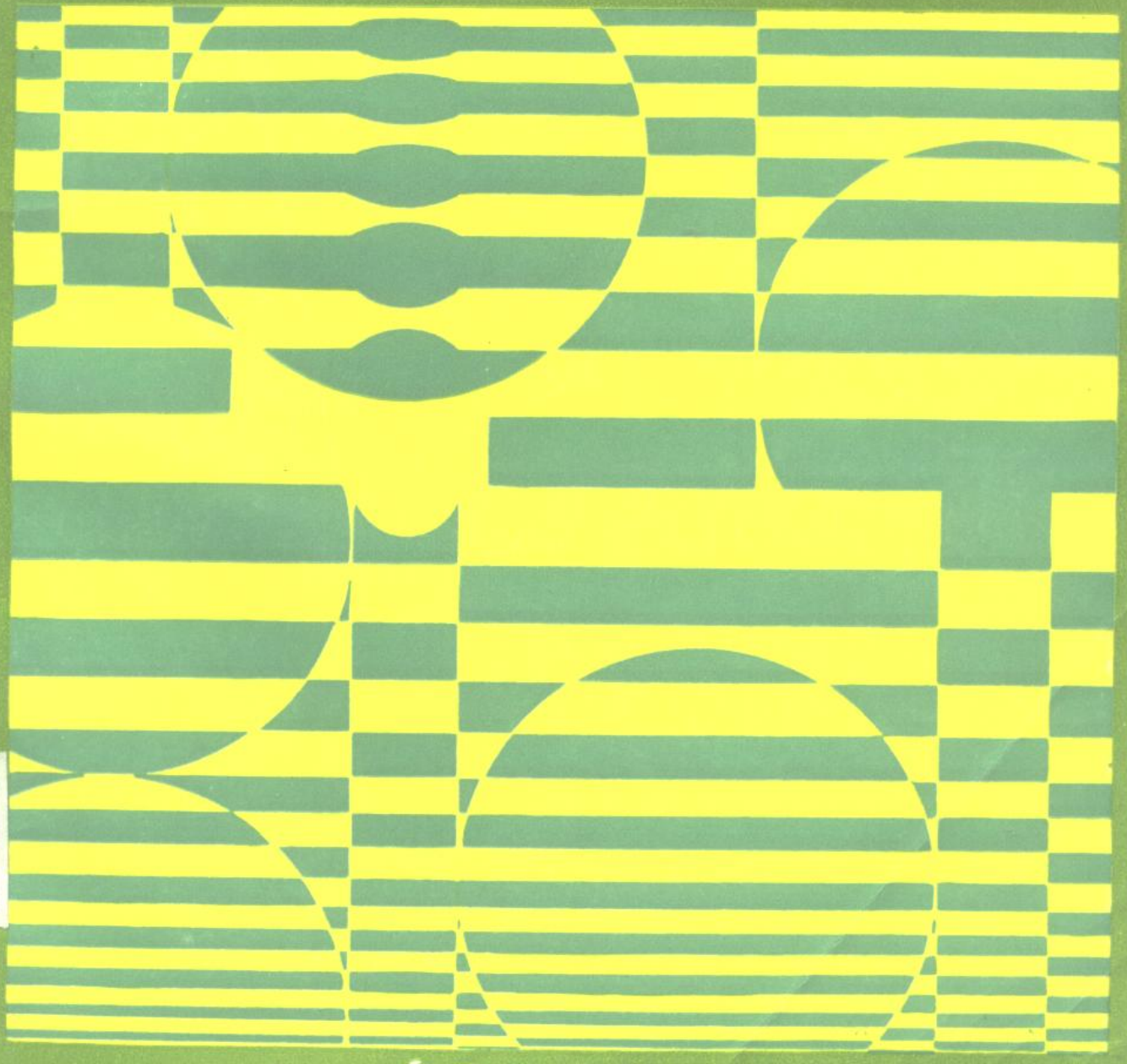
高等学校
工科电子类 规划教材

数据结构

●周岳山



電子工業出版社

数据结构

周岳山 编

电子工业出版社

内 容 简 介

本书介绍了计算机学科中常用的数据结构，内容包括线性表、栈队、列、串、数组、广义表、树、图、查找及排序等。

本书以培养与提高学生的基本专业素质及综合应用能力为目标，侧重内容的先进性、实用性及科学性。书中全面引入并应用面向对象的方法具体实现某些结构，每章均引入一个实例，将其抽象、改造和拓宽并归结为一个程序设计问题或一个演示程序，便于学生理解和掌握教学内容。

