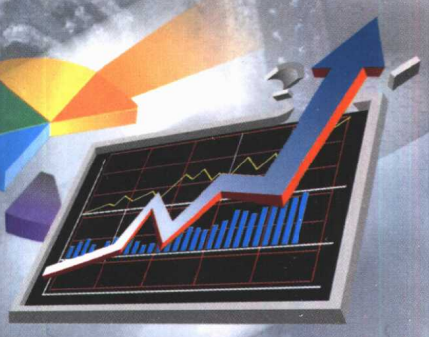
高等学校计算机科学与技术教材

算法与数据结构

(C与C++描述)

陈松乔 肖建华 刘丽华 陈 可 编著



清华大学出版社
http://www.tup.tsinghua.edu.cn
北方交通大学出版社
http://press.njtu.edu.cn

高等学校计算机科学与技术教材

Algorithm & Data Structure

(Program Design In C & C++)

算法与数据结构

（C与C++描述）

陈松乔 肖建华 刘丽华 陈可 编著

清华大学出版社
北方交通大学出版社
北京 • BEIJING

内 容 提 要

本书系统地介绍了算法和数据结构的有关概念、原理、方法和技巧。全书共分 9 章。第 1 章介绍算法和数据结构的基本概念，然后按照线性表、树、图、排序和查找的顺序，详尽简述各种数据结构的概念。对各种数据结构的存储结构和算法用 C/C++语言给出了其抽象数据类型定义，并对给出的算法进行了初步的算法分析。

全书内容新颖，力求理论联系实际、深入浅出和循序渐进。每章均附有习题。

本书主要作为高等学校计算机科学与技术专业本科“算法与数据结构”课程教材，亦可作为其他相关专业的教学用书，或作为从事软件开发人员的参考书和培训教材。

图书在版编目（CIP）数据

算法与数据结构：C 与 C++描述 / 陈松乔等编著. —北京：北方交通大学出版社，2002.8
（高等学校计算机科学与技术教材）
ISBN 7-81082-073-7

Ⅰ. 算… Ⅱ. 陈… Ⅲ. ① 电子计算机-计算方法-高等学校-教材 ② 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（2002）第 046452 号

丛 书 名：高等学校计算机科学与技术教材
书　　名：算法与数据结构（C 与 C++描述）
编　　著：陈松乔　肖建华　刘丽华　陈可
责任编辑：段连平
排版制作：北京依福星印刷咨询有限责任公司
印 刷 者：北京瑞哲印刷厂
装 订 者：北京瑞哲印刷厂
出版发行：北方交通大学出版社　　邮编：100044　　电话：010-62237564　51686045
　　清华大学出版社　　邮编：100084
经　　销：各地新华书店
开　　本：787×1 092　1/16　　印张：16.5　　字数：412 千字
版　　次：2002 年 8 月第 1 版　　2002 年 8 月第 1 次印刷
书　　号：ISBN 7-81082-073-7 / TP・25
印　　数：5 000 册　　定价：23.00 元

前　言

“算法与数据结构”是计算机科学与技术专业的主要专业基础课程之一。本教材为适应高等教育和远程教育的需要，由从事数据结构教学和实践经验丰富的教师编写而成。

“算法与数据结构”的教学要求是：学会构造和分析数据的结构，并能对具体应用问题的逻辑结构、存储结构及其算法进行综合分析和选择，学会编写效率更高的程序，初步掌握算法的时间和空间复杂度分析。算法与数据结构课程的学习过程是程序设计方法和技巧的训练过程，也是复杂程序设计的训练过程。本书采用C语言描述结构和算法，对于抽象数据类型采用类C++语言进行定义和实现，从而使得本书抽象数据类型的定义简明清晰、数据结构更易于理解。为了加强学生的上机编程训练，考虑到远程教育的学习特点，本书所给的C语言程序，力求简明扼要、结构清晰简单、易于上机实现。所有的程序都在Turbo C 2.0编译调试通过。

