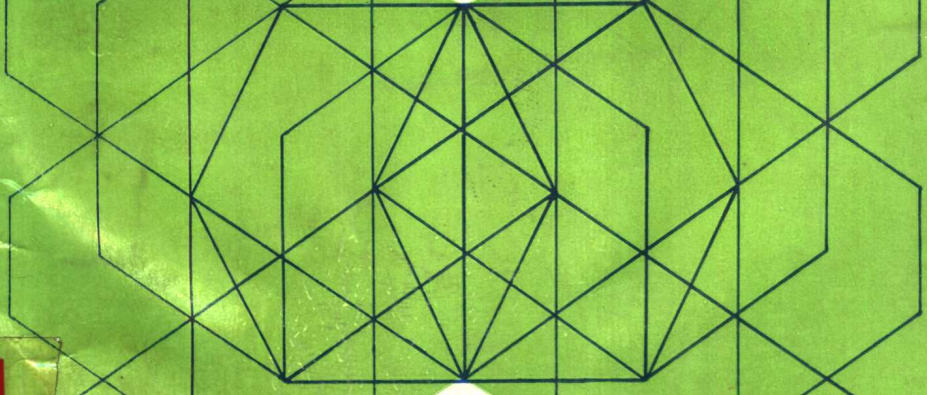
数据结构

施伯乐　蔡子经
孟佩琴　张乃玲



SHU JU
JIE GOU

复旦大学出版社

数据结构

施伯乐　蔡子经
孟佩琴　张乃玲　编

复旦大学出版社

内 容 提 要

本书共分八章：线性表、数组、串、内部排序、树、图、查找和外部排序。复旦大学计算机系从1980年开始以此为教材，并不断更新成一本比较完整的、系统的教科书。本书中大量算法都有实用价值，并用最流行的PASCAL语言描述。

本书是计算机各专业及与软件有关专业的基础课教材，可作为大专院校教师的教学参考书，以及数据处理工作者和软件应用技术人员的参考资料。

数据结构

施伯乐　蔡子经
孟佩琴　张乃玲　编

复旦大学出版社出版

新华书店上海发行所发行　复旦大学印刷厂印刷

开本850×1168　1/32　印张11.75　插页0　字数336000

1988年8月第1版　1988年8月第1次印刷

印数1—6000

ISBN7—309—00123—0/T·04　定价2.55元

序　言

随着计算机科学和技术的发展，计算机的功能和运算速度不断地提高，它应用于信息处理的范围日趋扩大，需要表示、存贮和处理的信息也越来越多。因此，与之相应地，计算机的加工处理对象也从简单的数值发展到一般的符号，进而发展到更复杂的数据结构。

数据结构是研究一些数据的集合，这些数据可以是数，也可以是符号，我们称之为数据元素，或称之为结点。在计算机中要表示、存贮、处理这些数据，数据结构就是根据数据的性质、数据元素间的关系，研究如何表示、存贮、操作(查找、插入、删除、修改)这些数据的技术。因此数据结构是计算机科学和技术中的一门基础课程，它是设计和实现编译程序、操作系统、数据库系统及其他程序系统的重要基础。

数据结构的表示和操作都涉及到算法，如何描述数据的结构和讨论有关的算法，又涉及到程序设计语言，我们选择了 PASCAL 语言为工具，因为 PASCAL 语言不仅具有丰富的数据类型，而且是一种较好地体现结构程序设计原则的语言。

全书共分八章。前三章是数据结构的基础，其内容是线性表、数组、串(字符串)的存贮结构和基本操作。在这三章中，作为重点的第一章介绍顺序存贮和链接存贮中的队、栈表示和操作。

第四章至第七章是数据结构的主要内容，这几章分别介绍排序(或称分类)、树、图、查找。第四章介绍排序，讨论把数据元素按关键字大小排序的各种算法，算法的好坏标准按算法的时间复杂性来区分；第五章介绍树，它是比线性表更复杂的数据结构，有较大实用价值，这章讨论一般树的结构和操作，对于二叉树作了更详细的讨论；第六章介绍图，它是比树更复杂的数据结构，这章讨论图的表示、遍历，及图在最短路径、拓扑排序和关键路径方面的应用；第七章介绍查找，查找是其他操作的基础，这章讨论了查找树、解答树、B 树的表示及有关的算法；