本书主要为适应高等职业教育而著，同时也可作为大专院校计算机专业必修课的教材及计算机科技人员与爱好者的自学参考书。

图书在版编目(CIP)数据

数据结构/朱振元，朱承编著. —西安：西安电子科技大学出版社，2000.7

高职系列教材

ISBN 7-5606-0878-7

Ⅰ.数… Ⅱ.①朱… 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 32027 号

责任编辑 云立实 戚文艳

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)8227828 邮 编 710071

http://www.xduph.com E-mail：xdupfxb@pub.xaonline.com

经 销 新华书店

印 刷 西安长青印刷厂

版 次 2000 年 7 月第 1 版 2000 年 7 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印张 20.5

字 数 484 千字

印 数 1～4 000 册

定 价 21.00 元

ISBN 7-5606-0878-7/TP·0462

前　言

本教材系按电子工业部的工科电子类专业教材 1991～1995 年编审出版规划，由中等专业学校计算机专业教学指导委员会征稿并推荐出版。责任编委凌林海。

本教材由上海电子技术学校周岳山担任主编，贵州无线电工业学校黄大胜担任主审。

本课程的参考学时数为 72 学时，其主要内容为数据的逻辑结构、物理结构以及对每种结构所定义的运算及应用。其中包括线性表、链表、数组、串、树和图结构以及相应的存储表示。本课程还介绍了程序设计中大量存在的查找和排序问题。本教材偏重于数据结构的算法和算法的应用，对于较复杂的算法都配有形成算法的流程框图。教材中出现的算法用类 Pascal 语言描述，重要的算法还给出 Pascal 程序，在算法的描述中，以非递归程序为主，同时给出算法的递归表示。为了便于阅读教材和提高对数据结构的应用和程序的调试、动手能力，在每章的结束都有应用举例和 Pascal 程序，全书的结尾还附有上机实习和课程设计的要求。使用本教材时应注意讲述数据结构的基本概念、形式和存储方式，着重描述每一种数据结构的算法及算法的基本应用。本课程是实践性较强的课程，要注意算法及算法应用的上机调试。

本教材由卢传友编写第七、八、九、十章，周岳山统编全稿。参加审阅工作的还有沈美琴，倪秉营等同志，他们都为本书提出许多宝贵意见，这里表示诚挚的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

1993 年 6 月

目　录

第一章 绪 论

当今世界已经进入了信息时代，自第一台计算机问世以来，计算机的发展已远远超出人们对它的估计。计算机的应用已逐步深入到社会的各个领域，其加工处理的对象也从简单的数值、字符发展到具有一定结构关系的数据。在情报检索、企业管理、数据库、网络通讯、人工智能等领域，对数据的处理尤为显得重要。如何使计算机能高效率、高可靠性的准确处理数据，就需要对数据的组织、数据内部的逻辑关系加以深入地研究，这就促使了“数据结构”的形成和发展。

§1.1 数据结构的发展

《数据结构》作为一门独立的课程体系是从1968年才开始的。但数据结构的有关内容，如表处理、串处理、树结构等已散见于计算机的其他课程，如操作系统、编译原理、表处理语言中，都缺乏规范性和系统性。由于当时主要使用的计算机语言是以数值处理为对象，对单数据进行复杂的运算时，有较高的效率；但对于非数值的运算，如非数值处理对象（数据库）的查询、加工、修改带来一定困难。从而形成了以数据为中心的程序设计方法，这对形成和发展数据结构起了一定的推动作用。当时，数据结构包含的内容相当庞杂，其内容几乎和图论、表、树的理论相等同。之后又扩充为网络、代数结构、集合论、关系等方面。这就是目前形成的两大课程：数据结构和离散数学。近年来，由于数据库系统的不断发展，在数据结构课程中又增加了文件管理的内容。