本书适应了计算机技术的迅速发展，加强了抽象数据类型等概念的介绍，面对目前应用较少或其他课程教材已有的内容（例如：外部排序、文件系统、存储管理等）进行了精简，提高了本书的先进性和适用性。全书共分9章。第1章，简述数据结构的研究对象和内容、抽象数据类型的基本概念、算法概念及算法分析方法；第2～5章，主要介绍线性表，其中包括顺序表、堆栈、队列、链表、数组、广义表和字符串等数据结构及应用；第6章，讨论树，内容有一般树、二叉树、Huffman树等数据结构及其应用；第7章，讨论图，主要介绍图的一般概念、图的遍历、拓扑排序、最小生成树、最短路径和关键路径算法；第8章，讨论排序，介绍排序的基本概念，重点讨论5种基本内部排序算法；第9章，讨论查找，介绍一般查找方法和Hash表。

全书由中南大学计算机科学与技术专业博士生导师陈松乔教授组织编著和审定，第2～4章由刘丽华副教授编写，第8章由陈可老师编写，其余各章由肖建华博士编写。

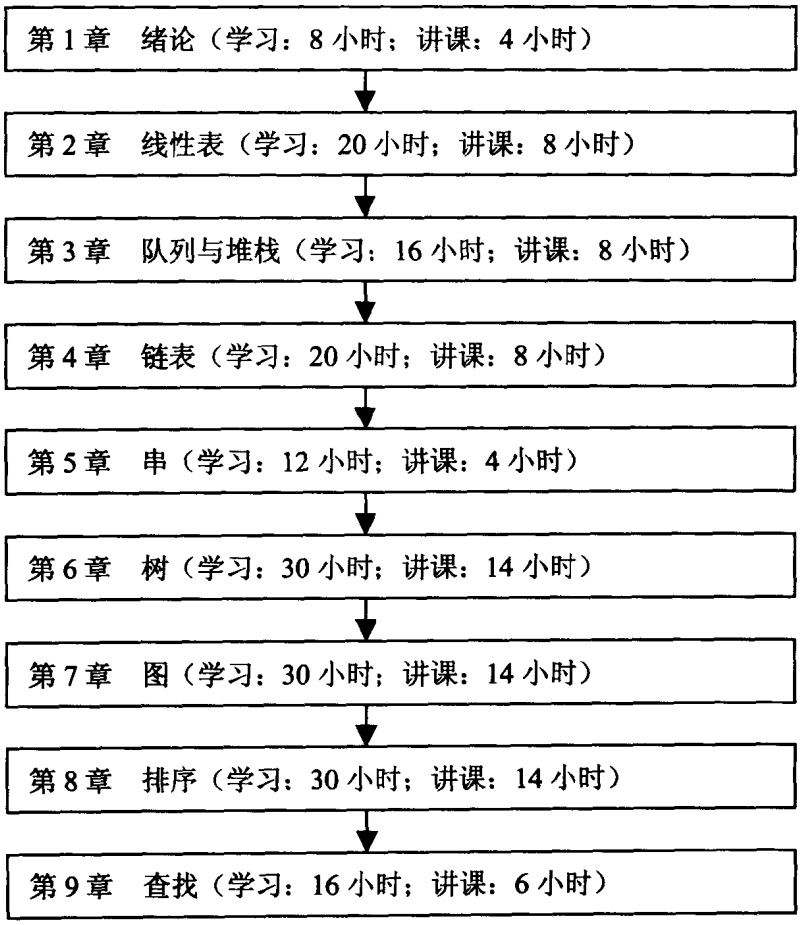
由于作者水平所限，时间仓促，书中的错误和不当之处在所难免，敬请各位读者批评指正。

