高等院校非计算机专业教材

（根据教育部对计算机基础教学的三个层次的要求编写）

数据结构与数据库应用基础教程

沈朝辉 / 主编
沈朝辉 / 赵 宏 / 王 刚 编著
沈琴婉 / 主审

南开大学出版社

高等院校非计算机专业教材
（根据教育部对计算机基础教学的三个层次的要求编写）

数据结构与数据库应用基础教程

沈朝辉 主编
沈朝辉　赵宏　王刚　编著
沈琴婉 主审

南 开 大 学 出 版 社
天 津

图书在版编目(CIP)数据

数据结构与数据库应用基础教程 / 沈朝辉，赵宏，王刚编著. —天津：南开大学出版社，2007.3

高等院校非计算机专业教材

ISBN 978-7-310-02673-9

Ⅰ.数... Ⅱ.①沈...②赵...③王... Ⅲ.①数据结构—高等学校—教材②数据库系统—高等学校—教材

Ⅳ.TP311.1

中国版本图书馆 CIP 数据核字(2007)第 027475 号

南开大学出版社出版发行

出版人:肖占鹏

地址:天津市南开区卫津路 94 号　　邮政编码:300071

营销部电话:(022)23508339　23500755

营销部传真:(022)23508542　　邮购部电话:(022)23502200

*

河北昌黎太阳红彩色印刷有限责任公司印刷

全国各地新华书店经销

*

2007 年 3 月第 1 版　　2007 年 3 月第 1 次印刷

787×1092 毫米　16 开本　20.125 印张　506 千字

定价:32.00 元

如遇图书印装质量问题,请与本社营销部联系调换,电话:(022)23507125

内容提要

本书是高校计算机基础教育第二层次的教材，供第一层次计算机文化基础与 C / C++的后继课程使用。全书共分三部分十八章，第一部分数据结构与算法基础的主要内容包括：算法基础、数据结构概述、线性表及其存储结构、栈和队列、树与二叉树、图、查找与排序；第二部分数据库设计及应用基础的主要内容包括：数据库的基本概念、关系数据库设计、Access 2003 基础；第三部分 Visual Basic 程序设计基础的主要内容包括：Visua1 Basic 6.0 概述、VB 数据类型、运算符和表达式、VB 内部函数及数据的输入输出、窗体和常用控件、程序控制结构和构造数据类型、过程、菜单编辑器和多窗体程序设计、VB 与数据库。

本书由多年从事计算机软件基础教学的教师编写。在编写过程中，特别注意遵循由浅入深、繁简适当的原则，采用图文并茂的形式，重在应用、加强基础、结合大量实例，深入浅出地讲解数据结构与算法、数据库应用、Visua1 Basic 6.0 的基本操作和面向对象编程技术。

本书可供高校非计算机专业学生作为与软件基础有关课的教材，也可供从事计算机工作的技术人员及其他需要学习软件基础知识的读者自学使用。

前 言

21 世纪是信息时代，是科学技术高速发展的时代。伴随着计算机技术与网络技术的结合，人类的生产、生活和思维方式发生了深刻的变化，不论现在和将来，计算机知识都是人类文化的一个重要组成部分。

在我国经济高速发展的今天，计算机知识更新的速度之快让入目不暇接，新理论、新技术、新工具更是层出不穷。几十年来，计算机技术的变化虽然很快，但有规律可循。有些基础知识对于常用的计算机基本没有改变，发生较大变化的是计算机的应用模式和人机交互模式。所以，在计算机教学中，我们一方面应注意让学生了解并掌握当前主流软件的应用，增加网络和有关信息处理的知识，使学生了解信息化社会对大学生的要求；另一方面，还要让学生掌握比较成熟的、稳定的、适用面广的理论和应用知识。这样，大学生在毕业后，能够尽快适应信息化社会的需求，又有一定的后劲跟上信息化社会突飞猛进的发展。对于当代大学生来说，只知道用还不够，还应不断地拓宽自己的知识面，夯实基础。这就要求从事计算机教学的人员，不断更新教学内容，加强学生的基础训练。

本书中的数据结构与算法是软件开发的基础，数据库及其应用是数据库技术的应用典范，而数据库技术是当前计算机理论和应用中发展极为迅速、应用非常广泛的领域之一。作为数据库管理系统的 Access 2003 是微软公司推出的最新关系数据库开发平台，是新一代微机版数据库管理系统的突出代表。作为实用图形界面软件开发环境的 Visual Basic 6.0 也是微软公司推出的一种面向对象程序开发的最容易学习的优秀编程语言。

