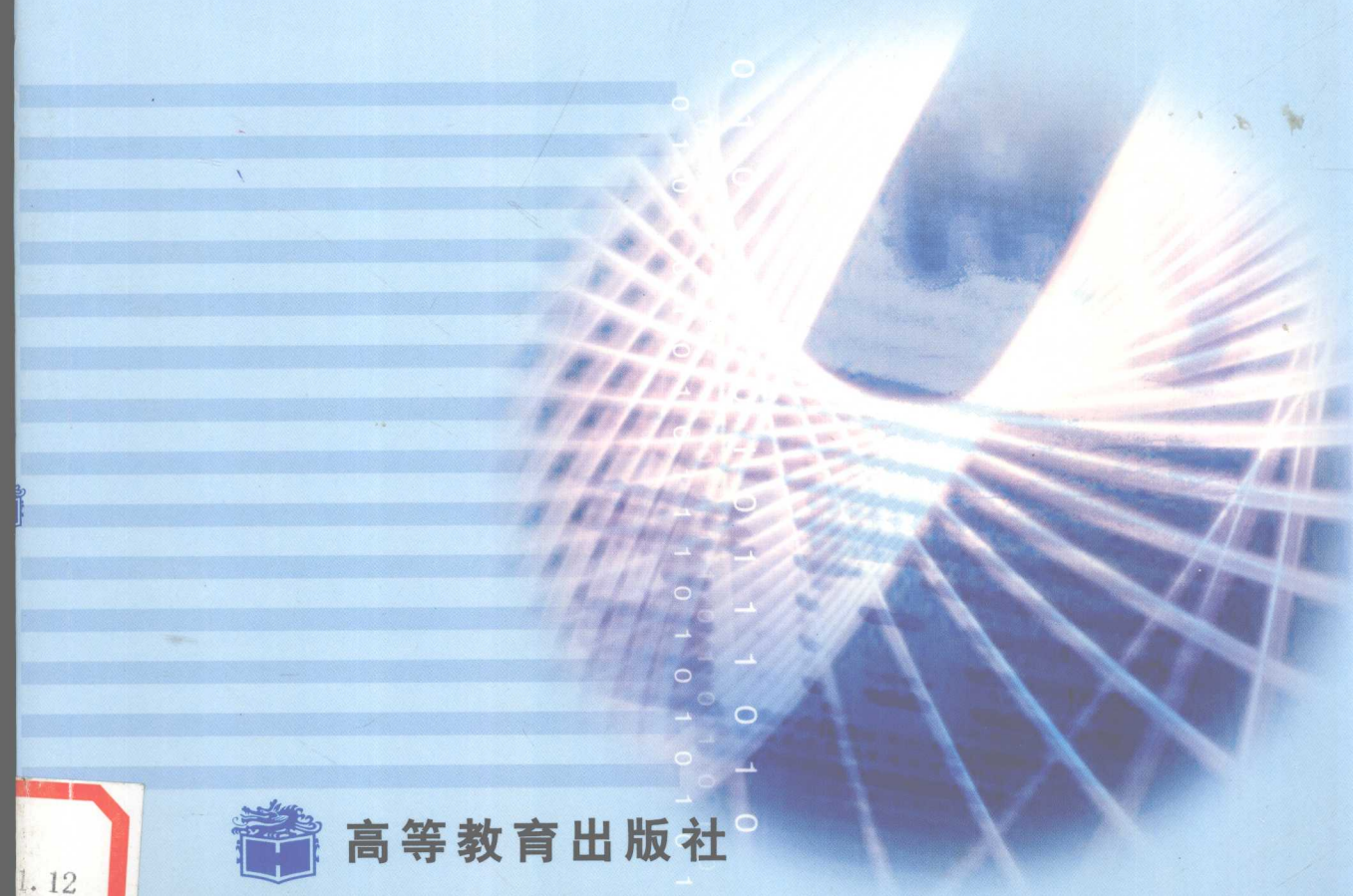
普通高等教育“十五”国家级规划教材

数据结构

（第二版）

陈 雁 主编



高等教育出版社

普通高等教育“十五”国家级规划教材

数据结构

（第二版）

陈　雁　主编

高等教育出版社

内容提要

本书是普通高等教育“十五”国家级规划教材。

本书主要内容包括：绪论、线性表和数组、栈和队列、树、图、排序、查找、数据结构程序设计等，最后以 Java 语言为例，介绍了面向对象程序设计的数据结构。在每章后均附有习题及上机实习题，以便学生巩固所学知识。书后配有光盘，其中包含一些过程的演示，帮助学生理解重点、难点内容。

