应用数据结构

北京工业大学

计算机科学系

1981·7

应用数据结构

北京工业大学

计算机科学系

1981·7

目　　录

序言

数据结构研究因其是一种必备的基础知识而作为一门重要的计算机科学课程出现在大学的学生课表中。无论是在应用程序的设计和实施阶段还是在用来构造程序设计系统技术基础的基本构造阶段，数据结构都是程序设计的组成部分。例如，编译程序、操作系统以及文本编辑程序都使用着本书所介绍的数据结构的技术。

人们必定会利用数据结构来发展有关系统实现和应用程序设计的各种技术的，因此它是一种面向应用的课题。我们将用实际的例子来论述各种数据结构的技术问题、权衡问题以及适用性问题。

为了便于管理，就需要对数据进行构造。构造的类型有赖于所期望的管理。反之，管理算法的效果也有赖于数据结构是否选得恰当。为了有效地构造数据，最根本的不仅是要了解各种技术，而且也要了解在什么时候使用哪一种技术。我们将介绍基本的数据管理问题，搜索、分类以及更新；也将介绍与具体数据结构（例如栈处理和稀疏表）相关联的特殊算法。

在最初几章里，我们是通过例子从直观上进行阐述的。第五章尽可能有条理地介绍关于数据结构的形式基础。本书都是在读者自然的直观认识上来展开每一个题目的。我们只是在便介绍更为明晰或要给出严格的、度量算法性能的工具时才用到准确的数学形式。第十章涉及到数据结构的应用，可以用它来增强对全书概念的理解。我们希望这种介绍方式将会逐步地引出对数据结构的形式研究来。

第一章介绍了几个例子，这会很快地激发起读者研究数据结构的兴趣来。你将会注意到我们采用了一种独特的工具来做为程序设计语言，即只要可能就用不很正规的英语短句来描述算法；另外我们对与结构程序设计相关联的算法描述总是坚持使用逐步完善的办法，因此程序设计语言（PL/I和SNOBOL）只是做为一种使内容更丰富的目的而给出来的。

第二章通过介绍非构造数据——即串——而开始研究数据结构。我们选用了PL／I和SNOBOL来说明串是如何出现在高级语言里的，这些问题在附录A和B里得到了详述。通过叙述联接、分隔、子串、

定位以及置换等串的运算而引出数据的组织问题，这些问题将随第三章讲述的技术而依次得到解决。

第三和第四章包括所有的线性数据结构：表、栈、队和双向队。表的紧凑性被用来做为衡量其紧密度〔也就是其存贮效率〕的标准，插入和删除这样的运算花费是很大的，除非限制在第四章所陈述的情况下。

第五章通过图论而开始讲述结构的形式理论。图的一般种类将包括线性的和非线性的表。特别地，通过限制图的位置而形成树结构模型。

第六章用对贮存在辅助存贮器上的数据结构的介绍而结束对结构的研究，这种外部的结构称之为文件。由单键和多重键文件结构的讨论表明文件结构和多重链表之间有点差别。

第七、八和九章包括基本的数据管理算法。分类是一个非常大的题目，因此我们只介绍最重要的技巧；搜索也是一个很大的题目，因此我们只讨论最显而易见的算法，在二分搜索技术中引出平均比较次数是本书的独特之处。由于存贮管理有很多的分支，我们致力于最常使用的方法，并且是首次在一本教课书里介绍了存贮管理的分析模型。

第十章包括十个数据结构的应用，这些应用可以用来说明某一具体点，也可以看作是应用方面的最后一章。本章从程序的设计到程序的调试，最后到程序的执行，然而任一应用都能够单独地加以处理。

第十一章是第五章所介绍的图论模型进一步的形式化，数学程度较好的读者可阅读第十一章。

附录A、B和C可以用来丰富使用PL/I程序设计的计算机科学课程。通过例子我们能看出PL/I的构造是很适用于表处理和存贮分配的。

我们觉得本书在若干方面是独特的。记号选择得尽可能准确地表示我们需要涉及的各种概念。在第二章，我们使用联接来表示两个串的结合。遗憾的是没有与拆散可比较的词，因之我们使用了分隔来代替非联接、不联接等等。

在第五章里我们讲述了表的密度概念。这是度量存贮器利用效率的好方法，也是择定各种紧缩表的好方法。以原子（atom）来指示连接表的单位以及以NIL来表示某个表的结尾。这是从LISP借用来的办法。你将会注意到，在后面的各章里我们更喜欢术语结点和顶点。诸如指针、链接或穿成串等术语是大家都共同接受的，但链（相当于链表）、搁置（相当于队列）以及堆积（相当于栈）则是不常使用的。

