普通高等教育“十五”国家级规划教材配套参考书

数据结构与算法
——学习指导与习题解析

张　铭　赵海燕　王腾蛟

高等教育出版社

普通高等教育“十五”国家级规划教材配套参考书

数据结构与算法

——学习指导与习题解析

张　铭　赵海燕　王腾蛟

高等教育出版社

内 容 提 要

数据结构与算法课程的学习目的是，根据应用问题的性质选择合理的数据结构，在合理的时间、空间复杂度限制下编程加以解决。认真地完成习题和上机题，是学好本课程，提高程序设计质量的重要环节。

本书配合我社出版的面向21世纪课程教材《数据结构与算法》的使用，为读者学习数据结构与算法课程给予指导。全书共14章，其中，第1～12章总结了本课程重要的内容知识点、学习重点和难点，某些章节还对相关知识点进行了扩展；前13章从题意分析、典型错误、数据结构、算法代码、算法分析等多个角度给出了主教材中212道习题和53道上机题的综合分析和参考解答，并新收入了覆盖各章知识点的170多道习题和40多道上机题供读者练习；第13章内容基本上选自ACM国际大学生程序设计竞赛题，强化算法实现和上机实习能力；第14章以1999～2005年北京大学计算机系研究生入学考试数据结构试题及解答为主，辅助读者自学与自测。

本书可作为普通高等院校计算机及相关专业数据结构与算法课程的教学参考书，也可供参加计算机硕士、计算机博士、软件工程硕士入学考试的考生参考使用，还可供计算机应用技术人员参考使用。

图书在版编目(CIP)数据

数据结构与算法：学习指导与习题解析/张铭，赵海燕，王腾蛟．—北京：高等教育出版社，2005.10

ISBN 7-04-017829-X

Ⅰ.数… Ⅱ.①张…②赵…③王… Ⅲ.①数据结构—自学参考资料②算法分析—自学参考资料

Ⅳ.TP311.12

中国版本图书馆CIP数据核字(2005)第113377号

策划编辑 倪文慧 **责任编辑** 倪文慧 **封面设计** 于文燕 **责任印制** 陈伟光

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街4号	**免费咨询**	800-810-0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
			http://www.landraco.com.cn
经　　销	北京蓝色畅想图书发行有限公司		
印　　刷	北京民族印刷厂		
开　　本	787×1092　1/16	**版　　次**	2005年10月第1版
印　　张	32.25	**印　　次**	2005年10月第1次印刷
字　　数	670 000	**定　　价**	39.50元

物料号　17829-00

前　言

计算机科学已经深入应用到各个领域，不仅有效地解决了各种工程和科学计算中的数值计算问题，而且也有效地解决了许多文本处理、信息检索、数据库管理、图像识别、人工智能等非数值的数据处理问题。

数据结构是计算机学科的一个重要分支研究领域，它是算法分析与设计、操作系统、软件工程、数据库概论、编译技术、计算机图形学、人机交互等专业基础课和专业课的先行课。其他计算机科学领域及有关的应用软件都要使用到各种数据结构，例如：语言编译要使用栈、散列表及语法树；操作系统中用队列、存储管理表及目录树等；数据库系统运用线性表、多链表及索引树等进行数据管理；而在人工智能领域，依求解问题性质的差异将涉及到各种不同的数据结构，如广义表，集合、搜索树及各种有向图，等等。

美国 IEEE 和 ACM 的教学计划 CC2001 把“算法与数据结构”列入计算机以及信息技术相关学科专业的本科必修基础课程。在我国，“数据结构与算法”已经作为理工科非计算机专业必修的信息技术基础课程之一。世界上许多科技人员对学习、研究数据结构都非常重视，对于从事计算机科学及其应用的科技工作者来说，数据结构更是必须透彻地掌握的重要基础，有助于程序员更有效地组织数据、设计高效的算法、完成高质量的程序，以满足错综复杂的实际需要。

从字面上来看，数据结构就是指数据间的相互关系。具体到计算机环境，谈到任何一种结构时，都自然地联系着作用在这种类型的数据上的运算（即函数），为了在计算机上执行这些运算，有必要把这些数据以某种方式存储在计算机中。因此，可以认为，所谓数据结构，就是由某种逻辑关系组织起来的一批数据，按一定的存储方法被存储于计算机中，并在这些数据上定义了一个运算的集合。也就是说，数据结构具有 3 个方面：数据的逻辑结构、数据的存储结构和数据的运算。

