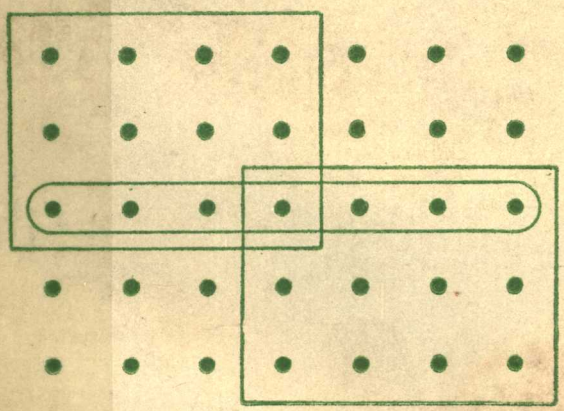
数据结构与算法

PROCEDURE mergsort(n,r,rl);



徐绪松　编著

武汉大学出版社

数据结构与算法

徐绪松　编著

武汉大学出版社

内 容 提 要

本书阐述各种数据结构的基本概念、逻辑特性和存贮表示，给出在各种结构上进行插入、删除、分类、搜索的算法，并介绍基本的算法分析方法和算法设计技术。

书中作者根据“算法＋数据结构＝程序”这一思想，进一步探讨数据结构与算法之间的密切联系，首次将两者有机地结合起来。本书可作为大专院校计算机专业和经济管理、系统工程、情报检索、应用数学等专业的教材或参考书，也可供从事这些专业的工作人员阅读。

数据结构与算法

徐绪松 编著

武汉大学出版社出版

（武昌 珞珈山）

新华书店湖北发行所发行 湖北省嘉鱼县印刷厂印刷

787×1092mm 1/16 14.75印张 320千字

1987年9月第一版 1987年9月第一次印刷

印数：1—5000册

ISBN7—307—00124—1/O·11

统一书号：13279·49 定价：2.20元

前　言

计算机科学的迅速发展，使它不再局限于解决数值计算的各种问题，而且更广泛地应用于非数值计算的各个领域。例如，企业管理、经济系统工程、管理信息系统、情报检索等。在这些领域中，一般有大量的数据需要处理，即要依照数据之间的结构关系建立许多表，对这些表进行访问，进行某种运算，根据需要对它们进行修改，或在适当的时侯把它们销毁。实现这些工作就要弄清数据之间呈现出来的结构关系，了解它们在计算机内的存贮方式，掌握在某种结构上进行删除、插入、搜索、分类等的运算方法。为此就要研究数据结构及其相应的算法。

数据结构与算法之间有着密切的联系。不了解施于数据上的算法就无法决定数据结构。反过来，算法的结构和选择在很大程度上依赖于作为基础的数据结构。可以说，数据结构为算法提供了工具，而算法则是运用这些工具来实施解决问题的最优方案。因此作者将数据结构与算法有机地结合在一起，根据多年来讲授《程序设计》、《数据结构》、《算法设计与分析》、《数据库》和《管理信息系统》等课程的教学体会，编著了本书奉献给读者。

本书共分十章。第一章介绍数据结构、算法的基本概念及算法描述的类PASCAL语言。第二至五章从“算法＋数据结构＝程序”的思想出发，介绍几种数据结构的基本特性、存贮表示和有关算法，以及相应的应用程序。第六至八章从数据结构和算法相结合的角度，介绍用数据结构解决实际问题的技术——分类和搜索。第九章介绍算法复杂性的分析方法。第十章介绍对大型问题进行算法设计的六个基本策略思想。第二至五章可以作为程序设计技能的训练，第六至十章可作为算法设计能力的训练。每章都编有习题供读者选用。

本书中的各种算法用类PASCAL语言描述，可以根据需要翻译成上机运行的高级语言程序。另外书中还提供了一些应用程序的例子，选用BASIC语言编制，可上机运行。

本书在内容上力求做到图文并茂，深入浅出，脉络清楚。对每个算法都阐述其设计思想和实现方法，并用具体例子加以说明。凡学过一门高级程序设计语言的人，都能看懂书中内容。相信本书对读者会有一定的启发作用。