为了顾及到所叙述的名字，在树的搜索算法中我们不采用当前流行使用的前序、后序和尾序等提法。例如，在前序的场所我们采用根结点－左－右－再重复（NLR－再重复）的提法。原因是使用我们的术语时就不用记住是从树的哪一个结点开始搜索了。

只有在可以为理解公式带来明显好处的分析方法里我们才使用简单的概率和计算。通常，在读者不会弄不清数学来源时，我们就给出性能公式。

算法都是用英语和代数的方式来描述的。可以把它们写成为较严密的模型，并且能毫无困难地把这些模型用某种程序设计语言来实现。在实现过程中将用到结构程序设计的各种技术。有经验的程序设计人员将很容易辨认出这种办法，而一个初学程序设计的人也很容易在他（或她）的职业中形成自身的风格习性。

作者讲授数据结构多年，正是出于下述两个目的而写这本书的：为与数据结构相关联的那些课题提供真实的资料以及提供一本好读的书。我们拟定出了在深度上合适的文献，并确定在本书中将达到什么地步以及在最基本的情况下应该读些什么。有关的文献都可在参考书目中找到。

在使用本教材早期手稿的几批学生的帮助下，使我们达到了第二个目的。他们的贡献是很大的，我们常常是由于听取了他们提出的各种建议而使本书更简明易懂了。

对于他们极有价值的帮助，我们愿意一一感谢在本书各个发展阶段复审了此书的人：Wisconsin大学的David R.Musser，Tennessee大学的Sara R.Jordan，Indiana大学的Daniel

P. Friedman, Purdue大学的Victor B. Schneider，以及Southwestern Louisiana大学的Edwavd Katz。

也还要很愉快地感谢Missouri-Rolla大学的学生和教师，Southwestern Louisana大学的学生，以及May Heatherly所进行的无可挑剔的打字工作。此外，我们还要感谢Terry Walker，John Hamblen，Bob Flandrena，以及Bran Smith的帮助。

T. G. Lewis

M. Z. Smith

1. 什么是数据结构?

1.1 直观的认识

初学程序设计的新手通常都只与少量的数据打交道，并且也不注重贮存信息的有效方法。他们或是选用较比容易的程序设计方法，或是选用他们较为熟悉的方法。然而在着手处理复杂的问题或是包含大量数据的问题时，程序设计人员却必须组织好解决问题的步骤。否则就有可能浪费掉极有价值的计算机时间和存贮量。

按照经典的Von Neumann 模型，大多数计算机都具有顺序排列的存贮器。这意味着我们可以根据单元的次序来勾画出存贮器，每一个单元都带有一个如图1.1中所示的唯一的数值地址

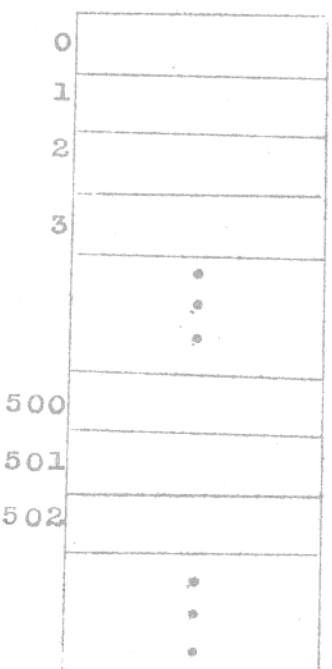

图1.1 存贮区示意图

只要不去清除我们希望保存的值，放在存贮器里的数据通常是不会发生什么问题的。然而，当我们试图使用数据并意识到需要将存贮器中数据组织得更有条理时就有可能产生出困难来。例如，假定我们有一组名字，把它们列成表时名字中有的是最后一个字在前面，而有的则是第一个字在前面。为了简明起见，假定每个名字放入一个存贮单元（事实上每一个名字可能需要几个单元才行，这当然就更会有一个组织的问题了）。通过考察图1.2我们将会发现，用字母顺序排列名字也并不容易，因为必须首先找到名字的最后一个字。

