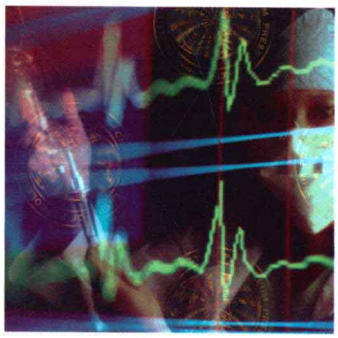
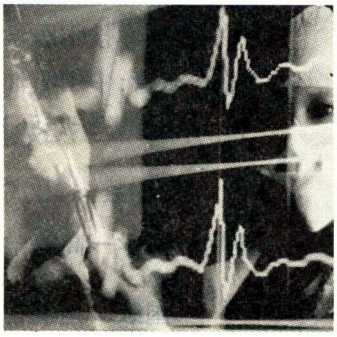
数据库技术医学应用实践教程



主　编　雷国华　胡西厚
副主编　王德臣　杨春波
邢慧丽　薛　慧

中国石油大学出版社

数据库技术医学应用实践教程



主　编　雷国华　胡西厚
副主编　王德臣　杨春波
邢慧丽　薛　慧

编　者（按姓氏笔画排序）：
王玉凤　王金才　王希杰　王德臣　刘海青
邢慧丽　李龙森　李秀敏　杨春波　胡西厚
徐　静　谢　芬　雷国华　薛　慧　魏　飞

中国石油大学出版社

图书在版编目（CIP）数据

数据库技术医学应用实践教程/雷国华，胡西厚主编. —东营：中国石油大学出版社，2012.8
ISBN 978-7-5636-3788-1

Ⅰ.①数… Ⅱ.①雷… ②胡… Ⅲ.①数据库系统—应用—医学—高等学校—教学参考资料 Ⅳ.①R319

中国版本图书馆 CIP 数据核字（2012）第 178264 号

数据库技术医学应用实践教程

主　　编：雷国华　胡西厚
责任编辑：魏　瑾

出 版 者：中国石油大学出版社（山东 东营，邮编 257061）
网　　址：http://www.uppbook.com.cn
电子邮箱：weicbs@163.com
印 刷 者：山东省东营市新华印刷厂
发 行 者：中国石油大学出版社（电话 0546-8391810）
开　　本：185 mm×260 mm　印张：17.5　字数：448 千字
版　　次：2012 年 9 月第 1 版第 1 次印刷
定　　价：31.80 元

前言 Preface

数据库技术是计算机信息系统与应用系统的核心技术，已成为绝大多数高等学校非计算机专业的必修课。根据教育部高等学校非计算机专业计算机基础课程教学指导委员会提出的有关“数据库技术及应用”课程的教学要求，教育部高等学校医药类计算机基础课程教学指导分委员会对医药类高等学校“数据库技术与应用”课程的教学内容提出了新的要求和规范，本书就是据此编写而成的。编者力图以教育部高等学校医药类计算机基础课程教学指导分委员会制定的《数据库技术及应用课程大纲》为主线，以培养学生分析问题、解决问题的能力为出发点，以技术应用为核心，创建应用实例。

本书在编写过程中注重基础理论与 Visual FoxPro 数据库应用技术的相辅相成。“知识准备”部分提供了支撑实践过程的理论知识点，“知识巩固”和“知识巩固答案及解析”部分通过练习题及其答案解析让学生巩固这些知识点，“实践内容”部分提供了技术实践的过程、利用数据库技术解决实际问题的手段和技术方法。

本书的主要内容包括算法与数据结构的基础理论，Visual FoxPro 基本操作，数据操作实践，结构化查询语言（SQL）应用实践，面向过程程序设计实践，查询、视图、表单、菜单、报表等面向对象技术应用实践，数据库应用系统开发案例等。通过“住院管理系统”的开发，描述了数据库应用系统开发的方法、过程及步骤，功能实现方面给出了详尽的代码。编者虽极尽规范，但由于个人思维的差异，难免出现错误和不足，书中代码仅供参考。