1968年出版的美国研究计算机科学的著名教授唐·克努特（D. K. Knuth）所著《计算机程序设计技巧》的第一卷《基本算法》是第一本较系统地阐述数据的逻辑结构和存储结构的著作。七十年代初以来，系统软件和大型应用程序不断推出，结构程序设计思想成为程序设计方法的主要内容，人们对数据和结构的研究更加广泛和深入。从七十年代中期到八十年代初各种版本的数据结构著作不断涌现。

《数据结构》在计算机科学中有着十分重要的地位，在我国计算机专业教学计划中是核心课程之一。它是学习、设计和实现编译程序、操作系统、数据库、人工智能和其他应用程序及系统程序的基础。

§1.2 什么是数据结构

数据结构（Data Structures）包括二方面的内容：数据和结构。数据是计算机程序进行加工处理的对象。在计算机技术发展的初期，由于计算机主要用于数值的计算，处理的对象大多是数量较少，结构简单，但计算过程复杂的数值数据，程序设计的重点是在程序的设计技巧上。随着计算机科学技术的发展，计算机的应用领域不断扩大，以数值为基础的数据概念被大大的扩展，越来越多的非数值数据需要处理，如财务管理中的大量表格、人事档案管理中

的文字处理，这不仅涉及到数值类型的变化，而且还关系到数据中结构的处理和对于这些结构在程序设计中的描述。有关资料表明，当前非数值计算的处理已占用了80%以上的计算机计算时间。如何认识数值结构在程序设计中的重要性是学习和研究计算机技术的一个重要课题。我们可以通过以下的例子加以简要的说明。

最简单的数据结构的例子是一维数组表示。例如，要计算100个数 $a_1,a_2\cdots,a_{100}$ 的和，一个原始而简单的办法是把它们逐一相加，即：

$$S=a_1+a_2+\cdots+a_{100}$$

显然，用以上的赋值语句来描述是极不方便的。我们可以把这100个数从排列的结构上组成一个一维数组，用 Pascal 语言描述就是 array A[1：100]。在此基础上，计算这100个数和的程序段可以描述为：

```
Sum：=0
for i：=1 to 100 do
    Sum：=Sum+A[i];
end
```

计算结果在 Sum 中。这个程序段比 S：$=a_1+a_2+\cdots+a_{100}$ 这样的语句要简明得多，况且在程序设计中不允许书写省略号。这里实际上包括了两方面的内容：第一是数据结构A[1：100]表示的数组；第二是用数组为数据结构的形式所描述的算法。从而达到了使程序设计简明、易读。

较复杂的一个例子是用数据结构中树结构完成对应用程序中的数据处理。以下给出的是某学校管理程序中程序流程的框架如图1-1和某班级学生成绩表如图1-2。

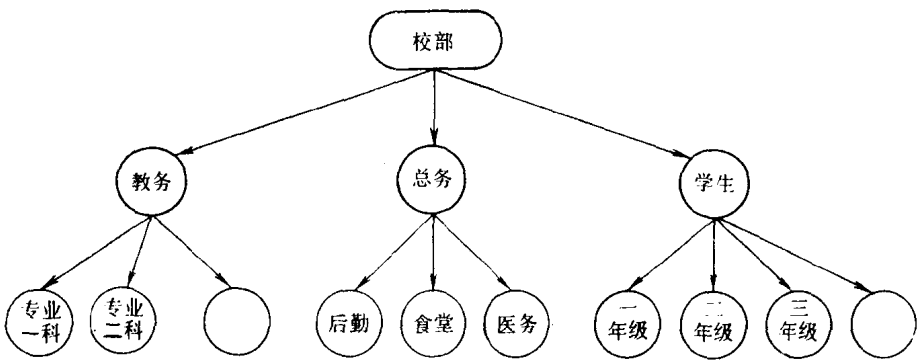


图1-1　学校管理程序流程框图

用计算机实现对学校管理，经课题调查、研究，按管理的模式确定程序的数据处理模式和流程，即图1-1，以及对具体部门数据的结构形式，如图1-2的学生成绩表。若在学生成绩管理中要查询某班、某学生的学习情况。其操作的逻辑关系为：从校部即根结点入口，选择查询的三个流向，确定“学生”这个结点的数据流；又通过“学生”这个结点选择某个年级，再从某个年级这个结点最终确定需查询的某个班级这个结点，该结点在数据结构中称作为叶结点，而整个结构就称为树结构。这种以树的形式所表示的数据结构不仅反映了整个管理的层次非常清晰，数据流向也一目了然。从程序设计角度看数据处理方便，结构化程序设计清楚，

