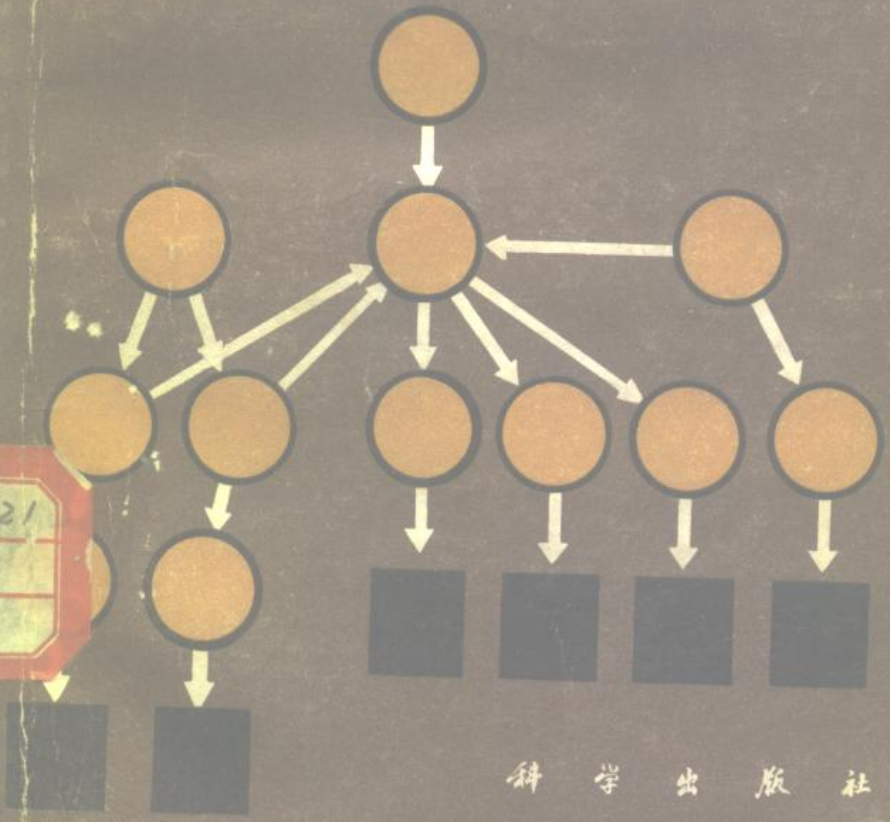
数据结构与程序设计技术

〔西德〕H. H. 莫勒 著



科学出版社

数据结构与程序设计技术

〔西德〕H. H. 莫勒 著

孙永强 张 然 译

科 学 出 版 社

1984

内 容 简 介

本书原版为德文，后由 Camille C. Price 译成英文．此书是根据英文版转译的．

全书共四章．第一章介绍本书所用到的预备知识，后三章系统地介绍了三种最基本的数据结构——并列表、树、图以及这些结构的各种存贮方法和处理方法．书中通过大量精选的实例介绍这些数据结构的典型应用，并附有实用程序．

本书的特点是以统一的观点讲述数据结构的基础，重点突出、条理清晰、深入浅出、易于阅读．

本书可作为大专院校计算机软件和硬件专业的数据结构课程的教科书或教学参考书，也可供有关专业的科技人员参考．

Herman H. Maurer

DATA STRUCTURES AND PROGRAMMING TECHNIQUES

Prentice-Hall, Inc.

数据结构与程序设计技术

〔西德〕H. H. 莫勒 著

孙永强 张 然 译

责任编辑 李淑兰 那莉莉

科学出版社出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1981 年 8 月第 一 版 开本：787×1092 1/32

1984 年 8 月第二次印刷 印张：9 7/8

印数：9,051—30,130 字数：225,000

统一书号：15031・351

本社书号：2202・15—8

定价：1.55 元

英译本前言

本书译自德文，是以卡斯鲁赫大学教授 H. Maurer 博士指导的课程为基础编写的. 这门课程的讲义在 H. W. Six 的协助下，经过全面的修改和扩充，以《Datenstrukturen und programmier-verfahren》为书名出版.

