高等职业学校教材

数据结构

闵 敏 朱辉生 主编

高等教育出版社

高等职业学校教材

数据结构

闵敏　朱辉生　主编

高等教育出版社

内容提要

本书以 Turbo C 为算法描述语言，以数据结构的分类为主线索，讲述了线性表（含堆栈、队列）、串与特殊矩阵、树与二叉树、图形结构及基本的排序和查找方法。

本教材语言浅显简洁，可作为各类高等职业学校计算机及相关专业数据结构课程的教材，也可作为从事计算机工作的技术人员自学或计算机等级考试的参考用书。

图书在版编目（CIP）数据

数据结构/闵敏，朱辉生主编．—北京：高等教育出版社，2001.6

高等职业学校教材

ISBN 7-04-008899-1

Ⅰ.数… Ⅱ.①闵…②朱… Ⅲ.数据结构-高等学校：技术学校-教材 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字（2000）第 86732 号

责任编辑 李 波 封面设计 王凌波
版式设计 马静如 责任校对 王效珍 责任印制 张小强

数据结构
闵敏 朱辉生 主编

出版发行 高等教育出版社
社 址 北京市东城区沙滩后街 55 号 邮政编码 100009
电 话 010-64054588 传 真 010-64014048
网 址 http://www.hep.edu.cn
http://www.hep.com.cn

经 销 忻华书店北京发行所
印 刷 北京市鑫鑫印刷厂

开 本 787×1092 1/16 版 次 2001 年 6 月第 1 版
印 张 10.5 印 次 2001 年 6 月第 1 次印刷
字 数 230 000 定 价 14.80元

前　言

数据结构是计算机及应用专业的一门主干课，它不仅是计算机学科的理论基础之一，也是软件设计的技术基础。它主要研究信息在计算机中的组织和表示方法。无论是硬件和系统软件的设计，还是应用软件的开发，无不涉及数据结构的知识。

数据结构被学生视为较难学的课程之一。适合高等职业学校计算机及应用专业的数据结构教材也为数不多。本教材力图作一些有意义的尝试，本着基础、实用的原则来编排教材内容，以通俗易懂的语言讲解概念和算法，考虑到计算机的发展趋势及学生的接受能力，采用 Turbo C 作为算法的描述语言。为改善学生对该课程存在的“听得懂但不知如何做”的状况，每章后面安排了适合本章内容及特点的习题和实验选题，以供学生课后复习及练习；由于数据结构和程序设计二者之间的关系极为密切，本教材中的算法基本上就是一个完整的 Turbo C 函数，且大多数算法均给出了较为详细的注释以帮助学生阅读和理解。

本书共分八章，分别介绍了线性表、栈和队列、串与特殊矩阵、树和二叉树、图以及常用的排序和查找方法。通过学习，学生应能够理解数据结构的基本概念，掌握线性结构、树形结构及图形结构的基本存储方式、基本算法及简单应用，熟练掌握常用排序和查找的算法，并能进行简单的算法分析。

使用本书进行教学，建议学时分配如下：

章节内容	理论时数	实验时数	总学时数
第一章　绪论	2		2
第二章　线性表	6	4	10
第三章　栈和队列	6	2	8
第四章　串与特殊矩阵	4		4
第五章　树与二叉树	6	4	10
第六章　图	6	2	8
第七章　排序	6	4	10
第八章　查找	6	2	8
复习课(习题课)	4		4
总学时数	46	18	64

本教材由多年从事数据结构教学的老师编写，全书由闵敏任主编，朱辉生任副主编，其中第一章、第五章由闵敏编写，第二章、第四章由刘爱华编写，第三章由洪昕编写，第六章由朱立华编写，第七章、第八章由朱辉生编写。白少布老师审读了全部书稿，并提出了许多宝贵意见和建议。在此表示衷心的感谢。