作　者

2002年8月于中南大学

学 习 指 南

① 先修课程：计算机基础、高等数学、离散数学、高级语言程序设计及其课程设计等；
② 参考学时和学分：建议学习时限约为 182 小时，其中讲课 80 小时，本课程计 4 学分；
③ 学习进度安排和学习流程：



目　录

第1章 绪 论

数据是信息的载体，在计算机科学中数据是指所有能输入到计算机中并由计算机程序处理的符号的总称。随着计算机技术的发展，计算机处理的数据已不再局限于数值数据，而是更多地用于表示文字、图像、声音等非数值数据。因此，大多数计算机程序不仅要完成运算，还要进行数据的存储、检索和处理。从存储空间和运行时间的角度来看，这些程序必须有效地组织信息，即采用有效的逻辑结构和存储结构，以支持高效的信息处理。随着计算机技术迅速发展，人们越来越深刻地认识到，程序设计的实质就是构造一种好的结构加上设计一种好的算法，即人们常说的“程序设计＝数据结构＋算法”。所以，研究数据的结构和算法以有效地支持程序的实现就成了计算机科学的核心问题之一。而数据结构课程研究的主要内容就是数据的逻辑结构、存储结构和操作及其算法，因此，数据结构课程已成为计算机科学与技术专业的重要专业基础课程之一，同时也是其他相关专业的重要必修或选修课程。

1.1 基本概念与术语

本书中将用到很多有关数据的概念和术语，首先介绍一些基本概念和术语。

1.1.1 数据和数据结构

1. 数据

数据（Data）是对客观事物的符号表示，在计算机科学中是指能输入到计算机并能由计算机程序进行处理的符号的总称。例如，数值计算中求积分程序的处理对象是整数和实数，而文件处理程序或语言编译程序所处理的对象是字符。因此，整数、实数、字符都是数据。在多媒体程序中，其处理对象是声音、图像等。虽然计算机不能直接处理声音和图像，但声音和图像经过数字化或编码后，就变成能被计算机程序处理的对象，因此，声音和图像都是数据。对计算机科学而言，数据的含义极为广泛，除上面列举的整数、实数、字符、声音和图像等数据之外，它还具有更广泛的内涵。

2. 数据元素

数据元素（Data element）是数据的基本单位，在程序中通常是作为一个整体来进行处理的。一个数据元素通常由若干个数据项（Data item）组成，而数据项是数据的不可分割的最小单位。

例如，在图书检索程序中，一本书的信息就是数据元素，它由书名、作者名、出版社和分类号等数据项组成，这些数据项是不可分割的最小单位，比如，分类号尽管由若干字符和数字组成，它不能再分割，因为分割后，它没有任何含义。

3. 数据对象

数据对象（Data object）是性质相同的数据元素的集合。任何计算机程序不会只处理一

个数据元素，其处理对象通常是相同性质的数据元素集合。例如，图书检索程序处理的是某图书馆内所有图书，而不是某一本图书。

4. 数据结构

数据结构（Data structure）是相互之间存在的一种或多种特定关系的数据元素的集合。数据结构这个概念至今还没有一个统一的定义，本书给出的定义，实质表明了数据结构是由数据对象及其相互之间的关系两大部分组成，即有关系的数据对象集合才能是数据结构，这是一种较粗浅的定义，要进一步了解其含义还需要从下面三个方面来理解数据结构。

（1）结构

结构即数据元素之间的关系。根据数据元素之间关系的不同特性，通常有下列四类基本结构形式。

① 集合　数据元素之间除了“属于同一个集合”之外没有任何其他关系。如图 1-1 所示。

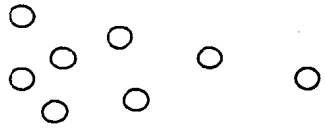


图 1-1　数据的集合关系

② 线性结构　数据元素之间存在着一对一的关系（线性关系），即除第一个和最后一个数据元素外，其他每个数据元素有且仅有一个直接前驱和一个直接后继。如图 1-2 所示。

