为最好，在实际应用中常常优于其他排序算法

②时间复杂度为$O(n^2)$的方法有：直接插入排序、冒泡排序、简单选择排序；其中以直接插入排序最好；特别是那些对关键字近似有序的记录序列更是如此。

③时间复杂度为$O(d(n+r))$的方法：基数排序

36、排序算法空间复杂度对比

空间复杂度指的是排序过程中所需的辅助空间大小

①插入类（直接插入、折半插入、希尔）、选择类（简单选择、堆）、冒泡排序）的空间复杂度为$O(1)$，即仅需要借助常数个辅助空间

②快速排序：平均、最好情况为$O(\log_2 n)$，最坏情况为$O(n)$，为栈所需的辅助空间，用于实现递归。

③归并排序：空间复杂度为$O(n)$，所需辅助空间最多，用于元素复制

④基数排序需附设 r 个队列（r 个队头指针、r 个队尾指针），则空间复杂度为$O(r)$

37、哪些排序算法会受初始关键字分布的影响？

当待排记录序列按关键字序列有序时、直接插入排序、冒泡排序可达到$O(n)$的时间复杂度。

而对于快速排序而言，这是不好的情况，此时的时间性能退化到$O(n^2)$

38、排序算法的稳定性

稳定性指的是对于两个关键字相等的记录，它们在序列中的相对位置，在排序之前和排序之后，没有改变。

①插入排序、冒泡排序、归并排序和基数排序是稳定的排序方法

②简单选择排序、快速排序、希尔排序和堆排序都是不稳定的排序方法。其中快速排序、堆排序最不稳定。

39、排序算法的比较（35、36、37、38 结合）

40、比较直接插入排序算法和希尔排序算法的不同点

直接插入排序算法是稳定的，更适合于原始元素基本有序的情况。若采用折半查找而不是顺序查找待插入元素的插入位置时，可减少元素比较的次数，但移动元素的次数没有减少；在采用顺序查找待插入元素的插入位置时也适用于链式存储结构。

希尔排序算法是不稳定的，元素的总比较次数和移动次数都比直接插入排序时要少，n 越大时，效果越明显；增量序列 d 可以有不同的取法，但有两个共同的特征，即最后一个增量必须是 1，增量序列中的值没有除 1 之外的公因子，希尔排序不适用于链式存储结构。

41、指出堆和二叉排序树的区别。

以小根堆为例，堆的特点是双亲节点的关键字必然小于等于孩子节点的关键字，而两个孩子节点的关键字没有次序规定。而二叉排序树中，每个双亲节点的关键字均大于左子树节点的关键字，每个双亲节点的关键字均小于右子树节点的关键字，也就是说，每个双亲节点的左、右孩子的关键字有次序关系。

42、堆排序是否是一种稳定的排序方法？为什么？

堆排序是一种不稳定的排序方法。因为在堆的调整过程中，关键字进行比较和交换所走的是该节点到叶子节点的一条路径，因此对相同的关键字而言，就可能出现排在后面的关键字被交换到前面来的情况。