本书适合于高等职业学校、高等专科学校、成人高校、本科院校举办的二级职业技术学院，也可供示范性软件职业技术学院、继续教育学院、民办高校、技能型紧缺人才培养使用，还可供本科院校、计算机专业人员和爱好者参加使用。

图书在版编目(CIP)数据

数据结构 / 陈雁主编. —2 版. —北京：高等教育出版社，2004.11

ISBN 7-04-015741-1

Ⅰ. 数… Ⅱ. 陈… Ⅲ. 数据结构-高等学校-教材 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2004）第 107360 号

策划编辑 冯 英 **责任编辑** 俞丽莎 **封面设计** 王凌波 **责任绘图** 朱 静
版式设计 胡志萍 **责任校对** 康晓燕 **责任印制** 孔 源

出版发行	高等教育出版社	购书热线	010-64054588
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
经 销	新华书店北京发行所		
印 刷	北京乾洋印刷有限公司		
		版 次	2001 年 9 月 第 1 版
开 本	787×1092 1/16		2004 年11月 第 2 版
印 张	11.75	印 次	2004 年11月 第 1 次印刷
字 数	270 000	定 价	23.20 元（含光盘）

物料号：15741-00

出版说明

为加强高职高专教育的建材建设工作，2000年教育部高等教育司颁发了《关于加强高职高专教育教材建设的若干意见》（教高司[2000]19号），提出了“力争经过5年的努力，编写、出版500本左右高职高专教育规划教材”的目标，并将高职高专教育规划教材的建设工作分为两步实施：先用2至3年时间，在继承原有教材建设成果的基础上，充分汲取近年来高职高专院校在探索培养高等技术应用型专门人才和教材建设方面取得的成功经验，解决好高职高专教育教材的有无问题；然后，再用2至3年的时间，在实施《新世纪高职高专教育人才培养模式和教学内容体系改革与建设项目计划》立项研究的基础上，推出一批特色鲜明的高质量的高职高专教育教材。根据这一精神，有关院校和出版社从2000年秋季开始，积极组织编写和出版了一批“教育部高职高专规划教材”。这些高职高专规划教材是依据1999年教育部组织制定的《高职高专教育基础课程教学基本要求》（草案）和《高职高专教育专业人才培养目标及规格》（草案）编写的，随着这些教材的陆续出版，基本上解决了高职高专教材的有无问题，完成了教育部高职高专规划教材建设工作的第一步。

2002年教育部确定了普通高等教育“十五”国家级教材规划选题，将高职高专教育规划教材纳入其中。“十五”国家级规划教材的建设将以“实施精品战略，抓好重点规划”为指导方针，重点抓好公共基础课、专业基础课和专业主干课教材的建设，特别要注意选择一部分原来基础较好的优秀教材进行修订使其逐步形成精品教材；同时还要扩大教材品种，实现教材系列配套，并处理好教材的统一性与多样化、基本教材与辅助教材、文字教材与软件教材的关系，在此基础上形成特色鲜明、一纲多本、优化配套的高职高专教育教材体系。

普通高等教育“十五”国家级规划教材（高职高专教育）适用于高等职业学校，高等专科学校、成人高校及本科院校举办的二级职业技术学院、继续教育学院和民办高校使用。

教育部高等教育司

2002年11月30日

第二版前言

本书第一版是教育部高职高专规划教材，于2001年初出版，前后共印刷了5次。广大读者已对该教材予以充分的认可，同时也提出了许多宝贵的修改意见。近年来，计算机专业的教学改革也对本课程提出了新的要求。此次新版，我们依据教育部高职高专精品教材、立体化教材要求，同时吸取了读者意见和教改经验，对原书的内容和教辅材料进行了修改、调整和充实。