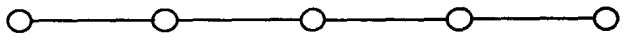


图 1-2　数据的线性关系

③ 树形结构　数据元素之间存在一对多的关系，也即一个数据元素可以与一个或多个数据元素有关系，其结构形式有如倒生长的树。如图 1-3 所示。

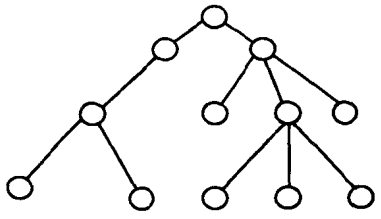


图 1-3　数据的树形关系

④ 图状结构或网状结构　数据元素之间存在多对多的关系。在这种关系中，数据元素之间关系不受任何限制。如图 1-4 所示。

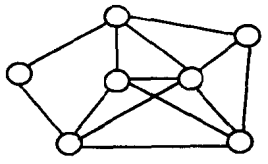


图 1-4　数据的图状关系

（2）数据结构的组成部分

从数据结构的形式定义知，数据结构实质是一个二元组：

数据结构＝<数据对象，关系>

但这种二元组并没有真正反映数据结构的内涵，因此，它不能成为数据结构的标准定义。

一般来说，数据结构由四部分组成：数据对象、数据的逻辑结构、数据的存储结构和数据的操作运算。数据的逻辑结构是对数据元素之间关系的数字形式描述。而上述二元组中的“关系”仅是数据的逻辑结构，讨论数据的逻辑结构的目的是处理数据元素，因此，仅有数据的逻辑结构还不够，而需要研究在计算机内如何表示这种逻辑结构。

（3）数据的存储结构

数据结构在计算机中的存储方式称为数据的物理结构，本书中又称为存储结构，它包含数据元素及数据元素之间关系的表示。在计算机中表示信息的最小单元是二进制数的一位，称为位（bit）。那么表示一个数据元素时需要若干个位，这若干个位组成位串，称这种位串为结点（Node）或元素（Element）。而结点中对应于数据项的子位串称为数据域（Data field）。

数据的存储结构有两种不同形式：顺序存储结构和链式存储结构。顺序存储结构的特点是利用结点在存储器中的相对位置来表示逻辑关系。而链式存储结构借助于指向结点存储地址的指针（Pointer）表示数据元素之间的逻辑关系。例如，复数的存储就可以采用这两种存储方式。若用顺序存储结构，则实部用一个四字节长的内存空间表示，而虚部用接下来的一个四字节长的内存空间表示。如图 1-5 所示。

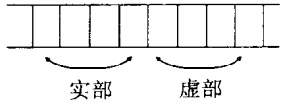


图 1-5　复数的顺序存储

若用链式结构表示复数，则实部用一个四字节长的内存空间，但接下来的二个字节长的内存空间存储虚部所在的地址。如图 1-6 所示。

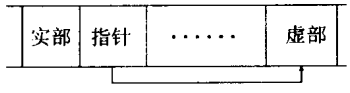


图 1-6　复数的链式存储

图 1-7 为复数－3.5＋2.4i 的两种表示方法在内存中的映像。

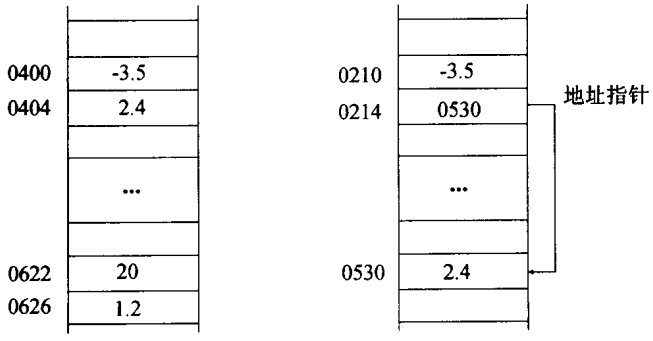


图 1-7　复数的两种存储方式的内存映像

1.1.2 数据结构的研究内容

从学科方面来讲，数据结构是一门研究程序设计问题中操作对象以及它们之间关系和运算的学科。具体地讲，数据结构研究下列四个方面的问题：

- 数据的逻辑结构，即数据元素之间的逻辑关系；
- 数据的物理结构，即数据在计算机中的存储方式；
- 定义在数据结构上一组操作及其完成算法；
- 算法的效率分析，主要分析算法的时间和空间效率。