由于编者水平有限，书中难免存在错误或不妥之处，敬请批评指正。

编者

2000 年 12 月

目　录

第1章 绪 论

【学习目标】

1. 了解数据结构课程的内容及其重要性；

2. 理解与数据结构相关的各个基本概念，尤其是数据的逻辑结构、物理结构及算法的特性及评价标准；

3. 初步掌握算法效率的分析方法。

随着计算机应用领域的不断扩大，软件的开发作为一项系统工程的概念已逐渐被越来越多的人所接受。根据软件工程的实践理论，软件的生存期可分为制定计划、需求分析、软件设计、程序编码、软件测试和运行维护六个阶段。其中软件设计是软件工程的技术核心，是程序编码的基础。而在软件设计过程中，数据结构的设计是很重要的一个方面。所谓数据结构是指数据的各个元素之间某种关系的集合。数据的组织方式及复杂程度可能多种多样；但典型的数据结构种类却是有限的，“数据结构”是一门介绍典型的数据结构及与之相关算法的课程，它主要涉及到软件开发的设计阶段、部分编码阶段及分析阶段。

在现实生活中，有许多的实际应用问题都可以用数据结构的相关概念和算法来解释或描述。例如，欲在若干个城市之间建立一个通信网，假设每两个城市之间的通信线路的成本可以预先估算，则应设计一个方案使得任意两个城市之间均可进行通信且总的成本最低。像这样的一个实际应用问题其实就是数据结构中的“求最小生成树”问题。再比如对于一个酒店管理系统，如果忽略客人的特殊要求，要想使每个客房使用的概率均等，则可将所有客房编成一个队列，每次有客人人住时先从队头选定客房，对于客人已经退出的客房则可将其排到队尾，在这样的一个分配系统中，可最大限度地保证每个客房均有可能有客人人住。“队列”也正是数据结构中所讨论的一种重要的数据结构。再如，一般情况下，在十字路口只需设红、绿两种颜色的交通灯便可维持正常的交通秩序，但在多叉路口，则需设置多种颜色的交通灯。那么应该如何设置这些交通灯，才能既保证车辆不会互相碰撞，同时又能达到车辆的最大流通，要解决这一问题，则必须运用到“数据结构”中称之为“图”的数据结构。

本章主要介绍与数据结构相关的一些概念及对算法进行初步分析的方法，从而为后面各章的学习打下基础。

1.1　数据结构的基本概念

1.1.1　数据、信息及数据的表示与处理

1. 数据与信息

数据是指能被计算机存储、加工的对象。信息则是经计算机加工后产生的有意义的数据。尽管二者有差异，但在习惯上，若不是特别说明，数据与信息经常通用。

2. 数据的表示

数据的表示分为机外表示和机内表示两种。机外表示指数据在实际问题中的表现形式；机内表示则指数据在计算机存储器中的存在形式。通常意义上的数据表示即指将数据从机外表示转化为机内表示。

3. 数据处理

用适当的可执行语句编制程序，以便让计算机去执行对数据机内表示的各种操作，从而实现处理的要求，即得到所需结果，这项工作称为数据处理。事实上，数据表示和数据处理也正是解决具体问题的两个最基本的任务。

4. 数据元素与数据项

数据元素是数据的基本单位，在程序中作为一个整体加以考虑和处理。因此，运算的基本单位是数据元素（又可称为元素、结点、记录等）。数据元素通常又是由若干个数据项构成，数据项（有时也可称为字段、域）是数据的不可分割的最小单位。

事实上，数据、数据元素和数据项反映了数据组织的三个层次。数据可由若干个数据元素构成，而数据元素又可由若干个数据项构成。

例 1－1 学生成绩登记表。

一张学生成绩登记表是由许多学生的成绩记录组成的，每个学生的成绩构成一个记录，即一个数据元素，而每个学生记录中包含的学号、姓名、课程名、分数等则为数据项，也称为字段或数据域。通过计算机实现添加、删除或修改学生的成绩则称为数据处理。

1.1.2　数据的结构

在任何的实际问题中，数据元素都不可能是孤立存在的，它们之间必然存在着某种联系（或称关系），这种联系称为结构。在不同的问题中，数据与数据之间的关系可能是不同的，但从数据结构的观点来看，数据的结构可分为数据的逻辑结构和数据的物理结构。

1. 数据的逻辑结构

数据的逻辑结构就是数据的组织形式，即数据元素之间逻辑关系的整体。而所谓数据的逻辑关系则是指数据元素之间的关联方式或称邻接关系。比如，对于例 1-1 中的学生登记表，数据元素之间的关系是一对一的线性关系。