本书力求继续保持高职高专的特点，对原教材的主要内容、框架结构未作大的变动。主要内容包括数据结构的基本概念、线性表和数组、栈、队列、树、图、排序、查找等，最后介绍了一些实用算法和面向对象程序设计中数据结构的描述方法，以便学生进一步学习和提高。本书中程序部分主要采用C语言编写。与第一版相比，我们所作的修改，主要是以下几个方面：

（1）本书对文字进行了重新润色、对内容进行了局部调整。对第5、6、7、9等4个章节修改的内容较多，同时对各章节的例子、习题进行了调整、增加，还注重了与计算机水平考试、各种资格论证相结合。

（2）重新制作了学生用盘。光盘内容主要包括每章的要点和主要内容，以及实例的动画演示和模拟试卷等。

（3）增加了教师用盘。其主要包括电子教案、教师指导手册、素材库、习题答案等。

（4）第9章中面向对象的程序设计语言原用C++语言，现考虑到要与国际接轨，所以改为用Java语言实现，并增加了较多用Java实现数据结构的实例。

本教材第1章～第5章由顾小晶撰写，第6章～第7章由陈雁撰写，第8章～第9章由曾海撰写，带“*”章节为可选章节，读者可根据情况自行选择；本书附带的光盘由顾成喜制作；教师用盘主要由顾小晶完成。全书由陈雁统稿。

北京联合大学鲍有文教授在百忙中抽时间担任了主审并提供了许多宝贵的修改建议，同时作者也得到了所在单位苏州市职业大学的领导和同行的支持，特别是过怡老师做了大量的资料收集等工作，在此一并表示衷心的感谢。

由于编者的水平，本书难免有不足之处，恳请广大读者批评指正。

作　者

2004年8月于苏州

第一版前言

本书是教育部高职高专规划教材，是依据教育部《高职高专教育数据结构课程教学基本要求》编写的。本书系统地介绍了各种数据结构的特点、存储结构和有关算法，并采用流行的C语言描述算法。主要内容包括数据结构的基本概念、线性表、栈、队列、树、图、查找、排序等，最后部分介绍了一些实用算法和面向对象程序设计中数据结构的描述方法，以便学生进一步学习和提高。

本教材力求体现高职高专的特点，本着理论够用，注重实用的原则，对传统的数据结构课程的教学内容进行了调整，对一些不常用的数据结构和算法进行了简化和忽略；对常用的数据结构和算法除进行详细介绍外，还在主要章节增加了应用实例，引导学生在理解基本内容的基础上，学习数据结构的使用。为便于教学，每章后面配有习题和上机实习题，并有专门章节进行数据结构程序设计指导，介绍算法书写、上机实习规范、程序调试技巧、实习报告整理等，还专门配有演示主要算法和应用实例的软件。全书概念表述清楚、简洁，内容由浅入深，强调实践环节，利于教学与自学。

本书可作为高等职业学校、高等专科学校、成人高等学校及本科院校举办的二级职业技术学院计算机类专业或信息类相关专业的教材，也可作为非计算机专业学生选修课或辅修课的教材，还可作为计算机应用人员和工程技术人员的自学参考书。

本教材第1章至第5章由顾小晶撰写；第6、7章由陈雁撰写；第8、9章由曾海撰写；本书附带的光盘由顾成喜制作。全书由陈雁统稿。

南京大学陈佩佩副教授在百忙中抽时间担任主审并提出了许多宝贵的修改建议，高等教育出版社给予了热情的帮助与指导。编写过程中，作者得到了所在单位苏州市职业大学的领导和同行的支持。在此一并表示衷心的感谢。

由于编者水平有限，书中难免有一些不足之处，请读者批评指正。

作　者

2001年3月于苏州

目　录

第1章 绪 论

计算机科学是一门研究信息表示、组织和处理的科学，而信息的表示和组织直接关系到处理信息时的效率。随着计算机产业的迅速发展和计算机应用领域的不断扩大，计算机应用已不仅仅限于早期的科学计算，而是更多地用于控制、管理和数据处理等方面，所以，随之而来的便是处理的数据量越来越大，数据类型越来越多，数据结构越来越复杂。因此，针对实际问题，例如要编制一个高效率的处理程序，那么就需要解决如何合理地组织数据，建立合适的数据结构，设计好的算法，来提高程序执行的效率这样的问题。“数据结构”这门学科就是在此背景下逐步形成和发展起来的。