本书对各部分的重点和难点，从不同角度、不同层次由浅入深地进行讲解，使读者通过阅读典型例题受到一定启发，深入而全面地理解数据结构与算法、数据库应用及面向对象程序设计的基本概念，并通过独立完成一定数量的习题和上机实习操作，激发学生对先进科学技术的向往，启发学生对新知识的学习热情，培养学生创新意识，提高学生的创新能力，锻炼学生的动手实践能力。

本书适用于学时（含上机）为 60～80 的课程按排。另外，本书各部分的内容相对独立，自成体系。若课时较少，可根据教学的需要酌情取舍。

书中第一部分数据结构与算法基础的第 1 至 7 章由沈朝辉编写，第二部分数据库设计及应用基础的第 8 至 10 章由赵宏编写，第三部分 Visual Basic 程序设计基础的第 11 至 18 章由王刚编写。全书由沈朝辉统稿，沈琴婉教授主审。

在编写本书的同时，我们还编写了与之配套的《数据结构与数据库应用基础实习指导与习题集》一书。

本书编写过程中，得到南开大学信息学院计算机基础教学部的沈琴婉、周玉龙等老师和南开大学出版社张蓓同志的大力支持和帮助，责任编辑李冰对书稿提出了许多宝贵意见，并对书中的错误一一加以改正，在此一并致以诚挚的感谢！

由于编者水平所限，书中错误与不妥之处，敬请读者批评指正。

编 者

2006 年 12 月

E-mail 地址：shenzh@nankai.edu.cn

目　录

第一部分　数据结构与算法基础

第二部分 数据库设计及应用基础

第三部分 Visual Basic 程序设计基础

第一部分　数据结构与算法基础

第1章　算　法

本章主要介绍算法的基本概念、算法复杂度和算法的描述方式。

1.1　算法的基本概念

所谓算法是指解题方案的准确而完整的描述。

对于一个问题，如果可以通过一个计算机程序，在有限的存储空间内，运行有限长的时间而得到正确的结果，则称这个问题是算法可解的。但算法不等于程序，也不等于计算方法。当然，程序也可以作为算法的一种描述，但程序通常还需考虑很多与方法和分析有关的细节问题。为了设计出高质量的解题方案，首先要熟悉算法的基本特征、算法基本要素和算法设计的基本方法。

1. 算法的基本特征

算法应具有可行性、确定性、有穷性和拥有足够的情报等基本特征。

（1）可行性：指算法描述的每一个步骤必须能够实现，且能够达到预期的目的。

（2）确定性：指算法中的每一个步骤，必须经过明确的定义，不允许有二义性。

（3）有穷性：指算法必须能在有限的时间内做完，即算法必须能在执行有限个步骤之后终止。

（4）拥有足够的情报：一个算法是否有效，还取决于为算法所提供的情报是否足够。通常，算法中的各种运算总是要施加到各个运算对象上，而这些运算又可能具有某种初始状态，这是算法执行的起点或是依据。因此，一个算法执行的结果，总是与输入的初始数据有关，不同的输入将会有不同的结果输出。当输入不够或输入错误时，算法本身也就无法执行或导致执行出错。一般来说，当一个算法拥有足够的情报时，此算法才是有效的。

2. 算法的基本要素

一个算法通常由两种基本要素组成：一是对数据对象的运算和操作；二是算法的控制结构。

（1）算法中对数据对象的运算和操作

通常，计算机可以执行的基本运算和操作有以下四类：

①算术运算：主要包括加、减、乘、除等运算；

②逻辑运算：主要包括“与”、“或”、“非”等运算；

③关系运算：主要包括“大于”、“小于”、“等于”、“不等于”等运算；

④数据传输：主要包括赋值、输入、输出等操作。

算法的设计一般都应从上述四种基本操作考虑，按解题要求从这些基本操作中选择合适的操作组成解题的操作序列。

（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。算法的控制结构给出了算法的基本框架，它不仅决定了算法中各操作的执行顺序，而且也直接反映了算法的设计是否符合结构化原则。描述算法的工具通常有传统流程图、N-S 结构化流程图、算法描述语言等。一个算法一般都可以用顺序、选择、循环三种基本控制结构组合而成。

3．算法设计的基本方法

算法设计的基本方法有列举法、归纳法、递推、递归、减半递推技术和回溯法。

（1）列举法

列举法的基本思想是，根据提出的问题，列举所有的情况，并用问题中给定的条件检验哪些是需要的，哪些是不需要的。因此，列举法常用于解决“是否存在”或“有多少可能”等类型的问题，例如求解不定方程的问题。

通常，在设计列举算法时应尽可能减少列举量，只要对实际问题进行详细的分析，将与问题有关的知识条理化、完备化、系统化，从中找出规律，就可以大大减少列举量。

（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。由于在归纳的过程中往往不可能对所有的情况进行列举，因此，最后由归纳得到的结论还只是一种猜测，还需要对这种猜测加以必要的证明。