数据结构是计算机科学与技术的核心课程。早期的数据结构几乎是表、树、图的同义语，发展到现在，数据结构课程还包括排序、查找、文件等方面的内容，该课程不仅研究常规的数据结构，还要研究抽象数据类型。由于数据结构研究的数据要在计算机中处理，因此，不仅要考虑数据本身的数学性质，还要重点考虑数据的存储结构及相应操作的实现算法，这一点丰富了数据结构的研究内容，使数据结构课程成为计算机科学技术专业的核心课程。

学习数据结构的目的，就是提高程序设计理论和技术，提高计算机程序的效率，这些技巧包括数据的存储方法和算法。程序设计的实质就是为问题设计一个好的结构，加上设计一种好的算法，只有这样才会有高效的程序。有人可能认为，随着计算机的日益强大，程序的运行效率好像变得越来越不那么重要了。事实上并非如此，计算机功能越强大，人们就越想去尝试更复杂的问题，而更复杂的问题需要更大的计算量，这就使得对高效率程序的需求更加明显。因此，今天的计算机科学与技术专业的学生必须训练出彻底理解隐藏在高效程序设计后面的理论和技术，提高设计高效可靠程序的能力。这种能力无法从日常的生活经验获得，只有通过不断地计算机程序设计训练才能提高。

1.2 抽象数据类型

在高级语言编程中，任何一个变量、常量或表达式都有一个它所属的确定的数据类型。这些数据类型明显或隐藏地规定了数据取值范围及定义在其上的运算，因此，数据类型是值的集合和定义在值集上的一组操作的总称。例如，C 语言的整型数据，其值集为 [-32768，32767] 区间内的整数（这个区间依赖于机器字长），而定义在其上的操作为加、减、乘、除和取余等运算。按“值”的不同特性，可以把数据类型分为两类：原子类型和结构类型。

原子类型的值是不可分解的。例如，C 语言中的整型、实型、字符型、指针型等都是原子类型的数据类型。这种类型较少，一般高级语言中定义的基本数据类型就可以满足实际需要。当然，有时为了某种特定目的，还可以定义新的原子类型，例如，“位长 100 的整数”就是一种原子类型。

结构类型的值是由若干成分（原子类型或结构类型）按某种结构组成的。例如，一本书的值可以定义为书名、作者、出版社、出版时间和分类号等成分值组成；又如，C 语言的数组是语言惟一定义的一种结构类型，它由若干分量组成，而每个分量可以是整型、实型等原子类型，也可以是数组。因此，结构类型的值是可分解的。

对于数据结构和结构类型的区别，可以这样简单理解：数据结构是具有相同结构值的集

合，而数据类型可以看成由数据结构和定义在其上的一组操作组成。

在计算机科学里，数据类型的概念不局限高级语言中，在计算机硬件系统、软件系统以及许多应用环境中都提供数据类型。引入“数据类型”的目的，对于普通用户来说，他们不必了解这些类型和操作的实现方法，而只需知道输入什么后，按操作规则计算机会输出什么；有效地达到信息的封装，使用户不必也不需要去了解封装内的复杂细节；简化用户对概念的理解，使计算机硬件、软件和应用系统能被一般用户所理解，有利于计算机的普及。高级语言中的这种数据类型成功地实现了数据和操作的抽象化，但这种数据类型对数据关系和操作能力的抽象还不是很强，为了提高抽象能力和更进一步了解数据类型的本质，有必要引入抽象数据类型的概念。

