程序员级高级程序员级
软件知识

施伯乐　主编
潘锦平　主审

中国计算机软件专业技术资格和水平考试
统编辅导教材

清华大学出版社

程序员级高级程序员级
软件知识

施伯乐　主编
潘锦平　主审

清华大学出版社

内 容 提 要

本书是中国计算机软件专业技术资格和水平考试委员会组织编写的计算机软件专业技术资格和水平考试统编辅导教材之一。它以1991年考试大纲为依据，对程序员级和高级程序员级考生必备的软件知识作了详尽的讲解。主要内容包括：数据结构、操作系统、数据库系统、程序语言和语言处理程序、计算机网络、软件工程、计算机安全知识等计算机科学的主要领域中的基本概念、典型方法和技术，对考生应考有重要参考价值。

本书可作为参加中国计算机软件专业技术资格和水平考试的考生应考的主要参考书，也可供大专院校师生和广大工程技术人员学习参考，并可供各部门举办辅导班作为教材。

(京)新登字158号

程序员级高级程序员级软件知识

施伯乐 主编

潘锦平 主审

☆

清华大学出版社出版

北京 清华园

北京顺义曙光印刷厂印刷

新华书店总店科技发行所发行

☆

开本：787×1092 1/16 印张：19.5 字数：492 千字

1992年5月第1版 1992年12月第2次印刷

印数：20001—30000

ISBN 7-302-01055-2/TP·390

定价：12.00元

出版说明

当今国际间的竞争是综合国力的竞争，关键又是科学技术的竞争，说到底是人才的竞争。而电子信息技术又是当今国际间竞争的热点，它渗透和影响着现代化社会生活的各个方面。而科学技术（包括电子信息技术）是要靠人去掌握、去应用、去发展的，国家的强盛、民族的振兴靠人才，人才的培养靠教育。所以，党中央要求我们把经济建设转到依靠科技进步和提高劳动者素质的轨道上来。

培养人才要坚持多种形式、多种途径，大力开展岗位培训，不断提高职工队伍的技术和专业水平。计算机软件专业技术资格和水平考试制度，就是为了加速我国电子信息技术的广泛应用和软件事业的发展，科学考核和合理使用人才，促进计算机软件人才的国际交流与合作，进一步深化职称改革。这种考试是于1985年首先在上海、云南、四川三省市实行的，1987年发展为部分省、区、市的联合考试，到1988年全国已经有31个省、区、直辖市和计划单列市参加程序员、高级程序员两个级别的联合考试，1989年发展为程序员级、高级程序员级、系统分析员级三个级别的联合考试，1990年在全国统一组织实施了软件专业技术职务任职资格（水平）考试。1991年3月计算机软件专业技术资格和水平考试工作会议，对考试《暂行规定》作了修改，新的《暂行规定》把“资格”考试与“水平”考试从性质上和考试级别上作了区分，“资格”考试除了保留“程序员”、“高级程序员”的级别外，增加了“初级程序员”的考试级别，而“水平”考试则仍为“程序员”、“高级程序员”、“系统分析员”三个级别。同时在报考条件与考试的内容和评分标准等方面，也都作了相应的规定。1991年的软件专业技术资格和水平考试就是依照新的《暂行规定》进行的。实践证明这是一种严格的认定考试制度，给应试者提供了一次均等的机会。软件专业技术资格和水平考试，每年举行一次，实行全国统一组织、统一大纲、统一试题、统一评分标准。

中国计算机软件专业技术资格和水平考试委员会（以下简称考委会）把考试和培训两项相辅相成的工作担当起来是责无旁贷的。为了使全国参加统一考试的考生得到较好的辅导，编一套全国统一的辅导教材是完全必要的。1991年9月考委会在北京主持召开了辅导教材编审委员会会议，对统编辅导教材的编写、出版作了部署，成立了统编辅导教材编审委员会，并确定了这套教材六本书的主编和主审，他们是：《初级程序员级软硬件知识》（主编：刘尊全；主审：吴克忠）、《程序员级、高级程序员级硬件知识》（主编：王爱英；主审：周明德）、《程序员级、高级程序员级软件知识》（主编：施伯乐；主审：潘锦平）、《计算机综合应用知识》（主编：罗晓沛；主审：侯炳辉）、《程序员级、高级程序员级程序设计》（主编：张福炎；主审：郑国梁）、《1990年度—1991年度试题分析与解答（初级程序员级、程序员级、高级程序员级、系统分析员级）》（主编：张然；主审：施伯乐）。我们的本意是要把这套全国统编辅导教材搞成为具有正确性、科学性、系统性，而且针对性强，在一段时间内相对稳定的好教材。虽然吸取了北京、上海、成都等地已有的辅导教材的长处，但由于时间紧，经验不足，错误在所难免，请读者和有关专家能给予指正。