2. 逻辑结构的分类

根据数据元素之间关系的不同特性，逻辑结构一般可分为：集合、线性结构、树形结构、图形结构四类，图 1－1 为这四类结构的示意图。

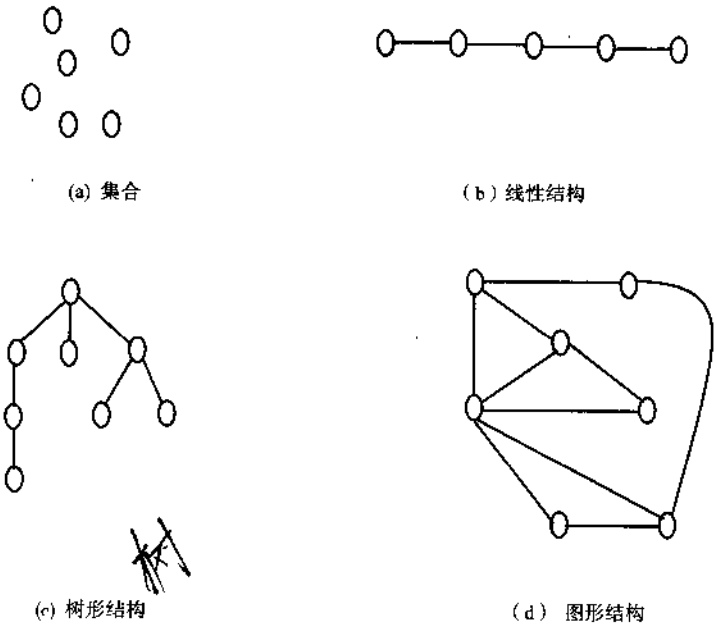

图 1－1 四类逻辑结构示意图

由图 1-1 可知，集合中的任何两个结点（即数据元素）之间均没有逻辑关系，因此是一种松散的组织形式；线性结构中的结点按逻辑关系依次排列，具有一对一的关系（如学生成绩表）；树形结构具有分支、层次的特性，是一对多的关系（如一个部门的行政关系）；图形结构中各数据元素的关系则较复杂，是多对多的关系（如城市交通网）。

说明：

(1) 逻辑结构与其所含结点个数、数据元素本身的形式、内容及元素的相对位置无关。

(2) 在逻辑结构上所进行的操作称为运算。按操作的效果来分，运算可分为加工型（操作改变了逻辑结构，如插入、删除、排序等）和引用型（其操作未改变原逻辑结构，如查找）。

(3) 常见的数据运算有：插入、删除、更新、排序、查找等。

(4) 在不会引起混淆的情况下，通常将数据的逻辑结构简称为数据结构。

3. 数据的物理结构

数据的物理结构又称数据的存储结构，是指将具有某种逻辑结构的数据存于计算机内，

这些机内表示的数据所体现的结构称为存储结构。换而言之，数据的物理结构是指数据的逻辑结构在计算机中的表示，也称映像，它包含数据元素的映像和关系的映像。

存储结构中应包含的主要内容：

(1) 存储结点，每个存储结点中应保存一个数据元素；

(2) 数据元素之间关联方式的表示，即逻辑结构的机内表示；

(3) 其他，有时为方便运算的实现而设置的虚结点等。

根据数据元素之间关联方式的不同，数据的存储结构一般可分为顺序存储方式、链式存储方式、索引存储方式及散列存储方式四种。

顺序存储方式：每个存储结点只含一个数据元素，所有存储结点依次存放在一组地址连续的存储单元中。顺序存储方式的特点是数据元素之间的逻辑关系是用存储结点间的位置关系来表示的，即逻辑上相邻，物理上也相邻。

链式存储方式：每个存储结点中除包含一个数据元素外，还包含一组指针，指向与本结点有逻辑关系的结点。因此该物理结构下的数据元素之间的逻辑关系是用附加的指针来表示的。

索引存储方式：每个存储结点只含一个数据元素且所有存储结点连续存放，另外再建立一个用于指示逻辑记录和物理记录之间一一对应关系的索引表。

散列存储方式：每个结点包含一个数据元素，各结点根据散列函数的指示存储在相应的存储单元中。

1.2　算法及其分析

1.2.1　算法的基本概念

运用计算机来解决实际问题时，当完成了数据表示这项工作后便可进入数据处理阶段，即编写适当的程序使计算机能完成对数据的相应操作，但程序是建立在算法的基础之上的，因此有一个好的算法就显得格外重要。