本章将引出数据、数据结构和算法等基本概念以及介绍评价算法的一般方法。

1.1 数据结构的基本概念

1.1.1 引言

在程序设计课程的学习中，通常用计算机解题的基本方法是：分析问题，确定数据模型；设计相应的算法；编写程序；反复调试程序直至得到正确的结果。有些问题的数据模型可以用具体的代数方程、矩阵等来表示。然而，更多的实际问题是无法用数学方程来表示的。下面给出三个简单的例子加以说明。

表 1-1-1 是一个学生基本情况表。表中有 30 个记录，按学号顺序排列，它们之间存在一对一的关系，这是一种线性结构，其主要操作有查找、修改、插入或删除等。

表 1-1-1 学生基本情况表

学 号	姓 名	性 别	班 级	…
9905001	李力	男	99101	…
9905002	杜军	男	99101	…
9905003	程霄寒	女	99101	…
⋮	⋮	⋮	⋮	…
⋮	⋮	⋮	⋮	…
⋮	⋮	⋮	⋮	…
9905030	方勇	男	99101	…

图 1-1-1 表示的是某高校的专业设置情况。在图 1-1-1 中可以把一所高校名称看成树根，把下设的若干个系名看成它的树枝中间结点，把每个系的若干个专业方向看成树叶，这就形成一个树形结构。树形结构通常用来表示结点的分层组织，结点之间是一对多的关系。对于树形结构的主要操作是遍历、查找、插入或删除等。

图 1-1-2 是一个描述若干个城镇之间的公路网。图中每个顶点代表一个城镇，边表示城镇

之间的道路，图 1-1-2 中各个顶点和边的关系是一种多对多的关系。通常把具有这种关系的结构称之为图形结构。例如要从某个原料产地把原料运往各加工地，就需要制定一个用图形结构来描述的运输方案使得运输费用最省。

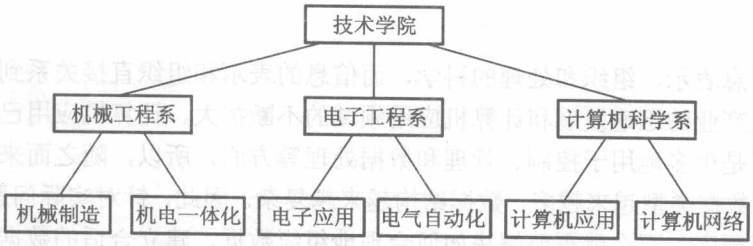


图 1-1-1 树形结构示例

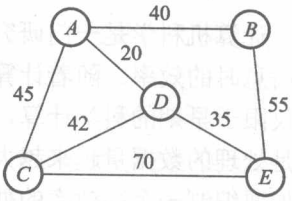


图 1-1-2 图形结构示例

类似的还有图书档案类问题、棋类对弈问题、通信网络问题等。对于这些非数值问题的描述都可归纳为上述的表、树和图之类的数据结构，这类数据结构中元素和元素之间都存在着相互的关系。因此，数据结构是一门研究非数值计算的程序设计中，计算机的操作对象以及它们之间的关系和操作等的学科。

1968 年，美国的 D.E.Kunth 教授开创了数据结构的最初体系,他的名著《计算机程序设计技巧》较为系统地阐述了数据的逻辑结构和存储结构及其操作。随着计算机科学的飞速发展和应用领域的不断扩大，到 20 世纪 80 年代初期，数据结构的基础研究日臻成熟，已成为一门完整的学科。

“数据结构”是计算机专业的一门专业基础课。它为操作系统、数据库原理、编译原理等后续专业课程的学习奠定了基础。数据结构涉及各方面的知识，如计算机硬件范围内的存储装置和存取方法；计算机软件范围内的文件系统、数据的动态存储与管理、信息检索；数学范围内的算法知识；还有一些综合性的知识，如编码理论、算子关系、数据类型、数据表示、数据运算、数据存取等。因此，数据结构是数学、计算机硬件、计算机软件专业的一门核心课程。