教材编审委员会

（按姓氏笔画为序）

秘书组联络地址：北京市海淀区学院南路55号（邮政编码：100081）

前　言

“程序员级高级程序员级软件知识”一书涉及面很广，由于篇幅关系只能选择其主要内容，分七个方面作介绍，全书分七章，但从每一章来看其内容还是较多、较丰富，在一般大学往往是一学期的课程内容，因此编写的难度较高，本教材为此特邀请了各方面的专家共同编写，这些专家都是在教学第一线的有名望的学者，使其内容尽量做到既保持科学性又能深入浅出，由于时间较紧一定有很多不足之处，望能多加指正，下面就各章内容作简要说明。

第一章数据结构是本书以后几章的基础，由北京大学杨冬青撰写，本章对各种数据结构，线性表，树、图均给出了定义和简单的应用，并对有关的重要算法作了介绍，其进一步应用将在以后几章展开，对于算法的具体实现，望读者结合有关的程序语言加以实现。

第二章操作系统是计算机的最基本系统，由北京大学方裕撰写。本章对操作系统的核心及主要功能、作业管理、进程管理、存储管理、设备管理、文件管理作了详细的描述，读者通过各种管理的学习，能全面理解操作系统的内容。

第三章数据库系统在数据处理中有广泛的应用，由复旦大学丁宝康、施伯乐撰写，主要介绍数据模型、数据库体系结构、关系、网状数据库及语言，在此基础上介绍了数据库设计，其重点在关系数据库和SQL语言，读者通过本章学习，在建立数据库时应尽量应用数据库设计方法。

第四章程序语言和语言处理程序是理论性较强的一章，由华东师范大学徐国定撰写，本章主要介绍程序语言的基础知识和其汇编编译解释程序的基本原理，其重点在于编译程序原理，其内容有一定难度，望读者能多花些时间，通过习题能深入了解，对整个编译有一个全面的了解。

第五章计算机网络是有广泛应用的一章，由刘福滋和张公忠撰写，本章通过网络体系结构、局域网、局域网的协议标准等基本知识和基本概念的介绍，使读者具有从网络系统的总体高度来考虑问题的能力。

第六章软件工程是软件产业的重要支柱，由清华大学郑人杰撰写，本章以软件生命期为导引，扼要介绍软件开发计划、需求分析、设计、测试维护等方面的问题，读者通过学习能对软件开发的全过程有所了解，通过实践进一步掌握有关的方法。

第七章计算机安全知识概述是软件知识的重要内容之一，由方裕撰写，本章主要介绍计算机病毒、病毒的诊断治疗及预防，通过学习读者能对当前流行的计算机病毒有一个全面的了解。

目　录

第一章 数 据 结 构

“数据结构”是计算机各有关专业的一门重要基础课程。计算机科学领域中，尤其是系统软件和应用软件的设计和实现中都要用到各种数据结构。学习“数据结构”既为进一步学习其他软件课程提供必要的准备知识，又有助于提高软件设计和程序编制水平。本章将介绍数据结构的基本概念和一些常用的数据结构，阐明数据结构内在的逻辑关系，讨论它们在计算机中的存储表示，以及在数据结构上进行各种运算的执行程序，并作简单的算法分析。

1.1 概 述

1.1.1 数据

数据就是对现实世界的事物采用计算机能够识别、存储和处理的方式（例如数字、字符等）进行的描述。简言之，数据就是计算机化的信息。

数据元素是数据的基本单位，即数据集合中的个体。有些情况下也把数据元素称作结点、记录、表目等。一个数据元素可由一个或多个数据项组成，数据项是有独立含义的数据最小单位。有时也把数据项称作域、字段等。例如，可以将一个学生的有关信息作为一个数据元素，它由姓名、专业、学号等数据项组成。

1.1.2 数据结构

被计算机加工的数据元素不是互相孤立的，它们彼此间一般存在着某些逻辑上的联系，这些联系需要在对数据进行存储和加工时反映出来。因此，数据结构概念一般包括三个方面的内容：数据之间的逻辑关系、数据在计算机中的存储方式、以及在这些数据上定义的运算的集合。