本书旨在介绍使用电子数字计算机解决问题时适用的组织数据的方法. 这些方法将有助于设计出有效而周密的程序. 因此每个计算机工作者都应该熟练地掌握它们.

本书只要求读者对某种程序设计语言有所了解，不要求预先在数学或计算机科学方面具备专门的知识，因为本书是自成系统的.

在本书末汇总了书中用到的记号并作了注解. 本书自始至终使用的是在 1.2 节和 1.3 节引进的记号，这使得我们可以一致而精确地描述书中的内容. 这种表示方法十分简单，一般读者不必仔细研究记号的格式就能阅读.

全书使用程序段描述算法并介绍数据结构的处理. 使用汇编语言可以直接访问单个的存贮单元，但是由于没有一种标准的汇编语言可供使用，同时也为了方便起见，书中使用了高级语言——PL/I 语言的一个非常简单的子集（实际上只是略加扩充的 FORTRAN 语言）. 因此，稍微具有一些程序设计语言知识的读者都能读懂书中的程序. S. Reiniger 小姐测试了本书的 PL/I 程序，谨此致谢.

Camille C. Price

引　言

为解决给定的问题而设计的程序，其有效性、清晰性和复杂性同程序中用到的数据的组织方式，即所选取的数据结构有密切关系．本书旨在介绍最重要的数据结构，说明在哪种条件下用哪种数据结构最好，并指出在处理各种数据结构时最好使用什么样的程序设计方法．

在研究数据结构时，不仅要定义数据结构概念的本身，还要说明如何存贮和处理数据结构．因此，我们的做法是：首先根据数据结构概念的一般定义给出用图形表示数据结构的方法，然后引进计算机存贮器的一种模型，以及用图形表示存贮单元的方法；在此基础上介绍存贮数据结构的方法．我们用 PL/I 程序具体描述存放在计算机中的数据结构的处理算法．在程序中，用 PL/I 的"变量说明"取得计算机存贮器模型中的存贮单元．

第 1 章的内容是全书的基础．第 2 章介绍常用而简单的数据结构——**并列表**．第 3 章着重介绍极为重要的树结构．在第 4 章引进两类复杂的数据结构．

目　　录

1. 处理数据结构的一种模型

1.1 数学方面的预备知识

定义 1.1.1（集合和集合运算） 如果集合M由n ($n \geqslant 0$)个元素$a_1, a_2, \cdots, a_n$组成,则称集合M是**有限的**,并写成$M = \{a_1, a_2, \cdots, a_n\}$. 如果集合$M$的每个元素都有性质$P$*, 则可以把$M$写成$\{x|x$ 具有性质$P\}$. 不包含任何元素的集合称为**空集**, 记作ϕ. 非有限的集合称为**无限的**.

如果M是一个集合, a是M的一个元素, 则记作$a \in M$; 如果a不是M的元素,则记作$a \notin M$. 如果M和N都是集合, 并且M的每一个元素也都是N的元素,则称M是N的**子集**,记作$M \subseteq N$或$N \supseteq M$. 如果$M \subseteq N$, 并且N至少有一个元素不是M的元素,则称集合M是集合N的**正则**(Proper)子集, 记作$M \subset N$或$N \supset M$. 如果$M \subseteq N$且$N \subseteq M$, 则称集合M和集合N是相等的, 记作$M = N$.