定义 1.1　抽象数据类型（Abstract Data Type），简称 ADT，是指基于一个逻辑类型的数据类型以及这个类型上的一组操作。所谓类型是指一组值的集合。

每个操作由数据的输入和输出定义，一个 ADT 的定义并不涉及它的实现细节，这些实现细节对于 ADT 用户来说是隐藏的，隐藏实现细节的过程称为封装，由此，数据结构的实质就是 ADT 的物理实现——计算机上的实现。

和数据结构的形式定义一样，抽象数据类型可用以下的三元组表示：

$$ADT=\langle D,S,P\rangle$$

其中，D 是数据对象，S 是 D 上的关系集，P 是对 D 的基本操作集。与前面介绍的高级语言数据类型不同，这里的数据对象之间的关系更负责、更灵活，常用关系模型表示，而基本操作可在数据类型定义中根据需要自行定义。本书采用以下格式定义抽象数据类型。

```
ADT <抽象数据类型名> is
Data:
    <数据元素及其数据元素之间的关系描述>
Operations:
    <运算描述>
End <抽象数据类型名>
```

其中，数据对象和数据关系按严格的数学形式描述，而基本操作描述格式为：

```
基本操作名(参数表);  //操作功能描述
```

如果形式参数的前面有&，则操作结束后返回相应结果给实际参数。

本书采用 C 语言来实现抽象数据类型。由于是在 C 语言的虚拟层次上讨论抽象数据类型的表示和实现，故采用介于伪码和 C 语言之间的类 C 语言作为描述工具，本书中的程序一般都用标准 C 语句，只在一个抽象操作的抽象算法中才采用伪码描述。所有的程序和结构描述，只需进行很少的修改就可以在 C 或 C++上调试运行。

为了更好地理解抽象数据类型，例 1.1 给出复数的抽象数据类型的定义。

【例 1.1】 复数抽象数据类型的定义。

```
typedef int status;
ADT Complex_Number is
Data:
    D ={r,c|r,c∈R(实数集合)}
    R={<r,c>}
```

Operations:

void Init_Complex_Number(&T,v1,v2) //构造一个实部和虚部分别为 v1,v2 的复数 t

void Destroy_Complex_Number(&t) //销毁复数 t

status Plus_Complex_Number(t1,t2,&t) //复数相加运算，t=t1+t2；返回 t

status Minus_Complex_Number(t1,t2,&t) //复数相减运算，t=t1-t2；返回 t

status Multi_Complex_Number(t1,t2,&t) //复数相乘运算，t=t1*t2；返回 t

status Divide_Complex_Number(t1,t2,&t) //复数相除运算，t=t1/t2；返回 t

End Complex_Number

1.3 算法与算法分析

1.3.1 问题、算法和程序

程序设计人员需要不断地处理问题、算法和计算机程序，这是三个不同的概念，这一节将详细描述这些概念，最后还将讨论如何进行算法分析。

1. 问题

从直觉上讲，问题（Problem）就是一个给定的需要完成的任务，即对应一组输入有一组输出。作为计算机程序员，只有在问题被准确定义并完全理解后才能研究问题的解决方法。对问题的理解还必须包含该问题的一些限制，计算机要解决的任何一个问题，总有一些直接或间接的限制，比如说，内存、外存和运行时间的限制。但是，问题的定义中不能包含有关怎样解决问题的限制，但应该包含对任何可行方案所需资源的限制。

2. 算法

算法（Algorithm）是指解决问题的一种方法或者一个过程，是计算机操作步骤的集合。一个问题可以有多种算法，但一个给定的算法只解决一个特定的问题。本书对涉及到的许多问题给出了不同的算法，例如，对排序问题就给出 10 多种算法。