1．数据的逻辑结构

数据的逻辑结构只抽象地反映数据元素间的逻辑关系，而不管其在计算机中的存储表示方式。

数据的逻辑结构分为线性结构和非线性结构。若各数据元素之间的逻辑关系可以用一个线性序列简单地表示出来，则称之为线性结构，否则称为非线性结构。线性表是典型的线性结构，而树、图等都是非线性结构。

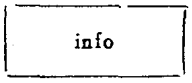
2．数据的存储结构

数据的存储结构是逻辑结构在计算机存储器里的实现。

为全面地表示一个逻辑结构，它在存储器中的映象应包括数据元素自身值的表示和数据元素之间的关系的表示两个方面。因此，存储在计算机中的数据结构，其结点的各域按性质可分成两大类，一类是存放自身值的域，例如姓名、专业、学号等，通常称之为自身信息域，可用标识符 info 表示这些域的全体；另一类是存放该结点与其它结点的关系的域，例如一个或多个指针，或其它形式的连接信息，通常称之为连接信息域，可用标识符 link 表示这

些域的全体。一般情况下，存储结构中结点的形式为：

这里说的是一般情况，而在某些存储结构中，结点可以不包括连接信息域，这时的结点形式为：



3．数据的运算

数据的运算是定义在数据的逻辑结构上的，但运算的具体实现要在存储结构上进行。数据的各种逻辑结构有相应的各种运算，每种逻辑结构都有一个运算的集合。常用的运算有检索、插入、删除、更新、排序等。

数据的运算是数据结构的一个重要方面，讨论任何一种数据结构时都离不开对该结构上的数据运算及实现算法的讨论。

1.1.3 主要的数据存储方式

有多种不同的方式来实现数据的逻辑结构到计算机存储器的映象。下面介绍两类最主要的存储方式，大多数数据结构的存储表示都采用其中一类方式，或两类方式的结合。

1．顺序存储结构

这种存储方式主要用于线性的数据结构，它把逻辑上相邻的数据元素存储在物理上相邻的存储单元里，结点之间的关系由存储单元的邻接关系来实现。

顺序存储结构的主要特点是：①结点中只有自身信息域，没有连接信息域，因此存储密度大，存储空间利用率高；②可以通过计算直接确定数据结构中第i个结点的存储地址 L_i，计算公式为：$L_i = L_0 + (i-1) * m$，其中 L_0 为第一个结点的存储地址，m 为每个结点所占用的存储单元个数；③插入、删除运算不便，会引起大量结点的移动，这一点在下一节还会具体讲到。

2．链式存储结构

链式存储结构就是在每个结点中至少包括一个指针域，用指针来体现数据元素之间逻辑上的联系。这种存储结构可把人们从计算机存储单元的相继性限制中解放出来，可以把逻辑上相邻的两个元素存放在物理上不相邻的存储单元中；还可以在线性编址的计算机存储器中表示结点之间的非线性联系。

链式存储结构的主要特点是：①结点中除自身信息外，还有表示连接信息的指针域，因此比顺序存储结构的存储密度小，存储空间利用率低；②逻辑上相邻的结点物理上不必邻接，可用于线性表、树、图等多种逻辑结构的存储表示；③插入、删除操作灵活方便，不必移动结点，只要改变结点中的指针值即可，这一点在下一节还会具体讲到。

除上述两种主要存储方式外，散列法也是在线性表和集合的存储表示中常用的一种重要存储方式，我们将在1.7.4节中介绍。

1.2 线 性 表

线性表是最简单、最常用的一种数据结构。线性表的逻辑结构是 n 个数据元素的有限序

列（a_1，a_2，…，a_n）。用顺序存储结构存储的线性表称作顺序表。用链式存储结构存储的线性表称作链表。对线性表的插入、删除运算可以发生的位置加以限制，则是两种特殊的线性表——栈和队例。若线性表中的元素都是单个字符，则称作串。

1.2.1 顺序表和一维数组

各种高级语言里的一维数组就是用顺序方式存储的线性表，因此也常用一维数组来称呼顺序表。下面主要讨论顺序表的插入和删除运算。

往顺序表中插入一个新结点时，可能需要往后移动一系列结点。若顺序表中结点个数为n，且往每个位置插入的概率相等，则插入一个结点平均需要移动的结点个数为n/2。

类似地，从顺序表中删除一个结点可能需要往前移动一系列结点。在等概率的情况下，删除一个结点平均需要移动的结点个数也是n/2。

1.2.2 链表

1. 线性链表（单链表）

线性链表就是链式存储的线性表，它的每个结点中含有一个指针域，用来指出其后继结点的位置。线性链表的最后一个结点没有后继结点，它的指针域为空（记为NIL或∧）。另外还需要设置一个指针head，指向线性链表的第一个结点。