全书由雷国华、胡西厚统稿，雷国华组编。第 1 章由薛慧编写，第 2 章由邢慧丽、谢芬编写，第 3 章由薛慧、谢芬编写，第 4 章、第 5 章和第 9 章由雷国华编写，第 6 章由刘海青编写，第 7 章由徐静编写，第 8 章由杨春波编写，附录由邢慧丽、李龙森编写。雷国华、徐静、薛慧对全书文字部分进行了审校，雷国华、王德臣、魏飞对代码部分进行了审校。本书所有应用实例内容均由王德臣、魏飞进行了上机测试。

本书是《数据库技术与医学应用》的配套用书，体系清晰，具有整体性、全面性和实用性，既可作为医药类高等学校“数据库技术与应用”课程的教材，也可作为全国计算机等级考试二级 Visual FoxPro 程序设计的自学教材或参考书，还可以作为卫生事业单位计算机用户或计算机技术人员学习数据库技术的自学用书。

目前计算机科学与技术日新月异，计算机在医学中的应用也在不断发展，再加上编者水平所限，书中难免存在不足、谬误和疏漏之处，恳请广大读者批评指正。

编　者

2012年9月

目录 Contents

第1章　算法与数据结构

1.1　知识准备

1.1.1　算法

算法是指解题方案的准确而完整的描述。算法不等于程序，也不等于计算机方法，程序的编制不可能优于算法的设计。

1. 算法的基本特征

1）有穷性（Finiteness）

算法的有穷性是指算法必须能够在执行有限个步骤之后终止。

2）确切性（Definiteness）

算法的确切性是指算法的每一个步骤必须有确切的定义。

3）输入项（Input）

一个算法有0个或多个输入，以刻画运算对象的初始情况。所谓0个输入是指算法本身给定了初始条件。

4）输出项（Output）

一个算法有一个或多个输出，以反映对输入数据的加工结果。没有输出的算法是毫无意义的。

5）可行性（Effectiveness）

算法中执行的任何计算步都可以被分解为基本的可执行的操作步，即每个计算步都可以在有限时间内完成（也称为有效性）。

2. 算法的基本要素

（1）算法中对数据的运算和操作有以下四类：

① 算术运算（包括加、减、乘、除等）。

② 逻辑运算（包括与、或、非等）。

③ 关系运算（包括大于、小于、等于、不等于等）。

④ 数据传输（包括赋值、输入、输出等操作）。

（2）算法的控制结构：算法中各操作之间的执行顺序称为算法的控制结构。

① 一个算法一般由顺序、选择和循环三种基本控制结构组合而成。

② 描述算法的工具通常有传统流程图、N-S结构化流程图和算法描述评议。

3. 算法设计的基本方法

1）列举法

列举法的基本思想是，根据提出的问题，列举出所有可能的情况，并用问题中给定的条件

检验哪些是满足条件的，哪些是不满足条件的。列举法通常用于解决“是否存在”或“有哪些可能”等问题。例如，我国古代的趣味数学题“百钱买百鸡”、“鸡兔同笼”等，均可采用列举法解决。

2）归纳法

归纳法的基本思想是，通过列举少量的特殊情况，经过分析，最后找出一般的关系。归纳是一种抽象，即从特殊现象中找出一般规律。但由于在归纳法中不可能对所有的情况进行列举，因此，该方法得到的结论只是一种猜测，还需要进一步证明。

3）递推法

递推，即从已知的初始条件出发，逐次推出所要求的各个中间环节和最后结果，其中，初始条件由问题本身给定，或通过对问题的分析与化简而确定。递推法的本质是归纳，递推关系式通常是归纳的结果。例如，裴波那契数列就是采用递推的方法解决问题的。

4）递归法

在解决一些复杂问题时，为了降低问题的复杂程序，通常是将问题逐层分解，最后归结为一些最简单的问题。这种将问题逐层分解的过程，并没有对问题进行求解，而是在解决了最后的那些最简单的问题后，再沿着原来分解的逆过程逐步进行综合，这就是递归的方法。

