普通高等教育“十一五”国家级规划教材

高职高专计算机系列规划教材

数据结构与实训

张红霞　白桂梅　主编

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育“十一五”国家级规划教材
高职高专计算机系列规划教材

数据结构与实训

张红霞　白桂梅　主编

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书为普通高等教育“十一五”国家级规划教材。全书系统地介绍了数据结构的主要内容，全书共分8章及附录（实训指南）。第1章介绍了数据结构和算法的基本概念，第2、3、4章介绍了线性表、堆栈和队列、串和数组等常用的线性结构，第5、6章介绍了树形结构和图形结构，第7、8章介绍了查找的常用算法和两个基本技术排序。附录中介绍了实训的相关知识，包括实训的步骤、实训报告规范和实训的环境。本书对每一种数据结构都详细阐述了基本概念、各种不同的存储结构及在不同存储结构上主要算法的实现，并给出很多典型例题，以帮助读者理解。

数据结构是一门实践性很强的课程，本书很注重理论与实践相结合，每章都由浅入深，循序渐进地给出了典型的例题、实训例题，以及与之相配套的、精心挑选的、难易搭配的习题和实训习题。通过习题与实训，使学生掌握所学知识，并能灵活运用所学知识解决实际问题。

本书叙述精练，概念清楚，注重实用，可作为高职高专院校计算机专业及相关专业数据结构课程的教材，也可供从事计算机应用开发的工程技术人员参考使用。

图书在版编目（CIP）数据.

数据结构与实训/张红霞，白桂梅主编. —北京：电子工业出版社，2008.4
普通高等教育“十一五”国家级规划教材. 高职高专计算机系列规划教材
ISBN 978-7-121-06163-9

Ⅰ. 数… Ⅱ. ①张… ②白… Ⅲ. 数据结构—高等学校：技术学校—教材 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2008）第 030137 号

策划编辑：吕 迈
责任编辑：徐 磊
印 刷：
装 订：北京市李史山胶印厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：18 字数：461 千字
印 次：2008 年 4 月第 1 次印刷
印 数：4 000 册 定价：26.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系电话及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

本书为普通高等教育 “十一五” 国家级规划教材。

写作背景

计算机技术的应用领域早已从数值领域拓广到非数值领域，程序所处理的数据也越来越复杂，程序设计的实质就是对确定的问题选择一种好的结构，加上设计一种好的算法。数据结构就是一门研究用计算机解决一系列问题特别是非数值信息处理问题时所用的各种数据的组织方法、数据的存储方法及在各种数据结构上执行操作的算法的学科。数据结构是计算机学科的核心专业基础课程，是计算机程序设计的重要理论和实践基础，在整个计算机专业教学体系中处于举足轻重的地位。

目前，数据结构的书、习题解析教材较多，但多是理论与实践分开的模式，且例题偏少，专门的数据结构与实训结合在一起的教材几乎没有。这本教材正好弥补这一空缺，希望通过理论与典型的习题、实训题目的讲解，使学生把理论和实际结合起来，给学生一些解题的思路和启发，帮助学生更好地理解和掌握课程内容，理解和掌握算法设计所需的方法和技术，为后续课程打下良好的基础。

教材特点

本书是根据作者多年教学实践经验，结合当前计算机技术的发展，参考近年的多本教材精心编写而成的，所选内容覆盖了数据结构的主要内容。全书共分 8 章及附录（实训指南）。第 1 章介绍了数据结构和算法的基本概念，第 2 至第 4 章介绍了线性表、堆栈、队列、串、数组等常用的线性结构，第 5、6 章介绍了树形结构和图形结构，第 7、8 章介绍了查找的常用算法和两个基本技术排序，附录中介绍了实训的相关知识，包括实训的步骤、实训报告规范、实训的环境。对每一种数据结构都详细阐述了基本概念、各种不同的存储结构及在不同存储结构上主要算法的实现，并给出很多典型例题，以帮助读者理解。