最后一章介绍外部排序，涉及到二级存贮设备，对于大的文件系统及数据库管理系统的研究有一定的价值。

本书原是编者为讲授数据结构而编的教材，作为复旦大学计算机科学系的重点课程教材，已使用很久。本书同时也可作计算机培训班的教材，为软件工作者所必备。

编　者

一九八七年四月

目　录

第一章 线 性 表

1.1 线性表及表的基本运算

1.1.1 线 性 表

最简单和最常见的数据对象是有序的数据结构，称为线性表。例如，一星期中的七天，依次是：

MONDAY, TUESDAY, WEDNESDAY, THURSDAY,
FRIDAY, SATURDAY, SUNDAY

这是有序的数据结构，构成线性表，其中 MONDAY, TUESDAY, …，称为线性表的结点。

再如下面的学生成绩登记表，是略为复杂一点的线性表的例子，见表 1.1。

其中每个学生的成绩情况在表中各占一行，每行的信息说明某个学生四门课程的学习成绩、总成绩及平均成绩。整个成绩登记表是线性的数据结构，表中的每一行称为一个结点（或记录），记录中的每个数据项，如学号、姓名、年龄、各科成绩等称为记录的域，或称为字段。上述成绩登记表与前面列出的一星期中的七天一样，也是线性表的结构。

一般来讲，线性表是由长度为 $n(n\geqslant 0)$ 的一组结点 $k_1, k_2, \cdots, k_{i-1}, k_i, k_{i+1}, \cdots, k_n$ 组成的有限序列。当 $n=0$ 时，线性表为空，称为空表。在非空的线性表 $(n>0)$ 中，有且仅有一个开始结点 k_1 和一个终端结点 k_n，除 k_1 和 k_n 外，表中的每一个结点 $k_i(2\leqslant i\leqslant n-1)$ 都有一个直接前趋结点 k_{i-1}（也称为前件）和一个直接后继结点 k_{i+1}（也称为后件）。表中只有一个结点没有直接前趋，即开始结点 k_1，同时，也只

有一个结点没有直接后继，即终端结点 k_n。

表 1.1

NO (学号)	NAME (姓名)	SEX (性别)	AGE (年龄)	POLI (政治)	MATH (数学)	PHYS (物理)	ENGL (英语)	SUM (总分)	AVER (平均分)
8001	ZHAO	FEMALE	21	81	91	91	81	344	86.0
8002	QIAN	MALE	22	82	92	83	91	348	87.0
8003	SUN	MALE	23	65	99	91	85	340	85.0
8004	LI	FEMALE	20	89	89	70	98	346	86.5
8005	ZHOU	FEMALE	20	75	90	80	81	326	81.5
8006	WU	MALE	24	70	80	82	96	328	82.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

抽象地可以把非空线性表写成

$(k_1,k_2,\cdots,k_{i-1},k_i,k_{i+1},\cdots,k_{n-1},k_n)$

可用一个二元组 $B=(K,R)$ 表示线性表的数据结构，其中 K 是结点的有限集合，而 R 是K 上的关系的有限集合，这里 R 只包含一种关系 N，且 K 的结点可以排列得使 $N=\{(k_{i-1},k_i)\mid 2\leqslant i\leqslant n\}$。

1.1.2 线性表的基本运算

在线性表上可以实行各种各样的运算与操作。线性表的基本运算，也就是常规操作有：

一、求线性表中结点的个数 n；

二、从左到右(或从右到左)读表；

三、找出线性表中的第i个结点，$1\leqslant i\leqslant n$；

四、找出表中结点 x 的位置（或找出某个域具有某个值的结点）；

五、在表的第 i 个位置上插入新的结 点 x，$1\leqslant i\leqslant n+1$，使编号

为i,i+1,…,n的结点变成编号为i+1,i+2,…,n+1的结点；

六、删除线性表的第i个结点，1≤i≤n，使编号为i+1,i+2,…,n的结点变成编号为i,i+1,…,n-1的结点；

七、依赖于结点的某个域或某些域的值，按域值大小的递增或递减次序，重新排列线性表中的结点。

不是任何时候都需要同时执行以上这些操作，往往只需要执行其中的一部分。当然，从上述这些基本操作出发，一些其他的操作，如：把两个或更多的线性表组合成一个线性表；或是把一个表分成若干个表；以及复制表等等都不难实现。