递归法分为直接递归和间接递归两种方法。如果一个算法直接调用自己，称为直接递归调用；如果算法 A 调用算法 B，而算法 B 又调用算法 A，则称为间接递归调用。

5）减半递推技术

减半递推即将问题的规模减半，然后重复相同的递推操作。例如，一元二次方程的求解。

6）回溯法

有些实际的问题很难归纳出一组简单的递推公式或直观的求解步骤，也不能使用无限的列举。对于这类问题，只能采用试探的方法，通过对问题的分析找出解决问题的线索，然后沿着这个线索进行试探。如果试探成功，就得到问题的解，如果不成功，再逐步回退，换别的路线进行试探，这种方法称为回溯法，例如人工智能中的机器人下棋。

4. 算法复杂度

1）算法的时间复杂度

算法的时间复杂度是指执行算法所需要的计算工作量。计算工作量用算法所执行的基本运算次数来确定。

（1）平均性态（Average Behavior）。

平均性态分析是指用各种特定输入下的基本运算次数的加权平均值来度量算法的工作量。设 x 是所有可能输入中的某个特定输入，$p(x)$ 是 x 出现的概率，$t(x)$ 是算法在输入为 x 时所执行的基本运算次数，则算法的平均性态定义为

$$A(n)=\sum_{x\in D_n}p(x)t(x)$$

即

$$A(n)=\sum_{i=1}^{n+1}p_it_i=\sum_{i=1}^{n}\frac{q}{n}i+(1-q)n=\frac{(n+1)q}{2}+(1-q)n$$

其中，D_n 表示当规模为 n 时，算法的所有可能输入的集合。

（2）最坏情况复杂性（Worst-case Complexity）。

最坏情况分析是指在规模为 n 时，用算法所执行的基本运算的最大次数来度量算法的工作量。它定义为

$$W(n)=\max_{x\in D_n}\{t(x)\}$$

例如，在具有 n 个元素的数列中搜索一个数 x，其平均形态和最坏情况分析如下：

平均性态分析：该数在数列中任何位置出现的概率是相同的，也有可能不存在，存在的概率为 q。如果有一半的机会存在，即概率 q 为 1/2，则平均性态为

$$A(n)=\frac{(n+1)\times\frac{1}{2}}{2}+(1-\frac{1}{2})\ n\approx\frac{3}{4}n$$

如果查找的元素一定在数列中，即每个数存在的概率都为 1，则平均性态为

$$A(n)=\frac{n+1}{2}\approx\frac{n}{2}$$

最坏情况分析：x 在数列的最后或不在数列中，即最坏情况复杂度为

$$W(n)=\max\{t_i\mid 1\leqslant i\leqslant n+1\}=n$$

2）算法的空间复杂度

算法的空间复杂度指执行算法所需要的内存空间，包括：

（1）算法程序所占的空间。

（2）输入的初始数据所占的存储空间。

（3）算法执行过程中所需要的额外空间，即算法程序执行过程中的工作单元以及某种数据结构所需要的附加存储空间。

1.1.2　数据结构的基本概念

数据结构是指相互有关联的数据元素的集合。

1. 数据结构研究的三个方面

（1）数据集合中各数据元素之间所固有的逻辑关系，即数据的逻辑结构。

（2）在对数据进行处理时，各数据元素在计算机中的存储关系，即数据的存储结构。

（3）对各种数据结构进行的运算。

2. 数据的逻辑结构

数据的逻辑结构是对数据元素之间的逻辑关系的描述。

（1）数据元素的集合通常记为 D。

（2）D 上的关系反映了数据元素之间的前后件关系，通常记为 R。

一个数据结构 B 可以表示为

$$B=(D, R)$$

3. 数据的存储结构

数据的存储结构是指数据的逻辑结构在计算机存储空间中的存放形式，也称数据的物理结构。数据存储时，不仅要存放数据元素的信息，而且要存储数据元素之间的前后件关系的信息。