本书的出版得到曾宪昌教授、邹海明教授、姚天顺教授、衷仁保教授、王以德副教授、胡湘凌老师、刘又诚老师的热情支持、指导，借此机会向上述同志以及为本书的出版付出了辛劳的所有同志致以衷心感谢！

作者水平有限，差错难免，敬请批评指正。

徐绪松

于武汉大学

目　录

第一章 绪 论

§ 1.1 数据结构和算法

电子计算机是一种信息处理装置，计算机程序要处理大量的信息和数据，也就是对一些信息表进行插入、删除、分类、搜索等运算。这些信息中的各数据元素之间有着一定的结构关系，它们之间的结构关系如何表示？这些数据和信息在计算机中如何存贮？以及处理这些数据有些什么样的技巧？这就是本书要研究的问题。

什么是数据结构？

首先了解一下什么是数据。直观地说，数据是描述客观事物的数字、字母和符号，是计算机程序使用和加工的“原料”。数据的基本单位是数据元素，性质相同的数据元素的集合叫做数据对象。数据对象中的元素彼此之间存在的相互关系叫做结构。

数据结构指的是数据之间的结构关系。具体来说，它包括数据的逻辑结构和数据的物理结构。

数据的逻辑结构——仅考虑数据元素之间的逻辑关系。它包括线性结构（如线性表、栈、队）和非线性结构（如树、二叉树和图）。

数据的物理结构——指数据元素在计算机存贮器中的表示，即存贮结构。比如向量、链表。

一种逻辑结构通过映象便得到它相应的存贮结构。同一种逻辑结构可以映象成不同的内部存贮结构。反过来，数据的存贮结构一定要反映数据之间的逻辑关系。

为了更具体一些，我们举一个例子。

有一叠朴克牌，希望在计算机中表示这一叠朴克牌的内容（也就是这一组信息）。

在计算机内一组信息是由一组结点组成。在这里我们用一个结点表示一张牌，为了说明这张扑克牌的内容，必须将它的花色（梅花、方块、红心、黑桃）、点数、正反面、名称表示出来。同时还要将这张牌的下一张牌表示出来。

为此，用五个域组成一个结点。如图 1.1 所示。

图1.1 结点的格式

其中TAG表示牌的正、反面（用0、1表示），SUIT表示花色（用1、2、3、4、分别表示梅花、方块、红心、黑桃），RANK表示点数，NEXT表示下一结点的地址（即组成该

结点的那些存贮单元的首地址），TITLE表示这张牌的名称，用五个字符表示。

如图1.2所示的一叠朴克牌，它的逻辑结构是线性表：（方块2，梅花3，黑桃10（反）），在计算机中相应的存贮结构用链表表示，如图1.3所示（也可以用别的存贮方式）。

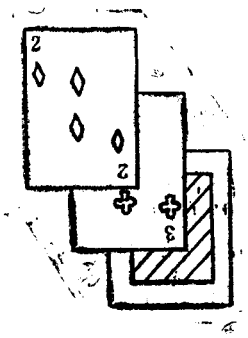


图1.2

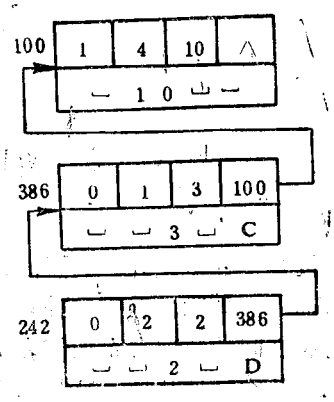


图1.3

数据结构直接影响计算机进行数据处理的效率。比如一个自动电话查号管理系统，需要将一个电话号码簿（它记录了n个人的姓名和相应的电话号码）存贮在计算机中，并要求设计一个算法，当给定任何一个人的姓名时，能打印此人的电话号码。若电话号码簿中无此人，也要报告没有这个人的标志。为解决这个问题，我们按姓名的字典顺序，用向量方式存贮姓名及相应的电话号码。若要查找某人的电话号码，就可根据姓名的第一个字母，去查找以该字母开头的姓名，这样查找将是方便而快速的。但是，如果姓名和对应的电话号码在计算机中的排列次序没有任何规律，只能逐一查找，这将相当费时。且当城市很大，有电话的人又很多时，那么自动电话查号管理系统，采用这种逐一查找方法就行不通了。

