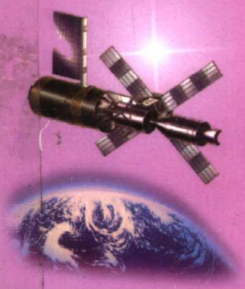

丛书主编　陈松乔

高职高专计算机专业规划教材

数据结构

主　编　余绍军

中南大學出版社

SHUJUJIEGOU

高职高专计算机专业规划教材

丛书主编　陈松乔

数据结构

主　编　余绍军
副主编　刘　佳　陈　畅
编　委　姜　瑜　陈文辉
余绍军　刘　佳
陈　畅

中南大学出版社

图书在版编目(CIP)数据

数据结构/余绍军主编．—长沙：中南大学出版社，2004.8
ISBN 7-81061-917-9

Ⅰ.数… Ⅱ.余… Ⅲ.数据结构 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2004)第 078472 号

数据结构

主　编　余绍军

□责任编辑　李昌佳
□出版发行　中南大学出版社
社址：长沙市麓山南路　邮编：410083
发行科电话：0731-8876770
传真：0731-8710482
□经　　销　湖南省新华书店
□印　　装　中南大学湘雅印刷厂

□开　　本　787×1092 1/16　□印张 13.75　□字数 328 千字
□版　　次　2004 年 8 月第 1 版　□2004 年 8 月第 1 次印刷
□书　　号　ISBN 7-81061-917-9/TP·037
□定　　价　20.00 元

高职高专计算机专业规划教材编委会

前　言

数据结构是计算机专业的一门专业基础课，也是一门核心课程，在计算机专业课程中起着承前启后的作用，数据结构为操作系统、编译原理、数据库系统、计算机图形学、人工智能等后续课程奠定基础。

本书分为8章：第1章为绪论，讨论数据结构和算法的基本概念，以及时间复杂度的估算方法；第2章为线性表，讨论线性表的逻辑结构、线性表的顺序存储结构和链式存储结构、数组和稀疏矩阵的存储结构及其基本操作的实现；第3章为栈和队列，讨论栈和队列的特点及其各种存储结构与基本操作的实现，并给出了相应的应用实例；第4章为串，讨论串的各种存储结构及其基本操作的实现；第5章为树和二叉树，讨论树和二叉树的定义、性质、表示、存储结构以及二叉树的基本操作，讨论哈夫曼树的基本概念及其应用，讨论二叉排序树的概念及基本操作；第6章为图，讨论图的各种存储结构和遍历的实现；第7章为线性表的查找，讨论各种常用的查找方法及其实现；第8章为排序，讨论各种排序方法及其实现。

本书有3个附录，附录一提供了7个实验，附录二给出了各章习题的部分参考答案，附录三提供了历年计算机水平与资格考试的数据结构试题及参考答案。

本书所有的数据结构和算法都采用C语言描述，全部算法都通过了上机调试。对附录一的每个实验都给出了用C语言编写的源程序。对习题中较复杂的算法在附录二中给出了用C语言编写的源程序。

本书在内容的选取、概念的引入、文字的叙述以及例题、习题和实验的选择等方面充分考虑了目前高职高专学生的知识结构、能力结构和素质结构，较好地落实了教育部《关于加强高职高专教育教材建设的若干意见》和《高职高专教育专业人才培养目标及规格》等文件精神，编写时力求做到由浅入深、深入浅出和循序渐进，突出其实用性和应用性，注重培养读者分析问题和解决问题的能力。

本书为高职高专计算机专业教材，也可作为计算机专业的成人教育、自学考试和各类培训班的教材，对从事计算机应用的工程技术人员也是一本十分有价值的参考书。

本书由余绍军任主编，刘佳、陈畅任副主编，其中第1、2章由余绍军编写，第3、4章由刘佳编写，第5章由陈文辉编写，第6章由姜瑜编写，第7、8章由陈畅编写，刘佳对第1~8章中的所有算法进行了统稿，陈畅对各章的习题及附录进行了统稿，全书由余绍军统稿、修改和定稿，姜瑜、陈文辉参加了部分内容的统稿和修改。

