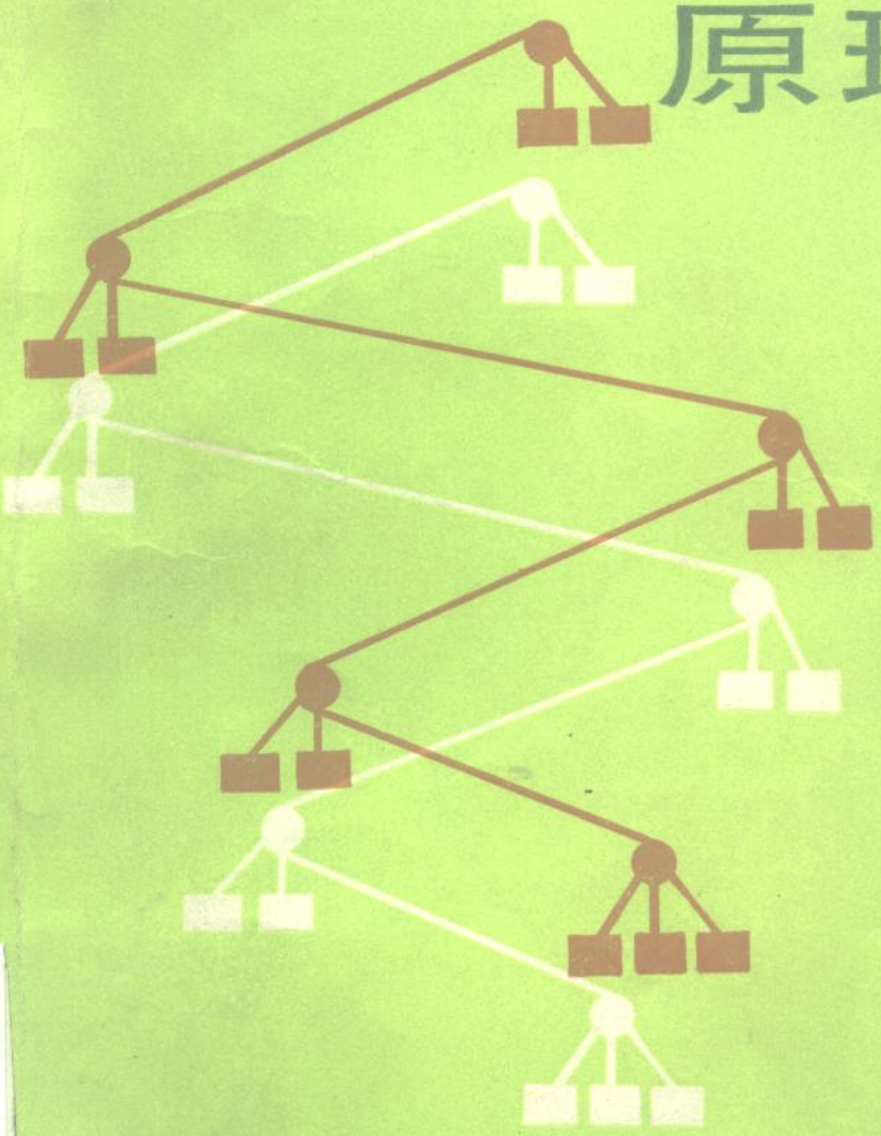
# 数据结构原理





上海科学技术文献出版社

# 数据结构原理

本书编译组 编

上海科学技术文献出版社

数据结构原理
本书编译组 编

*

上海科学技术文献出版社出版发行
（上海市武康路2号）
新华书店经销
昆山亭林印刷厂印刷

*

开本 850×1168 1/32 印张 13.75 字数 332,000
1988年8月第1版 1988年8月第1次印刷
印数：1—5,200

ISBN 7-80513-301-8/T·108
定价：6.85 元
《科技新书目》155—309

# 编 者 的 话

数据结构，是从事计算机科学及其应用的科技人员必须掌握的重要基础知识，也是计算机科学教学中一门核心课程。

编者根据我国大学计算机科学教学实际情况和自己的教学经验，参考了 IEEE 计算机科学与工程示范教学计划的"数据结构"内容，在翻译了 Ellis Horowitz and Sartaj Sahni所著并为百余所大学采用的 «Fundamentals of Data Structures in PASCAL» (1984)书的基础上，主要参考了 D. E. Knuth 著的 «The Art of Computer Programming» (Vol. 1, 3, 1973)和复旦大学出版的«数据结构讲义»，编译出«数据结构原理»，奉献给我国广大读者。