链表的一个重要特点是插入、删除运算灵活方便，不需移动结点，只要改变结点中指针域的值即可。图1.1显示了在单链表中指针P所指结点后插入一个新结点的指针变化情况，虚线所示为变化后的指针。

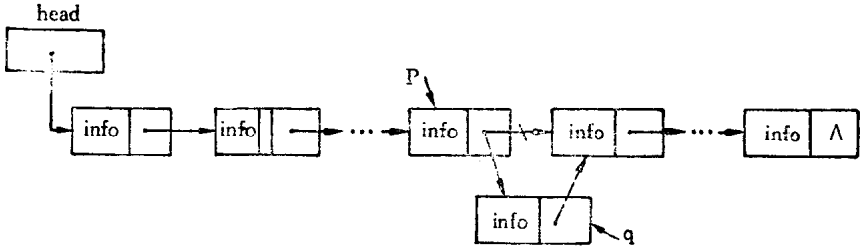


图 1.1 单链表的插入

插入运算的关键步骤为：

```
q↑.link:=P↑.link;
P↑.link:=q;
```

图1.2显示了从单链表中删除指针P所指结点的下一个结点的指针变化情况，虚线所示为变化后的指针。

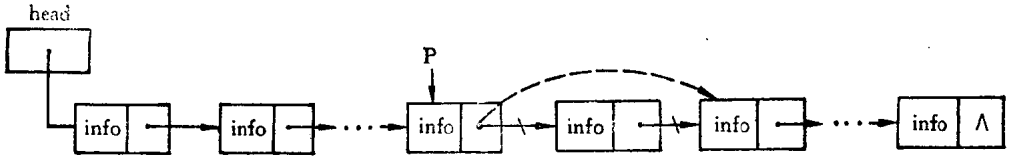


图 1.2 单链表的删除

删除运算的关键步骤为：

```
q:=P↑.link;
P↑.link:=q↑.link;
```

注意，做删除运算时改变的是被删结点的前一个结点中指针域的值。因此，若要求查找且删除某一结点，则应在查找被删结点的同时，记下它的前一个结点的位置。

在线性链表中，往第一个结点前面插入新结点和删除第一个结点会引起表头指针 head 值的变化。通常可以在线性链表的第一个结点之前附设一个结点，称为头结点。头结点的数据域可以不存储任何信息。头结点的指针域存储指向第一个结点的指针，如图 1.3 所示。这样，往第一个结点前面插入新结点和删除第一个结点就不影响表头指针 head 的值，而只改变头结点的指针域的值，因此就可以和其它位置的插入、删除同样处理了。

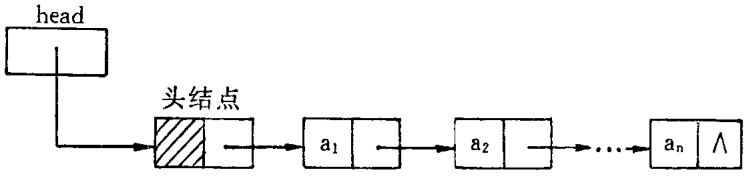


图 1.3 带头结点的线性链表

2. 双链表

在单链表中，从任何一个结点能通过 link 域找到它的后继，但不能找出它的前驱。如果在链表的每个结点中包括两个指针域，其中 rlink 指向结点的后继，llink 指向结点的前驱，就可以方便地进行向后和向前两个方向的查找了。这样的链表称作双链表，如图 1.4 所示。

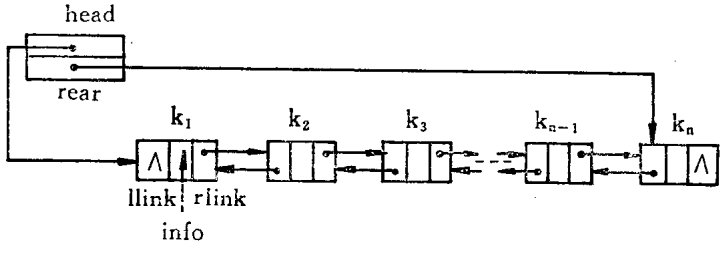


图 1.4 双链表

在双链表中，如果要删除指针变量所指的结点，只需修改该结点前驱的 rlink 字段和后继的 llink 字段，即

```
P↑.llink↑.rlink←P↑.rlink;
P↑.rlink↑.llink←P↑.llink;
```