由于编者水平有限，书中难免有错误或不当之处，恳请专家和广大读者批评指正。

编　者

2004年7月

目　录

第1章 绪 论

教学目标

通过本章的学习，熟悉各名词和术语的含义，理解数据结构及其有关的概念，了解算法的基本特征和算法的评价标准，掌握估算算法的时间复杂度的方法。

运用计算机对一批数据进行处理时，必须解决好三个方面的问题：第一，根据数据之间的逻辑关系如何组织这批数据；第二，如何将这批数据存储在计算机的存储器中；第三，对于存储在计算机中的这批数据可以进行哪些操作，如何实现这些操作，对同一问题的不同操作方法如何进行评价。这些问题就是数据结构这门课程所要研究的主要问题。在本章先介绍数据结构及其有关的概念，然后讨论算法的基本特征、评价标准及算法的时间复杂度的估算方法。

1.1 数据结构

在阐述数据结构的概念之前，先对数据结构中常用的几个基本概念和术语给出确切的定义。

1.1.1 基本概念和术语

1. 数据

数据是人们利用文字符号以及其他规定的符号对现实世界的事物及其活动所做的抽象描述。或者说数据是能被计算机识别、存储和加工处理的一切信息。数据一般可分为数值型数据和非数值型数据，如数学中的整数、实数等都是数值型数据，字符、表格、图形、图像和声音等都是非数值型数据。

2. 数据元素

数据元素是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。有时

一个数据元素可以由若干个数据项组成，数据项是具有独立含义的最小单位。数据元素有时也叫做结点、元素、顶点、记录等。例如，在数据库管理系统中，数据库中的一条记录就是一个数据元素，这条记录中的每一个字段就是构成这个数据元素的数据项。

3. 数据对象

数据对象是性质相同的数据元素的集合，它是数据的一个子集。

4. 数据类型

数据类型是高级程序设计语言中的一个基本概念，是对数据的取值范围、数据的结构以及允许对这些数据进行的操作的一种描述，它规定了程序中操作对象的特性。或者说数据类型是程序设计语言中各变量可取的数据种类。在程序设计语言中，每个变量、常量或表达式都应该属于某种确定的数据类型。

因此，可以把数据类型定义为：数据类型是由若干个值组成的集合以及定义在这个集合上的一组操作的总称。或者说，数据类型是在程序设计语言中已经实现了的数据结构。实际上，数据类型规定了在程序执行期间变量、常量或表达式所有可能的取值范围，以及允许进行的操作。例如，在 C 语言中，如果说一个变量是整数类型的变量，实际上就规定了这个变量的取值范围只能是[-maxint, maxint]上的整数，其中 maxint 是计算机所允许的最大整数。而且，这个变量在该取值范围内所进行的操作只能是加法、减法、乘法、整除和取余。

数据类型可分为简单类型和结构类型。简单类型中每个数据都是无法再分割的整体，结构类型是由简单类型按照一定的规则构造而成，并且结构类型中还可以包括结构类型。例如，在 C 语言中，整数、实数、字符、指针等都是无法再分割的整体，它们所属的类型都是简单类型；数组是一种结构类型，它是由若干个相同类型的数据顺序排列而成的一种数据类型；结构体也是一种结构类型，它是由若干个不同类型的数据顺序排列而成。

1.1.2 数据结构的定义

什么是数据结构，目前还没有一个公认的定义，对数据结构的概念，可以从以下两个角度来理解：一是把数据结构作为一门独立开设的课程，分析其在计算机专业中的地位、性质、任务以及它研究的内容等；二是把数据结构作为计算机所处理的数据的组织方式来理解。