学号	姓名	Pascal语言	电工	程序设计	数据结构	微机原理
1	张平	82	76	91	82	74
2	李园	90	85	86	88	79
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
35	赵晶	75	82	78	75	83

图 1-2　学校成绩表

软件的维护和管理也容易。至于查询的最后一个结点，某班的学生成绩表就是数据结构中最常见的线性表。在以线性表所表示的学生成绩表对某个学生的学习情况进行查询也是很容易实现的。用以上所涉及的树和线性表来表示和实现学校管理的应用软件是合适的。

在学校的整个管理系统中会出现下列的问题：当校部在管理模式要增加一个部门，如校办工厂。在这个树型结构中如何插入，这就导致了树结构中结点的插入问题；另外，如果某个专业科由于专业调整而被取消了，这时必然会影响该树的树型，实际上就是树结构结点的删除问题；当增加一个专业时，这棵树中就增加了一个结点。为了适应这种数据的查找、插入、删除的运算，设计了对应的算法。

通过以上的示例可以看到：现实的世界是信息世界，所谓信息就是客观存在的反映，而数据就是信息的表现形式和描述，这种描述是为了能被计算机所识别、存储和处理。这种描述一般可归结为数字、字符和各种符号的集合。如上例中的教师、学生、成绩、设备等。值得注意的是数据结构中所研究的数据不是孤立的，而是表现为相互关联的数据。如某学生的姓名以及该学生的学科成绩的数值都作为数据存在，其数据与数据之间的关系为：学生姓名与班级名对应，成绩与学科相匹配。这种数据元素之间存在的相互关系就是数据的结构，也称作为数据之间的逻辑结构。学校管理系统中各部门的上、下级关系表现为树结构中的层次关系；学生成绩与各学科的联系表现为数据之间的线性关系。把数据和表示数据之间的关系，即结构在计算机中的存储形式，称作数据的物理结构或存储结构。

至此可以认为：数据结构就是研究数据的逻辑结构和物理结构，以及它们之间的相互关系和所定义的算法在计算机上运行的学科。本书中将要讨论的逻辑结构有线性关系的表、栈、队列、数组、串和链、非线性关系的包括树和图的结构。它们分别以顺序和非顺序的映象关系，在内存中表示不同的存储结构。本书还将讨论并给出上述结构的算法和算法的基本应用，同时附有相应的 Pascal 程序。

§1.3　算法语言的描述和算法分析

学习数据结构是为了更有效的提高程序设计的效率，使所设计的程序结构合理、层次清晰、内存占有量少、运行时间短。其中，关键环节之一是数据结构算法的设计和对该算法在运行中的时间分析。

§1.3.1 描述算法的语言

数据结构既是一门理论性学科，更重要的是一门实践性很强的课程。通过该课程的学习，能运用数据结构的基本概念以便更好地应用算法进行程序设计。在讨论各种数据结构的基本运算时，除了讨论和给出基本算法外，还给出相应的算法流程框图。流程框图不仅有文字说明，而且还配有对应的表达式，力图使读者在掌握和理解了框图所表示的程序设计思想后，能较方便地用读者熟悉的计算机语言来编制源程序。

为了便于理解和掌握算法的设计思想和实质，选择一种合适的算法语言来描述是一个重要的问题。以结构化程序设计思想为基础，就标准的 Pascal 语言而言，其结构严谨、条理清晰，但该语言首部说明太细，约定太多，不便于书写表达。本书采用的是既保持 Pascal 语言结构严谨的特点，又用易于书写、便于阅读和理解的类 Pascal 语言。该语言略去了对于变量的说明和一些约定。这样，读者可以把注意力集中到算法的实质，而不是把精力花费在某种高级语言的许多具体约定上。如果我们希望在计算机上实现类 Pascal 语言的算法，可以很容易通过人工翻译，转换成具体的能在该机器上执行的源程序。当然，如果有一个进行预处理的翻译器，由机器实现自动翻译，那就更方便了。

以下我们对类 Pascal 语言作一些必要的说明和举例。

1. 所有的算法都以如下过程形式表示。

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

例：循环计算 100 个数的和。

```
PROCEDURE SUM(A)
    BEGIN
      S：=0
      for i：=1 to 100 do
          S：=S+A(i)
      end
      write(S)
END SUM
```