对于这个电话号码簿，当需要增加一个姓名及相应的电话号码时，或者某人调到其他城市工作，他的电话折掉时，就需要在已安排好的结构上进行插入或删除。因此还必须对数据结构定义某些运算，并设计出相应的算法，使得经过这些运算后，得到的新结构仍然是原来的结构类型。因此，我们还要进一步研究算法。

所谓算法，就是解决某一问题的办法，更确切地说，就是对于某一特定的问题，算法给出了解决这个问题的一系列（有穷的）操作，而每一操作都有它的确定性的意义（使得计算机能够遵照它的指示工作），并在有限时间（有穷步骤）内算出结果。另外，一个算法应有多个输入量，它是问题给出的初始数据，经过算法的实现（一系列操作），它有一个或多个输出量，这是算法输入运算的结果，即问题的解答。

我们研究算法就是研究算法设计技术，以及算法复杂性的分析，以选取一个好的算法。

§1.2 类PASCAL语言

前面谈到对于数据结构的研究都要讨论在各种数据结构上的基本运算，并给出相应的

算法，这就需要选取程序设计语言来描述算法。本书中我们选择类PASCAL语言。这是一种示意性的语言。用这种语言写出的算法，很容易翻译成真正的高级语言文本。之所以采用它，是因为它的表现能力强，且用它描述的算法简单、易读、一目了然。

下面对它的一些主要语句进行说明。

（1）所有算法都写成如下过程形式：

```
PROCEDURE    过程名（参量表）;
BEGIN
  语句组
END;
```

其中，参量表可以含有若干个参量；语句组由一个或一个以上的语句组成，用“；”号作语句的结束符，但最后一个语句结束时不带分号。

（2）基本语句：

赋值语句

变量名：＝表达式；

条件语句

```
IF  条件  THEN  语句 1;
```

或

```
IF  条件  THEN  语句 1
          ELSE  语句 2;
```

它们的含义（或执行方式）用下框图表示：

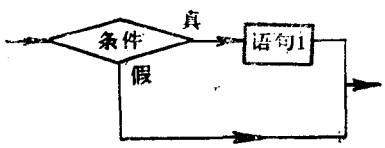


图1.4

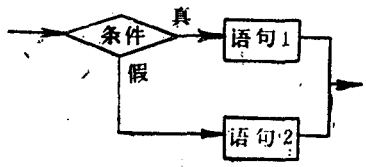


图1.5

为了方便起见，我们将使用几种循环语句。

WHILE DO 语句

```
WHILE  条件  DO  语句;
```

其含义如下：

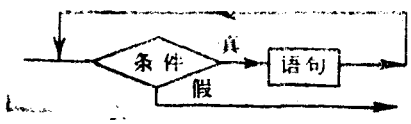


图1.6

只要条件成立，做DO后面的语句；一旦条件不成立，就做下面的语句。

REPEAT——UNTIL语句

```
REPEAT
```

语句组

UNTIL 条件;

它的意义是

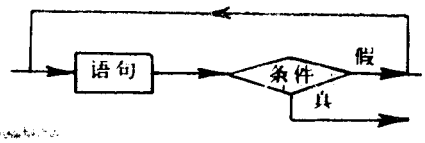


图1.7

反复执行语句组，直到条件满足为止。

FOR 语句

FOR 变量：=初值 TO 终值 DO 语句;

或

FOR 变量：=初值 DOWNTO 终值 DO 语句;

在这两个语句中，前者的终值大于初值，增量为1；后者的终值小于初值，增量为－1。

情况语句

CASE

条件1：语句1;

条件2：语句2;

…

条件n：语句n;

ELSE 语句n＋1

END;