数据结构作为一门独立的课程在国外是从 1968 年才开始设立的，我国从 1978 年开始，各高等院校才先后开设这门课程，现在数据结构已是计算机专业的一门综合性的专业基础课。它是操作系统、编译原理、数据库系统、计算机图形学、人工智能等课程的先行课。数据结构不仅涉及计算机硬件的研究范围，而且和计算机软件的研究有着密切的关系，可以说，数据结构是介于数学、计算机硬件和计算机软件三者之间的一门综合性课程，可以用一个定义描述如下：数据结构是研究非数值计算的程序设计问题中计算机的操作对象以及它们的关系和操作的一门学科。

如果从数据元素之间的组织形式来看，可以认为数据结构指的是计算机所处理的数据元素之间的组织形式和相互关系。在任何问题中，数据元素都不是孤立存在的，它们之间必定存在着某种内在的或者是根据需要人为定义的关系，这种关系就是这些数据元素的数据结构。

根据数据元素之间的不同特性，可以把数据结构分为四种基本类型：

(1) 集合。集合中的数据元素之间除了“同属于一个集合”的关系外，没有其他任何关系。它是数据结构的一种特例，本书不予讨论。

(2) 线性结构。线性结构中的数据元素之间存在一个一对一的关系，即除了第一个元素外，其他的每个元素都有惟一的一个前驱，除了最后一个元素外，每个元素都有惟一的一个后继。

(3) 树形结构。树形结构中的数据元素之间存在一个一对多的关系，即每个结点最多只有一个前驱，但每个结点都可以有多个后继。

(4) 图形结构。图形结构中的数据元素之间存在多个多对多的关系，即各个结点可以有多个前驱，也可以有多个后继。

数据结构的四种基本类型如图 1 - 1 所示。有时也可以把数据结构分为两种类型：线性结构和非线性结构。非线性结构包括树形结构和图形结构。

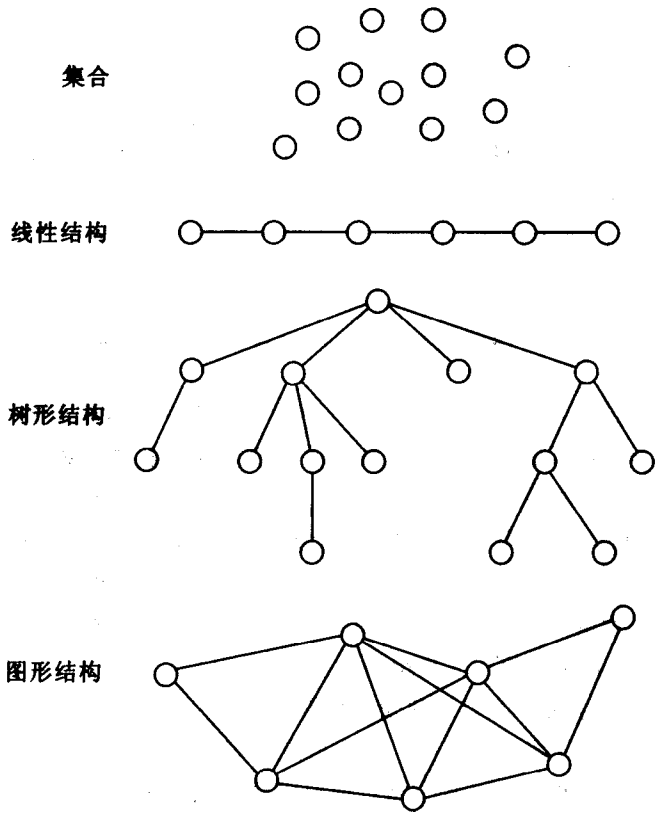

图 1 - 1 四类基本结构关系图

数据结构可分为逻辑数据结构和存储数据结构，也可简称为逻辑结构和存储结构。数据的逻辑结构是指数据结构中数据元素之间的逻辑关系，是用户根据需要而建立起来的。数据的存储结构是指数据元素在计算机的存储器中的存储方式。常用的存储结构有以下四种类型：顺序存储结构、链式存储结构、Hash 存储结构和索引存储结构。顺序存储结构和链式存