由于该课程内容丰富，技术性与实践性强，学习量大，学生学习起来有一定困难。学生对基本知识还可以看懂听明白，但一到自己解答习题尤其是算法设计题时，觉得无从下手，做起来特别费劲，和教学内容不能很好地衔接。因此，学生必须在掌握理论知识的同时，加强上机实训才能学好这门课程。上机实训对学生是一种全面综合训练，是与课堂听讲、课下复习和作业练习相辅相成的不可缺少的一个教学环节。为了帮助学生更好地学习本课程，理解和掌握算法设计所需的技术，为整个专业学习打好基础。本书很注重理论与实践相结合，突出应用，每章都由浅入深，循序渐进地给出了典型的例题、实训例题，以及与之相配套的精心挑选、难易搭配的习题和实训习题，在内容的编排方面尽量选取经典实例，力求新颖，吸引学生的兴趣；对每个具体的实训题目要求学生给出完整的问题描述、数据结构描述、算法描述、程序源代码及调试分析结果。通常，实训题中的问题比平时的习题复杂一些，也更接近于实际，可进一步培养学生的动手能力；另一方面，能使书上的知识变“活”，起到深化理解和掌握教学内容的目的。通过实训内容的训练，培养学生的数据抽象能力，如何把现实世界的客观问题转换为在计算机内的表示形式，学会分析研究计算机加工

的数据的结构特性，以便为应用涉及的数据选择适当的逻辑结构、存储结构及相应的算法，突出构造性思维训练的特征，提高学生组织数据、编写高质量的应用程序的能力，并为后续课程的学习打下良好的理论基础和实践基础。

学习方法指导

本课程的研究对象是数据，它是一门理论性和逻辑性较强的课程，程序设计语言（如C语言）是本课程的一个非常重要的先修课程，程序设计语言是作为一个工具来说明本课程的一些相关思想和算法的，因而，程序设计语言掌握的熟练程度，直接影响到本课程的学习。

本课程的学习主要分为两大块知识的学习，即数据结构和算法，其中数据结构包括线性结构、树形结构和图形结构；算法包括查找与排序。对不同的知识模块，在学习上应该有不同的学习方法。

1．数据结构的学习

线性结构、树形结构和图形结构是数据与数据之间关系的3种体现方式，计算机要对数据进行处理，首先必须搞清楚它所处理的对象之间的相互关系，然后对不同关系的数据进行不同的存储和处理。因而，这一块知识是本课程的重点知识。它的学习要点是：

（1）每一种数据结构的逻辑描述；

（2）每一种数据结构的存储形式，其中包括顺序存储与链式存储两种形式的结构描述；

（3）每一种数据结构的基本算法。

2．算法的学习

算法是计算机处理数据的操作步骤的描述，算法的好坏直接影响到程序的执行效率，尤其是查找与排序算法，是在实践中使用频率非常高的两类算法，因而对这两类算法进行分析和研究是非常有必要的。它的学习要点是：

（1）每一种算法的思想；

（2）每一种算法的实现；

（3）每一种算法的评价。

本课程的重、难点

重点：本课程的学习重点是数据的逻辑结构与存储结构；线性表、堆栈和队列的基本运算及典型应用；树的存储表示，二叉树的遍历及其应用；图的存储，图的遍历；基本的查找技术，内排序技术及各种排序技术的比较，基本的算法分析方法。

难点：本课程的难点是二叉树的非递归遍历；线索二叉树；哈夫曼树及其应用；图的存储及应用；哈希查找；堆排序与基数排序。

本书讲授学时数为60学时左右，实训学时数为20学时以上。教师可根据学时数和学生的实际情况选讲本书的例子。

本书由张红霞、白桂梅任主编。书中第2、3、4章由白桂梅编写，第1、6章由王丽伟编写，第5、7章、附录由张红霞编写，第8章由王勤编写。张红霞审阅第1、2、3、4章，白桂梅审阅第5、6、7、8章，全书由张红霞统稿。

由于编者水平有限，虽然在编写过程中不遗余力，但书中疏漏和错误之处在所难免，恳请广大同行和读者不吝指正。

编　者

2007-10-11

目　录

第1章 概　　论

《数据结构》是计算机专业的核心课程，它研究的内容、使用的术语，以及算法的度量标准将是本章所要介绍的。

1.1 引言

1.1.1 什么是数据结构

数据结构包含两方面的内容，其一是构成集合的数据元素，其二是数据元素之间存在的关系。数据结构也就是带有结构的数据元素的集合，结构指的是数据元素之间的相互关系，即数据的组织形式。由此可见，计算机所处理的数据并不是数据的杂乱堆积，而是具有内在联系的数据集合。如表 1-1 所示的成绩表就是一个数据结构（线性表），其中每一行表示一个数据元素，表中的所有行构成了一个数据元素集合，数据元素之间具有线性结构关系（详见第 2 章）。如图 1-1 所示是一软件公司员工职务组织结构图，其中每一个方框表示一位员工的信息（如职工号、姓名、性别、年龄和电话等），即一个数据元素，全部员工的信息构成了一个数据元素集合，数据元素之间具有树形结构关系（详见第 5 章）。如图 1-2 所示是城市交通示意图，其中每一个顶点表示一座城市，即一个数据元素，全部顶点构成了一个数据元素集合，每一条边表示两座城市间的交通线路，之间的距离用边上的值表示，数据元素之间具有图结构关系（详见第 6 章）。