两个集合M和N的**并集**、**交集**和**差集**分别记作$M \cup N$, $M \cap N$和$M - N$, 它们的定义如下:

$$M \cup N = \{x|x \in M \text{ 或 } x \in N \text{ 或二者皆成立}\}$$

$$M \cap N = \{x|x \in M \text{ 且 } x \in N\}$$

$$M - N = \{x|x \in M \text{ 且 } x \notin N\}$$

如果两个集合的交集是空集, 则称它们是**不相交的** (disjoint). 如果M是一个集合, $M_1, M_2, \cdots, M_n$是两两不相交

---

* 此处应补充一句: 且定义域中具有性质P的元素都属于集合M.——译者注

的集合序列，又有 $M = M_1 \cup M_2 \cup \cdots \cup M_n$，则称$M_1$，$M_2$，$\cdots$，$M_n$是$M$的一种**划分**(Partitioning)。

定义 1.1.2（n 元组和序列） 由元素 $a_1, a_2, \cdots, a_n (n \geqslant 0)$组成的 **n 元组**，$A$是长度为 n 的元素序列，记作 $A=(a_1, a_2, \cdots, a_n)$，或 $A = a_1, a_2, \cdots, a_n$。假设 $A = (a_1, a_2, \cdots, a_n)$ 是一个 n 元组，如果 $b = a_i$，则称 b 是A的**第 i 个分量**，其中 i 是 b 在A中的位置。我们用符号形式把它写成 $\pi_A b = i$，或简写作 $\pi b = i$。

我们常常把集合方面的符号用到序列上。具体地说，我们把空序列(零元组)记为 ϕ；当 a 在序列A中时，记作 $a \in A$；否则记作 $a \notin A$。如果 $\{x|x \in A\} \cap \{x|x \in B\} = \phi$，则称序列$A$和$B$是不相交的。如果序列$B$由序列$A$中抽出的 $n(n \geqslant 0)$个元素组成，则称B是A的一个子序列。(因此，3，4，2是6，3，0，12，4，2，7的一个子序列，而不是6，4，0，0，12，3，7，2的子序列。)如果序列A是序列B的子序列，而B又是A的子序列，则称A和B是**相等的**，记作 $A = B$。

定义 1.1.3（笛卡尔积和关系） 定义两个集合M和N的笛卡尔积（记作 $M \times N$）为如下的有序偶的集合：

$$M \times N = \{(x, y)|x \in M \text{ 且 } y \in N\}$$

设M和N都是集合，我们把 $M \times N$ 的每一个子集都称为在 $M \times N$（或在M上，如果 $M = N$ 的话）上的一个**关系**。

设 r 是M上的一个关系。如果 $(a, b) \in r$，则称 a 是 b 的关于 r 的**前件**(predecessor)，b 是 a 的关于 r 的**后件** (successor)。我们用 arb 表示 $(a, b) \in r$，用 $a \not r b$ 表示 $(a, b) \notin r$。

关系 r 的**定义域**和**值域**分别记作 domain(r) 和 range(r)，它们的定义如下：

$$\text{domain}(r) = \{x|(x, y) \in r\}$$

$$\text{range}(r) = \{y|(x, y) \in r\}$$

关系 r 的**逆** (inverse) 记作 r^{-1}，定义为

$$r^{-1}=\{(y,x)|(x,y)\in r\}$$

设 r 是 M 上的一个关系. 如果对于每一个 $a\in M$ 都有 $(a,a)\in r$，则称 r 是**自反的** (reflexive)；如果对于任何 $a\in M$，$(a,a)\in r$ 都不成立，则称关系 r 是**非自反的** (antireflexive). 如果 $(b,a)\in r$ 时必有 $(a,b)\in r$，则称关系 r 是**对称的**(symmetric). 如果当 $(a,b)\in r$ 且 $(b,c)\in r$ 时必有 $(a,c)\in r$，则称关系 r 是**传递的** (transitive).

定义 1.1.4（传递关系的基） 设 r 是 M 上的一个传递关系，且 $b\subseteq r$. 如果下式成立，则称关系 b 是关系 r 的**基** (basis)，又称关系 r 是关系 b 的**传递体** (transitive hull).

$$r=\{(x,y)|\text{在}M\text{中有元素}x_0,x_1,x_2,\cdots,x_n(n\geqslant 1),\text{使得}(a)\,x_0=x,\ (b)\,x_n=y,\ \text{且}\ (c)\,(x_{i-1},x_i)\in b\ (i=1,2,,\cdots,n)\}$$

定义 1.1.5（等价关系和等价类） 如果 M 上的一个关系 r 是自反的、对称的和传递的，则称 r 是**等价关系** (equivalence relation). 如果 r 是一个等价关系，且 $(a,b)\in r$，则称 a 和 b **是等价的**；若集合 M 具有等价关系 r，则 M 是一些两两不相交的集合的并集；这些集合叫作**等价类** (equivalence classes).