对"数据结构"来说，感兴趣的问题是用什么方式来表示线性表，以便这些基本操作能有效地实现。

1.2 线性表的顺序存贮

1.2.1 顺序存贮的数组实现

在计算机内，可以利用不同的方式表示线性表，这里先介绍一种最简单、最普通的方法——顺序存贮，即用一组连续的存贮单元依次存贮线性表中的结点。

不失一般性，不妨考虑线性表中的所有结点都属于同一类型，这样，每个结点在存贮器中占用空间的大小是相同的。假设记第一个结点 k_1 的存贮开始地址是 αk_1，并设每个结点占用机器 s 个存贮单元(机器字)，那么结点 k_i 的地址可以用 αk_1 和整数 i 通过公式

$$\alpha k_i = \alpha k_1 + s(i-1)$$

计算出来，其中 αk_i 是存放结点 k_i 的开始地址。

对于这种存贮方式，要访问第 i 个结点特别显得方便，因为可以直接计算出 k_i 的开始地址 αk_i。但是要插入一个结点、或删除一个结点，或对线性表进行重新排序，必然引起结点的大量移动，故而比较麻烦。

具体地说，线性表的顺序存贮可以用PASCAL语言的数组来表示：

```
VAR list: ARRAY [1..listsize] OF object;
```

这里假设线性表的结点个数始终不超过 listsize(一个整型常量),数组元素的成分类型为 object,它可以是简单的类型,如 integer, char 或 real,也可以是构造类型,如数组类型或记录类型。为简单起见,以后若不作特别说明,我们总是把 object 作为 integer 类型来讨论。

从前面例举的学生成绩登记表中已经看到,当线性表中的结点类型是记录类型时,一个结点可以由许多个数据项构成,每个数据项表示结点的某种属性,由结点的数据项构成记录的各个域,通常把能唯一地确定结点的那些数据项,称为关键字或简称键,也就是把记录中能唯一地标识记录的域称为关键字,关键字可以由一个或多个域组成,通常,由一个域组成。

例如 1.1 中提到的学生成绩登记表,其结点类型 object 可用记录类型来定义:

```
TYPE object = RECORD
                no: integer;
                name,sex:PACKED ARRAY[1..6]OF char;
                age: 20..30;
                poli, math, phys, Engl:0..100;
                sum: integer;
                aver: real
              END;
VAR studlist: ARRAY [1..50] OF object;
```

其中记录的域 no,它代表学生的学号,由它可以唯一确定是哪一个学生,这里 no 就是关键字。

为了简化问题,我们还可采用索引表的方法。把结点的关键字从线性表中抽出来,建立一张索引表,索引表的每个记录由两个域组成:关键字和指针(见图1.2.1),其中指针指向对应的线性表的结点的存贮开始地址。由于在索引表中包含了这种指针域,通过它可以快速地找到线性表任一点的地址。事实上,这时线性表本身是否还是按顺序存贮方式存贮已无关紧要,若索引表仍采用顺序存贮,那末对线性表进行插入、

删除和排序的操作都只要移动索引表中的结点，而不必去移动线性表本身。

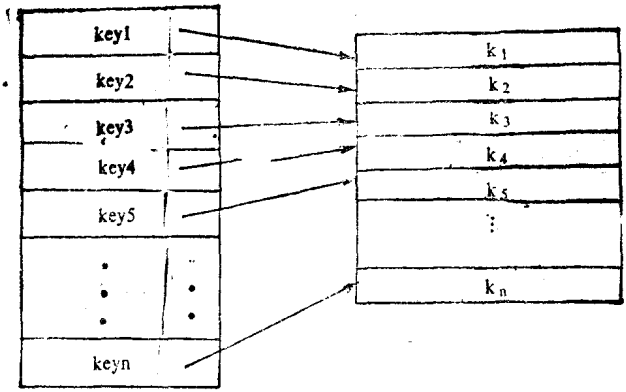


图1.2.1 索引表示意图

如果线性表的结点不属同一类型，每个结点所占用的存贮空间就可能不相等。这时若采用顺序存贮，因为结点的地址不容易统一计算，存取就不能快速进行。但若采用上述的索引表的方法，存取仍只需对索引表进行，上述困难就不复存在。