一个算法应该具有以下几条性质。

- 正确性：它必须完成期望的功能，把每一次输入转化为正确的输出。
- 具体性：一个算法必须由一些具体步骤组成。具体意味着每一步骤必须是机器可读的而且是可执行的。
- 确定性：下一步应执行的步骤必须明确。选择语句（C 语言中的 if 和 switch 语句）是任何算法描述语言的组成部分，它允许对下一步执行的语句进行选择，但选择过程必须是确定的。
- 有限性：一个算法必须由有限步组成。如果一个算法的描述是无限步组成的，那么就不能将算法写出来，更不可能将它变成计算机程序来实现。大多数描述语言均提供一些重复行为的方法，如循环（C 语言中的 while、do...while 和 for 语句都是循环语句）。循环结构具有简短的描述，但是实际执行的次数会是许多次，它由输入来决定。

- 可终止性：算法必须可终止，即不能进入死循环。

3. **程序**

程序（Program）是对一个算法使用某种程序设计语言的具体实现。

本书中几乎所有算法都给出了C语言程序，由于是用语言按语法规则书写的，因而一个算法可以有多个不同的程序实现。本书中常常混淆“算法”和“程序”，把它们看成同一概念。但根据算法和程序的定义，可以知道并不是所有计算机程序都是算法，例如，一个有死循环的程序就不是算法，但任何一个算法至少对应一个计算机程序。

1.3.2 算法分析

一个问题有多种算法求解，怎样选择呢？当然应该选择“最好”的算法来求解问题，但什么算法是“好”算法呢？这需要从程序设计的目的出发来分析该问题。计算机程序设计有两个核心目标：

- 简便算法 设计一个容易理解及编码和调试方便的算法；
- 高效算法 设计一个能有效利用计算机资源的算法，即运行时间短，存储空间消耗少的算法。

在理想情况下，达到这两个目标的最终程序都是“最好”的程序，有时也可以说是“完美的”。但通常这两个目标是相互冲突的，高效算法对一般用户有时甚至对计算机专家来讲是难理解的，而容易理解的算法常常达不到高效的目的。本书所给出的算法是按“完美”要求进行设计的，因此本书中的算法基本是趋于“完美”的。如何达到简便算法的要求是软件工程课程追求的内容，不是本书目的。本书追求的目的是高效的算法，或者是在高效算法的基础上来实现简便算法。

估量一个算法效率的方法，称为算法分析。算法分析的目标在于尽可能少地占用计算机资源。计算机资源一般有两个：存储空间和执行时间。算法需用的存储空间大小称为算法的空间复杂度，算法的执行时间称为算法的时间复杂度。相对于空间复杂度来讲，对于大多数算法而言时间复杂度显得更重要，更需要研究。本书讨论最多的是算法的时间复杂度。

1.3.2.1 算法的时间复杂度

比较两种算法的时间复杂度，有两种办法。一种办法就是：用源程序分别实现这两种算法，然后输入适当的数据运行，测算两个程序各自的时间开销。但是，这个方法并不可行。第一，编写两个程序，测算两种算法将花费较多的时间和精力，而实际上至多只需要其中的一个程序；第二，仅凭实验来比较两种算法，很有可能因为一个程序比另一个“写得好”，而使得另一个算法由于“写得不好”使其真正质量没有得到很好的体现；第三，测试数据的选择可能对其中的一个算法有利；第四，即使是较好的那种算法也超出了原来预算的时间开销，这就意味着还得重复一遍这样的过程——寻找一种新的算法，再编写一个程序实现它。

另一种办法可以解决所有这些问题，这就是渐近算法分析（Asymptotical algorithm analysis），或事先估算算法分析，简称算法分析（Algorithm analysis）。它可以估算出当问题规模变大时，一种算法及实现它的程序的效率和开销。这种方法实际上是一种估算方法，若两个程序中一个总是比另一个“稍快一点”时，它并不能判断那个“稍快一点”的程序的相