（3）递推

所谓“递推”，是指从已知的初始条件出发，逐次推出所要求的各中间结果和最后结果。其中初始条件或是问题本身已经给定，或是通过对问题的分析与化简而确定。

（4）递归

人们在解决一些复杂问题时，为了降低问题的复杂程度，一般总是将问题逐层分解，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，实际上并没有对问题进行求解，而只是当解决了最后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，这就是递归的基本思想。

递归分为直接递归与间接递归两种。如果一个算法 P 显式地调用自己，则称为直接递归。如果算法 P 调用另一个算法 Q，而算法 Q 又调用 P，则称为间接递归。

（5）减半递推技术

所谓“减半”，是指将问题的规模减半，而问题的性质不变；这里的所谓“递推”，是指重复“减半”的过程。

（6）回溯法

回溯法的基本思想是通过对问题的分析，找出一个解决问题的线索，然后沿着这个线索逐步试探，对于每一步的试探，若试探成功，就得到问题的解；若试探失败，就逐步回退，换别的路线再进行试探。

1.2 算法复杂度及算法的描述方式

一个算法的优劣，除了与算法复杂度有关外，还与描述方式有牵连。本节主要介绍算法复杂度并简要介绍算法的描述方式。

1. 算法复杂度

算法复杂度主要包括算法的时间复杂度和算法的空间复杂度。

（1）算法的时间复杂度

所谓算法的时间复杂度，是指执行算法所需要的计算工作量。

算法的工作量，一般用算法所执行的基本运算次数来度量，而算法所执行的基本运算次数是问题规模的函数，即

算法的工作量＝T(n)

其中，n 是问题的规模。例如，两个 n 阶矩阵相乘，所需要的基本运算（即两个实数的乘法）次数是 n 的函数 T(n)。

用 C/C++描述的两个 n 阶矩阵 a[][]与 b[][]相乘的结果，存放到一个 n 阶矩阵 c[][]中的算法如下：

```
for(i=0;i<n;i++)                    //执行 n+1 次
  for(j=0;j<n;j++)                  //执行 n(n+1)次
    {
      c[i][j]=0;                    //执行 n^2 次
      for(k=0;k<n;k++)              //执行 n^2(n+1)次
        c[i][j]=c[i][j]+a[i][k]*b[k][j];   //执行 n^3 次
    }
```

一条语句在算法中被重复执行的次数，称为该语句的频度。

分析两个 n 阶矩阵相乘的算法可知，外层循环的控制变量 i 从 0 增加到 n 时，循环才会终止，故其频度为 n+1；第二层循环，是外循环的循环体，应该执行 n 次，而第二层循环本身要执行 n+1 次，故第二层循环的频度为 n(n+1)，语句 c[i][j]=0 的频度为 n^2；内循环又是第二层循环的循环体，应该执行 n^2 次，而内循环本身要执行 n+1 次，故内循环的频度是 $n^2(n+1)$，语句 c[i][j]=c[i][j]+a[i][k]*b[k][j]的频度为 n^3。该算法的所有语句的频度之和 $T(n)=2n^3+3n^2+2n+1$。

这里，$T(n)=2n^3+3n^2+2n+1$ 是 n 的函数，n 是两个 n 阶矩阵相乘问题的规模。

显然，$T(n)/n^3 \to 2$ （当 $n \to \infty$ 时）。

由高等数学中无穷大的比较的知识知，T(n)与 n^3 的量级（数量级）相同（即同为 3 阶无穷大），也就是说，时间复杂度 T(n)的量级为 $O(n^3)$，记为 $T(n)=O(n^3)$。

当问题的规模 n 趋向无穷大时，时间复杂度 T(n)的量级（阶）称为算法渐近时间复杂度。

显然，时间复杂度 T(n)的量级，实际上是由频度最大的语句（高次项）决定的。故在具体分析时，可做如下处理：

① 若语句很少执行且与规模 n 无关，则可忽略不计；

② 若所有语句都与规模 n 无关，则即使有上千条语句，其执行时间也不过是一个较大

的常数，故时间复杂度的量级也只是 $O(n^0)=O(1)$；

③ 一般可只考虑与程序规模有关的频度最大的语句，如循环语句的循环体、多重循环的内循环等。

在很多情况下，我们在讨论算法的时间复杂度时，没有必要精确计算出算法的执行次数（即频度），只要大致计算出算法时间复杂度的量级就可以了。常见时间复杂度的量级有：常数阶 $O(1)$、线性阶 $O(n)$、平方阶 $O(n^2)$、指数阶 $O(2^n)$、对数阶 $O(\log_2 n)$、线性对数阶 $O(n\log_2 n)$。通常认为，具有指数阶量级的算法是实际不可计算的，而量级低于平方阶的算法是高效率的。