定义 1.1.6（半序和拓扑分类） 如果 M 上的一个关系 r 是传递的和非自反的，则称 r 是**半序** (Partial ordering) 关系. 如果 $A=a_1,a_2,\cdots,a_n$ 是 M 的元素的一个序列，且当 $i<j$ 时有 $(a_i,a_j)\in r$，则称 A **是相对于 r 拓扑分类的** (topologically sorted relative to r).

【注】应该注意到，在任何一个具有半序关系 r 的有限集合中，至少有一个元素没有前件，也至少有一个元素没有后件.

定义 1.1.7（全序和分类） 如果 M 上的一个关系 r 是传递的且满足以下两个条件，则称 r 是**全序** (total ordering) 关系.

(1) 如果 $(a, b) \in r$ 且 $(b, a) \in r$，则 $a = b$.

(2) 对于任意两个 a 和 b，要末 $(a, b) \in r$；要末 $(b, a) \in r$；要末二者皆成立.

设 $A = a_1, a_2, \cdots, a_n$ 是由 M 的元素组成的序列，如果 $(a_i, a_{i+1}) \in r (1 \leqslant i \leqslant n-1)$，则称 A 是（相对于 r 的）**已分类的**或**已排序的**.

定义 1.1.8（最小值和最大值） 设 M 是数的集合，如果在 M 中存在一个最小的数，则称之为 M 的最小数，并记作 $\min(M)$. 类似地，用 $\max(M)$ 表示 M 的最大数，并且有 $\max(M) = -\min\{-x \mid x \in M\}$.

定义 1.1.9（函数） 设 f 是 $M \times N$ 上的一个关系. 如果对每一个 $a \in M$ 都刚好有一个 $b \in N$，使 $(a, b) \in f$，则称 f 是由 M 到 N 的一个**函数**，并记作 $f: M \to N$. 我们常用 $f(a) = b$ 代替 $(a, b) \in f$，或者简单地写成 $fa = b$. 如果只要 $a \neq b$ 就有 $f(a) \neq f(b)$，则称函数 $f: M \to N$ 是**一一对应的**.

【练习】

1.1.1 设 r 是一个关系：

$$r = \{(1, 2), (2, 3), (3, 5), (3, 4), (1, 3), (1, 4), (2, 4), (2, 5), (1, 5)\}.$$

(1) 试说明 r 是传递的.

(2) r 是不是等价关系？

(3) 试找出 r 的有四个元素的基 b.

1.1.2 设 r 是具有基 b 的传递关系

$$b = \{(1, 4), (2, 5), (5, 8), (0, 3), (3, 6), (6, 9), (4, 1), (8, 2), (9, 0)\}.$$

(1) r 是等价关系吗？

(2) 试找出 r 的值域.

1.1.3 设 r 是集合 $M = \{1, 2, 3, 4, 5, 6, 7\}$ 上的关系：

$$r = \{(3,1),(3,2),(4,6),(4,7),$$
$$(5,4),(5,6),(5,7),(6,7)\}$$

(1) 试说明 r 是半序的.

(2) 试找出M的元素的一个拓扑分类序列.

1.2 数据结构和表示数据结构的方法

计算机已被广泛地用来处理数据的集合. 我们可以把数据集合看成是数据**元素**或数据**结点**的集合，把它们的值看作是字符串的 n 元组. 计算机可以用各种方式分析和更改数据集合中的数据元素.

表示机器中部件的名称、现有数量和编号的字符串三元组，定义某一函数的自变量及其对应函数值的**数偶** (pair of numbers)，表示立约人的姓名、地址、开户银行和序号的字符串四元组等，都是作为结点的值的典型例子.