事实上，数据结构的三个侧面，以数据的逻辑结构和数据的运算定义更为重要。因为很多时候人们并不关心数据的存储结构和运算的具体实现。1983 年 Aho 等人把数据结构的存储与实现细节剥离，定义了抽象数据类型（简称“ADT”）的概念。抽象数据类型是定义了一组运算的数学模型。这种抽象的数据类型可以在较高级的算法中直接引用，而不用考虑其实现细节。这就使得设计者可以在不同的设计阶段采用不同的抽象数据类型作为设计的基础，在适当的抽象层次上考虑程序的结构和算法，从而很好地支持了逻辑设计和物理实现的分离，支持封装和信息隐蔽。

从教学的角度，应该以数据的逻辑结构为主线，顺序介绍线性结构、树形结构、图结构和

文件结构。在介绍每种数据结构时,再讨论其存储结构以及相关的算法。例如,对于线性表,考虑到其存储结构,分别讨论了数组和链表;考虑到其运算的特殊性,分别讨论了向量、栈和队列。对于一些比较重要的算法,再列出单独的章节来讨论,例如排序、检索、索引等。每一种数据结构与其相关的算法都有时间、空间开销和效率等问题,需要揉合一些基本的算法分析技术。基本问题包括:对于给定的一类问题,最好的算法是什么?需要多少存储空间和时间?空间与时间的折中方案是什么?存取数据最好的方法是什么?最好算法的最坏情况是什么?平均来说,算法的运行好到何种程度?算法一般分析到何种程度——即什么类型的问题可以用类似的方法处理?通过数据结构的学习,在面临新的设计问题时,应该知道怎样权衡时间与空间开销、设计出更有效的数据结构和算法,以适应问题的需要。

从另一个角度来看,Peter J. Denning 等人认为,计算机学科分为理论、抽象与设计这三个形态。我们把数据结构课程所涉及的主要方面整理为:

1. 理论

(1) 算法的数学基础。例如,有关图论、组合论等,特别是递归、递推、归纳等分析方法。

(2) 算法的时间和空间度量。

2. 抽象

(1) 排序、检索等重要问题类的有效算法,以及最好、最差和一般性能的分析比较。

(2) 重要的数据结构技术,例如:分治法(二分检索、快速排序、归并排序)、贪心算法(Huffman 编码、Prim 算法、Kruskal 算法、Dijkstra 算法)、动态规划(Prim 算法、Dijkstra 算法、Floyd 算法、最佳二叉搜索树)、栈的引用(深度优先周游)、队列的应用(广度优先周游),等等。

3. 设计

(1) 掌握并灵活应用常用的基本数据结构的抽象数据类型、各种基本存储方法及其主要的算法,例如,线性结构(包括一维数组、链表、栈、队列、字符串等)、二叉树、树、图、文件,等等。

(2) 排序、检索、索引等重要问题类的算法的选择、实现和测试。例如,各种排序方法的实验时间比较,散列、倒排索引、B 树等应用。

(3) 存储管理的实现与测试。例如,可利用空间表、无用单元收集、伙伴系统,等等。

也就是说,数据结构这门课程不仅仅要让学生掌握链表、树、图是如何实现的,还要进一步从理论、抽象、设计的角度来考虑问题。数据结构这门课程的三个教学目标是:第一,让学生懂得"数据结构 + 算法 = 程序",也就是说,程序不仅仅是算法的实现。求解问题要设计恰当的数据结构,并把它和求解算法结合起来;第二,是培养数据抽象的能力,不仅让学生了解链表、树、图等数据结构是如何实现的,特别要加强对数据逻辑关系的分析与认识;第三,要把数据结构与算法的理论分析与编程实践相结合,在实际应用中灵活运用。综合性的上机

