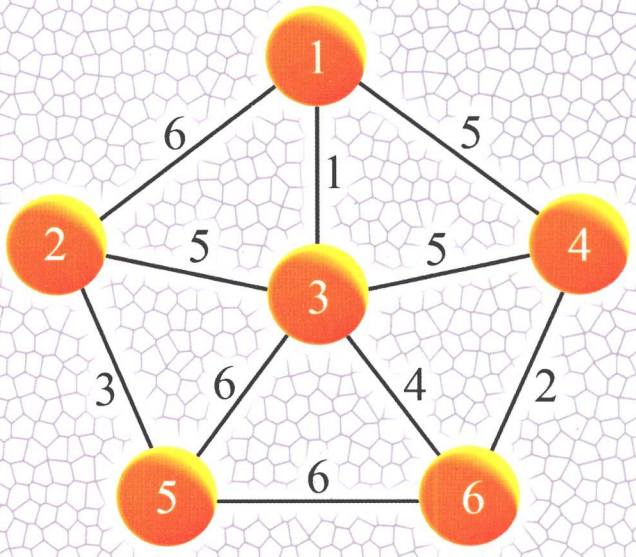
SHUJU JIEGOU JIEXI XITI KECHENG SHEJI

数据结构

解析·习题·课程设计

苏仕华　贾伯琪　黄学俊　编著



中国科学技术大学出版社

数据结构

解析·习题·课程设计

苏仕华　贾伯琪　黄学俊　编著

SHUJU JIEGOU

JIEXI XITI KECHENG SHEJI

中国科学技术大学出版社

内容简介

本书是依据“数据结构”课程教学大纲和实验要求编写的，先对知识点内容进行解析，对解题思路进行分析，再给出同步练习。附录部分给出了同步练习参考答案、课程设计实例、模拟试题及参考答案、部分自学考试试题。

本书可作为学习“数据结构”课程的辅导教材，也可作为广大自学者的参考用书。

图书在版编目(CIP)数据

数据结构：解析·习题·课程设计/苏仕华，贾伯琪，黄学俊编著. —合肥：中国科学技术大学出版社，2009.9

ISBN 978-7-312-02624-9

Ⅰ.数… Ⅱ.①苏…②贾…③黄… Ⅲ.数据结构—高等学校—教学参考资料 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2009)第 165896 号

出版 中国科学技术大学出版社

安徽省合肥市金寨路 96 号，邮编：230026

http://press.ustc.edu.cn

印刷 合肥现代印务有限公司

发行 中国科学技术大学出版社

经销 全国新华书店

开本 710 mm×960 mm 1/16

印张 23.5

字数 453 千

版次 2009 年 9 月第 1 版

印次 2009 年 9 月第 1 次印刷

定价 35.00 元

前　言

“数据结构”是计算机专业的必修、主干课程之一，也是信息等相关专业的必修课程。它旨在使读者学习分析研究计算机加工的数据对象的特性，学会数据的组织方法，以便选择合适的数据逻辑结构和存储结构以及相应的运算（操作），将现实世界中的问题在计算机中表示和处理，这是进一步训练良好的程序设计技能的过程。在教学和学习过程中，解题能力和技巧的训练是一个重要的环节。为了帮助学习这门课程的读者，在学好和巩固知识的同时，加强实践动手能力的训练以及应对各种考试，我们编写了本书。

笔者在长期讲授“数据结构”这门课程中体会到，每次在布置课程设计或作业时，总是要苦思冥想，选什么样的题，如何评分等难以定夺，为此可能要伤透脑筋；而读者往往在如何利用书本中的基本知识和方法解决一些实际问题以及对问题的求解进行算法设计时感到难以下手。实践证明，在理解课程内容与能够较好地解决实际问题之间存在着明显差距，而算法设计完成的质量与基本程序设计素质的培养是密切相关的。要想理解和巩固所学的基本概念、原理和方法，牢固地掌握所学的基本知识、基本技能，达到融会贯通、举一反三的目的，就必须多做、多练、多见（见多识广）。正是为了达到上述目的，在本书中用一些实际的应用，对一些重要的数据结构和算法进行解读。经过循序渐进地训练，读者就可以掌握更多的程序设计技巧和方法，提高分析问题解决问题的能力。

本书遵循“数据结构”课程的教学大纲要求，内容共分 10 章，每章先介绍知识结构，再进行知识解析与解题思路的分析，并给出同步练习题及与本章内容相关的课程设计实验题；在附录中分别给出同步练习的参考答案、课程设计实验题解实例以及多套模拟试题及参考答案。本书也是中国科学技术大学出版社出版的《数据结构与算法解析》一书的配套教学参考书。