BILL ABRAHAM
CoOKE MARTHA
ANN JOHNSON
PALMET JO
HAMILTON H.J.
SAM PAUL

图1.2　按字母顺序排列的一个名字表

对于一个人来讲，找出名字中的最后一个字要比计算机来得容易。因为人知道Bill、Martha以及Ann是第一个字。然而，我们怎样才能把这个告诉给计算机呢？表上最后一个人又怎么样？他的名字是Sam Paul呢还是Paul Sam？除非能假定所有名字的最后一个字都首先出现，否则任何人按字母顺序来排列该表都会有困难。

结构就是一群东西中各个元素之间的关系。政府、学校以及商店都有其规定的组织结构。在计算机科学中，数据项之间的关系就是数据结构。结构这种概念能用来表示某种抽象的性质吗？例如用其来作为描述数据的方法或访问数据的方法？或者它正是数据项贮存在计算机存贮器里所采用的方法？在本书中，我们将介绍不同的数据结构，给出其术语，讨论各自的存贮方法。我们也将介绍很多恰当的例子，以便说明数据结构的应用。现在让我们来讨论若干种结构类型。你或

许是熟悉它们的，但可能并未能认识到这是一种数据结构的分类。

表（譬如图1.2所给出的）是数据结构的一种类型。各个数据项在两个方面有联系：首先，它们全是名字；其次，它们被贮存在顺序的存贮单元中。为了访问此表，我们必须知道表的起端及其末端或其长度。

当我们着手往表上添加或删除数据项时，表就会出现一些基本问题。例如，把要往食品表或顾客表增加的内容添加到表的末端上去，是一件简单的事情。然而我们把一个名字添加到地址表去时，就可能要按字母排列的顺序来保持名字，于是只得将其写在间隔处或写得小一些，如图1.3所示的那样。

从非计算机型的表中删除一项同样是简单的事情。我们只须要把整个这一行元素都划掉就行了。倘若我们有足够的纸，那么在一个无序表里增加或抹掉名字不会有什么困难。但如果我们的表是有序的话，插入或删除就可能会有很多的问题了。在需要增加、撤除或修改名字的通讯簿情况里，变更和维持一个按字母顺序排列的表就会受到篇幅的限制。实际上就会不时地要求得到新的通讯簿，因而要进行最新的拷贝。

由于表的最简形式是项的有序排列，因此把表存放在计算机存贮器里是非常容易的事情，它很自然地与计算机存贮器的有序性相对应。相对于表，表格（或矩阵）是二维的。做为一个例子，考虑如图1.4所示的表格，它给出了购买1、3或12个单位的物品时的价格。假如一个存贮单元只放一个表目，那么4×7个表目的表格需要28个单元。

大多数高级语言多有贮存表格的内部过程，访问单独的项目时，程序设计人员只需要给出它的行和列号就行了。例如，要得到第1175号物品12个单位的价格，我们就访问第3行第4列，从而找到是13.76。在汇编语言或不能自动完成贮存表格的语言上工作时，就需准备好我们自己的策略。通常，表格可按列或按行存放（见图1.5）。如果我们知道表格的确切尺寸（行数和列数），那么就能计算出任何项目所在的位置（见练习3）。

(a) 往食品表的末尾添加内容

milk
eggs
bread
beef
ham

(b) 往地址表添加内容

Adams,Bill 212 Greenwood
Carter,Lyne 1728 Bernay
Craig Eleanor 507 woodlawn
Dillard,H.209 Woodland
Drysdale,Jane 111 Pine

图1.3 往表上添加内容

物品号码	这些数量的价格		
	1	3	12
1062	.99	2.84	10.50
1048	1.88	5.60	22.38
1175	1.29	3.17	13.76
1287	2.35	7.01	25.11
1296	10.12	28.92	118.05
1408	5.06	14.32	58.73
1450	7.21	20.86	80.90

