数据结构+算法

夏克俭　编著

国防工业出版社

·北京·

图书在版编目(CIP)数据

数据结构+算法/夏克俭编著 .—北京：国防工业出版社,2001.2

ISBN 7-118-02419-8

Ⅰ.数… Ⅱ.夏… Ⅲ.①数据结构 ②电子计算机-计算方法 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2000)第 54229 号

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号)

(邮政编码 100044)

三河市腾飞胶印厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 19 434 千字

2001 年 2 月第 1 版 2001 年 2 月北京第 1 次印刷

印数:1-3000 册 定价:26.00 元

前　言

随着计算机在我国广泛而深入的应用，在分析和开发计算机系统软件和应用软件的过程中，迫切需要掌握各种数据结构（或数学模型）的描述方法以及相应的处理算法。因此，数据结构与算法知识是计算机相关学科的学生和从事计算机应用的技术人员必备的专业基础知识。

著名的瑞士计算机科学家沃思（N.Wirth）提出：数据结构 + 算法 = 程序。其中数据结构指的是数据及其相互关系的表示，包括数据的逻辑结构和存储结构，实际上是研究从具体问题中抽象出来的数学模型如何在计算机存储器中表示的问题；而算法是数据处理的方法，研究如何在相应的数据结构上施加运算来完成所要求的任务。如果关于问题的数据表示及数据处理都实现了，也就等于完成了相应的程序设计。

本书系统地介绍了常用的数据结构类型，着重讨论数据结构在计算机存储器中的表示，以及在数据的存储结构上如何实现相关的算法，同时对算法的时间复杂度也进行了必要的分析。

全书内容包括十章：第一章绪论，围绕数据结构和算法介绍一些基本概念；第二至第四章讨论线性结构——线性表、栈和队列、字符串等；第五章讨论数组和广义表，它们可以看做线性表的扩充；第六章讨论层次结构——树，重点是二叉树结构及相关算法；第七章讨论网状结构——图，重点是图的存储结构及基本算法；第八、第九两章讨论数据处理中的查找和排序方法，关键在于提高算法的运行速度；第十章介绍文件的基本概念及结构。同时书中许多章节提供了数据结构的应用实例。

随着计算机软件技术的发展，数据结构的内容日渐增多，但始终围绕着一条主线，即“线性结构——层次结构——网状结构”；而对每种类型的数据结构，又都是从数据的逻辑结构、存储结构和算法这三个方面来展开的。掌握了这一点，就能较好地掌握数据结构的相关知识。

本书中的算法用 C 语言函数形式描述。我们的教学实践工作表明，掌握数据结构的内容并不困难，难点在于算法的设计与分析。作者对书中的重点算法一般先给出思路，然后是 C 语言描述和算法分析，书中实例均进行过相应程序的上机实践。相信这对读者分析与设计算法，以及编制 C 语言应用程序会有较大的帮助。

本书是我们在多年的教学基础上编写而成的，研究生徐生震、蒋伟杰协助完成了书稿的整理工作和上机实践。

本书经中国人工智能学会理事长、北京科技大学涂序彦教授审阅，在此表示衷心的感谢。

由于作者水平有限，书中不足之处或错误在所难免，恳请读者批评指正。

夏克俭　北京科技大学信息工程学院

2000 年 7 月于北京

目　录

第一章 绪 论

目前，计算机特别是微型计算机业在飞速发展，其应用领域早已不限于科学计算，而是广泛深入到社会的各个部门。生物、军事、行政管理、教育卫生、工厂企业、商业银行和邮电通信等部门，都在不同程度地使用计算机，完成先前由人工或机械所从事的任务，大大提高了工作质量和工作效率。计算机在各部门的应用大致可归纳为以下几类：

(1)科学计算与分析(Scientific Calculation and Analysis)，如用计算机进行方程求解、卫星轨道计算、天气预报和情报分析等。

(2)计算机管理(Management Information System)，如部门的物质、施工、财务、产品销售和档案诸方面的计算机管理。这方面目前以“管理信息系统(MIS)”为代表，主要从事数据处理和辅助决策方面的工作。