表 1-1　成绩表

学　号	姓　名	性　别	数　学	数据结构	英　语	计算机组成原理
0504028	秦占军	男	88	76	78	86
0504029	武晓云	女	90	82	84	79
0504030	关国江	男	92	90	86	88
…						

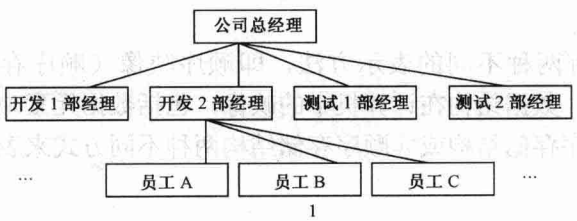

图 1-1　公司员工职务组织结构图

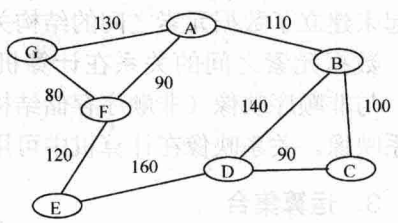

图 1-2　城市交通示意图

1.1.2 数据结构研究什么

数据结构可定义为一个二元组：Data_Structure=（D,R）

其中 D 是数据元素的有限集，R 是 D 上关系的有限集。

数据结构具体应包括 3 个方面：数据的逻辑结构、数据的物理结构和数据的运算集合。

1. 逻辑结构

数据的逻辑结构是指数据元素之间逻辑关系的描述。

根据数据元素之间关系的不同特性，通常有下列 4 种基本的逻辑结构，如图 1-3 所示。

（1）集合结构。结构中的数据元素之间除了同属于一个集合的关系外，无任何其他关系。

（2）线性结构。结构中的数据元素之间存在着一对一的线性关系。

（3）树形结构。结构中的数据元素之间存在着一对多的层次关系。

（4）图状结构或网状结构。结构中的数据元素之间存在着多对多的任意关系。

由于集合的关系非常松散，因此可以用其他的结构代替。本教材所讨论数据的逻辑结构如图 1-4 所示。

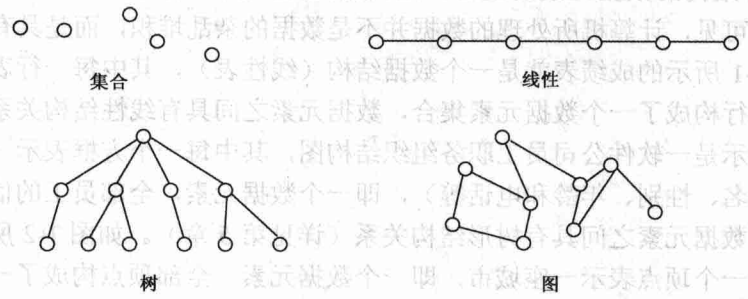

图 1-3　4 种基本逻辑结构

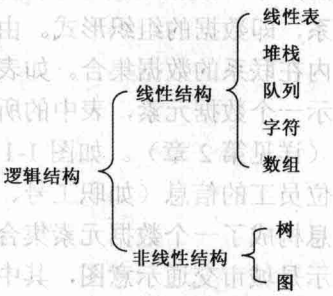

图 1-4　本教材所讨论的逻辑结构

2. 存储结构

存储结构（又称物理结构）是逻辑结构在计算机中的存储映像，是逻辑结构在计算机中的实现（或存储表示），它包括数据元素的表示和关系的表示。有数据结构 Data_Structure=（D,R），对于 D 中的每一个数据元素都对应有存储空间中的一个单元，D 中全部元素对应的存储空间必须明显或隐含地体现关系 R。逻辑结构与存储结构的关系为，存储结构是逻辑结构的映像与元素本身的映像。逻辑结构是抽象，存储结构是实现，两者综合起来建立了数据元素之间的结构关系。