1. 算法的概念及其分类

(1) 算法的定义　算法是对特定问题求解步骤的一种描述，它是指令的有限序列。

(2) 算法的特性　算法有 5 项特性。

●有穷性：必须在执行有限步之后结束，且每一步都可在有限时间内完成；

●确定性：算法中每一条指令必须有确切的含义，不能有二义性，并且，在任何条件下，算法只有唯一的一条执行路径，即对相同的输入只能得出相同的输出；

●可行性：一个算法是可行的，是指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；

●输入：有零个或多个输入；

●输出：有一个或多个输出。

(3) 算法的分类 尽管算法的设计可以避开具体的程序设计语言，但在描述算法时则必须要借助于某种描述形式，根据算法的描述方式不同，可将算法分为以下三类：

●程序：这种算法可直接在计算机上运行；

●伪语言算法：采用某种“伪程序设计语言”描述的算法，如类 Pascal 语言；

●非形式算法：用自然语言，同时可能还使用了程序设计语言或伪程序设计语言描述的算法。

2. 算法的评价标准

一般而言，解决一个实际问题可采用不同的算法，设计人员当然希望采用其中好的算法，因此如何评价算法就显得尤为重要。通常从以下几个方面对算法作出评价。

(1) 正确性 能正确地实现预定的功能；

(2) 易读性 易于阅读和理解，方便调试、修改和扩充；

(3) 健壮性 当遇到非法数据时，算法应能做出适当的反应或处理（如能给出一个表示出错的信息并返回到适当的地方重新执行），而不会产生不希望的结果；

(4) 高效率 具有较好的时空性能，即具有较高的执行效率和较低的空间代价。

值得注意的是在上述四个方面中，正确性最为重要，一个不正确的算法是毫无意义的，其他几个方面可能会发生矛盾，因此评价算法应根据实际问题的需要有所侧重，但一般而言，算法不能过分追求高效率，应在保证正确性、易读性和健壮性的前提下，尽可能提高其执行效率。

1.2.2 算法效率的分析

算法的效率一般以算法执行时在时间和空间资源方面的消耗量来衡量。在时间方面，可使用事后统计和事前估计两种办法，但事后统计的缺陷较为明显：只能在算法具体执行之后才能了解，而且所得结果依赖于计算机的软件和硬件资源（如同一算法在不同的机型上执行或采用不同的实现语言都会直接影响到运行时间），因此一般采用算法的时间复杂度这个与软、硬件资源无关的量度来进行事前估算。这一量度虽不能精确地确定算法的执行时间，但可确定当问题的规模增长时，算法所需时间的增长趋势。

算法的时间复杂度和空间复杂度合起来便称为算法的时空性，简单地说，所谓时间复杂度是指算法所包含的计算量，而空间复杂度则是指算法所需要的存储量。

1. 时间复杂度的估算

在一般情况下，一个算法的时间复杂度是算法的输入规模的函数。算法的输入规模也称为问题的规模，通常为该算法输入数据的个数。如：n 个整数结点的排序算法，其问题的规模便为 n；两个 n 阶方阵的相乘算法，习惯上将其问题规模定为 n。

另外，在一个算法中所涉及到的操作往往有多种，在分析时间复杂度时，可选择其中的一种或几种进行计算，被选取的操作便称为原操作，原操作应该是与其重复执行次数和算法的执行时间成正比的操作，即在算法的执行过程中执行次数最多的操作（在大多数情况下，

选择最深层循环内的语句)。

设问题的规模为 n，原操作重复执行的次数与 n 之间的函数为 $f(n)$，则算法的时间复杂度 $t(n)$一般就记为：$t(n)=O(f(n))$ 。

简单地讲，若 n 表示问题的规模，算法的时间复杂度为 $O(n)$，则表示算法所需的时间与 n 成正比，对于一般的情况，称算法时间复杂度 $t(n)$为 $O(f(n))$时，意思指该算法的执行时间与函数 $f(n)$的增长趋势相一致。即若算法的执行次数 $f(n)$为 $3n^2+5n+1$，则可认为该算法的时间复杂度为 $O(n^2)$。