项目对于达到第三个目标是很有帮助的。在达到前两个目标时，学生就基本具备了解决现实未知问题的能力，再辅以必要的综合上机项目训练，可以达到第三个目的。

总之，需要强调以下四个方面的知识和能力：(1) 掌握并能够灵活应用基本数据结构的抽象数据类型、存储方法、主要的算法，特别是线性结构、二叉树、树、图、文件等；(2) 掌握并应用常用的排序、检索和索引算法和方法；(3) 掌握基本的算法设计和分析技术，并结合具体的设计，对所设计的数据结构和算法进行分析；(4) 在进行程序设计、调试、测试的项目训练(即上机实习训练)中，注意综合应用，将所学到的数据结构和算法知识应用到对具体数据对象的特性分析。结合实际例子进行设计，选择合适的数据结构和存储结构以及相应的算法。显然，合理地组织数据、有效地表示数据、有效地处理数据，这三者是提高程序设计质量的关键因素，它们为提高程序的清晰性和易读性等创造了基础。

与本书配套的主教材《数据结构与算法》(高等教育出版社 2004 年出版，本书引用时统称“教材”)自出版以来，受到了众多读者的欢迎。由于教材中每一章中都设计了比较新颖的习题和综合上机实习题(即项目实习)，不少读者迫切希望出版配套的习题解答，作为教学和自学参考，以更好地完成数据结构与算法这门理论和实践结合比较紧密的课程。

配合主教材的使用，作者编写本书，为读者学习数据结构与算法课程给予指导。本书分为 14 章，第 1 ~ 12 章总结了主教材重要的数据结构和算法知识点、学习重点和难点，某些章节还对相关知识点进行了扩展。全书前 13 章从题意分析、典型错误、数据结构、算法代码、算法分析等多个角度，给出了主教材中 212 道习题的和 53 道上机题的综合分析和参考解答，并新收入了覆盖各章知识点的 170 多道习题和 40 多道上机题供读者练习。尤其是第 13 章，选择 ACM 国际大学生程序设计竞赛的题目作为例题和习题，从穷举法、搜索和剪枝、动态规划、栈和图的应用等算法的角度加强编程实习训练，并且给出了法雷序列上机实习报告样例。第 14 章为考试题及其解答，包括 2004 年秋季学期的期中、期末考题和答案，以及 1999 ~ 2005 年北京大学计算机系硕士研究生入学考试数据结构试题及解答，有助于读者了解考核的内容和形式，并自测学习效果。

有些读者反映主教材内容比较多，也比较深。事实上，读者可以根据需要进行裁减。2004 年秋季学期我们几位作者给北京大学信息学院学生试讲该教材时，对微电子、电子学的学生，只讲授了第 1 ~ 10 章内容，而且不要求学生掌握非递归的二叉树周游、穿线二叉树、等价类的并查算法、外排序的选择树算法。对计算机专业的二年级本科生，则稍微加快进度，在接近一半的学期课程时间就讲完了前 6 章，重要的章节都安排比较综合的上机实习题；在开始讲解排序与检索的同时，开始安排学期综合上机实习题搜索引擎(见上机题 12.10)；最后第 11 ~ 12 章比较高级的数据结构内容只是讲座性的内容，重点介绍了稀疏矩阵、广义表、字符树、AVL 树，没有新的上机实习，而学生也面临期末考试复习了，这样的安排更有利于学生调节学习时间。给电子商务专业的学生讲授本课程时，根据大部分学生来自法律、经济等文科专业的特点，我们把最基本的内容安排在前 6 章讲解完，然后简单介绍了第 7 章的内排

序,其他内容让感兴趣的学生自学。从期末考试和学生评估结果来看,电子商务专业的学生们反映课程内容非常充实,知识点掌握得很牢固。

从作者多年的教学经验来看,只有经过大量的实践训练,才能真正深入理解数据结构与算法的精髓,熟练掌握各种经典的数据结构和算法,更有效地组织数据、设计高效的算法、完成高质量的程序,以满足错综复杂的实际需要。本书对于广大教员、复习和准备参加考试的学生具有最大的参考价值。但是,对于正在学习"数据结构与算法"课程的学生来说,应该以掌握基本的数据结构和算法知识为主,适当培养自己动手编写数据结构和算法来解决应用问题、分析算法的时间与空间代价的能力,不应过于依赖现成的习题分析和解答。当然,自己先动手解答,然后再与参考答案进行对比分析,还是值得提倡的学习方法。