为了提高学生分析问题解决问题的能力，除了对主要知识进行了分析外，每章中都选择了一个或两个课程设计实例，这些设计内容丰富、涉及面广、难易适当，给学习“数据结构”这门课程的读者以启发，以达到掌握相关知识和开阔视野的目的。大部分设计题目都做了解析，并给出了参考算法和源程序代码，放在附录中。

本书由苏仕华承担主要编写工作，参加本书编写工作的还有贾伯琪、黄学

俊等。

在本书的编写过程中，中国科学技术大学计算机系黄刘生教授、自动化系刘振安教授对本书提出了许多宝贵的意见和建议，另外在本书中还参考了大量作者的书籍和资料，笔者在此一并致以诚挚的谢意。

由于作者水平所限，加之时间仓促，书中难免存在一些缺点和错误，殷切希望广大读者及同行们批评指正。

编　者

目　录

第1章 概 论

本章的学习任务主要是熟悉各名词、术语的含义；掌握各种基本概念，特别是数据结构的逻辑结构、存储结构、运算3方面的内容以及这3方面的相互关系；熟悉C语言的书写规范；理解算法的5个要素的确切含义，即有穷性、确定性、可行性、输入和输出，从而掌握计算语句频度和估算算法时间复杂度的方法等，为学习数据结构打下基础。

1.1 知识解析

本章主要介绍数据、数据元素、数据对象、数据结构、存储结构和数据类型等概念；算法设计的基本要求；算法描述、算法分析以及估算算法时间复杂度的方法。本章是全书的开始，基本概念较多，但容易理解，所以对其中的内容一般常以单选题和填空题的形式进行考核；而算法时间复杂度的求解多半是与算法设计和分析联系在一起考核。

1.1.1 基础知识

数据(Data)是信息的载体，也是计算机程序加工的“原料”，它能够被计算机识别、存储和加工处理。随着计算机软件、硬件的发展以及计算机应用领域的扩大，数据的含义也随之拓广了，它不仅仅可以是数字和字符串，而且还可以是图形、图像、声音等。

数据元素(Data Element)是数据的基本单位。数据元素也称为元素、结点、顶点或记录。有时一个数据元素可以由若干个数据项(也可称为字段、域、属性)组成，数据项是具有独立含义的最小标识单位。

数据结构(Data Structure)指的是数据之间的相互关系，即数据的组织形式。虽然至今没有一个关于数据结构的标准定义，但它一般包括以下3个方面的内容：

◆ 数据元素之间的逻辑关系，也称为数据的逻辑结构；

◆ 数据元素及其关系在计算机存储器内的表示，称为数据的存储结构；

◆ 数据的运算，即对数据施加的操作。

数据的逻辑结构是从逻辑关系上描述数据，它与数据的存储无关，是独立于计算机的。数据的存储结构是逻辑结构用计算机语言的实现(亦称为映象)，它是依赖于计算机语言的。数据的运算是定义在数据的逻辑结构上的，每种逻辑结构都有一个运算的集合，最常用的运算有检索、插入、删除、更新和排序等。

数据的逻辑结构有两大类：线性结构和非线性结构。

(1) 线性结构

线性结构的逻辑特征是：若结构是非空集，则有且仅有一个开始结点和一个终端结点，并且所有结点都最多只有一个直接前趋和一个直接后继。如线性表就是一个典型的线性结构。

(2) 非线性结构

非线性结构的逻辑特征是：一个结点可能有多个直接前趋或多个直接后继。如树形结构和图形结构就是典型的非线性结构。

数据的存储结构可用以下 4 种基本的存储方法实现：

(1) 顺序存储方法

该方法是把逻辑上相邻的结点存储在物理位置上相邻的存储单元里，结点间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为顺序存储结构。

(2) 链接存储方法

该方法不要求逻辑上相邻的结点在物理位置上亦相邻，结点间的逻辑关系是由附加的指针字段表示的。由此得到的存储表示称为链式存储结构，通常要借助于程序语言的指针类型来描述它。

(3) 索引存储方法

该方法通常是在存储结点信息的同时，还建立附加的索引表。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)，关键字是能惟一标识一个结点的那些数据项。若每个结点在索引表中都有一个索引项，则该索引表称之为稠密索引。若一组结点在索引表中只对应一个索引项，则该索引表称为稀疏索引。