注意：今后若无特别说明，就将算法的时间复杂度量级近似地看作算法的时间复杂度。也就是说，一般情况下算法的时间复杂度是指算法的时间复杂度量级。

【例 1-1】说明下列各个程序段的时间复杂度。

①
```
t=a;   //交换 a 和 b 的内容
a=b;
b=a;
```

②
```
sum=0;  //求 n 以内所有 2 的幂次数的和，即 1+2^1+2^2+…+2^k，2^k≤n
for(i=1;i<=n;i*=2)
    sum+=i;
```

③
```
for(i=0;i<n;i++)     //给二维数组 a[][]赋值 i+j
  for(j=0;j<n;j++)
     a[i][j]=i+j;
```

说明如下：

在程序段①中，三条语句的执行次数均为 1，与规模 n 无关，故可知其时间复杂度为 $O(1)$。

在程序段②中，执行次数最多的语句是循环体 sum+=i，它执行的次数未知，显然不是 n 次，若设为 k 次，由于 $2^k \leq n$，所以有 $k \leq \log_2 n$ 故时间复杂度为 $O(\log_2 n)$。

在程序段③中，执行次数最多的语句是内循环的循环体 a[i][j]=i+j，该语句执行了 n^2 次，所以时间复杂度为 $O(n^2)$。

在具体分析一个算法的工作量时，还会存在这样的问题：对于一个固定的规模，算法所执行的基本运算次数，还可能与特定的输入有关，而实际上又不可能将所有可能情况的算法所执行的基本运算次数都列举出来。例如：在长度为 n 的一维数组中，查找值为 x 的元素，若采用顺序搜索法，即从数组的第一个元素开始，逐个与被查找值 x 进行比较。显然，若第一个元素值恰好等于 x，则只需比较一次，就得到结果；若最后一个元素值等于 x，或所有元素的值都不等于 x，则需要比较 n 次，才能得到结果。因此，在这个问题的算法中，其基本运算（即比较）的次数与具体的被查找（输入）值 x 有关。考虑到这种情况，通常采用平均性态和最坏情况复杂性两种方式，来确定一个算法的工作量。

① 平均性态

所谓平均性态，是指在规模为 n 时，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（即输入为 x 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：

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

其中，D_n 表示当规模为 n 时，算法执行时所有可能输入的集合。这个式子中的 t(x) 可以通过分析算法来加以确定，而 p(x) 必须由经验或用算法中有关的一些特定信息来确定，通常是不能解析地加以计算的。如果确定 p(x) 比较困难，则会给平均性态分析带来困难。

② 最坏情况复杂性(Worst-Case Complexity)

所谓最坏情况复杂性，是指在规模为 n 时，算法所执行的基本运算的最大次数，定义为

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

显然，W(n) 的计算要比 A(n) 的计算方便得多。由于 W(n) 实际上是给出了算法工作量的一个上界，因此，它比 A(n) 更具有实用价值。

下面通过一个例子来说明算法复杂度的平均性态与最坏情况复杂性。

【例 1-2】采用顺序搜索算法（即从数组的第一个元素开始，逐个与被查找值 x 进行比较）从长度为 n 的一维数组中查找值为 x 的元素。试用平均性态和最坏情况复杂性两种方式，分析确定顺序搜索算法的工作量。

首先考虑平均性态方式。

设被查找项 x 在数组中出现的概率为 q。当需要查找的 x 为数组中第 i 个元素时，则在查找过程中，需要做 i 次比较；当需要查找的 x 不在数组中时（即数组中没有 x 这个元素），则需要与数组中所有的元素进行比较。即

$$t_i=\begin{cases} i, & 1\leqslant i\leqslant n \\ n, & i=n+1 \end{cases}$$

其中 i=n+1 表示 x 不在数组中的情况。

假设需要查找的 x 出现在数组中每个位置上的可能性是一样的，则 x 出现在数组中每一个位置上的概率为 q/n（因为前面已经假设 x 在数组中的概率为 q），而 x 不在数组中的概率为 1-q。即

$$p_i=\begin{cases} q/n, & 1\leqslant i\leqslant n \\ 1-q, & i=n+1 \end{cases}$$

其中 i=n+1 表示 x 不在数组中的情况。

因此，用顺序搜索法，在长度为 n 的一维数组中，查找值为 x 的元素，在平均情况下需要做的比较次数为

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

如果已知需要查找的 x 一定在数组中，此时 q=1，则 A(n)=(n+1)/2。这就是说，在这种情况下，用顺序搜索法在长度为 n 的一维数组中查找值为 x 的元素，在平均情况下需要检查数组中一半的元素。

如果已知需要查找的 x 有一半的机会在数组中，此时 q=1/2，则

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