其意义是

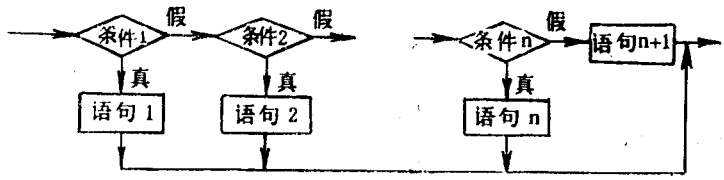


图1.8

取其中一种情况，即条件i成立，便做语句i。

调用过程语句

CALL 过程名（参量表）;

输入输出语句

read（变量表）; write（变量表）;

第二章　线性表和向量

§ 2.1　线性表及其存贮结构

§ 2.1.1　线性表

在计算机程序中最常用、最简单的一种数据结构是线性表，它是一种线性结构。

线性表是n（≥0）个数据元素$a_1, a_2, \cdots, a_n$的有限序列，它的结构性质在本质上只有元素间的线性关系。

上面提到的数据元素，或者叫结点，或者叫记录，是独立的信息。它可以是一个单独的符号。如：英文字母表（A，B，…，Z），数据元素是一个字母；也可以由若干个数据项组成，如：一个企业单位的全体职工档案登记表（图2.1），这里数据元素由姓名、职工号、性别、职务、工资五个数据项组成。

姓　名	职工号	性　别	职　务	工　资
陈　琳	820721	男	厂　长	104
王玉英	820722	女	工程师	136
刘　萍	820723	女	会计师	85
张　健	820724	男	工　人	77
⋮	⋮	⋮	⋮	⋮

图2.1　职工档案登记表

具有大量记录的线性表称作文件。如上列职工档案登记表构成一个职工档案文件。

在线性表定义中，还提到一个线性关系。所谓线性关系，即在线性表中，除最后一个元素外，每个元素有且仅有一个直接后继。

如线性表：

$(a_1, a_2, \cdots, a_i, \cdots, a_n)$。$a_2$是$a_1$的直接后继。$a_n$是最后一个元素，没有直接后继。一般来说，$a_i$的直接后继是$a_{i+1}$，直接前趋是$a_{i-1}$。

数据元素在线性表中的位置只取决于它们自己的序号，即a_1是它的第一个结点，a_2是它的第二个结点，……。

线性表中数据元素的个数，称作线性表的长度。

通常，对线性表可以进行如下基本运算：

（1）求出线性表的长度；

（2）存放一个新的数据元素于表的第i个位置；

（3）在第i－1和第 i 个元素之间插入一个新的数据元素；

（4）删除第 i 个数据元素；

（5）将两个或两个以上的线性表合并成一个线性表；

（6）将一个线性表折成两个以上的线性表；

（7）复制一个线性表；

（8）按某个特定的值查找线性表；

（9）对线性表中的数据元素按某个数据项值递增（或递减）的顺序重新排列。

§ 2.1.2 向量

研究数据结构，一个很重要的方面是如何将抽象的数据结构映象到内部存贮结构上去，而使得我们所需要的运算能有效地执行。将线性表存放到计算机的存贮器中，一个常用的方式是顺序分配。也就是说，用一组连续的存贮单元，依次存放线性表中的结点，且每个结点占用相同数目的机器字。这种内部存贮结构叫向量。

向量是计算机存贮器中一组相毗邻的元素。每个元素是由个数相同且又毗邻的存贮单元组成。我们用 v 表示向量，用v[i]表示向量的第 i 个分量。向量的第 i 个分量是线性表第 i 个元素 a_i 在计算机存贮器中的映象，即v[i]：＝a_i。