本书第7、9、10、12、13、14章由张铭教授执笔;第1、2、3、11章由赵海燕副教授执笔;第4、5、6、8章由王腾蛟副教授执笔。其中第1-12章习题解答部分的原始习题和上机题直接选自与本书配套的《数据结构与算法》主教材,作者在此对许卓群教授、杨冬青教授、唐世渭教授衷心致谢。本书第13章的内容基本上选自李文新副教授主持的北京大学ACM ICPC在线评测系统(简称POJ),其网址是http://acm.pku.edu.cn/JudgeOnline/,作者对此表示感谢。本书的很多内容曾经由我们三位作者在2004年度给北京大学信息学院讲授的"数据结构与算法"课程中使用,感谢同时执教的冯梅萍副教授的建设性贡献,在共同执教过程中,我们一起讨论课程重点、交流授课经验,集体编写期中和期末考试试题、讨论参考答案并制定评分标准。下列同学对书稿和算法代码提出了许多宝贵的修改意见,在此一并表示感谢:参与课程辅导的程兰兰、朱兴国、赵静、银平、邓鹏、丁树凯、梁希云、陈昆、郑闽睿、苏志华、李双峰、杜昆鹏、魏可伟、伍赛、赵培翔、李丽、王蜀安、钟松、王宁等历届硕士研究生,听课的阳萌、柳超、王位春、何啸、杨宇、杨健、饶向荣、江云亮等历届本科生。

尽管本书经过了作者反复讨论和推敲,并经历了2004年秋季学期的教学检验,但限于水平,难免有不妥之处,希望读者不吝赐教。为了读者联系方便,作者已经部分开放了北京大学信息学院数据结构网站http://db.pku.edu.cn/mzhang/ds/index.htm。网站主要内容有:多媒体演示讲稿、张铭主讲的主课视频、标准C++模板编写的可执行的源程序代码。课程网站还有一个BBS讨论版,读者可以在BBS上提出问题、建议和意见,作者将组织助教回答问题、讲解习题中的要点和难点等。

张铭 赵海燕 王腾蛟

2005年9月于北京大学

郑重声明

目　录

第1章 概 论

“数据结构与算法”是计算机科学与技术学科的一门专业基础课程，它为后续的多门专业课程提供了必要的知识和技能准备，例如，程序设计语言及其编译技术要使用栈、散列表及语法树，操作系统会用到队列、存储管理表及目录树，数据库系统会用到线性表、多链表以及索引树等基本数据结构及其相关算法等。

教材第1章主要讨论了在其后续各章都要用到的一些预备知识和相应的设计原则。

1.1 知识点总结

1.1.1 学习数据结构的目的和目标

数据结构课程的主要目的是介绍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，并结合各种数据结构，讨论对它们进行的各种运算的实现算法。很多算法实际上是对某种数据结构施行的一种变换，研究算法也就是研究在实施变换过程中数据结构的动态性质。

1.1.2 什么是数据结构

数据结构包括数据的逻辑结构、数据的存储结构和数据的运算三个方面，即涉及数据之间的逻辑关系、数据在计算机中的存储方式和在这种数据结构上的一组操作三个方面。

1. 数据的逻辑结构

数据的逻辑结构反映了人们对数据的含义解释。一个逻辑结构可以用一组数据结点和一个关系集合 R 来表示：

$$(K,R)$$

其中，K 是由有限个结点组成的集合，R 是一组定义在集合 K 上的二元关系。在此需要重点掌握的内容包括结点的类型、结点的结构等几部分。

结点的类型可以是基本数据类型，也可以是复合数据类型。常见的程序设计语言中使用了5种基本的数据类型：整数类型、实数类型、布尔类型、字符类型、指针类型，它们的共同特点是：在程序访问数据时，把基本数据类型看成一个整体，而不会把基本类型的一个组成部分看成具有独立含义的数据。

根据关系集 R 中关系 r 的分类，结点的结构一般可分为线性结构、树形结构和图结构。这种结构分类揭示了枯燥数据之间的相互关系，给出了关系本身的一般性质，对于理解数据结构以及正确设计算法都是很重要的。