1.1.2 数据结构有关概念及术语

数据（Data）是描述客观事物的数字、字符以及所有能输入计算机中并被计算机程序处理的符号的集合。计算机输入和输出的数据除了数字以外，还有字符串，即用英文、汉字或其他符号组成的词组、语句以及表示图形、声音、光和电的符号等。

数据元素（Data Element）是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。例如，图 1-1-1 树形结构示例中的一个专业就是一个数据元素。数据元素可以由一个或多个数据项组成，例如，表 1-1-1 中每个学生的信息就可作为一个数据元素，每个数据元素由学号、姓名、性别、出生日期等数据项组成，数据项（Data Item）是具有独立含义的数据的最小单位。

数据对象（Data Object）是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合{0，±1，±2，…}，字母字符数据对象是集合{'A'，'B'，…，'Z'}。

数据结构（Data Structure） 是指相互之间存在着一种或多种关系的数据元素的集合。严格地讲，数据结构应包括两方面的内容：数据的逻辑结构和物理结构。数据的逻辑结构是指各数

据元素之间的逻辑关系，是用户按使用需要建立起来的数据的组织形式，是独立于计算机的。例如，前文中表 1-1-1 和图 1-1-1 表示的就是数据的逻辑结构。数据的物理结构又称为数据的存储结构，是指数据的逻辑结构在计算机内的表示方法，即存储形式。对机器语言来说，这种存储形式是具体的，但在高级语言的层次上，就可以借助于它的数据类型来描述这种存储形式的实现细节。常见的存储结构有顺序存储结构和链式存储结构。

数据结构课程主要研究数据的逻辑结构、相应的存储结构以及定义在它们之上的一组运算，并要求设计出相应的算法，同时还必须分析执行算法时的时空效率。

数据类型（Data Type）是程序设计语言中所允许使用的变量种类。一个数据类型不仅定义了相应变量可以设定的值的集合，而且还规定了对变量允许进行的一组运算及其规则。例如，C 语言中的整型（int），它可以设定的值的集合是[–32 768，32 767]，主要运算有+、–、*、/、%（取模运算）等。从硬件的角度看，它们的实现涉及“位”、“字节”、“字”、“位的运算”等。但对用户来说，只需了解整数的各种运算的抽象特性，而不必了解计算机实现这些运算的细节，这样就可使用高级语言进行程序设计。所以，通常可以把数据类型看作是程序设计语言中已经实现了的数据结构。

抽象数据类型（Abstract Data Type，ADT）是指基于一类逻辑关系的数据类型以及定义在这个类型上的一组操作。在软件设计中，抽象数据类型通常包含定义、表示和实现三部分。

抽象数据类型的定义可以看作是描述问题的模型，它独立于具体实现。例如，一个线性表的抽象数据类型可定义如下：

ADT Linear_List{
数据元素　所有 a_i 属于同一数据对象，i=1,2,…,n　（n≥0）
逻辑结构　所有数据元素 a_i 存在次序关系（a_i,a_{i+1}）,a_1 无前驱，a_n 无后继
基本操作　设 L 为 Linear_List 类型的线性表
　　InitList（L）；　建立一个空的线性表 L；
　　Length（L）；　求线性表 L 的长度；
　　GetElem（L,i）；取线性表 L 中的第 i 个元素；
　　Locate（L,x）；　确定元素 x 在线性表 L 中位置；
　　Insert（L,i,x）；　在线性表 L 中第 i 个元素之前　（或之后）插入一个新元素 x；
　　Delete（L,i）；　删除线性表 L 中的第 i 个元素；
}ADT Linear_List

抽象数据类型的定义是软件设计者之间的接口。在软件设计中凡是需要使用抽象数据类型的地方，就可以方便地根据抽象数据类型的定义来使用它。下面是一个利用线性表的抽象数据类型实现其他算法的例子。

例 1.1　已知两个集合 *A* 和 *B*，现要求一个新的集合 *A*=*A*–*B*。

分析：假设用线性表 *LA* 和 *LB* 分别表示集合 *A* 和 *B*，只要将同在 *LA* 和 *LB* 中的元素从 *LA* 中删除即可。算法描述如下：

```
void  Difference(Linear_List LA, Linear_List LB){
  n=Length(LB);
  for(i=1;i<=n;i++){
```

```
    x=GetElem(LB,i);
    k=Locate(LA,x);
    if(k!=0) Delete(LA,k); /* x在LA和LB中 */
  }
  }/* Difference*/
```