本书在编译过程中，本着由简及繁，由易到难，循序渐进，前后衔接自然的原则。力求做到概念清楚，深入浅出，问题交待清楚，充分体现数据结构的特点和系统性。

全书共分九章，内容丰富，各章都备有适量的习题供读者练习。大部分算法都用 PASCAL 语言编写，有些已在计算机上通过。在教学中取得了较好的效果，得到了学生的好评。因此，本书的通用性和实用性都很强。

参加本书翻译和编写工作的有：钮和荣(第一、二章)，孟佩琴(第三、四、八章)，蔡子经(第五、七章)，王槐珍(第六、九章)，缪瑞清自始至终参加了编译工作，全书由罗文化组织编译并最后修改定稿。

美国加里福尼亚大学伯克利分校叶晓石女士提供了原书；在编译过程中得到了复旦大学施伯乐教授的支持和指导。值此对他们谨表谢意。

限于编者的水平，书中错误和不妥之处，敬请读者批评指正。

1986 年 10 月

# 导　论

在六十年代初期，还没有单独开设“数据结构”这门课程，而数据结构的一些基本内容大多分散在“编译原理”和“操作系统”课程中。六十年代中期，开始设置“表处理语言”(List Processing Language)课程，其主要内容包括SLIP，IPL—V，LISP和SNOBOL等表处理系统。1968年出版了D.E.克努特所著的《计算机程序设计技巧》第一卷基本算法(THE ART OF COMPUTER PROGRAMMING, Volume 1/Fundamental Algorithms, D. E. Knuth)，他论证了几乎任何一种语言都能采用表处理语言那样的技术。1968年，美国明确规定“数据结构”为计算机科学专业的一门核心课程；1982年，又被美国IEEE计算机协会列为计算机科学与工程示范教学计划中的一门核心科目。

数据结构研究对象是：1.数据的逻辑结构、物理结构以及两者之间的关系，对每种结构定义相应的算法；2.设计算法，分析算法的效率，用某种高级程序设计语言描述给出的各种算法过程；3.数据结构在计算机科学和软件工程中的应用。

根据我国的教学实际情况和我们的教学实践，参考了计算机科学与工程示范教学计划中所规定的“数据结构”内容。本书的基本内容为：特定的数据结构(数组，表，树，图)；数据结构的实现及操作(数组的运算，表和树的查找，内部排序，外部排序)；文件(存取方法，索引技术，文件的组织形式)。书中还列举了一些能较好地说明问题且程序结构优良的实际例子。

算法设计是数据结构中非常重要的一部分内容，为了便于读者理解且掌握算法的基本思想和实质，以及上机实习，需要选择一种合适的高级程序设计语言描述算法过程。根据学生的基础和高级程序设计语言的可用性，我们选用了PASCAL语言来描述算法

过程，它的优点是易于书写、阅读和理解，使读者集中精力于算法的实质上，而不是把精力花在高级程序设计语言上。

数据结构主要以离散数学和程序设计为基础，运用集合论、图论和概率统计等方面的基础理论来分析、处理数据结构和算法。数据结构这一学科形成的时间不太长，还处于发展阶段，它的内容将不断地发展和充实，计算机硬件和软件的发展将对它产生重大影响。

通过学习本书，在下述几方面得到特别训练：

1．在较高的抽象程度上，确定数据结构和选用正确的算法；

2．设计一个较适用的数据结构的实现方案；

3．编写结构优良的程序；

4．分析程序执行的时间复杂性。

本书可作为计算机科学专业的“数据结构”课程教材，也可作为研究生以及计算机科学工作者的参考书。

# 目　录

# 第一章 数 组

数组是大家所熟悉的知识，故从数组开始来学习数据结构是合适的。数组是一种应用最广泛的数据结构，数组是更复杂数据结构的基本构造模块，几乎可用数组表示任何一种复杂的结构。在这一章中，我们要介绍一维和多维数组，同时还要介绍三角矩阵、带状矩阵和稀疏矩阵。

## 1.1 一维数组

数组是由有限个同类元素所组成的有序集合。有序性是指可以识别数组的第一、第二、……、第 $n$ 个元素；数组的元素是同类的，即数组是由具有相同数据类型的元素所组成。