线性结构在程序设计中应用最为广泛。它是一种满足全序性和单索性等约束条件的有向关系。全序性是指，线性结构的全部结点两两皆可以比较前后；单索性是指，每一个结点 x 都可以存在惟一的一个直接后继结点 y。树形结构，也称树结构，是一种层次结构，其关系 r 称为层次关系或“父子关系”。树形结构的最高层次的结点称为根结点(root)，只有它没有父结点。树形结构存在很多变种，如二叉树、堆结构等，它们都有各自独特的应用。图结构也称网络结构，其关系 r 没有任何约束。

2. 数据的存储结构

数据的存储结构是建立一种由逻辑结构到存储空间的映射：对于逻辑结构 (K,r)，其中 $r\in R$，对它的结点集合 K 建立一个从 K 到存储器 M 的单元的映射：$K\rightarrow M$，其中每个结点 $j\in K$ 都对应一个惟一的连续存储区域 $c\in M$。

常用的基本存储映射方法有顺序方法、链接方法、索引方法和散列方法。

顺序存储把一组结点存放在按地址相邻的存储单元里，结点间的逻辑关系用存储单元的自然顺序关系来表达，即，用一块存储区域存储线性数据结构。顺序存储方法为使用整数编码访问数据结点提供了便利。因为，顺序存储方法的存储空间除了存储有用数据外，没有用于存储其他附加的信息，所以顺序存储结构一般也被称为紧凑存储结构。

链接法是在结点的存储结构中附加指针字段来存储结点间的逻辑关系。链接法中数据结点包括两部分：数据字段存放结点本身的数据，指针字段存放指向其后继结点的指针。链接方法适用于那些需要经常进行增删结点的复杂数据结构。

索引法是顺序存储的一种推广，用于大小不等的数据结点的顺序存储。通过建造一个由整数域 Z 映射到存储地址域的函数，把整数索引值映射到结点的存储地址，从而形成一个存储一串指针的索引表，每个指针指向存储区域的一个数据结点。

作为索引法的一种延伸和扩展，散列法利用散列函数进行索引值的计算，然后通过索引表求出结点的指针地址。

1.1.3 抽象数据类型

作为描述数据结构的一种理论工具，抽象数据类型(Abstract Data Type，ADT)可以看作是定义了一组操作的一个抽象模型，其特点是把数据结构作为独立于应用程序的一种抽象代数结构，在很大程度上可以使人们独立于程序的实现细节来理解数据结构的主要性质和约束条件。

抽象数据类型是模块化思想的发展，有助于人们从更抽象的高度去讨论算法和数据结构的问题。

1.1.4 算法及其特性

算法是一个十分古老的研究课题。简单来说,它是为求解问题而给出的指令序列。一个算法可能有若干个输入,这些输入数据在算法开始时提供一组量。对算法的描述应该精确地说明这些输入的个数、类型以及它们应满足的初始条件,算法的每个步骤必须被明确描述,并且可行,不能有二义性。

算法的一般性质包括以下几点:

(1) 通用性 对于那些符合输入类型的任意输入数据,都能根据算法进行问题求解,并保证计算结果的正确性。

(2) 有效性 组成算法的每一条指令都必须是能够被人或机器确切执行的。

(3) 确定性 算法每执行一步之后,对于它的下一步,应该有明确的指示。即,保证每一步之后都有关于下一步动作的指令,不能缺乏下一步指令或仅仅含有模糊不清的指令。

(4) 有穷性 算法的执行必须在有限步内结束。

在实际应用中,算法的表现形式千变万化,但许多算法的设计思想具有相似之处。归纳起来,常用的算法大致可分为以下几类:

(1) 穷举法 基本思想是在一个可能存在可行状态(可行解)的状态全集中依次遍历所有的元素,并判断是否为可行状态。

(2) 贪心法 基本思想是试图通过局部最优解得到全局最优解。

(3) 分治法 基本思想是把一个规模较大的问题划分成相似的小问题,各个求解,再得到整个问题的解。

(4) 回溯法 基本思想是一步一步向前试探,有多种选择时任意选择一种,只要可行就继续向前,一旦失败时就后退回来选择其他可能性。

(5) 动态规划法 基本思想是把大问题分解为若干小问题,通过求解子问题来得到原问题的解。由于这些子问题相互包含,为了复用已计算的结果,常把计算的中间结果全部保存起来,自底向上多路径地求解计算原问题的解。