人们感兴趣的是想要知道数据集合中结点间相互有什么关系. 例如，当结点表示机器的部件时，最好说明某部件是另一部件的构成部分，或者说明在进行改装时具有某些编号的部件被全部安装在一起. 当结点规定函数值时，可以把每个结点同具有下一个最大的函数值的结点联系在一起. 如果结点表示立约人，则可能需要指出哪些立约人在同一城市工作.

在数据集合中，结点之间的联系可以用关系表示，结点和关系结合在一起叫作**数据结构**.

定义 1.2.1 (数据结构) **数据结构** B 是一个二元组 $B = (K, R)$，其中 K 是结点的有限集合，而 R 是 K 上的关系的有限集合. 结点 $k \in K$ 的值[1]记作 ωk，它是字符串 n 元组 $(n \geqslant 0)$;

1) 在不致发生误解的场合下，将不区分结点和结点的值.

$\omega_i k$ 表示结点 k 的(值的)第 i 个分量，用符号表示为 $\omega k = (\omega_1 k, \omega_2 k, \cdots, \omega_n k)$。

定义 1.2.2(记号约定) 设 $B = (K, R)$ 是一个数据结构，$r \in R$ 是一个关系，k, k' 是结点. 如果 $(k, k') \in r$，则称 k' 是 k 的后件，k 是 k' 的前件，k 同 k' 是相邻的，k 指向 k' (都是相对于 r 来说的). 如果没有结点 k' 使 $(k, k') \in r$，则称 k 是(相对于 r 的) **终结点**；如果没有结点 k' 使 $(k', k) \in r$，则称 k 是(相对于 r 的)**始结点**. 既不是终结点，也不是始结点的结点 k，称为**内结点**. 如果 k 是结点且 $\omega k = \phi$(即 k 是零元组)，则称 k 是**指针**. 指向始结点的指针称为**始指针**，指向终结点的指针称为**终指针**. 对数据结构的**处理**的含义是：求出或改变结点的值，或者向 B 中添加结点或由 B 中删除结点.

定义 1.2.3(数据结构的图形表示) 设 $B = (K, R)$ 是一个数据结构. 产生 B 的图形表示的方法如下：对于每个结点 $k \in K$，画一个图形. 当 $\omega k = \phi$ 时划一个圆，否则画一个中间写着 ωk 的方框. 如果结点 k 指向另一个结点 k'(相对于关系 r)，则用有向线段联结 k 和 k' 所对应的图形. 我们选用不同类型的线段表示不同的关系. 有时最好能给对应于结点的图形命名，并把名字写在图形的外面.

例 1.2.1 数据结构 $B = (K, R)$ 具有 10 个结点，$K = \{k_1, k_2, \cdots, k_{10}\}$ 和两种关系 $R = \{T, N\}$，其中

$$\omega k_1 = (\text{machine}, 40, 612)$$
$$\omega k_2 = (\text{motor}, 2, 802)$$
$$\omega k_3 = (\text{block}, 3, 105)$$
$$\omega k_4 = (\text{filter}, 2, 117)$$
$$\omega k_5 = (\text{tank}, 10, 118)$$
$$\omega k_6 = (\text{brakes}, 60, 230)$$
$$\omega k_7 = (\text{ignition}, 30, 408)$$

$$\omega k_8 = (\text{housing}, 17, 507)$$

$$\omega k_9 = (\text{frame}, 40, 230)$$

$$\omega k_{10} = (\text{plates}, 1, 702)$$

$$T = \{(k_1, k_2), (k_1, k_8), (k_2, k_3), (k_2, k_7), (k_3, k_4), (k_3, k_5), (k_3, k_6), (k_8, k_9), (k_8, k_{10})\}$$