储结构是本书讨论的重点，Hash 存储结构将在第 7 章进行讨论，索引存储结构本书不讨论。

由此可见，数据结构一般包括以下三个方面的内容：

(1) 数据元素之间的逻辑关系，即数据的逻辑结构。

(2) 数据元素及其关系在计算机存储器内的表示，即数据的存储结构。

(3) 数据的运算，即对数据元素所进行的操作。

因此，也可以把数据结构定义为：对计算机处理的一批数据，首先按某种逻辑关系把它们组织起来，再按一定的存储表示方式把它们存储在计算机的存储器中，然后给这批数据规定一组操作，这就叫做一个数据结构。

可以通过下面的例题来进一步理解上述概念。

例 1-1 表 1-1 是一张学生成绩表。

表 1-1 学生成绩表

学号	姓名	高等数学	英语	数据结构	程序设计	平均成绩
0205001	刘世清	73	75	80	72	75
0205002	王志伟	68	80	76	80	76
0205003	张小林	80	85	83	88	84
0205004	赵 刚	60	53	67	65	61
…	…	…	…	…	…	…

这张学生成绩表就是一个数据结构，表中的每一行为一个数据元素(或结点、记录、元素)，它由学号、姓名、各科成绩和平均成绩等数据项组成。表中数据元素之间的逻辑关系是一对一的关系，这个数据结构是线性数据结构。将这张表存储在计算机存储器中的方法主要有两种：其一采用顺序存储结构，将表中的每个数据元素按学号的顺序依次存储在一片连续的存储单元中；其二采用链式存储结构，动态地给每一个数据元素分配一个存储单元，然后用指针将每个数据元素链接起来。对这张表的数据元素可以进行查找、插入、删除等操作。

例 1-2 图 1-2 是某学院的人事组织示意图。

图 1-2 中的学院、各系、各教研室及每位职工都是一个数据元素或结点，它们之间是一对多的逻辑结构关系，除了学院结点外，每个结点都有且只有一个直接前驱，表示职工的结点都是终端结点，除了终端结点外每个结点都有多个直接后继，这些结点的逻辑结构就像一棵倒立的树，这种数据结构是一种树形结构。要把这些结点存储在计算机的存储器中不仅要考虑结点的存放顺序，而且还要考虑那些反映职工各种情况的数据项(如姓名、年龄等)，它的存储结构将在第 5 章讨论。