如果要在P所指结点后插入q所指的新结点，只需修改P所指结点的 rlink 字段和原后继的llink字段，并置q所指结点的 llink 和 rlink 值，即

```
q↑.llink←P;
q↑.rlink←P↑.rlink;
P↑.rlink↑.llink←q;
P↑.rlink←q;
```

插入和删除运算前后指针的变化情况如图1.5和图1.6所示。

3. 可利用空间表

要在链表上执行插入和删除操作，就有一个新的结点从何处来又回到何处去的问题。为

此，设立另一种线性链表，称作可利用空间表，它与要操作的线性链表具有同样的结构，所不同的是，它的结点信息域是空的。用一个指针FREE指向可利用空间表，如图 1.7 所示。

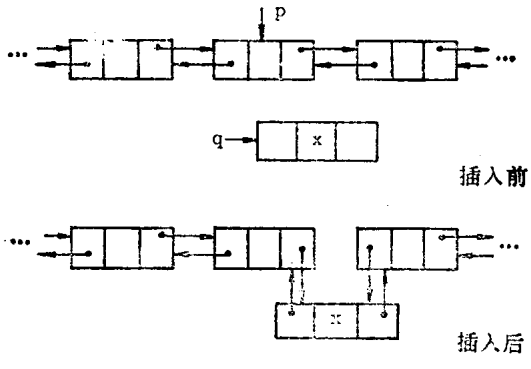


图 1.5 双链表的插入

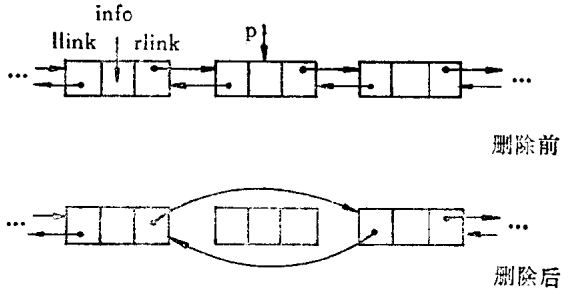


图 1.6 双链表的删除

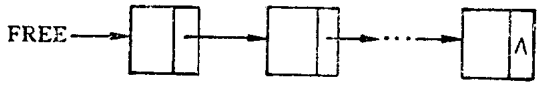


图 1.7 可利用空间表

可利用空间表的作用是管理可用于链表插入的结点。当链表插入需要一个新结点时，就从可利用空间表中删除第一个结点，用这个结点去做链表插入；当从链表中删除一个结点时，就把这个结点插入到可利用空间表的第一个结点前面。

1.2.3 栈

栈是一种特殊的、限定仅在表的一端进行插入和删除运算的线性表，这一端称为栈顶(top)，另一端则叫做栈底（bottom）。表中无元素时称为空栈。栈中有元素 a_1，a_2，…，a_n，如图 1.8 所示，称a_1是栈底元素，a_n是栈顶元素。新元素进栈要置于a_n之上，删除或退栈必须先对a_n进行，这就形成了“后进先出”（LIFO）的操作原则。

栈的物理储存可以用顺序存储结构，也可用链式存储结构。

栈的运算除去插入（称作推入）和删除（称作托出）外，还有取栈顶元素，检查栈是否为空，清除（置空栈）等。栈是使用最为广泛的数据结构之一，表达式求值、递归过程实现都是栈应用的典型例子。

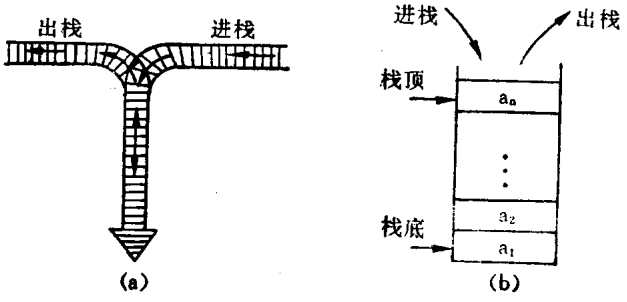


图 1.8 栈结构

1.2.4 队列

队列是一种特殊的、限定所有的插入都在表的一端进行、所有的删除都在表的另一端进行的线性表。进行删除的一端叫队列的头，进行插入的一端叫队列的尾，如图 1.9 所示。在