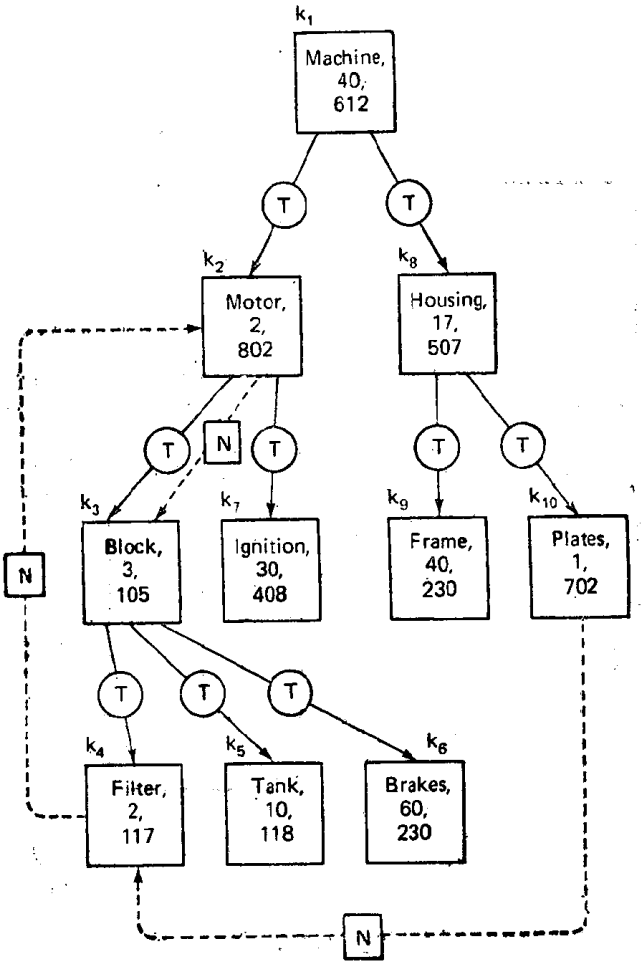


图 1.2.1

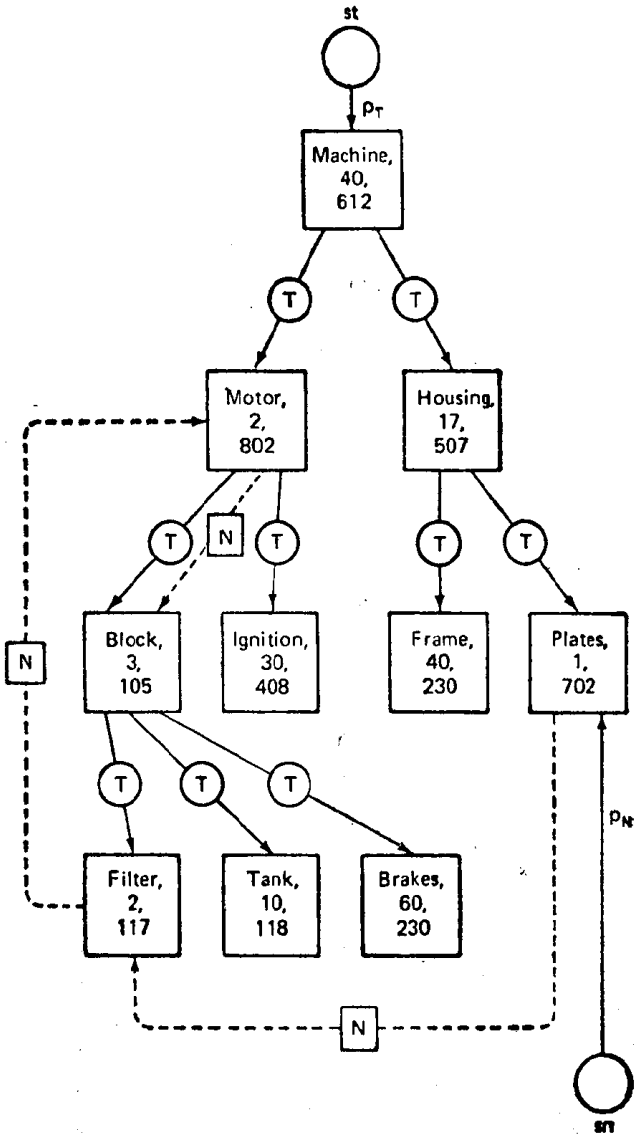

st
p_T
Machine,
40,
612
T
T
Motor,
2,
802
Housing,
17,
507
T
N
T
T
T
N
Block,
3,
105
Ignition,
30,
408
Frame,
40,
230
Plates,
1,
702
T
T
T
p_N
Filter,
2,
117
Tank,
10,
118
Brakes,
60,
230
N
sn