假设线性表的第一个元素的存贮地址（指起始地址）是LOC（a_1），每个元素需占用1个存贮单元，则表的第 i 个元素存贮地址为：

$$LOC(a_i)=LOC(a_1)+(i-1)*1$$

线性表的顺序分配情况如图2.2所示。

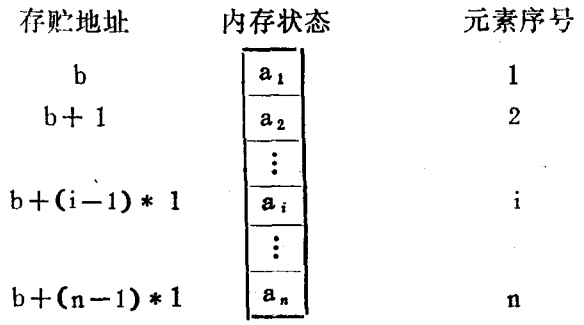


图2.2 线性表的顺序存放

在顺序分配时，用上界为n的向量表示长度为n的线性表。

§ 2.1.3 线性表的插入和删除运算

对线性表的插入和删除运算，在不同的存贮结构中，实现的方法也不同。这节我们仅介绍在线性表为顺序分配的情况下的插入、删除运算。

设有长度为n的线性表：a_1，a_2，…，a_n。插入运算是把新的元素 x 插在a_{i-1}和a_i之间。在进行这种运算时，需要做哪些操作呢？如图2.3所示，要把x插在a_{i-1}和a_i之间，首先要空出a_i上面的位置，这就要把从a_n到a_i的元素依次向下移动一个位置；再把x插到空出的位置上；由于插入一个元素后，线性表的长度发生了变化，故最后要修改容积。

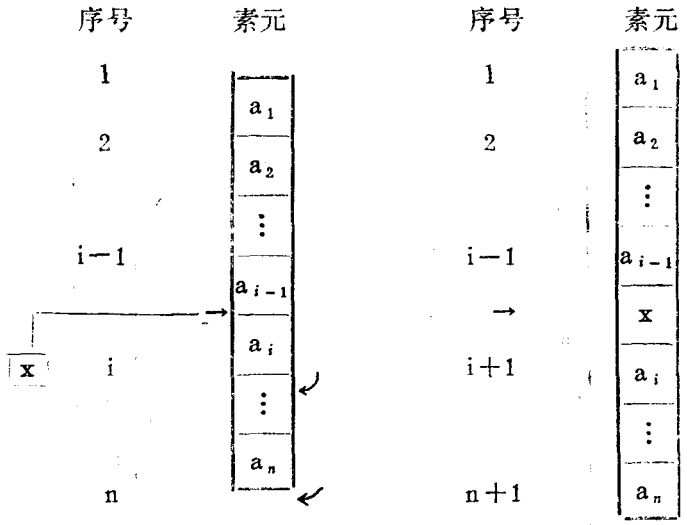


图2.3 在线性表中插入元素x

将这些操作用类PASCAL语言描述出来，便得到顺序存放的线性表的插入算法。

```
PROCEDURE insertsqlist(v, n, i, x);
    {在长度为n的线性表的第 i 个元素之前插入一个元素 x,v为存贮线性表的向量，v 的上界m>n}
BEGIN
1. IF i≤n
2. THEN [FOR j:=n DOWNTO i DO
                v[j+1]:=v[j];
3.          v[i]:=x;   n:=n+1]
END;
```

线性表的删除运算是把第 i 个元素从线性表中删去并送入某一单元 y 。具体来说，实现删除运算，需要做如下三步操作。

见图2.4，首先把元素a_i送入y单元；再删去a_i，为了保持结点间的连续性，我们把从a_{i+1}到a_n的元素依次向前移动一个位置；由于线性表删除一个元素后，长度减少，最后需修改容积。

相应的算法描述如下：

```
    PROCEDURE deletesqlist (v, u, i, y);
    {在长度为n的线性表中删除第i个元素送入y单元}
BEGIN
1. IF i>n
```

```
2.  THEN  write  ( 'No this element' )
3.  ELSE  [y:=v[i]; FOR j:=i  TO  n-1  DO
                        v[j]:=v[j+1];
          n:=n-1]
END;
```