其中过程名是 SUM，参量是数组 A，并假设有 100 个数在 A(1：100)中，write(S)表示输出累加 100 个数的结果。

2. 在过程中的任何处均可用一对花括弧括起来的中文注解，以便对算法或算法中的重要点作必要的说明，帮助读者阅读和理解。

3. 输入和输出，假定存在两个标准过程。

read(变量表)；　　write(变量表)。

4. 基本语句

(1)赋值语句

变量名：=表达式

例:A：=10;　　A(i)：=x+y;

(2)条件语句

if　条件　then　语句1(有多语句用[ ]表示)或　if　条件　then　语句1　else　语句2

例:if i=1 then x：=y+5;

```
if i=1 then x：=y+5
      else  [x：=y+10; z：=x+y]
```

(3)循环语句

有四种实现循环方式的语句可供使用。

```
a. while 条件 do
 ·      语句组
  end
```

例:i：=1;　j：=1

```
while i〈 〉5 do
      i：=i+1
      j：=j+i
end
```

在执行 while 语句时,只要条件成立,就执行 do 之后的语句或语句组,直到条件不成立时,循环才结束。值得注意的是如果条件一开始就不成立,那么循环体中语句就根本不执行,跳过循环而执行下面的语句。也就是说根据给出的条件可能一次也不执行 while 语句的循环体,也可能执行 while 语句中循环体一次或多次。

```
b. repeat　语句组　unti 条件
```

例:read(i);　j：=1

```
repeat
  i：=i+1
  j：=j+1
until i=5
```

reapet 语句的特点是无论给出的初值和条件如何,它至少执行一次 reapet 与 until 之间的语句组一次,然后直到 until 条件成立才停止重复执行,而去执行 repeat 后面的语句。在上例中可看出,无论 read(i)读入的是什么初值,都必须执行 reapet 后的语句组。

```
c. for 语句有二种形式
for　变量：=初值　to　终值　by　步长　do
    语句组
end
```

或

```
for　变量：=初值　downto　终值　by　步长　do
    语句组
end
```

前者的步长是正值且终值大于等于初值,后者的步长是负值且终值小于等于初值,两者的步长为+1时,by 步长可省略。从给定的初值、终值、步长可以确定 for 循环语句中的语句组一

次也不执行、只执行一次或执行多次。例如当步长为正，初值大于终值则循环体中的语句一次也不执行；如初值等于终值则仅执行一次；如终值大于初值则执行一次或一次以上。

例：有一数组 A(1：n)，要求对此数组输入数据，然后输出该数组的数。

```
for i：=1 to n do
  read A(i)
end
for i：=n downto 1 by -1 do
  write (A(i))
end
```

该程序段执行结果是输出的数据的秩序正好与输入数据的秩序相反。

d. loop 语句组 forever

这是一个无限循环语句，只有在语句组中出现 exit 命令时，程序才跳出循环，把控制转给 loop 语句之后的第一个语句执行。exit 可以有条件的使用，也可以无条件的使用。

例：当 i 的值有条件的等于 n 时，程序跳出 loop 循环，输出 i 的值。

```
i：=1
loop
  i：=i+1
  if i=n then exit
forever
write(i)
```

loop 循环语句在不执行 exit 语句时，程序将是死循环，因此在使用时应该谨慎。

(4)情况语句

```
case
  条件 1：语句 S1
  条件 2：语句 S2
    ⋮
  条件 n：语句 Sn
  (else 语句 Sn+1)
end
```

case 语句用于多路分支的判断，它可以方便地根据具体情况从若干个给定的条件中作出选择，而不必多次使 if-then-else 语句，使程序设计简单明了。

例：输入一个数值，从而确定执行的语句。

```
read(i)
case
  i=1 ： write('A')
  i=2 ： write('B')
  i=3 ： write('C')
  else write('D')
end
```

在执行该程序段时，若输入 i 的值是 2，则输出字母'B'，若输入 i 的值不是 1、2、3；则输出字母'D'，若程序段作一个修改，删去 else 这一句，若输入 i 的值也不是 1、2、3，则执行有什么变化，请读者考虑。