数组的最简单形式是一维数组，或称为向量。我们可用下图描述由 $n$ 个元素组成且被命名为 $A$ 的一维数组，元素的下标指明了该元素在该数组的排列中的位置。对于一个数组只能指定一个名字，而用数组名后跟用中括号括起来的下标表示数组中的元素。比如，$a[i]$ 是数组 $A$ 中带有下标值 $i$ 的元素，它拥有一个与数组 $A$ 中所有元素相同数据类型的值。有时，我们也可以把 $a[i]$ 写成 $a_i$。

| $a[1]$ | $a[2]$ | … | $a[i]$ | … | $a[n]$ |
|---|---|---|---|---|---|

例如，我们可以把某个城市在 24 小时内每小时的温度记录构成一个数组，数组的元素是按当天相应的时间进行排列的。如果我们给这个数组命名为 temp，那末 temp$[i]$ $(1\leqslant i\leqslant 24)$ 是第 $i$ 小时所记录的温度。如果数组中元素的数据类型是实数，则我们可以给出某个具体的 temp 数组如下：

| temp[1] | temp[2] | … | temp[12] | … | temp[24] |
|---|---|---|---|---|---|
| 9.0 | 9.5 | … | 12.3 | … | 8.9 |

又例如，某学校里某班级的学生成绩表是按学号顺序排列的，我们可用一个一维数组来描述这个班级学生的成绩表(见表1.1.1)，此时学号与数组元素的下标相一致。

表 1.1.1

| 学号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 成绩 | 82 | 79 | 94 | 65 | 76 | 85 | 73 | 70 | 88 | 92 |

如上所述，一维数组的每个元素是由一个值和一个下标值所确定的。下面给出一个典型的 PASCAL 说明：

**var** $a$: **array**[$l..u$] **of real**;

它指明数组的名字是 $a$，$a$ 是一个一维数组，数组 $a$ 中的所有元素都是实数，数组 $a$ 的下标是从 $l$ 开始一直到 $u$，$l$ 是下标的下界，$u$ 是下标的上界。如果数组 $a$ 的第一个元素被存放在地址为 $\alpha$ 的存贮单元中，那么很容易求得数组 $a$ 中其它元素的存贮地址。

数组元素　$a[l]$　$a[l+1]$　$a[l+2]\cdots a[i]\cdots a[u]$

存贮地址　$\alpha$　$\alpha+1$　$\alpha+2\cdots\alpha+i-l\cdots\alpha+u-l$

如果用 $\alpha a_i$ 表示元素 $a_i$ 的存贮地址，则可用如下的地址计算公式求得数组中各元素的存贮地址：

$$\alpha a_i=\alpha a_l+i-l=\alpha+i-l$$

## 1.2 二维数组

通常称一个 $m\times n$ 阶的矩阵(图 1.2.1)：

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

图 1.2.1　矩阵

为一个二维数组。它的每个元素由一个值和一组下标$(i,\ j)$值来确定。其中，$i=1,\ 2,\ \cdots,\ m$；$j=1,\ 2,\ \cdots,\ n$。每一组下标$(i,\ j)$都唯一地对应一个值 $a_{i,j}$。如果把二维数组看作是一张有序的表，

那么以上矩阵可表示成：

$$A=(A_1,\ A_2,\ \cdots,\ A_m)$$

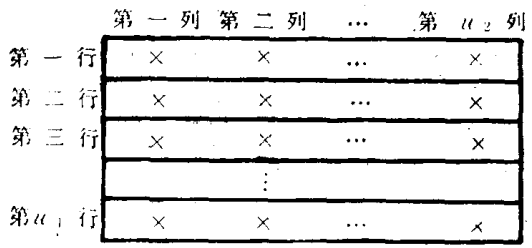
其中，每个 $A_p(p=1,\ 2,\ \cdots,\ m)$ 均是一个一维数组：