（1）一种数据结构可根据需要采用不同的存储结构。

（2）常用的存储结构有顺序、链接和索引等。

4. 数据结构的图形表示

如图 1-1 所示，数据结构的图形表示有两个元素：

（1）中间标有元素值的方框表示数据元素，称为数据结点。

（2）用有向线段表示数据元素之间的前后件关系，即有向线段从前件结点指向后件结点。没有前件的结点称为根结点，没有后件的结点称为终端结点，也称叶结点。

5. 线性结构与非线性结构

如果在一个数据结构中一个数据元素都没有，则称该数据结构为空的数据结构。根据数据结构中各数据元素之间前后件关系的复杂程度，一般将数据结构分为两大类：线性结构与非线性结构。

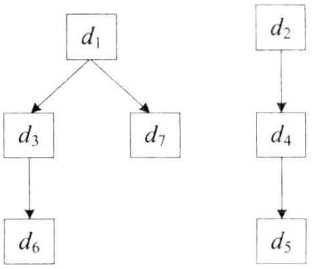


图 1-1　数据结构的图形表示

1）线性结构

线性结构又称为线性表。如果一个非空数据结构满足下列两个条件则称为线性结构：

（1）有且只有一个根结点。

（2）每一个结点最多有一个前件，也最多有一个后件。

如：栈、队列和线性链表都是线性结构。

注意：在线性结构表中插入或删除元素，该线性表仍应满足线性结构。

2）非线性结构

不满足线性结构条件的数据结构称为非线性结构。如：树、二叉树和图都是非线性结构。

对于空的数据结构，如果对该数据结构的运算是按线性结构的规则处理的，则该数据结构属于线性结构，否则属于非线性结构。

1.1.3　线性表及其顺序存储结构

1. 线性表的基本概念

（1）线性表由一组数据元素构成，数据元素的位置只取决于自己的序号，数据元素之间的相对位置是线性的。

注意：这里的数据元素并不仅仅指一个数据，而是指广义的数据元素，如矩阵、学生记录表等。

（2）在复杂线性表中，由若干项数据元素组成的数据组合称为记录，而由多个记录构成的线性表称为文件。

（3）非空线性表的结构特征：

① 有且只有一个根结点 a_1，无前件。

② 有且只有一个终端结点 a_n，无后件。

③ 除根结点与终端结点外，其他所有结点有且只有一个前件和一个后件。结点个数 n 称为线性表的长度，当 n=0 时称为空表。

2. 线性表的顺序存储结构

线性表的顺序存储结构是指用一组地址连续的存储单元依次存放线性表中的数据元素。线性表的顺序存储结构具备如下两个基本特征：

（1）线性表中的所有数据元素所占的存储空间是连续的。

（2）线性表中各数据元素在存储空间中是按逻辑顺序依次存放的。

a_i 的存储地址为 $\mathrm{ADR}(a_i)=\mathrm{ADR}(a_1)+(i-1)k$，其中，$\mathrm{ADR}(a_1)$ 为第一个数据元素的地址，k 代表每个数据元素所占的字节数。

3. 线性表的运算

线性表可以进行以下运算：插入、删除、查找、排序、拆分、合并、复制和逆转等。

1）线性表的插入运算

线性表的插入运算是指在表的第 i 个位置元素之前，插入一个新结点 x，使长度为 n 的线性表 $(a_1, \cdots, a_{i-1}, a_i, \cdots a_n)$ 变成长度为 $n+1$ 的线性表 $(a_1, \cdots, a_{i-1}, x, a_i, \cdots, a_n)$。对顺序存储表而言，需要改变的是从第 i 个元素起到第 n 个元素的存储位置，即从第 i 个到第 n 个元素向后移动一个位置，共需移动 $n-i+1$ 个元素。

2）线性表的删除运算

线性表的删除运算是指将表的第 i 个结点删去，使长度为 n 的线性表 $(a_1, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_n)$ 变成长度为 $n-1$ 的线性表 $(a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n)$。对顺序存储表而言，需要改变从第 $i+1$ 个元素起到第 n 个元素的存储位置，即从第 $i+1$ 到第 n 个元素向前移动一个位置，共需移动 $n-i$ 个元素。

在以顺序方式存储的线性表中插入或删除一个数据元素，平均约需移动表中一半的元素，这个数目在线性表的长度较大时是很可观的。因此，顺序存储表仅适用于不常进行插入或删除操作、表中数据元素元素相对稳定的线性表。