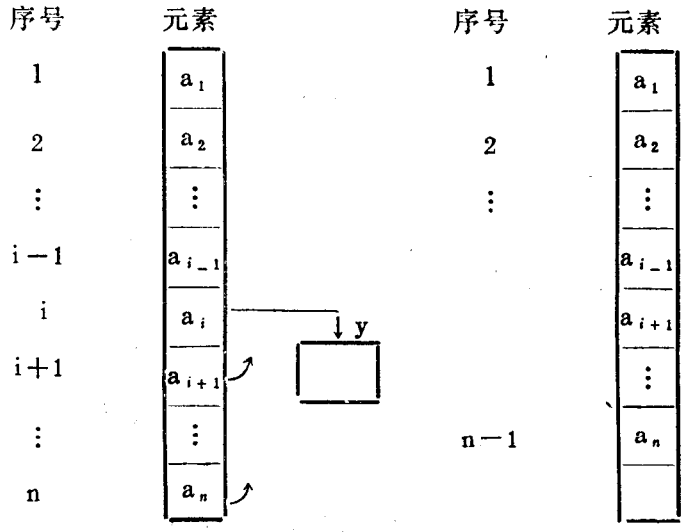


图2.4　在线性表中删除元素a_i送入Y

可能读者已经看到，若对顺序分配的线性表做很简单的插入和删除运算，在移动元素时就要花费大量的时间，有人计算过，大约需要移动表中的一半元素。这就是以向量作线性表的存贮结构的弱点。

§ 2.1.4　线性表的应用实例——仓库管理系统

将仓库中存放的一批货物，按顺序分配存贮在计算机中，是如图2.5所示的一个结点序列。试编制一BASIC程序对这张表进行四种运算：①打印第i个结点的值，②删除结点k（地址是I）的后继结点，③在结点k后插入一个新结点，④打印结点k的前趋结点和后继结点。

首先开辟一些存贮单元：

用　D$(M)——存放货物名称
　　D_1(M)——存放货物数量
　　D_2(M)——存放货物编号
　　I——下标地址
　　N——实现分支用

并用　　N$，$N_1$，$N_2$——存放新结点的值
　　L——已存放的结点数
　　J——控制循环的变量。

然后根据所要进行的四种运算，画出程序流程图。如图2.6所示。

地 址	货物名称	数 量	编 号
1	Machine	40	612
2	Motor	2	802
3	Block	3	105
4	Filter	2	117
5	Tank	10	118
6	Brakes	60	230
7	Ignition	30	408
8	Housing	17	807
9	Frame	40	230
10	Plate	1	702

图2.5 按顺序分配存贮的一批货物

最后我们写出它的BASIC程序

```
10 DIM D$(20), D1(20), D2(20)
20 FOR I=1 TO 10:READ D$(I), D1(I), D2(I):NEXT I:
   L=10:M=20
30 INPUT "N=" ; N:ON N GOTO 40, 70, 100, 150:END
40 INPUT "I=" ; I:IF I<1 OR I>L THEN 60
50 PRINT "PRINTER; " ; D$(I); D1(I); D2(I):PRINT:GOTO 30
60 PRINT "ERROR RETURN":PRINT:GOTO 30
70 INPUT "I=" , I:IF I>=L THEN 60:J=I
80 J=J+1:D$(J)=D$(J+1):D1(J)=D1(J+1):D2(J)=D2(J+1)
90 IF J=L-1 THEN 30:GOTO 80
100 INPUT "I, N$, N1, N2=" ; I, N$, N1, N2
110 IF L>=M OR I>L THEN 60:IF L=0 THEN 140:J=L
120 D$(J+1)=D$(J):D1(J+1)=D1(J):D2(J+1)=D2(J):J=J-1
130 IF J=1 THEN 140:GOTO 120
140 D$(I+1)=N$:D1(I+1)=N1:D2(I+1)=N2:L=L+1:GOTO 30
150 INPUT "I=" ; I:IF I<=1 OR I>=L THEN 60:PRINT
    D$(I+1); D1(I+1);
160 PRINT D2(I+1):PRINT D$(I-1); D1(I-1); D2(I-1):GOTO 30
170 DATA "MACHINE" , 40, 612, "MOTOR" , 2,802, "BLOCK" ,3,
    105, "FILTER" ,2,117
180 DATA "TANK" , 10, 118, "BRAKES" ,60, 230, "IGNITION" , 30
```