图 1.2.2

$R=\{N, T, Z\}$ 的数据结构 $B=(K, R)$，其中 $\omega k_i=i$.

$$N=\{(k_1, k_3), (k_1, k_9), (k_2, k_3), (k_3, k_1), (k_3, k_2), (k_4, k_5), (k_6, k_8)\}$$

$$T=\{(k_4, k_5), (k_3, k_4), (k_7, k_8), (k_1, k_3), (k_8, k_9), (k_5, k_7)\}$$

$$Z=\{(k_i, k_j)|i+j=7\}.$$

1.2.2 试找出练习 1.2.1 中相对于关系 N, T, Z 的全部始结点和终结点.

1.3 存贮数据结构的方法

在考虑用计算机实际存贮和处理数据结构时，最好从构造存贮数据结构的理论模型入手. 在这种模型中，我们设想每个存贮单元由两部分——**数据场** (data field) 和**指针场** (pointer field) 组成. 数据场用来存放结点 k 的值，指针场用来实现关系 r，即用来指出存放着 k 的后件（相对于 r）的一切存贮单元的存贮器地址. 可以为每个指针场设置一个**指示符**——它的值取 0 或取 1——以便给数据结构提供补充信息.

定义 1.3.1（存贮器） **存贮器**由有限个（q 个）**存贮单元**组成，并从 1 到 q 编号. 存贮单元 x 的位置叫作**地址**，记作 αx. 反过来，如果把地址为 t 的存贮单元记为 σt，则 $\alpha\sigma t=t$，并且 $\sigma\alpha x=x$. 存贮单元 x 由一个数据场（其值记作 δx）和 $m\geqslant 0$ 个指针场组成. 单元 x 的第 i 个指针场有一个值（记作 $\rho_i x$）和一个指示符（记作 $\tau_i x$），指示符的值通常为 0，此时不必把指示符明显地表示出来.

指针场的值[1] $\rho_i x$ 是整数 t 元组 ($t\geqslant 1$). $\rho_i x$ 的第 j 个

1) 在不致发生误解的情况下，指针场同其值之间或指针分量同其值之间不作区别.

分量记作 $\rho_{ij}z$. 数据场的值 δz 是字符串 n 元组 ($n \geqslant 0$), 它的第 j 个分量记作 $\delta_j z$. 除非特别声明,指针场的值 $\rho_i z$ 只有一个分量,且 $\rho_{i1} = \rho_i$. 如果只考虑一个指针场 ρ_1, 则把 ρ_1 简写成 ρ.

定义 1.3.2(存贮数据结构) 存贮数据结构 $B = (K, R)$ 是指: 为每个 $k \in K$ 唯一分配一个存贮单元 z, 使 $\delta z = \omega k$. 为结点 k 分配的存贮单元 z 记作 ξk, 且 $\xi k = z$.