1.1.4　栈和队列

1. 栈的概念

栈是限定在一端进行插入与删除的线性表。允许插入与删除的一端称为栈顶，用 TOP 表示；不允许插入与删除的另一端称为栈底，用 BOTTOM 表示。栈按照“先进后出”（FILO）或“后进先出”（LIFO）的方式组织数据。栈具有记忆作用。

2. 栈的运算

在程序设计语言中，用一维数组 S(1:m) 作为栈的顺序存储空间，其中 m 为栈的最大容量。S(BOTTOM) 通常为栈底元素（栈非空的情况下），S(TOP) 为栈顶元素。TOP=0 时表示栈空，TOP=m 时表示栈满。

1）入栈运算

入栈运算是指在栈顶位置插入一个新元素。首先将栈顶指针加 1（即 TOP 加 1），然后将新元素插入到栈顶指针指向的位置。当栈顶指针已经指向存储空间的最后一个位置时，说明栈空间已满，不可进行入栈操作，这种情况称为栈“上溢”错误。

2）退栈运算

退栈运算是指取出栈顶元素，并赋予一个指定的变量。首先将栈顶元素赋予一个指定的变量，然后将栈顶指针减 1（即 TOP 减 1）。当栈顶指针为 0 时，说明栈空，不可进行退栈操作，这种情况称为栈的“下溢”错误。

3）读栈顶元素

读栈顶元素是指将栈顶元素赋予一个指定的变量。这个运算不删除栈顶元素，只是将它赋予一个变量，因此栈顶指针不会改变。当栈顶指针为 0 时，说明栈空，读不到栈顶元素。

3. 队列

队列是指允许在一端（队尾）进行插入，而在另一端（队头）进行删除的线性表。REAR 指针指向队尾，FRONT 指针指向队头。队列是“先进先出”（FIFO）或“后进后出”（LILO）的线性表。队列运算包括：

1）入队运算

在队尾插入一个元素，只涉及队尾指针 REAR 的变化。

2）退队运算

在队头删除一个元素，只涉及队头指针 FRONT 的变化。

4. 循环队列

循环队列就是将队列存储空间的最后一个位置绕到第一个位置，形成逻辑上的环状空间，供队列循环使用。

循环队列的初始状态为空，即 REAR=FRONT=m，这里的 m 即为队列的存储空间，但当循环队列满时也有 REAR=FRONT，即在循环队列中当 REAR=FRONT 时，不能确定队列是满还是空。在实际使用循环队列时，为了能区分队列是否为空，通常还需增加一个标志 s，s=0 表示队列空，s=1 表示队列非空。

1）入队运算

首先若 s=0，则置 s=1，然后 REAR=REAR+1，当 REAR=m+1 时，置 REAR=1，将新元素插入到 REAR 指向的位置。

s=1 且 FRONT=REAR 表示队列满，不能进行入队运算，这种情况称为“上溢”。

2）退队运算

首先若 s=1，则 FRONT=FRONT+1，当 FRONT=m+1 时，置 FRONT=1，将队头指针指向的元素赋给指定的变量。若 FRONT=REAR，则置 s=0。

当循环队列为空（s=0）时，不能进行退队运算，这种情况称为“下溢”。

1.1.5 线性链表

1. 线性链表的基本概念

数据结构中的每一个数据结点对应于一个存储单元，这种存储单元称为存储结点，简称结点。结点由两部分组成：

（1）数据域，用于存储数据元素值。

（2）指针域，用于存放指针，指向前一个或后一个结点。

在链式存储结构中，存储数据结构的存储空间可以不连续，各数据结点的存储顺序与数据元素之间的逻辑关系可以不一致，而数据元素之间的逻辑关系是由指针域确定的。

2. 线性链表的分类

1）线性单链表

若指针域为 HEAD，则 HEAD 称为头指针，HEAD=NULL（或 0）称为空表。线性单链表可以从表头开始向后扫描链表中的所有结点，而不能从中间或表尾结点开始向前扫描位于该结点之前的元素。