队列中，新元素总是加入到队尾，每次删除的总是队列头上的，即当前“最老的”元素，这就形成了先进先出（FIFO）的操作原则。

队列的物理存储可以用顺序存储结构，也可以用链式存储结构。

队列的运算除了插入和删除外，还有取队头元素，检查队列是否为空，清除(置空队列)等。

在顺序方式存储的队列中实现插入、删除运算时，若采取每插入一个元素，队尾指示变量 R 的值加 1，每删除一个元素，队头指示变量 F 的值加 1 的方法，则经过若干插入，删除运算后，尽管当前队列中的元素个数少于存储空间的容量，但却可能无法再进行插入了，因为 R 已指向存储空间的末端。通常解决这个问题的方法是：把队列的存储空间从逻辑上看成一个环，当 R 指向存储空间的末端后，就把它重新置成指向存储空间的始端，如图1.10所示。

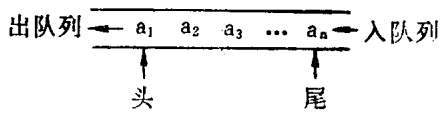


图 1.9 队列结构

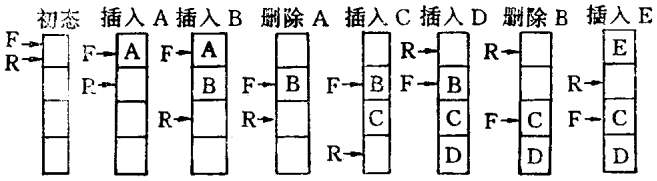


图 1.10 环状队列的插入和删除

队列在计算机中应用也十分广泛，硬设备中的各种排队器、缓冲区的循环使用技术、操作系统中的作业队列等都是队列应用的例子。

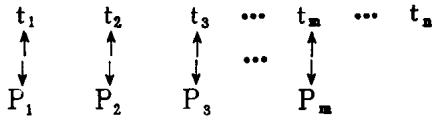
1.2.5 串

串（或字符串）是由零个或多个字符组成的有限序列，一般记为 $S = 'a_1\ a_2 \cdots a_n'$，其中 S 是串的名字，用单引号括起来的字符序列是串的值。零个字符的串是空串。串中字符的数目就是串的长度。a_i 是串中的字符，可以是字母、数字或其它字符。空串与空格构成的串如：‘ ’是不同的。

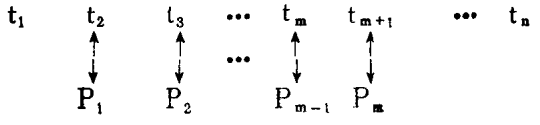
串的存储同样有顺序和链式两种。顺序存储时，既可以采用非紧缩方式，又可以采用紧缩方式。

串的基本运算有连接、赋值、求长度、全等比较、求子串、找子串位置以及替换等。有些程序设计语言以标准子程序（函数或过程）方式提供了这些运算。

在串的基本运算中，找子串位置（也称作模式匹配）是非常重要的，因为在与正文处理有关的许多应用中都需要做模式匹配，即在一串正文 T 中找一个子串 P 的出现位置。模式匹配的最简单的做法是：用 P 中字符依次与 T 中字符比较：



如果 $t_1 = P_1$，$t_2 = P_2$，…，$t_m = P_m$，则匹配成功，找到了子串位置；否则将 P 右移一个字符，用 P 中字符从头开始与 T 中字符依次比较：



如此执行下去，直到某一步匹配成功，或者一直将P 移到无法与T 继续比较为止，则匹配失败。上述的匹配算法非常简单，但效率很低，在最坏的情况下，总的字符比较次数为 m*(n-m+1)。因而在实际应用中，有多种改进的模式匹配算法。

1.3 多维数组、稀疏矩阵和广义表

1.3.1 多维数组的顺序存储

多维数组是一维数组的推广，多维数组中最常用的是二维数组。

$$A_{mn}=\begin{pmatrix} a_{11} & a_{12}\cdots a_{1n} \\ a_{21} & a_{22}\cdots a_{2n} \\ & \cdots \\ a_{m1} & a_{m2}\cdots a_{mn} \end{pmatrix}$$