图1.4 价格表

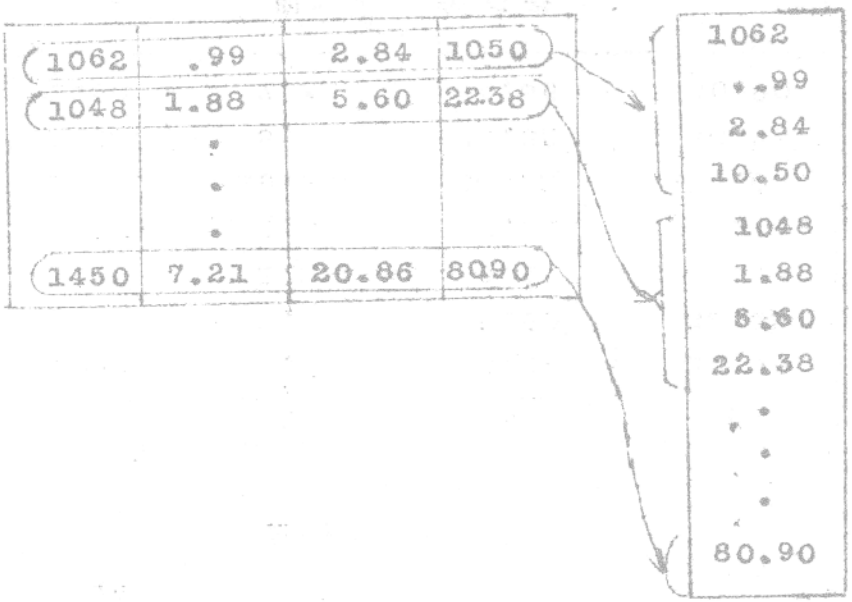

图 1.5　按顺序存贮单元存放表格

在进行表格处理时，我们必须考虑诸如插入或删除物品、按物品号整理表格、或者为寻求特定的物品而对表格进行搜索等等课题。这些要考虑的问题正是本书后面几章所要讨论的专门问题。

文件是数据结构的另一种类型。文件是一批记录，通常贮存在计算机存贮器外面的磁带或磁盘上。常把卡片箱的内容类比为文件，每一张硬纸卡片里的信息就是一个记录。由于文件通常都很庞大，因此在一次时间里只有其一部分能被读进主存。这些部分的尺寸结构也是很重要的，要使得存取时间减到最小。有一些文件，例如预订飞机票的各记录是动态的，人们经常建立或取消预约，因而这样的文件不时地在变化着。其它文件大体上是静态的，不必过多的注意，例如人口普查数据的变化仅每十年一次。

形式上更为复杂一些的又一种数据结构是树。树通过用来表示元素的分层组织。图1.6说明了在装配飞机时完成任务必须遵循的次序。最高层（1层）是所完成的产品；最低层（4层）包括装配线各任务，它们可以同时完成。4层的任务（肋材构架，金属板，电子仪器，涡轮机）必须在3层任务（机身、机翼和引擎）可以进行之前完成。因此，在此例中我们的工作是从树的底部到顶部的。

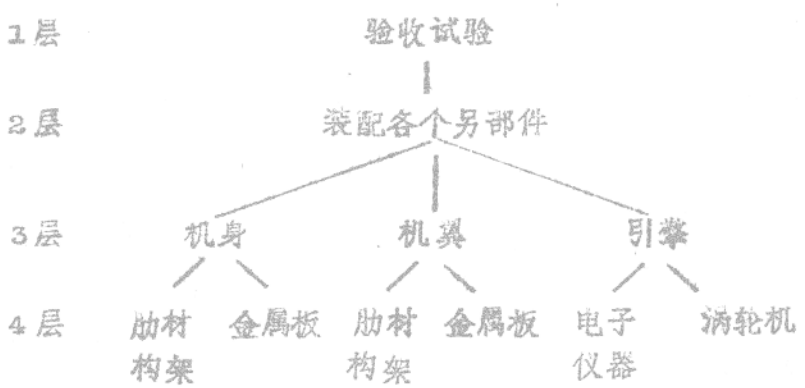

图1.6 有关飞机生产线的装配树

树也可在遊戏中用来表示正确的走法。例如在tic-tac-toe遊戏中我们有权先走。那就首先要作出选择，看是把棋子放在九个格的哪一个里面。在对手走了以后，我们就被限制于在七个方格中选择一个了。因此在一次遊戏中，我们是从树的一层到另一层，只用其一小部分。

来看图1.7，并假定你是“X”，而你的对手是“O”。开始，你可以选择把棋子放在左上方的方格、左边的方格、中央方格、右上方的方格或……，假定你最初选取了左边的方格。由于你是采用了树的中央分支，你的对手现在就可能回报（在第3层）以左上方、中央上方、右上方、右方以及中央。tic-tac-toe棋盘的对称性使我们能把树简化成如图1.7所示，这也就能够说明全部情形了。

当越来越多的and/or分支相应于随后的走步给出时，遊戏树就伸展开来。最佳遊戏策略就可用来决定选取哪一个“or”分支。例如，在tic-tac-toe中，最佳策略是选择提供获胜道路最大数目的那种“or”分支。在1层，获胜道路的最大数是由导致“X”在中央的那条“or”分支给出的（X在左上方只允许有三条直线，在左边只允许有两条直线，而在中央则允许有四条直条）。