2）线性双向链表

线性双向链表包含两个指针：左指针（Llink）——指向前件结点；右指针（Rlink）——指向后件结点。

3）循环链表

循环链表采用另一种链接方式，它的特点如下：

（1）在循环链表中增加一个表头结点，其数据域为任意或根据需要来设置，指针域指向线性表的第一个元素的结点。循环链表的头指针指向表头结点。

（2）循环链表中最后一个结点的指针域不是空的，而是指向表头结点。在循环链表中，所有结点的指针构成一个环状链。

3. 线性链表的基本运算

线性链表的基本运算包括查找、插入、删除等。

1.1.6　树与二叉树

1. 树

树是一种简单的非线性结构，所有元素之间具有明显的层次特性。

（1）在树结构中，每一个结点只有一个前件，称为父结点。没有前件的结点只有一个，称为树的根结点，简称树的根。每一个结点可以有多个后件，称为该结点的子结点。没有后件的结点称为叶结点。

（2）在树结构中，一个结点所拥有的后件的个数称为该结点的度，所有度中最大的值称为树的度。树的最大层次称为树的深度。

2. 二叉树

（1）非空二叉树只有一个根结点。

（2）每一个结点最多有两棵子树，分别称为该结点的左子树与右子树。

二叉树每个结点的度小于等于 2，而普通树中不限制结点的度数。二叉树是一个有序树，左、右子树不能颠倒

3. 满二叉树和完全二叉树

满二叉树是指除最后一层外每一层上的所有结点都有两个子结点的二叉树，其 k 层上有 2^{k-1} 个结点、深度为 m 的满二叉树有 2^m-1 个结点。

完全二叉树是指除最后一层外，每一层上的结点数均达到最大值，最后一层只缺少右边的若干结点的二叉树。

满二叉树也是完全二叉树，而完全二叉树不一定是满二叉树。

4. 二叉树的基本性质

（1）在二叉树的第 k 层上，最多有 2^{k-1}（$k\geqslant 1$）个结点。

（2）深度为 m 的二叉树最多有 2^m-1 个结点。

（3）度为 0 的结点（即叶结点）总是比度为 2 的结点多一个。

（4）具有 n 个结点的二叉树，其深度至少为 $[\log_2 n]+1$，其中 $[\log_2 n]$ 表示取 $\log_2 n$ 的整数部分。

（5）具有 n 个结点的完全二叉树的深度为 $[\log_2 n]+1$。

（6）设完全二叉树共有 n 个结点，如果从根结点开始，按层序（每一层从左到右）用自然数 $1, 2, \cdots, n$ 给结点进行编号（$k=1, 2, \cdots, n$），有以下结论：

① 若 $k=1$，则该结点为根结点，它没有父结点；若 $k>1$，则该结点的父结点的编号为 $\mathrm{INT}(k/2)$。

② 若 $2k\leqslant n$，则编号为 k 的结点的左子结点编号为 $2k$；否则该结点无左子结点（也无右子结点）。

③ 若 $2k+1\leqslant n$，则编号为 k 的结点的右子结点编号为 $2k+1$；否则该结点无右子结点。

5. 二叉树的存储结构

在链式存储结构中，用于存储二叉树中各元素的存储结点由两部分组成：数据域与指针域。

指针域有两个：一个用于指向该结点的左子结点的存储地址，称为左指针域；另一个用于指向该结点的右子结点的存储地址，称为右指针域。

对于满二叉树与完全二叉树可以按层序进行顺序存储。

6. 二叉树的遍历

（1）前序遍历（DLR）：首先访问根结点，然后前序遍历左子树，最后前序遍历右子树。

（2）中序遍历（LDR）：首先中序遍历左子树，然后访问根结点，最后中序遍历右子树。

（3）后序遍历（LRD）：首先后序遍历左子树，然后后序遍历右子树，最后访问根结点。

1.1.7 查找技术

1. 顺序查找

顺序查找是指从表的第一个元素开始，将给定的值与表中元素的关键字逐个进行比较，直到二者相等，查到所要找的元素为止，否则就是表中没有要找的元素，查找不成功。顺序查找的适用情况为：