例 1－2　试估算两个矩阵相乘的算法的时间复杂度。

算　法 1-1

```
void mul_matrix(a,b,c)
int a[n][n],b[n][n],c[n][n];
{ for (i=0;i<n;i++)
     for (j=0;j<n;j++)
       { c[i][j]=0;
         for (k=0;k<n;k++)
         c[i][j]=c[i][j]+a[i][k]*b[k][j]
       }
 }
```

上述算法的原操作为第三层循环中的赋值语句，其执行次数为 $n\times n\times n$，所以时间复杂度 $t(n)=O(n^3)$。

有时，一个算法的原操作的重复次数随问题的输入数据集的不同而不同，此时算法的时间复杂度的计算有两种方法：

(1) 平均时间复杂度　以算法在所有输入下的计算量的加权平均值作为算法的计算量。

(2) 最坏情况下的时间复杂度　以算法在所有输入下的计算量的最大值作为算法的计算量。

有关这类算法的时间复杂度的具体计算方法将在后面的相关章节中涉及。

2. 空间复杂度的估算

空间复杂度的概念与时间复杂度的概念类似。在执行一个算法时，除了需要存储空间来保存问题本身的数据外，可能还需要一些额外的空间来存储一些为实现算法所需信息的辅助空间，称之为附加空间，附加空间的大小就是空间复杂度。空间复杂度一般也是问题规模 n 的函数。

若附加空间相对于数据量来说是常数 c，则空间复杂度则记为 $O(1)$(有时也会记为 $O(c)$)，若附加空间与问题规模 n 呈线性关系，则记为 $O(n)$，其余以此类推。

常见的时间复杂度或空间复杂度有：$O(1)$——常数阶、$O(\log_2 n)$——对数阶、$O(n)$——线性阶、$O(n^2)$——平方阶、$O(n\log_2 n)$——线性对数阶。复杂度的阶数越低，算法的效率越高。

一般而言，时间和空间是相互矛盾的，但随着计算机内存的不断扩大，在算法评估时人们的注意力主要集中在时间复杂度上。

1.3 小 结

本章介绍了数据结构的一些基本概念和初步的算法分析方法。这些基本概念始终贯穿于全书，主要包括：体现数据组织三个层次的数据、数据元素和数据项；数据的逻辑结构和物理结构及各自的分类；算法的定义、特性及其评价标准。

本章的重点是数据结构的基本概念和算法的评价标准。数据的基本单位是数据元素，而数据元素又由若干数据项组成，数据项是数据不可分割的最小标识单位；根据数据元素之间关系的不同，数据的逻辑结构可分为集合、线性结构、树形结构、图形结构四类；根据数据元素之间关联方式的不同，数据的存储结构又可分为顺序存储方式、链式存储方式、索引存储方式及散列存储方式四种；设计算法时要注意算法的优劣，尤其要注意掌握时空复杂度这一概念及简单的时间复杂度的评估方法。

数据结构、算法和程序设计是彼此密不可分的三个方面。算法的设计可以不必考虑具体的程序设计语言，但算法的实现则必须借助于某种语言。著名计算机科学家、Pascal 语言的发明者 Nikaus Wirth 就曾提出过一个著名的公式：

算法+数据结构=程序

因此，在学习数据结构这门课程时，绝不能离开数据结构去抽象地分析求解的算法，也不能脱离算法去孤立地研究程序的数据结构。

1.4 实 训

1. 判断题

(1) 数据元素是数据的最小单位。

(2) 数据结构是带有结构的数据元素的集合。

(3) 数据的逻辑结构是与计算机的内存无关的。

(4) 算法的效率越高越好。

2. 简述题

(1) 试叙述数据、数据元素及数据项之间的关系。

(2) 试叙述数据的逻辑结构与物理结构的关系。

(3) 试说明数据的逻辑结构与存储结构的分类。

3. 设 n 为正整数，试计算下列各程序段的时间复杂度。

(1) i=1;k=0;
　　while (i<=n – 1) {k=k+10*i;i=i+1};

(2) i=1; k=0;
　　while (i<=n – 1) {i=i+1;k=k+10*i};