(5)调用过程语句

call 过程名（参量名）

使用该语句可以调用算法已设置的各种过程，该语句的使用可以使算法更具结构化，逻辑清晰。在算法或程序设计中，对常要使用的功能部分，可事先用过程描述，便于使用时调用。

(6)出错处理语句

error(字符串)

该语句主要使用在当算法出现异常情况时，表达当时程序执行的处理情况，如数组溢出时可表达为：

error('上溢')；

又若在栈空时可表达为：

error('stack-empty')

等。

(7)停止执行命令

stop

该语句使算法在任何情况停止执行。

对于类 Pascal 语言，尚有几点说明如下：

(1)本语言没有对数据类型作定义，而在算法中直接使用。在过程调用中也没有对形式参数区分。

(2)本语言允许一些简明的数学记号。如[X]表示对 X 值下取整，[X]表示对 X 值上取整等。

(3)过程允许嵌套调用和递归调用。

(4)本语言原则上不使用 goto 转向语句。

下面给出一个文件中数据匹配的算法，说明类 Pascal 语言如何使用和阅读。假如把文件中每个记录的关键值(区别每个不同记录的数据)均存放于数组。由于文件中的记录有时作成批处理。我们把要处理文件中的数据，称为主文件数据，记为 m(j)其中 $1 \leqslant j \leqslant q$，它是有序的，把成批待处理记录组成的新文件称事务处理文件，其事务处理数据为 t(i)，$1 \leqslant i \leqslant p$，它也是有序的，若 m(j)和 t(i)的值相等，则说明文件的记录和事务处理文件的记录相匹配。m(j)和 t(i)合并处理后产生新文件的记录 n(k)，r(l)是 t 文件中非处理文件，u(h)是 m 文件中非活跃文件。图 1-3 是成批数据匹配流程图，图 1-4 是数据匹配图。