(4) 散列存储方法

该方法的基本思想是根据结点的关键字，通过一个散列函数直接计算出该结点的存储地址。

1.1.2 算法描述

研究数据结构的目的在于更好地进行程序设计，而程序设计离不开数据的运算，这种运算的过程（或解题的方法）通常称为**算法**。

数据的运算是通过算法描述的。通俗地说，一个算法就是一种解题的方法。更严格地说，算法是由若干条指令组成的有穷序列，其中每条指令表示一个或多个操作。它必须满足以下5个准则：

① 输入：算法开始前必须给算法中用到的变量初始化，即一个算法的输入可以包含零个或多个数据。

② 输出：算法至少有一个或多个输出。

③ 有穷性：算法中每一条指令的执行次数是有限的，即算法必须在执行有限步后结束。

④ 确定性：算法中每一条指令的含义都必须明确，无二义性。

⑤ 可行性：算法是可行的，即算法中描述的操作都可以通过有限次的基本运算来实现。

因此，一个程序如果对任何输入都不会陷入无限循环，则它就是一个算法。

算法的含义与程序十分相似，但两者是有区别的。例如，一个程序就不一定满足有穷性。

一个算法可用自然语言、数学语言或约定的符号语言来描述。目前主要用两类语言描述算法，一种是类PASCAL；另一种是类C，类似于C语言，而又不完全相同。例如，要编写一个求n!（n的阶乘）的算法，实际上就是写一个C语言函数：

```
float fact(int n)
{ //求n! 就是从1开始连乘至n,即n! =1×2×3×…×n
   int i;
   float k=1.0;
   for(i=1;i<=n;i++)
     k=k*i;
   return k;
}
```

1.1.3 算法分析

求解一个问题可能有多种不同的算法，那么如何来评价这些算法的优劣好坏而从中选择好的算法呢？显然，算法的“正确性”是首先要考虑的。所谓一个算法

的正确性，是指对于一切合法的输入数据，该算法经过有限时间的执行都能得到正确的结果。除正确性外，还要考虑如下几点：

◆ 执行算法所耗费的时间，即时间复杂性；

◆ 执行算法所耗费的存储空间，主要是辅助空间，即空间复杂性；

◆ 算法应易于理解、易于编程、易于调试等，即可读性和可操作性。

其中，最主要的是时间复杂性。一个算法所耗费的时间，应是该算法中每条语句的执行时间之和，而每条语句的执行时间就是该语句的执行次数（也称频度）与该语句执行一次所需时间的乘积。

算法所耗费的时间采用**时间复杂度**来度量，一般不必精确计算出算法的时间复杂度，只要大致计算出相应的数量级，如 $O(1)$、$O(\log_2 n)$、$O(n)$或 $O(n^2)$等。

例如，计算下面算法的时间复杂度：

```
(1) for(i=0;i<n-1;i++)          n次
(2)   for(j=i+1;j<n;j++)        n(n-i)次
(3)     if(R[i]>R[j])           n(n-i-1)次
(4)     {  temp=R[i];
(5)        R[i]=R[j];
(6)        R[j]=temp;
        }
```

以上程序段中频度最大的语句是(3)，其频度为 $f(n)=n(n-i-1)$，所以该程序段的时间复杂度为 $T(n)\approx O(n^2)$。

一般地，常用的时间复杂度有如下关系：

$$O(1)\leqslant O(\log_2 n)\leqslant O(n)\leqslant O(n\log_2 n)\leqslant O(n^2)\leqslant\cdots\leqslant O(n^k)\leqslant O(2^n)$$

一个算法占用的空间是指在计算机存储器上所占用的存储空间，主要考虑在算法过程中临时占用的存储空间的大小，称之为**空间复杂度**，一般以数量级形式给出。

简单性是指算法的易读性等。

对于较复杂的算法，我们可以将它分成几个容易估算的部分，然后利用“O”的求和原则和乘法原则计算整个算法的时间复杂度。

大“O”下的求和准则：若算法两部分的时间复杂度分别为 $T_1(n)=O(f(n))$和 $T_2(n)=O(g(n))$，则 $T_1+T_2=O(\max(f(n),g(n)))$；又若 $T_1(m)=O(f(m))$，$T_2(n)=O(g(n))$，则 $T_1+T_2=O(f(m)+g(n))$。

大“O”下的乘法准则：若算法两部分的时间复杂度分别为 $T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$，则 $T_1\cdot T_2=O(f(n)\cdot g(n))$。