多维数组的所有元素并未排在一个线性序列里，要顺序存储多维数组就需要按一定次序把所有的数组元素排在一个线性序列里，常用的排列次序有行优先顺序和列优先顺序两种。二维数组在行优先顺序下元素排列次序为：a_{11}，a_{12}，…，a_{1n}，a_{21}，a_{22}，…，a_{2n}…，a_{m1}，a_{m2}，…，a_{mn}；在列优先顺序下，元素排列次序为：a_{11}，a_{21}，…，a_{m1}，a_{12}，a_{22}，…，a_{1n}，a_{2n}，a_{mn}。

给出任一数组元素的下标，可以直接计算数组元素的存放地址。二维数组元素的地址公式为：

(1) 行优先顺序下：

$$LOC(a_{ij})=LOC(a_{11})+((i-1)*n+(j-1))*l$$

(2) 列优先顺序下：

$$LOC(a_{ij})=LOC(a_{11})+((j-1)*m+(i-1))*l$$

式中，n和m分别为数组每行和每列的元素个数，l为每个数组元素占用的存储单元个数。

1.3.2 稀疏矩阵的存储

具有大量0元素的矩阵称作稀疏矩阵，对于稀疏矩阵可以进行压缩存储，只存储非0元素。

若非 0 元素的分布有规律，则可以用顺序方法存储非 0 元素，仍可以用公式计算数组元素的地址。例如，三角矩阵

$$A_{mn}=\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

当$i<j$时，$a_{ij}=0$。如果按行优先顺序列出其中的非零元素，便得到如下序列：

$$a_{11}\ a_{21}\ a_{22}\ a_{31}\ a_{32}\cdots a_{n1}\ a_{n2}\cdots a_{nn}$$

把它顺序存储在内存中，第一行到第 i-1 行共有非零元素的个数为

$$\sum_{k=1}^{i-1} k=\frac{i*(i-1)}{2}$$

因此非零元素 a_{ij} 的地址可用下式计算：

$$LOC(a_{ij}) = LOC(a_{11}) + \frac{i*(i-1)}{2} + (j-1),\ 1 \leqslant j \leqslant i \leqslant n$$

对于一般的稀疏矩阵，现介绍两种存储方法。

1. 三元组法

这个方法用一个线性表来表示稀疏矩阵，线性表的每个结点对应稀疏矩阵的一个非零元素，其中包括 3 个域，分别为该元素的行下标、列下标和值。结点间的次序按矩阵的行优先顺序排列（跳过零元素）。这个线性表用顺序的方法存储在连续的存储区里。

2. 十字链表法

用顺序的方法存储稀疏矩阵可以大大节省存储单元，但缺点是在非零元素增加或减少时，做插入或删除不便。链接存储则可以克服这一不足，十字链表法就是一种链接存储方法。

每个非零元素用一个结点表示，每个结点包含 5 个域，分别表示该元素的行下标、列下标、值，以及指向本行下一个非零元素的指针和指向本列下一个非零元素的指针。另外还有行指针向量和列指针向量，行指针向量有 m 个元素，分别指向各行的第一个非零元素结点。列指针向量有 n 个元素，分别指向各列的第一个非零元素结点。在这种表示下，各行的非零元素和各列的非零元素都分别链接在一起，共有 m + n 条链。对元素的查找可顺着所在行的链进行，也可以顺着所在列的链进行。例如稀疏矩阵

$$A = \begin{pmatrix} 3 & 0 & 0 & 0 & 7 \\ 0 & 0 & -1 & 0 & 0 \\ -1 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{pmatrix}$$