例 1-3 图 1-3 是五个城市之间的航线图，图中的数字表示两城市之间的距离。

图 1-3 中每个城市名就是一个数据元素或顶点，它们之间是一种多对多的逻辑关系，其逻辑结构是图形结构。它的存储结构将在第 6 章讨论。

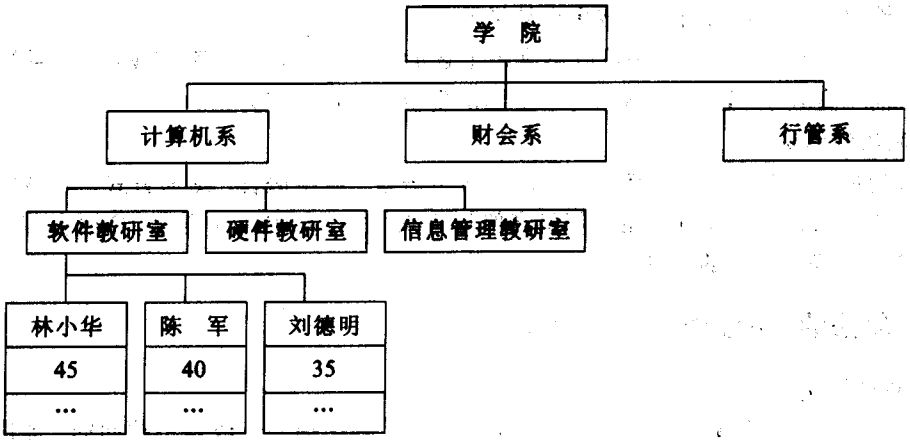

图1-2 人事关系示意图

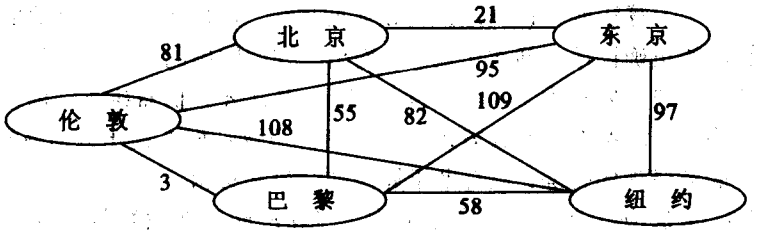

图1-3 城市之间的航线图

1.2 算法

1.2.1 算法的概念及描述

1. 算法的概念

简单地说，算法就是解决特定问题的方法。严格地说，算法是由若干条指令组成的有穷序列，其中每条指令表示计算机的一个或多个操作。例如，将一组给定的数据由小到大进行排序，解决这个问题的方法有若干种，因此每一种排序方法就是一种算法。

一个算法必须具有以下五个特性：

(1) 有穷性。一个算法在执行有限条指令后必须要终止，且每条指令都要在有限时间内完成。

(2) 确定性。算法中每条指令必须有确切的含义，不会产生二义性。

(3) 可行性。一个算法是可行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

(4) 输入。一个算法有零个或多个的输入，这些输入取决于某个特定的对象的集合。

(5) 输出。一个算法有一个或多个结果输出。

算法和程序有所不同，程序可以不满足上述的有穷性。例如，Windows 操作系统在用户未操作之前一直处于“等待”的循环中，直到出现新的用户操作为止。

2. 算法的描述

算法的描述有多种方式。例如，可以用自然语言、数字语言、图形方式、表格方式或其他约定的符号来描述，也可以用计算机高级语言来描述，如 Pascal 语言、C 语言等。本书采用 C 语言作为描述算法的工具。

1.2.2 算法的性能分析

1. 算法的评价标准

对于一个特定的问题，采用不同的存储结构，其算法描述一般是不相同的，即使在同一种存储结构下，也可以采用不同的求解策略，从而有许多不同的算法。那么，对于解决同一问题的不同算法，选择哪一种算法较为合适，以及如何对现有的算法进行改进，从而设计出更好的算法，这就是算法评价问题。评价一个算法的优劣主要有以下几个标准：

(1) 正确性。一个算法能否正确地执行预先的功能，这是评价一个算法的最重要也是最基本的标准。算法的正确性还包括对于输入、输出处理的明确的无歧义性的描述。

(2) 可读性。算法主要是为了人的阅读与交流，其次才是机器执行。即使算法已转变成机器可执行的程序，也需考虑人能较好地阅读理解。可读性好有助于人对算法的理解，这既有助于对算法中隐藏错误的排除，也有助于算法的交流和移植。

(3) 健壮性。算法应具有很强的容错能力，即算法能对非法数据的输入进行检查和处理，不会因非法数据的输入而导致异常中断或死机等现象。

(4) 运行时间。运行时间是指算法在计算机上运行所花费的时间，它等于算法中每条语句执行时间的总和。对于同一个问题如果有多个算法可供选择，应尽可能选择执行时间短的算法。一般来说，执行时间越短，算法的效率越高，性能越好。