1.2 解题思路

本章概念较多，但比较容易掌握，难点是如何分析算法的时间复杂度。下面将通过实例来解析算法分析的思路。

【例 1.1】 分析计算下面程序段的时间复杂度。

```
(1)   s=0;
(2)   for (i=1;i<=n;i++)
(3)     for (j=1;j<=n;j++)
(4)       s=s+1;
```

【分析】 按常规求算法时间复杂度，该算法的第(1)行频度为 1；第(2)、(3)、(4)行的频度分别为 $n+1$、$n(n+1)$、n^2，因此算法中所有频度之和为 $T(n)=n^2+(n+1)+n(n+1)+1=2n^2+2n+2$，即 $f(n)=n^2$，所以该算法的时间复杂度为 $T(n)=O(f(n))=O(n^2)$。

我们知道，执行一条赋值语句与 n 无关，时间复杂度为 $O(1)$，因此 $T_1(n)=O(1)$；对于第(3)条语句，$T_3(n)=O(n)$，如果利用前面的求和准则，有 $T_4(n)+T_3(n)=O(n)$；第(2)条语句 $T_2(n)=O(n)$，而(2)与(3)、(4)是循环嵌套，故有 $T_2(n)*(T_3(n)+T_4(n))=O(n^2)$；对第(1)条语句，$T_1(n)=O(1)$，它与其他语句之间的关系是顺序执行，所以该程序段的算法时间复杂度是 $T_1(n)+T_2(n)*(T_3(n)+T_4(n))=O(n^2)$，与前面常规方法求的完全一样。

【例 1.2】 求下面程序段的算法时间复杂度。

```
x=0;
for (i=2;i<=n;i++)
  for (j=2;j<=i-1;j++)
    x=x+1;
```

【分析】 由于算法的时间复杂度考虑的只是对于问题规模 n 的增长率，则在难以精确计算基本操作执行次数(或语句频度)的情况下，只需要求出它关于 n 的增长率或阶即可。上述程序段中，语句 x=x+1 执行次数关于 n 的增长率为 n^2，它是语句频度表达式 $(n-1)(n-2)/2$ 中增长最快的项，所以该程序段的算法时间复杂度为 $O(n^2)$。

【例 1.3】 求下面的起泡排序算法的时间复杂度。

```
change=1;
for(i=1;i<=n && change;i++)  {
```

```
    change=0;
    for (j=1;j<=n-i;j++)
      if (a[j]>a[j+1]){ a[j]与a[j+1]交换;change=1;}
}
```

【分析】 起泡排序是以"交换序列中相邻两个数"为基本操作。当a中初始序列为自小至大有序时，基本操作的执行次数为0，但循环还是执行了n-1次；当初始序列为自大至小有序时，基本操作的执行次数为n(n-1)/2。对这类算法的分析，一种解决的办法是计算它的平均值，即考虑它对所有可能的输入数据集的期望值，此时相应的时间复杂度为算法的平均时间复杂度。如假设a中初始输入数据可能出现n!种排列情况的概率相等，则起泡排序的平均时间复杂度$T_{avg}(n)=O(n^2)$。然而在很多情况下，各种输入数据集出现的概率难以确定，算法的平均时间复杂度也就难以确定。另一种更可行的也是更常用的办法是讨论算法在最坏情况下的时间复杂度。例如，上述起泡排序的最坏情况为a中初始序列为自大至小有序，则起泡排序算法在最坏情况下的时间复杂度为$T(n)=O(n^2)$。在本书以后各章中讨论的时间复杂度，除特别指明外，都是指最坏情况下的时间复杂度。

【例1.4】 计算判断一个正整数是否为素数的算法的时间复杂度。

```
i=2;
while((n/i ! =0 ) && (i< sqrt(n)))
  i++;
if(i > sqrt(n))
  printf("%d is peime".n) ;
else
  printf("%d is not prime",n);
```

【分析】 算法的时间复杂度是由嵌套最深层语句的频度决定的，判断是否是素数算法的嵌套最深层语句为"i++;"，它的频度由条件"((n/i ! =0)&&(i<sqrt(n)))"决定，显然i<sqrt(n)，即执行频度小于sqrt(n)，所以其时间复杂度是$O(\sqrt{n})$。

【例1.5】 分析计算n!的递归函数的时间复杂度。

```
float fact ( int n)
{
   if (n<=1)
     return 1;              // ①
   else
     return n*fact(n-1);    // ②
}
```