$$A_p=(a_{p,1},\ a_{p,2},\ \cdots,\ a_{p,n}),\ p=1,\ 2,\ \cdots,\ m$$

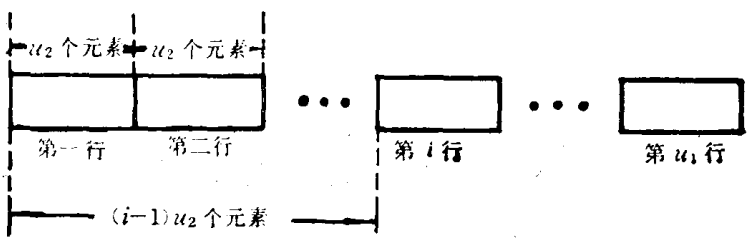
### 1.2.1 二维数组的存贮

在计算机科学中，通常用一批连续的存贮单元来表示数组，用这种方式来存贮数组一般称为顺序存贮。顺序存贮按其方式可分为两种：一种是按行序的顺序存贮，例如将数组 $A$ 按 $a_{1,1},\ a_{1,2},\ \cdots,\ a_{1,n},\ a_{2,1},\ a_{2,2},\ \cdots,\ a_{2,n},\ \cdots,\ a_{m,1},\ a_{m,2},\ \cdots,\ a_{m,n}$ 这样的次序进行存贮。由此可知，所谓按行序的顺序存贮就是按行序一行接一行地存贮。另一种是按列序的顺序存贮，例如将数组 $A$ 按 $a_{1,1},\ a_{2,1},\ \cdots,\ a_{m,1},\ a_{1,2},\ a_{2,2},\ \cdots,\ a_{m,2},\ \cdots,\ a_{1,n},\ a_{2,n},\ \cdots,\ a_{m,n}$ 这样的次序进行存贮。同理可以看出，所谓按列序的顺序存贮就是按列序一列接一列地存贮。

顺序存贮的优点是：只要知道数组元素下标组的值，就可以找




(a)




(b)

图 1.2.2 a[1..$u_1$, 1..$u_2$]按行顺序存贮

到该元素的存贮位置，这样就可以随机地存取每个元素。

一个二维数组 $a[1..u_1, 1..u_2]$可解释为有 $u_1$ 个行，每一行由 $u_2$ 个元素组成，现按行顺序存贮 $a$，这些行在内存中的表示如图 1.2.2 所示。

现假设每个数组元素只需一个存贮单元，$\alpha$ 为 $a[1, 1]$ 的地址，则 $a[i, 1]$的地址为 $\alpha+(i-1)u_2$，因为在第 $i$ 行的第一个元素之前有 $i-1$ 行，而每行有 $u_2$ 个元素。因此，可推得 $a[i, j]$的存贮地址为 $\alpha+(i-1)u_2+(j-1)$。

### 1.2.2 三角阵与带状矩阵

通常，二维数组也可以用矩阵来称呼。当矩阵的行数等于列数时，称该矩阵为方阵，如图 1.2.3 所示。

$$A=\begin{bmatrix} a_{1,1} & a_{1,2}\cdots a_{1,n} \\ a_{2,1} & a_{2,2}\cdots a_{2,n} \\ \vdots & \vdots \\ a_{n,1} & a_{n,2}\cdots a_{n,n} \end{bmatrix}$$

图 1.2.3 方阵

若 $A$ 满足 $a_{i,j}=a_{j,i}(1\leqslant i\leqslant n,\ 1\leqslant j\leqslant n)$，则称 $A$ 为对称阵。对称阵有近一半的元素是相同的。因此，若仍按一般存贮二维数组的方法来存放对称阵的每个元素，则对内存资源来说是很大的浪费。另外在实际问题中，也常常会遇到三角阵，这种矩阵其对角线以上或以下的元素均为零。对这种矩阵若也按一般存贮二维数组的方法将几乎近一半的零元素也存贮起来，这对存贮资源同样也是个很大的浪费。为此需研究如何节省存贮资源的方法。

一、三角阵

对于以上提及的对称阵和三角阵的存贮，我们将采用存贮三角阵的方法来实现，即只存贮对角线以下或以上的元素。设存贮的矩阵为 $A$，$A$ 是一个下三角矩阵(见图 1.2.4)，其元素为 $a_{i,j}(1\leqslant i\leqslant n,\ 1\leqslant j\leqslant n,\ i\geqslant j)$。

我们按行序：$a_{1,1}$，$a_{2,1}$，$a_{2,2}$，$a_{3,1}$，$a_{3,2}$，$a_{3,3}$，$\cdots$，$a_{n,1}$，$a_{n,2}$，$\cdots$，$a_{n,n}$ 存放 $A$ 阵对角线以下的全部元素。现设 $A$ 阵的每个元素只

$$A=\begin{bmatrix} a_{1,1} & & \\ a_{2,1} & a_{2,2} & \\ \vdots & & \ddots \\ a_{n,1} & a_{n,2}\cdots & a_{n,n} \end{bmatrix}$$