用十字链表法表示如图1.11所示。

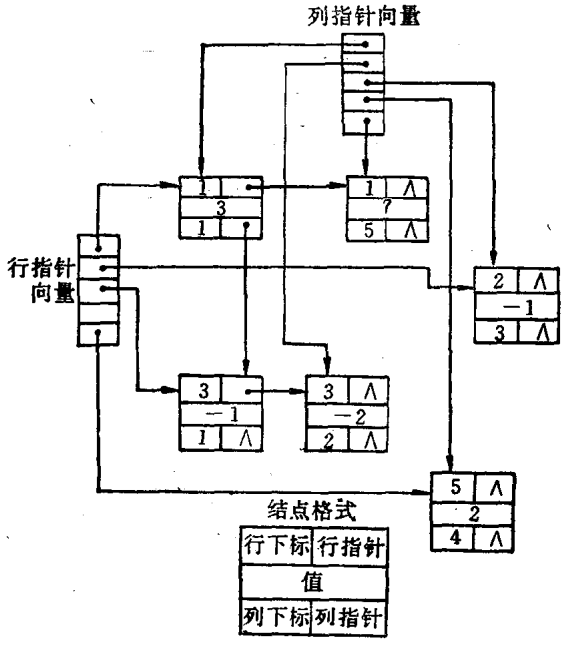


图 1.11 稀疏矩阵的十字链表法表示

1.3.3 广义表的定义和存储

广义表(又称列表)是线性表的推广，是由零个或多个单元素或子表所组成的有限序列。

广义表和线性表的区别在于：线性表的成分都是结构上不可分的单元素，而广义表的成分既可以是单元素，又可以是有结构的表。广义表一般记作

$$LS = (d_1, d_2, \cdots, d_n)$$

其中 LS 是广义表的名字，n是其长度，d_n是广义表的成分，习惯上当一个成分为单元素时，用小写字母表示，当一个成分为子表时，用大写字母表示。当广义表LS非空时，称第一个元素 d_1 为 LS 的表头，称其余元素组成的表 $(d_2, d_3, \cdots, d_n)$ 为 LS 的表尾。

例如：

A=( )

B=(e)

C=(a, (b, c, d))

D=(A, B, C)

分别是长度为0，1，2和3的广义表。

E=(a, E)

是长度为2的广义表，但它是一递归的广义表，相当于一无限的广义表

(a, (a, (a, …)))

由此可以看出，广义表有如下特征：①广义表的元素可以是子表，而子表的元素仍还可以是子表……；②广义表可被其它广义表所共享（引用），如上例中，D可通过子表A、B、C的名称引用它们的值，而不必列出每个子表的值；③广义表可以是递归的表，即广义表也可以是本身的一个子表，如上面的E。

和线性表类似，可对广义表进行的运算有查找、插入和删除等。但广义表的两个非常重要的基本运算是取广义表表头HEAD(LS)和取广义表表尾TAIL(LS)。

任何一个非空广义表，其表头可能是单元素，也可能是广义表，而其表尾必定为广义表。举例如下：

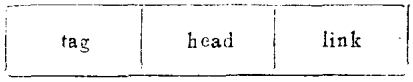
HEAD(B)=e, HEAD(D)=A

TAIL(B)=( ), TAIL(D)=(B, C)

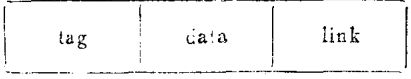
HEAD((B, C))=B, TAIL((B, C))=(C)

注意广义表( )和(( ))不同。前者为空表，长度为0，后者长度为1，可分解得到其表头和表尾均为空表( )。

广义表通常采用链接方式存储，广义表中每个元素用一个结点表示。子表结点形式为



其中，tag=1，表示为子表结点，head是指向子表第一个元素的指针，link是指向该子表同一层中下一个元素的指针。单元素结点形式为



其中tag=0，表示为单元素结点，data是数据元素值，link是指向该元素同一层中下一个元素的指针。

对于前面给出的广义表的例子，可给出它们的存储结构如图1.12所示。

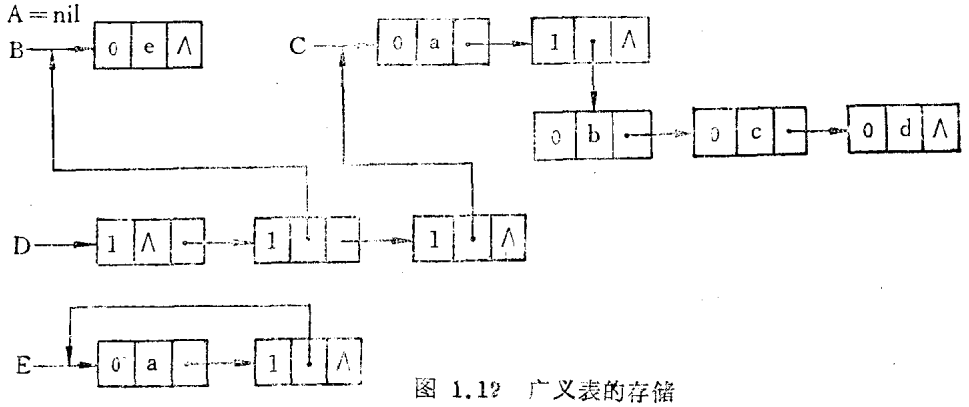


图 1.12 广义表的存储