【分析】 设fact(n)的运行时间函数是T(n)，该算法中语句①的运行时间是

O(1)，语句②的运行时间是 T(n−1)+O(1)。因此有

$$T(n)=\begin{cases}O(1) & n\leqslant 1\\ T(n-1)+O(1) & n>1\end{cases}$$

所以有

$$\begin{aligned}T(n)&=O(1)+T(n-1)\\&=2*O(1)+T(n-2)\\&=\cdots\\&=(n-1)*O(1)+T(1)\\&=n*O(1)=O(n)\end{aligned}$$

因此，算法 fact(n)的时间复杂度为 O(n)。

1.3 同步练习 1

一、单项选择题

1. 在数据结构中，从逻辑上可以把数据结构分为(　　)。

A. 紧凑结构和非紧凑结构　　B. 线性结构和非线性结构

C. 内部结构和外部结构　　D. 动态结构和静态结构

2. 若结点的存储地址与其关键字之间存在某种映射关系，则称这种存储结构为(　　)。

A. 顺序存储结构　　B. 链式存储结构

C. 索引存储结构　　D. 散列存储结构

3. 算法分析的两个主要方面是(　　)。

A. 正确性和简明性　　B. 时间复杂性和空间复杂性

C. 可读性和可维护性　　D. 数据复杂性和程序复杂性

4. 线性表采用链式存储结构存储时，要求内存中可用存储单元地址(　　)。

A. 可以是不连续的　　B. 部分地址必须是连续的

C. 必须是连续的　　D. 一定是不连续的

5. 算法指的是(　　)。

A. 计算机程序　　B. 解决问题的计算方法

C. 解决问题的有限运算序列　　D. 排序算法

二、填空题

6. 数据结构一般包括_____、_____和数据运算 3 个方面的内容。

7. 数据的逻辑结构可分为_____、_____两大类。

8. 数据的存储结构(物理结构)可以用_____、_____、_____及散列存储 4 种存储方法表示。

9. 选用算法除了首先考虑"正确性"外,还要考虑_____,_____,算法应易于理解、易于编程、易于调试 3 点。

10. 设有一批数据元素,为了最快地存取某元素,宜用_____结构存储;为了方便地插入一个元素,宜用_____结构存储。

三、应用题

设 n 为正整数,分别写出下面各程序段的时间复杂度。

11.
```
for (i=1;i<=n;i++)
{ y=y+1;
   for (j=1;j<=2*n;j++)
    x=x+1;
  }
```

12.
```
s=0;
while (n>=(s+1)*(s+1))
  s=s+1;
```

13.
```
x=1; sum=0;
for (i=1; i<=n; i++)
{ x=x*i;
  sum=sum+x;
}
```

14.
```
for(i=1;i<=n;i++)
    if(3*i<=n)
       for(j=3*i;j<=n;j++) {
        x++;
        y=3*x+2;
       }
```

15.
```
for(i=1;i<=n;i++)
     for(j=1;j<=i;j++)
        x=x+1;
          …
```

第2章 线 性 表

本章主要介绍线性表的逻辑结构及其存储表示方法，以及定义在逻辑结构上的各种基本运算及其在存储结构上如何实现这些基本运算。要求在熟悉这些内容的基础上，能够针对具体应用问题的要求和性质，选择合适的存储结构，设计出相应的有效算法，解决与线性表相关的实际问题。

2.1 知 识 结 构

对本章的考核内容要求较高，除对线性表的逻辑结构特征、线性表上定义的基本运算等内容要求了解之外，要求达到“综合应用”层次的较多，包括如下几个方面：

(1) 顺序表的含义及特点，即顺序表如何反映线性表中元素之间的逻辑关系；

(2) 顺序表上的插入、删除操作及其平均时间性能分析；

(3) 利用顺序表设计算法解决简单的应用问题；

(4) 链表如何表示线性表中元素之间的逻辑关系；

(5) 链表中头指针和头结点的使用；

(6) 单链表、双链表、循环链表链接方式上的区别；

(7) 单链表上实现的建表、查找、插入、删除以及链表的合并与分解等基本算法，分析其时间复杂度；

(8) 循环链表上尾指针取代头指针的作用，单循环链表上的算法与单链表上相应算法的异同点；

(9) 双链表的定义及其相关的算法；

(10) 利用链表设计算法解决简单的应用问题。

本章内容是全书的学习重点，对它的掌握直接影响后面许多章节的学习，因此本章大多数内容要求达到“综合应用”层次。对本章的学习，主要是通过对线性表的顺序存储和链式存储结构及在其基础上的基本运算算法的理解，并通过实例提