1.2.2 插入、删除与定位算法

线性表的插入运算是指在表的第 i 个($1 \leqslant i \leqslant n+1$)位置上，插入一个新的结点 x，使长度为 n 的线性表

$$(k_1, k_2, \cdots, k_{i-1}, k_i, k_{i+1}, \cdots, k_n)$$

变成长度为 $n+1$ 的线性表

$$(k'_1, k'_2, \cdots k'_{i-1}, k'_i, k'_{i+1}, \cdots, k'_{n+1})$$

其中新插入的结点为 $k'_i = x$，且

$$k'_j = \begin{cases} k_j & \text{当 } 1 \leqslant j \leqslant i-1 \text{ 时} \\ k_{j-1} & \text{当 } i+1 \leqslant j \leqslant n+1 \text{ 时} \end{cases}$$

即编号 $1, 2, \cdots, i-1$ 的结点保持不变，编号为 $i, i+1, \cdots, n$ 的结点改变成编号 $i+1, \cdots, n+1$ 的结点。在顺序存贮方式下实行插入运算时，若

插入位置i=n+1，即插入到表的末尾，那么只要在表的末尾增加一个结点就可；但是若1≤i≤n，则必须将表中第i个到第n个结点向后移动一个位置，这里共需移动n-i+1个结点。下面我们写出实现插入的算法，用PASCAL语言的过程说明来描述。

插入过程需要的说明如下：

```
CONST  m=1000;
TYPE   list=ARRAY [1..m] OF integer;
VAR    x,n: integer;
       v: list;
```

其中假设线性表容纳结点的最大数目是m=1000，线性表的长度为n，结点的类型为integer型。在数组v中，v[1]，v[2]，…，v[n]存放了线性表的各结点，v[n+1]，…，v[m]是备用的结点空间，在插入时，增加进去的结点需要存贮空间，就可以使用它们。

```
PROCEDURE sqinsert (i,x: integer; VAR v: list; VAR
                n: integer);
VAR j: integer;
BEGIN IF (i<1) OR (i>n+1) THEN
            writeln ('position does not exist')
    ELSE IF n>=m THEN writeln ('list is full')
        ELSE BEGIN
                FOR j:=n DOWNTO i DO
                v[j+1]:=v[j];{若i=n+1，这句不执行}
                v[i]:=x;
                n:=n+1
              END
END;
```

线性表的删除，其操作与插入的操作基本上是相似的。删除操作是指将线性表的第i个结点删去，使得长度为n的线性表

$$(k_1,k_2,\cdots,k_{i-1},k_i,k_{i+1},\cdots,k_n)$$

变成长度为n-1的线性表

$$(k'_1,k'_2,\cdots,k'_{i-1},k'_i,\cdots,k'_{n-1})$$

其中

$$k'_j=\begin{cases}k_j & \text{当 } 1\leqslant j\leqslant i-1 \text{ 时}\\ k_{j+1} & \text{当 } i\leqslant j\leqslant n-1 \text{ 时}\end{cases}$$

即编号为1,2,…,i-1的结点保持不变,将编号为i+1,…,n的结点变成编号为i,…,n-1的结点。对于顺序存贮,与插入运算相似,当i=n时,即删除表尾结点时,只要简单地将表的最后结点删去。但1≤i≤n-1时,必须将表中第i+1个到第n个这n-i个结点向前移动一个位置。

算法用到的说明及假设,与插入算法完全一样,删除的过程如下:

```
PROCEDURE  sqdelete (i:integer; VAR v:list;
                     VAR n:integer);
VAR j:integer;
BEGIN IF (i<1) OR (i>n) THEN
          writeln ('no this element')
   ELSE BEGIN FOR j:=i TO n-1 DO
          v[j]:=v[j+1];{i=n 时, 这句不执行}
          n:=n-1
        END
END;
```

我们再给出关于定位的算法,定位也就是找出结点x在表中的位置locate,如果某个结点$k_i=x$,则locate=i。算法的说明与假设同前,若在线性表中找不到结点x,则置locate=n+1,定位操作用下述函数说明来实现:

```
FUNCTION  locate (n,x: integer;  VAR v: list): integer;
LABEL  10;
VAR  j:integer;
BEGIN  FOR j:=1 TO n DO
         IF v[j]=x THEN BEGIN locate:=j;
                              GOTO  10
```