（1）线性表为无序表。

（2）表采用链式存储结构。

在最坏的情况下，顺序查找需要比较 n 次。

2. 二分法查找

二分法查找只适用于顺序存储的有序表，此处所述的有序表是指线性表中的元素按值非递减排列（即由小到大，但允许相邻元素值相等）。查找方法是将 x 与线性表的中间项比较：

（1）若 x 的值与中间项的值相等，则说明已查到，查找结束；

（2）若 x 小于中间项的值，则在线性表的前半部分以相同的方法查找；

（3）若 x 大于中间项的值，则在线性表的后半部分以相同的方法查找。

对于长度为 n 的有序线性表，最坏情况需比较 $\log_2 n$ 次。

1.1.8 排序技术

排序是指将一个无序序列整理成按值非递减顺序排列的有序序列。

1. 交换类排序法

两两比较待排序记录的关键字，发现两记录的次序相反即进行交换，直到没有反序的记录为止。冒泡排序和快速排序都属于交换类的排序方法。

1）冒泡排序法

对存放原始数据的数组，按从后往前的方向进行多次扫描，每次扫描称为一趟（Pass）。当发现相邻两数据的次序与要求的“递增次序”不符合时，即将这两个数据进行互换。这样，较小的数据就会逐单元向前移动，好像气泡向上浮起一样。假设线性表的长度为 n，则在最坏情况下，冒泡排序需要经过 $n/2$ 趟从前往后的扫描和 $n/2$ 趟从后往前的扫描，需要比较的次数为 $n(n-1)/2$。

2）快速排序法

任取待排序序列中的某个元素作为基准（一般取第一个元素），通过一趟排序，将待排元素分为左右两个子序列，左子序列元素的排序码均小于或等于基准元素的排序码，右子序列的排序码则大于基准元素的排序码，然后分别对两个子序列继续进行排序，直至整个序列有序。

2. 插入类排序法

每次将一个待排序的记录按其关键字大小插入到前面已经排好序的子序列中的适当位置，直到全部记录插入完成为止。

1）直接插入排序法

把 n 个待排序的元素看成一个有序表和一个无序表，开始时有序表中只包含一个元素，无序表中包含 $n-1$ 个元素，排序过程中每次从无序表中取出第一个元素，把它的排序码依次与有序表元素的排序码进行比较，将它插入到有序表的适当位置，使之成为新的有序表。最坏情况需要 $n(n-1)/2$ 次比较。

2）希尔排序法

将整个无序序列分割成若干小的子序列分别进行插入排序。

子序列的分割方法：将相隔某个增量 h 的元素构成一个子序列，在排序的过程中，逐次减小这个增量，最后当 h 减小到 1 时，再进行一次插入排序操作，即完成排序。

增量序列一般取 $h_t=n/2^k$（$k=1, 2, \cdots, [\log_2 n]$），其中 n 为待排序序列的长度。

3. 选择类排序法

每一趟从待排序的记录中选出关键字最小的记录，顺序放在已排好序的子文件的最后，直到全部记录排序完毕。

1）直接选择排序法

扫描整个线性表，从中选出最小的元素，将它交换到表的最前面，然后对剩下的子表采用同样的方法，直到子表为空为止。最坏情况需要 $n(n-1)/2$ 次比较。

2）堆排序法

最坏情况需要 $O(n\log_2 n)$ 次比较。

1.2　知识巩固

一、单项选择题

1. 以下数据结构中不属于线性数据结构的是_____。

A）队列　　B）线性表　　C）二叉树　　D）栈

2. 在一棵二叉树上第 5 层的结点数最多是_____。

A）8　　B）16　　C）32　　D）15

3. 算法的时间复杂度是指_____。

A）执行算法程序所需要的时间

B）算法程序的长度

C）算法执行过程中所需要的基本运算次数

D）算法程序中的指令条数

4. 下列叙述中正确的是_____。

A）线性表是线性结构　　B）栈与队列是非线性结构

C）线性链表是非线性结构　　D）二叉树是线性结构

5. 设一棵完全二叉树共有 699 个结点，则该二叉树中的叶结点数为_____。