(3) for (i=1;i<=n;i++)
　　for (j=1;j<=i; j++)
　　for (k=1;k<=j;k++)　x=x+1;

第2章　线　性　表

【学习目标】

1. 了解线性表的定义和基本运算；
2. 掌握两类存储结构（顺序和链式）的描述方法，以及基本操作的实现算法；
3. 能够分析线性表两种存储结构的特点和适用场合，从而学会在实际应用中选择恰当的存储结构。

2.1　线性表的定义和基本运算

2.1.1　线性表的概念

1. 线性表的定义

线性表是 $n(n\geq 0)$ 个具有相同特性的数据元素的有限序列，一般表示成 $(a_1, a_2, \cdots, a_n)$。其中，每个 a_i 代表一个结点（或称数据元素），不同情况下的数据元素可代表不同的具体含义。

例如，下面的L1（数据元素为字母）和L2（数据元素为整数）都是线性表：

L1：(A，B，C，…，Z)　L2：(50，68，81，96，110，127)

再如，表2－1所示的某单位职工工资表也是一个线性表（数据元素为一复杂信息）。

表2－1　某单位职工工资表

姓　名	工作部门	应发工资（元）	扣水电费（元）	实发工资（元）
徐小杰	人事部	950	80	870
蒋德明	研究部	1100	100	1000
刘　虹	纪检科	920	60	860
张　颖	生产部	1020	70	950
李晓红	销售部	1030	75	955
…	…	…	…	…

2. 线性表的相关概念

设有一线性表（$a_1, a_2, \cdots, a_n$），则：

(1) 表头与表尾　a_1称为表头（或称起始结点），a_n称为表尾（或称终端结点）。

(2) 直接前趋与直接后继　对任意一对相邻结点 a_i 和 a_{i+1}（$0<i<n$），称 a_i 是 a_{i+1} 的直接前驱，或者说 a_{i+1} 是 a_i 的直接后继。

(3) 线性表的长度　线性表中数据元素的个数。

(4) 空表　长度为 0 的线性表称为空表，可记为（ ）或∅。

3. 线性表的基本性质

对于一个非空线性表，除表头没有直接前趋外，其余结点有且仅有一个直接前趋；除表尾没有直接后继外，其余结点有且仅有一个直接后继。因此，在线性表中，所有结点之间的邻接关系是 1：1 的关系，线性表的逻辑结构是线性结构。

2.1.2　线性表的基本运算

线性表中的数据元素类型可以是字符、整数等简单类型，也可以是复杂的结构类型，长度可以增加或缩短，所以线性表是相当灵活的数据结构，对线性表中的数据元素可以进行各种运算。

本章所涉及的内容主要是线性表的一些典型运算，为了后面更为方便地讨论这些运算的实现算法，下面列出了这些基本运算的含义：

(1) 初始化线性表 Initiate(L)　设定一个空的线性表 L；

(2) 求表长 Length(L)　返回值为给定线性表 L 中数据元素的个数；

(3) 取表中的元素 Get(L,i)　若 $1 \leqslant i \leqslant$ Length(L)，返回线性表 L 中第 i 个数据元素的值；

(4) 定位操作 Locate(L,x)　若线性表中存在值和给定值 x 相等的数据元素，则返回该数据元素的位序。若满足条件的数据元素不唯一，取最小的位序；

(5) 插入操作 Insert(L,i,x)　若 $1 \leqslant i \leqslant$ Length(L)+1，则在线性表 L 中第 i 个位置上插入一个值为 x 的新结点；

(6) 删除操作 Delete(L,i)　若 $1 \leqslant i \leqslant$ Length(L)，删除线性表 L 中第 i 个数据元素；

(7) 判定空表 Empty(L)　若 L 为空表，则返回值为 1，表示“真”；否则返回 0，表示“假”；

(8) 表置空操作 Clear(L)　将已知的线性表 L 置为空表。

利用这些基本运算可实现线性表的各种复杂运算，如通过置空表操作和反复向表尾（或表头）插入数据元素，可在计算机上建立一个线性表的存储结构；通过反复执行删除第 i 个元素的操作，可删除线性表中从第 i 个元素开始的连续若干个元素等。