(6) 分支界限法 基本思想是在表示问题空间的树上进行系统搜索时采用广度优先策略,同时利用最优解属性的上下界来控制搜索的分支。

1.1.5 算法的执行效率及其度量

解决同一个问题一般存在多种算法。要想从这些算法中选择一个适合的算法作为解决方案,则需对算法进行度量和评价的方法。评价一个算法优劣的重要依据是渐近分析实现该算法的程序在计算机中执行时所需占用的机器资源的多少。两个重要指标为算法的空间代价(或称空间复杂度)和算法的时间代价(或称时间复杂度)。

算法的空间代价一般定义为求解问题的算法所需占用的存储空间大小。当问题的规模

以某种单位由1增至n时，求解该问题的算法所需占用的空间也以某种单位由$f(1)$增至$f(n)$，则称函数$f(n)$为该算法的空间代价。

算法的时间代价一般定义为求解问题的算法所需耗费的时间多少。当问题的规模以某种单位由1增至n时，求解该问题的算法所需耗费的时间也以某种单位由$g(1)$增至$g(n)$，则称函数$g(n)$为该算法的时间代价。

以算法的时间代价为例，算法的渐近分析就是估计当求解问题的规模n逐步增大时，时间开销$T(n)$的增长趋势。为简化时间和空间复杂度的度量，可以只关注于复杂度的量级，而忽略量级的系数。而从数量级大小来考虑，当n增大到一定值以后，$T(n)$计算公式中影响最大的就是n的幂次最高的项，其他的常数项和低幂次项都可以忽略。

渐近分析的结果是得到一个大O渐近表达式，简写为

$$\operatorname*{rate}_{n\to\infty} T(n) = O(f(n))$$

其中，$T(n)$是算法运行所消耗的时间或存储空间的总量，O是数学分析常用的符号“大O”，$f(n)$是自变量为n的某个具体的函数表达式，n反映求解问题的规模。即，如果存在正的常数c和n_0，当问题的规模$n \geqslant n_0$后，某算法的时间（空间）代价$T(n) \leqslant c \times f(n)$，则称该算法的时间（空间）代价为$O(f(n))$。

根据大O表示法的定义，有下列两个常用的计算规则：

（1）加法规则　设有两个程序段S_1和S_2，其时间代价分别为$T_1(n) = O(f_1(n))$和$T_2(n) = O(f_2(n))$，将两个程序段连接在一起得到$(S_1;S_2)$，则总的时间代价为$T(n) = T_1(n) + T_2(n) = O(f_1(n)) + O(f_2(n)) = O(\max(f_1(n), f_2(n)))$。

（2）乘法规则　假设有$T_1(n) = O(f_1(n))$和$T_2(n) = O(f_2(n))$，如果一个程序的时间代价为$T_1(n) = O(f_1(n))$，但其时间单位并不是最基本的，而是以某个子程序的执行代价为单位时间（$T_2(n) = O(f_2(n))$）来考虑的，那么这个程序的实际时间代价为$T(n) = T_1(n) \times T_2(n) = O(f_1(n)) \times O(f_2(n)) = O(f_1(n) \times f_2(n))$。

1.1.6　数据结构的选择和评价

求解一个问题时，如何设计或选择数据结构可以遵循下列一些原则：

（1）仔细分析所要解决的问题，特别是求解问题所涉及的数据类型和数据间的逻辑关系。

（2）数据结构的初步设计往往在算法设计之先。

（3）注意数据结构的可扩展性。包括考虑当输入数据的规模发生改变时，数据结构是否能够适应。同时，数据结构应该适应求解问题的演变和扩展。

（4）数据结构的设计和选择也要比较算法时空开销的优劣。

1.2　教材习题解答

1.1　设字符集为字母和数字的集合，字符的顺序为A,B,C,…,Z,0,1,2,…,9，请将下

列字符串按字典顺序排列、存储：PAB，5C，PABC，CXY，CRSI，7，B899，B9，并分析可以采取的存储方案。

解答：

【题意解析】

本题指定了字符的顺序，所以不能按照 ASCII 字符顺序来排序。

【典型错误】

（1）不按照题目给出的字符顺序进行排序，而按照 ASCII 字符顺序；

（2）没有给出排序结果；

（3）认为顺序存储法比较节约空间，事实上由于字符空隙，顺序存储法并不节约空间；

（4）没有比较各种方法的优劣。

【数据结构】

本题可以采用的存储结构有顺序（数组）和链表及索引等方式。

（1）如图 1.1 所示，用二维数组 array[NUMOFSTRING][MAX_LENGTH]存储，此题可定义为：

```
const int NUMOFSTRING = 8, MAX_LENGTH = 5。
```