(5) 占用空间。占用空间是指算法在计算机存储器上所占用的存储空间，包括存储算法本身所占用的存储空间，算法的输入、输出数据所占用的存储空间和算法运行过程中临时占用的存储空间。算法占用的存储空间是指算法执行过程中所需要的最大存储空间，对于一个问题如果有多个算法可供选择，应尽可能选择存储量需求低的算法。实际上，算法的时间效率和空间效率经常是一对矛盾，相互抵触，有时增加辅助存储空间可以加快算法的运行速度，即用空间换取时间，有时因为内存空间不够，必须压缩辅助存储空间，从而降低了算法的运行速度，即用时间换取空间。

通常把算法在运行过程中临时占用的存储空间的大小叫做算法的空间复杂度。算法的空间复杂度比较容易计算，它主要包括局部变量所占用的存储空间和系统为实现递归所使用的堆栈占用的存储空间。

2. 算法的时间复杂度

一个算法的运行时间是该算法中每条语句执行时间的总和，而每条语句的执行时间是该语句的执行次数(也叫语句频度)与执行一次该语句所需时间的乘积。由于同一条语句在不

同的机器上执行所需的时间是不相同的，也就是说执行一条语句所需的时间与具体的机器有关，所以要想精确地计算各种语句执行一次所需的时间是比较困难的。实际上，为了评价一个算法的性能，只需计算算法中所有语句执行的总次数。

任何一个算法最终都要被分解成一系列基本操作(如赋值、转向、比较、输入、输出等)来具体执行，每一条语句也要分解成具体的基本操作来执行，所以算法的运行时间也可以用算法中所进行的基本操作的总次数来估算。在一个算法中，进行简单操作的次数越少，其运行时间也就相对越少。为了便于比较同一问题的不同算法，也可以用算法中的基本操作重复执行的频度作为算法运行时间的度量标准。

通常把算法中的基本操作重复执行的频度称为算法的时间复杂度。算法中的基本操作一般是指算法中最深层循环内的语句，因此，算法中基本操作重复执行的频度 $T(n)$ 是问题规模 n 的某个函数 $f(n)$，记作：$T(n)=O(f(n))$。其中“O”表示随问题规模 n 的增大，算法执行时间的增长率和 $f(n)$ 的增长率相同，或者说，用“O”符号表示数量级的概念。例如，若 $T(n)=2n^2+3n+1$，则 $2n^2+3n+1$ 的数量级与 n^2 的数量级相同，所以 $T(n)=O(n^2)$。

如果一个算法没有循环语句，则算法中的基本操作的执行频度与问题规模 n 无关，记作 $O(1)$，也称常数阶。如果一个算法只有一重循环，则算法的基本操作的执行频度随问题规模 n 的增大而呈线性增大关系，记作 $O(n)$，也称作线性阶。常用的还有平方阶 $O(n^2)$、立方阶 $O(n^3)$、对数阶 $O(\log_2 n)$、指数阶 $O(2^n)$ 等。

下面举例来说明计算算法时间复杂度的方法。

例 1-4 分析以下程序段的时间复杂度。

```
for(i=1; i<n; i++)
{
  y=y+1;                          ①
  for(j=0; j<=(2*n); j++)
  x++;                            ②
}
```

解 该程序段中语句①的频度是 $n-1$，语句②的频度是 $(n-1)(2n+1)=2n^2-n-1$，则程序段的时间复杂度 $T(n)=(n-1)+(2n^2-n-1)=2n^2-2=O(n^2)$。

例 1-5 分析以下程序段的时间复杂度。

```
i=1;                              ①
while(i<=n)
    i=i*2;                        ②
```

解 该程序段中语句①的频度是 1，设语句②的频度为 $f(n)$，则有 $2^{f(n)}\leqslant n$，即 $f(n)\leqslant \log_2 n$，取最大值 $f(n)=\log_2 n$，所以该程序段的时间复杂度 $T(n)=1+f(n)=1+\log_2 n=O(\log_2 n)$。

例 1-6 分析以下程序段的时间复杂度。

```
a=0; b=1;                         ①
for(i=2; j<=n; i++)               ②
{
```