数据元素之间的关系在计算机中有两种不同的表示方法，即顺序映像（顺序存储结构）与非顺序映像（非顺序存储结构）。数据结构在计算机中的映像，包括数据元素映像和关系映像。关系映像在计算机中可用顺序存储结构或非顺序存储结构两种不同方式来表示。

3. 运算集合

讨论数据结构的目的是为了在计算机中实现所需的操作，施加于数据元素之上的一组操作构成了数据的运算集合，因此在结构上的运算集合是数据结构很重要的组成部分。

以表 1-1 所示的成绩表为例，该表的数据元素与数据元素之间是一种简单的线性关系，所以逻辑结构采用线性表。存储结构既可采用顺序存储结构，也可采用非顺序存储结构。对

于成绩表，当学生退学或转出时要删除相应的数据元素，转入学生时要增加数据元素，发现成绩输入错误时要修改。这里的增加、删除和修改就构成了数据的操作集合。

综上所述，数据结构的内容可归纳为 3 个部分：逻辑结构、存储结构和运算集合。按某种逻辑关系组织起来的一批数据，依一定的映像方式把它存放在计算机存储器中，并在这些数据上定义一个运算的集合，就构成了一个数据结构。

《数据结构》是一门主要研究怎样合理地组织数据，建立合适的数据结构，提高计算机执行程序所用的时、空效率的学科。数据结构课程不仅讲授数据信息在计算机中的组织和表示方法，同时也训练用高效的设计算法解决复杂问题的能力。《数据结构》属于计算机专业的核心专业基础课，对于程序设计者来说对该学科的掌握是非常必要的，数据结构贯穿程序设计的始末，缺乏数据结构功底的人，很难设计出高水平的应用程序。

1.2 数据结构的基本概念

1. 数据

数据是描述客观事物的数值、字符，以及所有其他能输入到计算机中，且能被计算机处理的各种符号的集合。简言之，数据就是计算机化的信息（或存储在计算机中的信息）。

2. 数据元素与数据项

数据元素是组成数据的基本单位，是数据集合的个体，在计算机中通常作为一个整体进行考虑和处理。一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最小单位（不可再分割）。如表 1-1 所示的成绩表，每一个学生的信息（每一行）是一个数据元素，每一个数据元素包含学号、姓名等 7 个数据项。

3. 数据对象

数据对象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 N={0,±1,±2,…}，字母字符数据对象是集合 C={'A','B',…,'Z'}。本节中的表 1-1 中的成绩表也可看做一个数据对象，由此可以看出，不论数据元素集合是无限集（如整数集）、有限集（如字符集），还是由多个数据项组成的复合数据元素（如成绩表），只要性质相同，都是同一个数据对象。

4. 数据类型

数据类型是一组性质相同的值集合，以及定义在这个值集合上的一组操作的总称。数据类型中定义了两个集合，值集合确定了该类型的取值范围，操作集合确定了该类型中允许使用的一组运算。例如，高级语言中的数据类型就是已经实现的数据结构。从这个意义上讲，数据类型是高级语言中允许的变量种类，是程序语言中已经实现的数据结构（即程序中允许出现的数据形式）。例如，在高级语言中的整型类型，它可能的取值范围是–32768～+32767，可用的运算为加、减、乘、除、乘方和取模。

按“值”的不同特性，高级程序语言中的数据类型可分为两大类。一类是非结构的原子类型，它的值是不可分解的，如 C 语言中的标准类型（整型、实型和字符型）及指针；另一类是结构类型，它的值是由若干成分按某种结构组成的，是可以分解的，并且它的成分

可以是非结构的，也可以是结构的。例如，数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。

5. 抽象数据类型

抽象数据类型是指基于一类逻辑关系的数据类型。抽象数据类型的定义取决于客观存在的一组逻辑特性，而与其在计算机内如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部使用。在某种意义上讲，抽象数据类型和数据类型实质上是一个概念。整数类型就是一个简单的抽象数据类型。“抽象”的意义在于数学特性的抽象。一个抽象数据类型定义了一个数据对象、数据对象中各元素间的结构关系，以及一组处理数据的操作（运算）。

抽象数据类型包括定义和实现两方面，其中定义是独立于实现的。定义仅给出一个抽象数据类型的逻辑特性，不必考虑如何在计算机中实现。抽象数据类型的特征是定义与其实现分离，使用（引用）与实现分离，从而达到封装和信息隐蔽。