在这个例子中，尚未涉及线性表和它的操作在工具语言层的具体实现，而是仅通过线性表的抽象数据类型的操作 Length、GetElem、Locate 和 Delete 完成了求集合 *A–B* 的算法。可见凭借抽象数据类型的支持，可以有效地研究问题、设计算法和分析算法。

抽象数据类型的表示和实现，可以通过某种程序设计语言的功能来完成，即利用程序设计语言中已有的基本数据类型或用户已定义的数据类型来说明新的数据类型（存储结构），组合新的操作。其具体方法可以采用传统的面向过程的程序设计方法，也就是根据数据的逻辑结构选定合适的存储结构，根据所定义的操作设计出相应的算法（函数）。这也是本书所采用的方法。另一种是采用面向对象的程序设计方法（Object_ Oriented Programming，OOP），在面向对象的程序设计语言中，存储结构的说明和操作函数的说明被封装在一个整体结构中，这个整体结构称之为“类”（Class），属于某个“类”的具体变量称之为“对象”（Object）。显然，OOP 与 ADT 的实现更加接近一致。在本书的第 9 章将对 OOP 作进一步的介绍。

抽象数据类型设计是软件设计的基础。在数据结构课程中讨论的各种基本数据结构，如线性表、栈、队列等，都可以作为设计更复杂软件的抽象数据类型。当应用程序中要用到的各种基本数据结构都已有设计好的、可重复使用的抽象数据类型时，以后的程序设计和程序维护将大大简化。

1.2 算法和算法描述

1.2.1 什么是算法

算法（A1gorithm）是对特定问题求解步骤的一种描述，它是指令或语句的有限序列。一个算法一般具有下列 5 个重要特性：

① 有穷性：一个算法必须总是在执行有穷步之后结束，即必须在有限时间内完成。

② 确定性：算法中的每一条指令必须有确切的含义，不能产生多义性。

③ 可行性：算法中的每一条指令必须是切实可行的，即原则上是可以通过已经实现的基本运算执行有限次来实现的。

④ 输入：一个算法有零个或多个输入，这些输入取自于特定对象的集合。

⑤ 输出：一个算法有一个或多个输出，这些输出是同输入有某个特定关系的量。

在计算机科学研究中，算法与数据结构是相辅相成的。解决某一特定类型问题可以选定不同的数据结构，而且选择恰当与否直接影响算法的效率。反之,一种数据结构的优劣要由实现各种操作时的算法来体现。

1.2.2 算法描述工具——类 C 语言

算法需要用一种语言来描述。为了便于理解和掌握算法的思想和实质，本教材采用类 C 语

言进行算法描述。类 C 语言实际上是对 C 语言的一种简化，保留了 C 语言的精华，忽略了 C 语言语法规则中的一些细节，这样描述出的算法清晰、直观、便于阅读和分析。

本书在用类 C 语言描述算法时作如下的约定：

① 问题的规模用 MAXSIZE 表示。具体尺寸可以由用户预先定义，例如

```
#define  MAXSIZE  l00
```

② 数据元素的类型约定为 ElemType。具体的类型可以由用户在使用时定义，例如

```
typedef  int  ElemType;
```

③ 数据的存储结构用类型定义（typedef）描述，在书写算法之前进行说明。

④ 算法以函数形式描述：

```
类型标识符  函数名（形式参数表）
  /* 算法说明 */
  {语句序列}
```

除了形参类型需要说明外，算法中其他变量的类型说明可省略不写，这样使算法的处理过程更加突出、明了。算法说明是一个完整算法中不可缺少的部分，它可以包括：算法的功能、数据的存储结构、形式参数的含义和输入输出属性等。

下面是书写算法的一般步骤。

设某班级有N个学生，现要求设计一算法将他们某一门课程的成绩按由高到低的顺序排列，并且按顺序输出他们的成绩和相应的学号。

分析：选用一维数组用来存储这 N 个学生的成绩和学号。每个数组元素有两个域：一个是学号域，一个是成绩域。

```
typedef struct{
   int  no;    /*学号域*/
   int  score;     /*成绩域*/
 }ElemType;
```