定义 1.3.3(符号约定) 如果两个存贮单元 z 和 z' 有 $\alpha z' = 1 + \alpha z$, 则 z 和 z' 是**相邻的**. 如果 $\alpha z < \alpha z'$, 则存贮单元 z 在 z' 的左边且 z' 在 z 的右边. 相邻的存贮单元的集合 M 称为**存贮区域**. 存贮区域 M 的第一个单元 z 称为存贮区域的**左端**, $\alpha z = \min\{\alpha z'' \mid z'' \in M\}$. 存贮区域的最后一个单元 z' 是存贮区域的**右端**, $\alpha z' = \max\{z'' \mid z'' \in M\}$. 如果存贮单元 z' 的地址 $\alpha z'$ 出现在存贮单元 z 的第 i 个指针场或数据场中, 则称 z 指向 z' (相对于第 i 个指针场或数据场). 如果 $B = (K, R)$ 是已存贮的数据结构, r 是一种关系, 则称结点 K 存贮在存贮单元 $\xi K = \{\xi k \mid k \in K\}$ 中. 在不致发生误解的情况下,"结点"和"存贮单元"两个术语可以互换, 例如, 结点 k 的地址(记作 αk) 是指对应于结点 k 的存贮单元的地址; 结点 k 的第 i 个指针场的值写作 $\rho_i k$ 或 $\rho_i \xi k$. 相对于关系 r 的始单元、终单元和内单元分别对应于始结点、终结点和内结点.

定义 1.3.2 只牵涉到如何存贮结点的值, 我们还必须考虑如何实现"关系", 实现关系的方法有两种: 一种是**顺序**实现另一种是**链接**实现.

定义 1.3.4(关系的顺序实现) 设 $B = (K, R)$ 是已存贮的数据结构. 如果对于每一对结点 $(k, k') \in r$, 都有 $\alpha k' = \alpha k + s$, 则称关系 r 是顺序实现的, 步长为 s. 对于步长为 s 的顺序实现来说, 相邻的结点存贮在地址之差为 s 的单元中. 步

长通常为 1，此时不必明显地把 s 表示出来.

定义 1.3.5（关系的链接实现） 设 $B=(K, R)$ 是已存贮的数据结构. 如果对于每一对结点 $(k, k')\in r$，都有 $\alpha k'\in \rho_i k$，则称关系 r 是用链接结构实现的（相对于第 i 个指针场）. 如果对于每一对结点 $(k, k')\in r$，都有 $\alpha k'=\rho_i k$，则称链接是**线性的**. 在以链接实现的关系中，结点 k 的第 i 个指针场给出 k 的一切后件的地址.

如果关系 r 是顺序实现的，则称对应的结点是顺序存贮的. 如果关系 r 是以链接结构实现的，则称对应的结点在存贮器中是链接在一起的.

定义 1.3.6（存贮单元的图形表示） 设 Z 是存贮单元的集合. Z 的图形表示可以由下述方式得到：对应于每个具有 m 个指针场的存贮单元 $z\in Z$，都画出一个方框，并把方框从左到右划分成 $m+2$ 个方框段. 第一个方框段中写上 αz，第二个方框段中写上 δz，在其余的 m 个方框段写上指针场的值 $\rho_1 z$，$\rho_2 z$，$\cdots$，$\rho_m z$. 若指示符 $\tau_i z$ 不等于 0，则在相应的指针场上打一个星号. 如果在存贮单元 z 的方框段 G 中出现有另一个存贮单元的地址 $\alpha z'$，则用有向线段把 G 同存贮单元 z' 的第一个方框段连结起来.

为了简单起见，有时略去对应于 αz，δz 或 $\rho_1 z$，$\rho_2 z$，$\cdots$，$\rho_m z$ 的某些方框段. 在某些情况下，可能有必要为对应于存贮单元的方框或方框段命名，这些名字应写在方框或方框段的外面.

例 1.3.1 图 1.3.1 说明存贮数据结构 $B=(K, R)$ 的一种方法. 数据结构 B 具有 5 个结点 $K=\{k_1, k_2, k_3, k_4, k_5\}$ 和一种关系 $r=\{(k_1, k_2), (k_2, k_3), (k_3, k_4), (k_4, k_5)\}$，$\omega k_1=$THAT，$\omega k_2=$THE，$\omega k_3=$THIS，$\omega k_4=$US，$\omega k_5=$WE. 由于 $\rho k_1=\alpha k_2$，$\rho k_2=\alpha k_3$，$\rho k_3=\alpha k_4$，$\rho k_4=\alpha k_5$，关系 r 是链接实现的.