图 1.2.4　三角阵

需一个存贮单元，$\alpha$ 为 $a_{1,1}$ 的地址，那么可以知道位于第 $i$ 行和第一列的元素 $a_{i,1}$ 的地址公式为：$\alpha+i(i-1)/2$。同理，我们可以得到位于第 $i$ 行和 $j$ 列的元素 $a_{i,j}(i\geqslant j)$ 的地址公式为：$\alpha+i(i-1)/2+(j-1)$（其中：$i$, $j=1, 2, 3, \cdots, n$, $i\geqslant j$）。用这种方法来存贮三角矩阵只需 $\sum_{i=1}^{n} i=n(n+1)/2$ 个存贮单元，这比 $n^2$ 个存贮单元要节省得多。

表 1.2.1 给出了一个三角阵及其存贮方法的实例。

例：设

$$A=\begin{bmatrix} 31 & 0 & 0 & 0 \\ 43 & 0 & 0 & 0 \\ 58 & 36 & 75 & 0 \\ 0 & 14 & 69 & 91 \end{bmatrix}$$

**表 1.2.1　三角矩阵及其存贮方式**

| 存贮地址 | 元　　素 | 存贮地址 | 元　　素 |
|---|---|---|---|
| 1 | 31 | 6 | 75 |
| 2 | 43 | 7 | 0 |
| 3 | 0 | 8 | 14 |
| 4 | 58 | 9 | 69 |
| 5 | 36 | 10 | 91 |

其中，$a_{1,1}$ 的地址是 1，即取 $\alpha=1$。其它 $a_{i,j}$ 的地址可以按公式：

$$\alpha+i(i-1)/2+(j-1)$$

计算得到。

如 $a_{4,3}$ 的地址为：$1+4(4-1)/2+(3-1)=1+6+2=9$

二、带状矩阵

对于 $n\times n$ 阶的方阵，若它的全部非零元素落在一个以主对角线为中心的带状区域，这个带状区域若包含主对角线下面及上面的各 $b$ 条对角线上的元素以及主对角线元素，那么称此方阵为半带宽为 $b$ 的带状矩阵。

若带状矩阵 $A$ 的元素记作 $a_{i,j}$，半带宽为 $b$，则当 $i-j>b$ 或 $j-i>b$ 时，$a_{i,j}=0$。$b$ 是比 $n$ 小的正整数。或者说，当 $|i-j|>b$ 时，$a_{i,j}=0$，则称 $A$ 是半带宽为 $b$ 的带状矩阵，称 $2b+1$ 为带状矩阵的带宽。图 1.2.5 给出一个半带宽为 3 的带状矩阵。

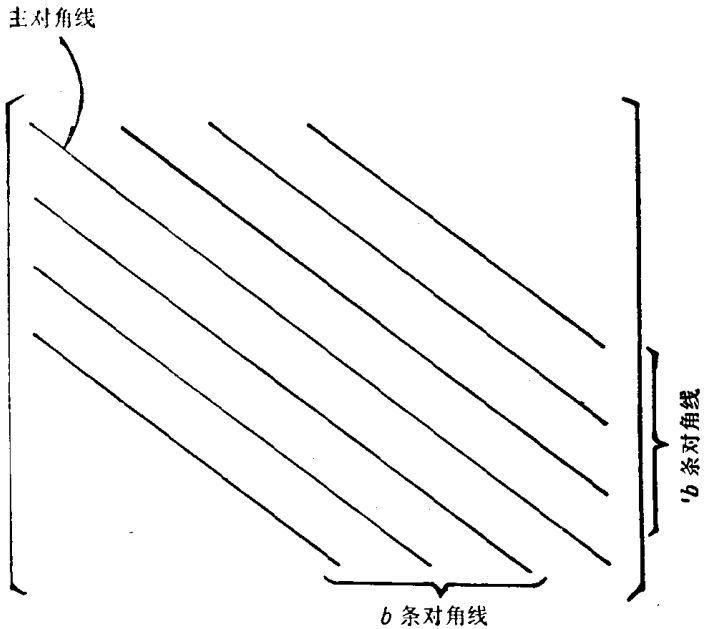



图 1.2.5　带状矩阵($b=3$)

存贮带状矩阵时，对于带状区域以外的元素，即 $abs(i-j)>b$ 的 $a_{i,j}$ 不存贮，而只存贮带状区域内的 $(2b+1)n-b(b+1)$ 个元素。但通常并不是把这些元素顺序地存贮在 $n(2b+1)-b(b+1)$ 个存贮单元中，而是按如下的方法存贮：即除第一行和第 $n$ 行外，每行都看作有 $2b+1$ 个元素，将带状矩阵存贮在 $n(2b+1)+2b$ 个