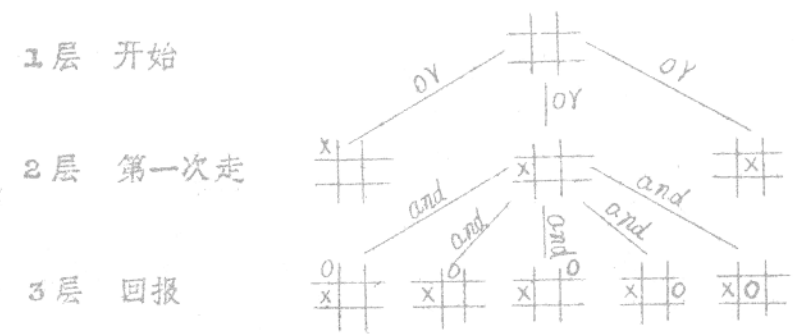

图1.7 部分tic-tac-toe遊戏树。只给出可能的对称走法。一个玩的人选“or”走法，其对手通过选“and”走法来玩。

在树的两个例子里应特别注意的是建立了层的概念。这是树的一个重要特征。层次可用来表达什么更为重要、什么必须先完成或者什么更为概括。这是一种单路径结构的概念化。从1层的项寻找最底层的每一项，有且仅有一条路径可走。若要有多路径结构，就需要一种称为图的组织了。

我们可以把图想像成是描绘城市以及它们之间连接的汽车司机行车图。通常，如果我们想要查找城市间的最短距离，就标上连接的长度，如图1.8所示。图在运输中有着广泛地应用，当要行经某几处且要求最少往返时，它就有助于找到最短、最快的路线。一个例子是安排校车或送货卡车的路线。

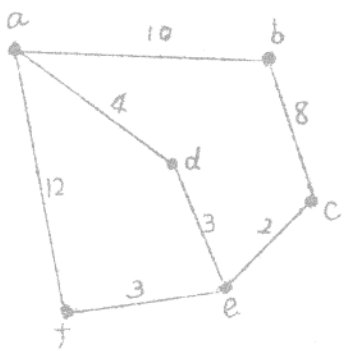

图1.8　边被标以数字、结点（顶点）标以字母的图

在此，我们将不讨论树或图在计算机存贮器里的存放问题。代之的是请你思考这样的问题：人们将如何把一种非顺序结构（例如树或图）存放到顺序的存贮单元中，并且能准确地表示出它的结构？

练习

1。解释下列术语：

（a）顺序存贮器；

(b) 结构;
(c) 文件;
(d) 遊戏树。

2。论述一种允许表增长的方法。对有序表和无序表而言，插入有何不同?

3。若一表格(例如图1.5中的那个)被存放在顺序存贮单元中，假定已知在表格中的行、列号，说说如何确定任何一个值?

4。画出如图1.9所示的Konigsberg问题的桥的图来。该问题促使Euler(1736)开始了图论的研究。

Konigsberg(现在的加里宁格勒)的居民打算星期日绕着市区开车兜风。但他们有一个问题：该市区被一条河流分隔成四部分，他们如何能够在返回出发点之前只一次就穿越这七座桥?(提示：假定把陆地聚集为图的结点，桥为图的边。)

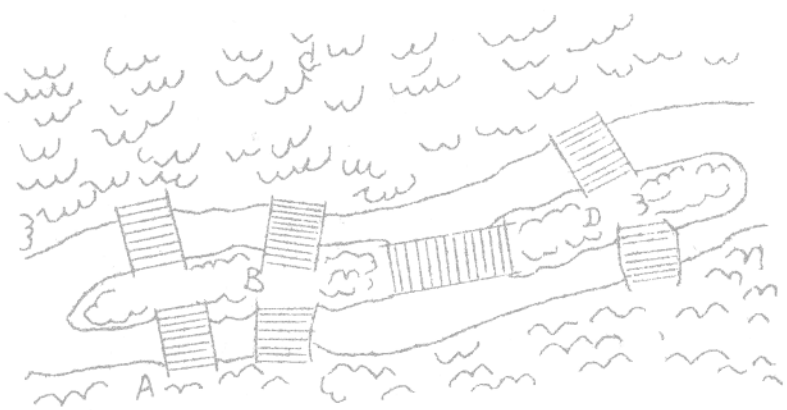

图1.9 连接陆地A，B，C和D的Konigsberg桥

5。画出表示国家、州以及州属城市间关系的树。你的通讯地址与此结构相适当吗?