B	8	9	9	\0
B	9	\0		
C	R	S	I	\0
C	X	Y	\0	
P	A	B	\0	
P	A	B	C	\0
5	C	\0		
7	\0			

图 1.1 字符串的顺序存储示意

优点：为紧凑存储结构，没有存储其他附加的信息，表示方法比较直观简单，代码实现十分容易。

缺点：为每个单词都开辟了同样长度的空间，造成了空间的浪费。

（2）用链表结构存储如图 1.2 所示，结点为结构，可定义为：

```
struct strings {
    char string[MAX_LENGTH];
    strings * pNext;
}
```

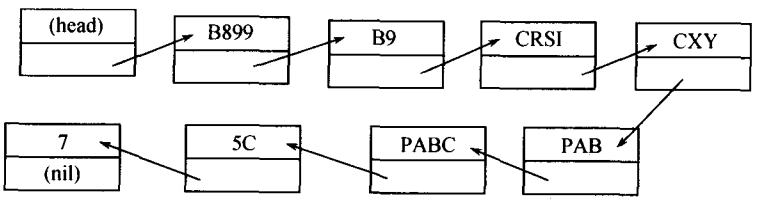

图 1.2 字符串的链式存储示意

优点：如果有后续工作，如反复增删结点，则效率很高，并且可以使用不连续的空间。

缺点：操作过程相对复杂，容易出错，而且排序过程需要从表头开始沿链索一个结点、一个结点地比较搜索，相当费时。

（3）索引存储是顺序存储的一种推广。使用一个字符串 char data[500]，将大小长度不

等的数据结点(单词)顺序存储其中。并使用一个字符指针数组 char * index[n]存储一系列指针,每个指针指向存储区域的一个数据结点。排序时,直接对 index 中的地址值进行调换修改即可,而不用修改 data 中的任何内容。

索引存储的优点是:由于单词长度不同,在存储时充分考虑了这个因素,可以节省空间;此外由于交换的不是单词本身而是单词的地址,可以节省时间,从时间和空间两方面得到了优化。

【排序结果】

B899,B9,CRSI,CXY,PAB,PABC,5C,7

1.2　有一个包括 100 个元素的数组,每个元素的值都是实数,请写出求最大元素的值及其位置的算法,讨论它所可能采取的存储结构。

解答:

【题意解析】

本题没有指明这 100 个实数是否存在相等的情况,因此这里考虑存在多个最大值的情形(即 100 个实数可能有相等的),为了保存其位置,利用 int pos[100](因为有可能这 100 个实数都是相同的)来保存最大值的所有位置。

【典型错误】

(1) 用 int 类型的数组来保存这 100 个元素,没有注意题目中说的是"每个元素的值都是实数";

(2) 求最大值时代码如下:

```
temp = 0;
for( int i = 0;i < 100;i++ )
    if( Num[ i ] > temp)
       temp = Num[ i ];
```

评注:这是错误的,例如,当 Num[i]都是负数时。

(3) 未考虑可能存在多个最大值的情况,只保存了一个最大值的位置。

【数据结构】

本题可以采用的存储结构有顺序(数组)、链表和索引。本题最好使用数组存储结构。由于其他存储方法需要记录附加信息,使得空间有效利用不能够最大化,因此在空间上顺序存储是最优的。时间上,无论如何此题都要遍历所有的数,因此 $O(n)$ 为可能的最优时间代价。加之此题的规模较小,因此使用大家最熟悉的顺序存储是较为优先的选择。

【算法描述】

(1) 由于最大值可能不止一个,甚至可能都是最大值,所以创建一个长度为 100 的整型数组 pos[100],用来记录最大值的位置。

(2) 初始情况,取这个数组的第一个位置为最大值所在的位置,存入变量 position 中。