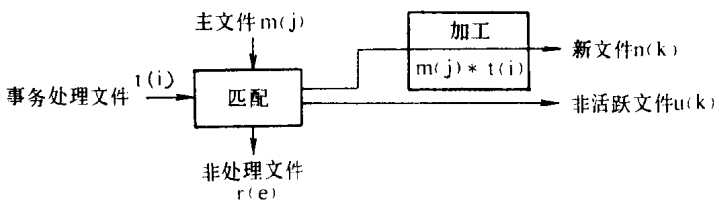


图 1-3　成批数据匹配流程图

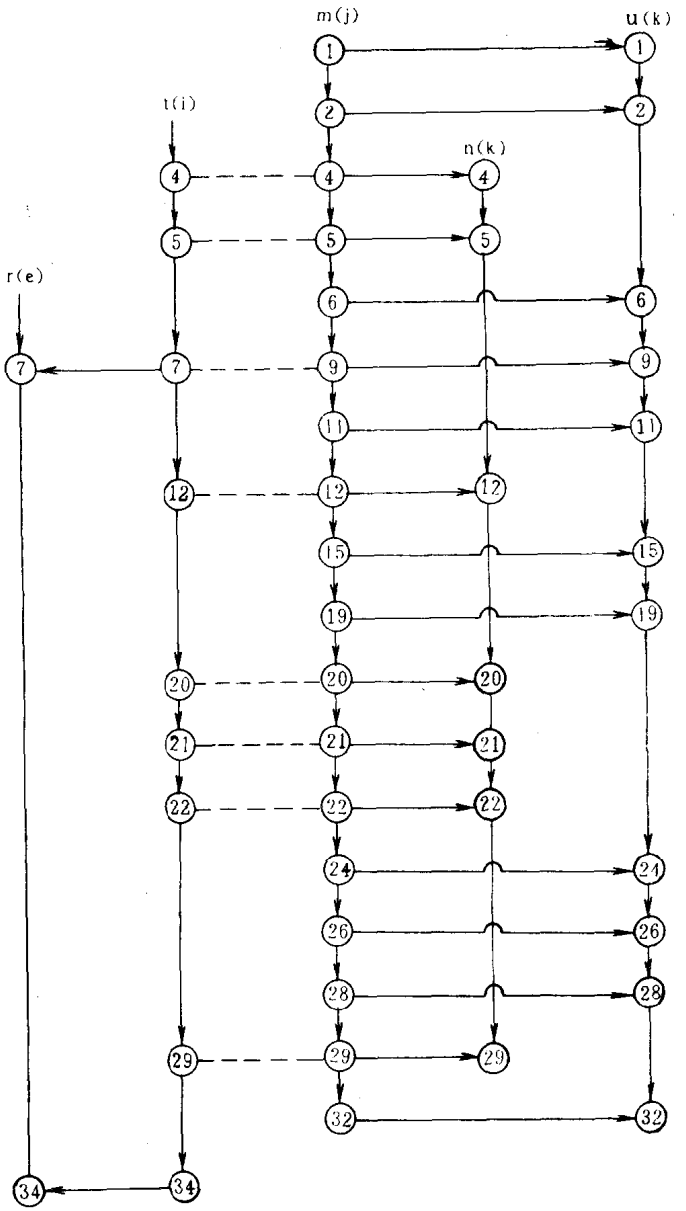


图 1-4　数据匹配图

这个成批数据匹配过程的程序可以按流程框图和类 Pascal 语言的书写规则，编写成如下的算法。算法中文件记录内的数据都以数组表示。

```
procedure PF · (m(q),t(p))
{m 为主文件数组,t 为事务处理文件数组,r 是非处理文件数组,n 是新文件。}
  k : =0; l : =0; j : =1; i : =1
  while i≤p do
        if j≤q
            then case
```

```
                t(i)=m(j):[k:=k+1 n(k):=t(i)*m(j) j:=j+1;i:=i+1]
                t(i)<m(j):[l:=l+1 r(l):=t(i) i:=i+1]
                else j:=j+1
            end
        else[l:=l+1;r(l):=t(i);i:=i+1]
end PF
```

若该算法中同时产生非活跃文件 u(h)，如何修改此算法，请读者在阅读该算法后自行修改和设计。

§1.3.2 算法分析说明

对数据结构中的算法或计算机的程序设计，算法分析是十分重要的，同时也是相当复杂的。本教材就此内容仅作一般的介绍。对于某一算法如何取得算法分析的结果不作进一步、深入地推导。重要的是对已经获得的算法分析的结果如何在应用程序中的使用。

那么，如何衡量和评价一个算法的优劣呢？假如所设计的算法在逻辑上是可行的，那么，虽然评价算法的标准很多，但是通常有三个方面的考虑因素：

(1)执行算法后，在计算机中运行所消耗的时间，即所需的机器时间。

(2)执行算法时，在计算机中所占存储量的大小，即所需的存储空间。

(3)所设计的算法是否易读、易懂，是否容易转换成其他可运行的程序语言。

从理想的角度讲，我们希望所设计的算法是一个既简单易懂，又占内存量少，运行时间短的程序。然而，在实际应用中，往往是一个看来很简便的程序，其运行时间要比形式上复杂的程序慢的多。譬如科学计算中求多阶方程的解，运行一个时间较短的程序，往往占用的内存空间较大；如数据库中数据文件的处理，所占用的时间就比较长。因此，不同的应用对算法有不同的选择。若该程序只使用一次或若干次，则力求算法简明易读，容易转换成执行的语言，同时便于上机调试；若该程序需要反复多次运行或多次调用，则应选用执行时间尽可能短的算法；若待处理的数据量甚大，而所使用的计算机存储空间相对较小，则选择的算法应主要考虑如何节省空间。考虑到算法实际分析的多重性和复杂性，本教材主要讨论算法的时间因素，以及如何简要计算该算法或程序的具体执行时间和估算出算法在执行时间上的量级。

一个算法的执行时间等于算法所有语句执行时间的总和。而任一语句的执行是该语句的总执行次数与执行一次所需时间的乘积。由于精确的计算出该算法的总执行时间是相当复杂的，而且是困难的。因此，只能根据给定的计算模式，粗略地估算该算法在执行时间上的量级。这里，我们用大写的“O”符号来表示量级的概念。

设有程序段表示两个 $n\times n$ 的矩阵相乘，其算法描述如下：

语句标号	语句	执行次数
(1)	for i:=1 to n do	$n+1$
(2)	for j:=1 to n do	$n(n+1)$
(3)	c(i,j):=0	n^2
(4)	for k:=1 to n do	$n^2(n+1)$
(5)	c(i,j):=c(i,j)+a(i,k)*b(k,j)	n^3