算法描述参见算法 1.1。

```
算法 1.1
void sort( ElemType  S[MAXSIZE], int n) {
 /* 对数组 S 中的 n 个数据按由大到小的顺序排序并输出,n<MAXSIZE */
    for (i=1;i<n;i++)
    for(j=i; j<=n; j++)
         if( S[i]. score<S[j]. score)
           { t=S[i]; S[i]=S[j]; S[j]=t; }
    for(i=1; i<=n;  i++)
  printf("%8d%8d%8d\n", i, S[i].no, S[i]. score);
}/*sort*/
```

1.3 算法评价

对于一个给定问题的求解，往往可以设计出若干个算法。那么如何评价这些算法的优劣呢？正确性是评价一个算法的首要条件。一个正确的算法是指在合法的数据输入下，能在有限

的运行时间内，得出正确的结果。此外主要考虑执行算法所耗费的时间和执行算法所占用的存储空间。

1.3.1 时间

一个算法的执行时间等于其所有语句执行时间的总和，而任一语句的执行时间为该语句的执行次数（也称语句频度 Frequency Count）与该语句执行一次所需时间的乘积。但当算法转换为程序之后，每条语句的执行时间取决于机器的硬件速度、指令类型以及编译所产生的代码质量，而这些因素是很难确定的。因此，通常只是将算法中基本操作重复执行的次数作为算法执行时间的量度。

算法中基本操作重复执行的次数根据算法中最大语句频度来估算，它是问题规模 n 的某个函数 $f(n)$，算法的时间量度记作 $T(n)=O(f(n))$，表示随问题规模 n 的增大，算法执行时间的增长率和 $f(n)$的增长率相同，称作算法的渐近时间复杂度(Asymptotic Time Complexity)，简称时间复杂度。时间复杂度往往不是精确的执行次数，而是估算的数量级，它着重体现的是随着问题规模 n 的增大，算法执行时间的变化趋势。

例如，在下列三个程序段中：

（a）x=x+1;

（b）for(i=1;i<=n;i++) x=x+1;

（c）for(j=1;j<=n;j++)

　　for(k=1;k<=n;k++) x=x+1;

语句 x=x+1;的频度分别为 1，n 和 n^2，则程序段(a)的时间复杂度可记为 $O(1)$；在程序段(b)中，因为赋值语句在 for 循环之中，所以要执行 n 次，其执行时间和 n 成正比，时间复杂度应记为 $O(n)$；在程序段(c)中，赋值语句要执行 n^2 次，其执行时间和 n^2 成正比，则时间复杂度应记为 $O(n^2)$。

现在来分析算法 1.1 的时间复杂度。算法中有一个二重循环，if 语句的执行频度为：

$$n+(n-1)+(n-2)+\cdots+3+2+1=n(n+1)/2$$

它的数量级为 $O(n^2)$。算法中输出语句的语句频度为 n，数量级为 $O(n)$。该算法的时间复杂度以 if 语句执行频度来估算(忽略输出部分)，则记为 $O(n^2)$。通常将这些时间复杂度分别称为常量阶、线性阶和平方阶，算法还可能呈现的时间复杂度有指数阶等。不同数量级时间复杂度的形状如图 1-3-1 所示。

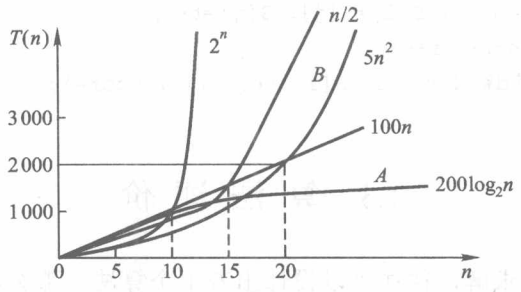


图 1-3-1 各种数量级的时间复杂度

从图 1-3-1 中可见，随着问题规模的增大，其时间消耗也在增大，但它们的增长趋势明显不同。如果对于一个问题设计两种不同算法，算法 A 的时间复杂度为 $O(\log_2 n)$，算法 B 的时间复杂度为 $O(n^2)$，由图 1-3-1 可知，随着问题规模 n 的增大，算法 B 所消耗时间迅速增大，而算法 A 增大趋向平缓，显然，算法 A 运行速度较快，可以认为算法 A 优于算法 B。