1.3 算法和算法的分析

1.3.1 算法及算法的描述

算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或多个操作。此外，一个算法还具有下列 5 个重要特性。

（1）有穷性。一个算法必须总是（对任何合法的输入值）在执行有穷步之后结束，且每一步都可在有穷时间内完成。

（2）确定性。算法中每一条指令必须有确切的含义，不能产生二义性。在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

（3）可行性。一个算法是可行的，即算法中描述的操作可通过已经实现的基本运算执行有限次来实现。

（4）输入。一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。

（5）输出。一个算法有一个或多个输出。这些输出是与输入有着一定关系的量。

《数据结构》中所讨论的算法，可用不同的方式进行描述，常用的有类 Pascal、类 C、类 C++、类 Java 程序设计语言，本教材以类 C 语言为描述工具。每一算法可用一个或多个简化了的 C 语言（即类 C）函数进行描述，简化是针对高级语言的语法细节所做的，如对函数内部的局部变量均不作声明而直接使用，交换两个变量 x、y 的值，不使用 3 条赋值语句，而仅简记为一条语句 x←→y；再如对结构体变量可以整体赋值，等等。需要注意的是，《数据结构》中的算法，因为用类 C 描述，所以不等同于 C 语言程序，若要上机运行某一算法，则必须将其完善为 C 语言程序。即增加 main( )函数，main( )函数中增添实现算法的函数调用语句，在实现算法的函数中补充、完善语法细节。

1.3.2 算法设计的要求

1. 正确性

正确性的含义是算法对于一切合法的输入数据都能够得出满足规格说明要求的结果，

事实上要验证算法的正确性是极为困难的，因为通常情况下合法的输入数据量太大，用穷举法逐一验证是不现实的。所谓的算法正确性是指算法达到了测试要求。

2. 可读性

可读性是指人对算法阅读理解的难易程度，可读性高的算法便于人之间的交流，有利于算法的调试与修改。

3. 健壮性

对于非法的输入数据，算法能给出相应的响应，而不是产生不可预料的后果。

4. 效率与低存储量需求

效率指的是算法的执行时间。对于解决同一问题的多个算法，执行时间短的算法效率高。存储量需求指算法执行过程中所需要的最大存储空间。效率与低存储量需求两者都与问题的规模有关，求 100 个人的平均分与求 1 000 个人的平均分显然不同。

1.3.3 算法的分析

1. 算法效率的度量

算法执行的时间是其对应的程序在计算机上运行所消耗的时间。程序在计算机上运行所需时间与下列因素有关：

（1）算法本身选用的策略；

（2）书写程序的语言；

（3）编译产生的机器代码质量；

（4）机器执行指令的速度；

（5）问题的规模。

度量一个算法的效率应抛开具体机器的软、硬件环境，而书写程序的语言、编译产生的机器代码质量、机器执行指令的速度都属于软、硬件环境。对于一个特定算法只考虑算法本身的效率，而算法自身的执行效率是问题规模的函数。对同一个问题，选用不同的策略就对应不同的算法，不同的算法对应有各自的问题规模函数，根据这些函数就可以比较（解决同一个问题的）不同算法的优劣。

2. 算法的时间复杂度

一个算法的执行时间大致上等于其所有语句执行时间的总和，语句的执行时间是指该条语句的执行次数和执行一次所需时间的乘积。语句执行一次实际所需的具体时间是与机器的软、硬件环境（机器速度、编译程序质量和输入数据等）密切相关的，与算法设计的好坏无关。所以，可用算法中语句的执行次数来度量一个算法的效率。

首先定义算法中一条语句的语句频度，语句频度是指语句在一个算法中重复执行的次数。以下给出了两个 n×n 阶矩阵相乘算法中的各条语句，以及每条语句的语句频度。

语句	语句频度
for（i=0；i<n;i++）	n+1
for （j=0；j<n;j++）	n^2+n

```
{c[i][j]=0;                          n^2
  for (k=0;k<n; k++)                 n^3+n^2
    c[i][j]=c[i][j]+a[i][k]*b[k][j];  n^3
}
```

算法中所有语句的总执行次数为 $T_n=2n^3+3n^2+2n+1$，从中可以看出，语句总的执行次数是问题的规模（矩阵的阶）n 的函数 f(n)（T_n= f(n)）。进一步简化，可用 T_n 表达式中 n 的最高次幂来度量算法执行时间的数量级，即算法的时间复杂度，记做：

$$T(n)=O(f(n))$$

上式是 T_n= f(n)中忽略其系数 n 的最高幂次项，它表示随问题规模 n 的增大算法的执行时间的增长率和 f(n)的增长率相同，称为算法的渐进时间复杂度，简称时间复杂度。如上算法的时间复杂度为 $T(n)=O(n^3)$。