1.3.2 空间

空间是指执行算法所需要的存储空间，包括算法本身所占用的存储空间、输入数据占用的存储空间以及算法在运行过程中的工作单元和实现算法所需的辅助空间。

空间复杂度（Space Complexity）可类似于时间复杂度的讨论。它与问题规模 n 的函数关系表示为：$S(n) = O(f(n))$。

习　题

1．简述下列术语：

数据元素　数据　数据对象　数据结构　存储结构　算法

2．举出一个数据结构的例子，叙述其逻辑结构、存储结构和运算三方面的内容。

3．试写一算法，由大至小依次输出顺序读入的三个整数 x，y 和 z 的值。

4．设某班级有 n 个学生，要求设计以下算法：（1）输入全班学生的学号和某一门课程的考试成绩；（2）将该课程的成绩按由高到低的顺序排列；（3）求出平均成绩和不及格学生的人数。最后设计 main()，要求调用以上算法后按从高到低的次序输出学生的学号和相应的成绩，输出平均成绩和不及格学生的人数。

5．分析下列算法的时间复杂度。

```
(1) int prime(int n)
   /*  判断 n 是否为素数，n≥0 */
   { for(i=2;i<sqrt(n);i++)
       if( n % i==0) return 0;
     return 1;
   }
(2) long sun(int n)
   /*  n 为一个正整数 */
  { s=0;
     for(i=1; i<=n;i++){
       for( p=1,j=1;j<i;j++)
           p=p*j;
       s+=p;
      }
    return s;
  }
```

第 2 章　线性表和数组

在本章和下一章中将主要讨论线性结构。线性结构的特点是数据元素之间的关系是线性的。数据元素可以看成是排列在一条线或一个环上。线性表（Linear_List）是最简单且最常用的一种数据结构。本章讨论线性表的逻辑结构、存储结构以及相关的操作。数组在某种意义上也可看作是一种线性结构，因此在本章一起讨论。

2.1　线性表的逻辑结构

2.1.1　线性表的定义

线性表是 n ($n \geqslant i \geqslant 0$)个数据元素组成的有限序列。一般记作：

$$L=(a_1,\ a_2,\ \cdots,\ a_i,\ \cdots,\ a_n)$$

表中的元素可以是一个数，一个符号或由多个数据项组成的复杂信息，但同一线性表中的元素必须属于同一数据对象。例如，英文字母表（A，B，C，D，…，X，Y，Z）和表 2-1-1 的学生登记表都是线性表。

表 2-1-1　学生登记表

学　号	姓　名	性　别	年　龄	班　级	…
9905010	程晓珉	女	18	计 99	…
9905012	方正飞	男	19	计 99	…
9905013	刘津津	男	19	计 99	…
…	…	…	…	…	…

线性表的逻辑结构是通过元素之间的相邻关系体现的：a_1 为开始结点，a_n 为终端结点；a_{i-1} 为 a_i 的直接前驱结点（Predecessor）（$2 \leqslant i \leqslant n$）；$a_{i+1}$ 为 a_i 的直接后继结点（Successor）（$1 \leqslant i \leqslant n-1$）。线性表中元素的个数（$n$）称为该表的长度。长度为零（$n$=0）的表称为空表。

2.1.2　线性表的基本操作

线性表是一种灵活的数据结构，可在任意位置上进行插入和删除，其表长也可根据不同的操作增长或缩短。对于线性表有以下几种常用的基本操作：

① InitList(L);　　建立一个空的线性表 L；

② GetElem(L,i);　取线性表 L 中的第 i 个元素；

③ Length(L);　　求线性表 L 的长度；

④ Locate(L,x);　确定元素 x 在线性表 L 中位置；

⑤ Insert(L,i,x);　在线性表 L 中第 i 个元素之前（或之后）插入一个新元素 x；

⑥ Delete(L,i);　删除线性表 L 中的第 i 个元素；