算法中所有语句的总执行次数 T_n 是问题规模 n 的函数，即 T_n= f(n)，其中 n 的最高次幂项与算法中称为原操作的语句的语句频度对应，原操作是算法中实现基本运算的操作，在上面的算法中的原操作是 c[i][j]=c[i][j]+a[i][k]*b[k][j]。一般情况下原操作由最深层循环内的语句实现。

T(n)随 n 的增大而增大，增长得越慢，其算法的时间复杂度越低。下列 3 个程序段中分别给出了原操作 count++的 3 个不同数量级的时间复杂度。

（1）count++ ；

其时间复杂度为 O(1)，称之为常数阶时间复杂度。

（2）for (i=1; i<= n; i++)

```
    count++;
```

其时间复杂度为 O(n)，是线性阶时间复杂度。

（3）for (i=1; i<= n; i++)

```
  for (j=1;j<= n; j++)
    count++;
```

其时间复杂度为 $O(n^2)$，是平方阶时间复杂度。

此外，算法能呈现的时间复杂度还有对数阶 $O(\log_2 n)$和指数阶 $O(2^n)$等。

3. 常见的时间复杂度

常见的时间复杂度有：O(1)常数阶、O(n)线性阶、$O(n^2)$平方阶、$O(n^3)$立方阶、$O(2^n)$指数阶、$O(\log_2 n)$对数阶和 $O(n\log_2 n)$。时间复杂度（从小到大排列）的比较如表 1-2 所示。

表 1-2　常用的时间复杂度的比较

$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	5096	65536
5	32	160	1024	32768	2147483648

4. 最坏时间复杂度

算法中基本操作重复执行的次数还随问题的输入数据集的不同而不同。例如，下面冒泡排序算法：

```
void Bubble(int a[], int n)
{/*对整数数组 a 中的 n 个元素从小到大排序*/
  int i=0, j;
  int change ;
do
{    change=0 ;
for(j=0; j<n-i-1; j++)
        if( a[j]>a[j+1])
{
a[j] ←→ a[j+1]; /*交换序列中相邻的两个整数*/
change=1;
}
i=i+1 ;
}
while(i<n-1 && change )
}
```

在这个算法中，“交换序列中相邻的两个整数”（a[j]←→a[j+1] ;）为原操作。当 a 中初始序列为自小到大有序，原操作的执行次数为 0；当初始序列为自大到小有序时，原操作的执行次数为 n(n−1)/2。对于这类算法时间复杂度的分析，一种解决的方法是计算它的平均值，即考虑它对所有可能输入数据集的期望值，此时相应的时间复杂度为算法的平均时间复杂度。然而在很多情况下，算法的平均时间复杂度是难以确定的，通常的做法是讨论算法在最坏情况下的时间复杂度。例如，冒泡排序在最坏情况下（初始序列为自大到小有序时）的时间复杂度就为 $T(n)=O(n^2)$。本教材中，如不进行特殊说明，所讨论的各算法的时间复杂度均指最坏情况下的时间复杂度。

5. 算法的空间复杂度

关于算法的存储空间需求，类似于算法的时间复杂度，采用空间复杂度作为算法所需存储空间的量度，记为：

$$S(n)=O(f(n))$$

其中 n 为问题的规模。一般情况下，一个程序在机器上执行时，除了需要寄存程序本身所用的指令、常数、变量和输入数据以外，还需要一些对数据进行操作的辅助存储空间。其中对于输入数据所占的具体存储量只取决于问题本身，与算法无关，这样只需要分析该算法在实现时所需要的辅助空间单元数就可以了。若算法执行时所需要的辅助空间相对于输入数据量而言是个常数，则称这个算法为原地工作，辅助空间为 O(1)。如果所占辅助空间量依赖于特定的输入，则除特别指明外，均按最坏情况来分析。

算法的执行时间和存储空间的耗费是一对矛盾体，即算法执行的高效通常是以增加存储空间为代价的，反之亦然。不过，就一般情况而言，常常以算法执行时间做为算法优劣的主要衡量指标。本教材对算法的空间复杂度不作进一步讨论。