(3)计算机实时控制(Real - Time Control)，如用计算机对轧钢流水线、瓶装线等生产过程的实时控制。

(4)计算机辅助设计/制造(CAD/CAM)及绘图，如电路、建筑、服装、汽车设计和机械方面的 CAD/CAM。

(5)计算机通信网络(Telecommunication Networks)，如利用计算机及网络技术，达到数据文件和资料等信息的远距离传输和共享。该领域以 Internet 网在世界范围的成功应用为典型范例。

(6)办公自动化(Office Automation)，如用计算机进行文字处理、报表生成、文件传送和召开电视会议等。

(7)人工智能(Artificial Intelligence)，即用计算机来模拟人的一些智能活动。“专家系统(ES)”、“决策支持系统(DSS)”和“机器人(Robot)”等是人工智能方面的具体应用。

(8)机器仿真(Computer Emulating)，即用计算机模拟一个真实活动的过程。如军事活动、生物活动和汽车驾驶模拟诸领域的计算机仿真。

(9)计算机辅助教学(CAI)和娱乐，如作曲软件、教授“数理化”和英语等知识的软件，以及棋、牌游戏软件等。这方面运用了多媒体技术后，图、文、声并茂，使学习更加生动有趣。

随着计算机应用的广泛深入，计算机的应用领域会越来越广，计算机要处理的数据更多样化，而数据之间的关系和对数据处理的要求也更为复杂。这就要求我们在从事具体的计算机应用时，要用较科学的方式描述、存储和处理数据，使计算机高效率地去完成预期的任务。

本章主要讨论数据结构、算法及算法分析等方面的一些基本概念。

1.1 数据结构的含义

数据结构(Data Structure)简称 DS,要弄清其含义,需了解以下几点:

1.计算机处理的对象(数据)已不再单纯是数值

例 1-1 图书管理中的数据问题,如表 1.1 所列。

表 1.1

	编号	书名	作者	出版社	出版日期	…
a_1	0001	C 程序设计及应用	李盘林	高等教育出版社	1998	…
a_2	0002	数据结构 + 算法	夏克俭	国防工业出版社	2000	…
a_3	0003	数据库原理与技术	周志逵	科学出版社	1998	…
a_4	0004	智能管理	涂序彦	清华大学出版社	1995	…
	⋮	……	…	……	……	…
a_n	⋮	……	…	……	……	…

表中每一行为一个数据元素(或记录),记为 $a_i(1 \leqslant i \leqslant n)$,则表 1.1 可表示为:

$$list = (a_1, a_2, \cdots, a_n)$$

显然,表 1.1 中每一数据元素包含的许多值是非数值性的(如文字、日期等数据),对其进行的操作(或运算)也不再是加、减、乘、除等数学运算,而是诸如查询(查找一本书的信息)、插入(增加一本书的信息)、修改(某书修订后,修改元素中的某些信息)、删除(某书不再版了,做删除标记)、分类(按某数据项的值建立索引)等等这样的操作。

例 1-2 大学系级行政机构,如图 1.1 所示。

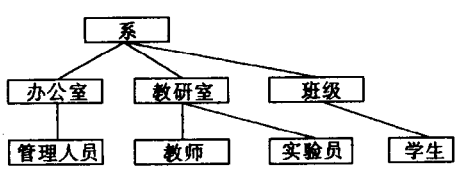


图 1.1

其中系、办公室、……、学生可视为数据元素;元素之间呈现的是一种层次关系,即系级下层机构为办公室、教研室和班级,而办公室、教研室和班级等单位又由若干个管理人员、教师、实验员和学生组成。

例 1-3 田径比赛的时间安排问题。

设田径比赛项目有:A(跳高)、B(跳远)、C(标枪)、D(铅球)、E(100m 跑)、F(200m 跑),参赛选手的项目表(每人限参加三项)如表 1.2 所列,问如何安排比赛时间,才能使得:

(1)每个比赛项目都能顺利进行(无冲突);

(2)尽可能缩短比赛时间。

此问题可归纳为图的“染色”问题:设项目 A ~ F 各表示一数据元素,以○表示。若两个项目不能同时举行,则将其连线(如项目 A 和 B 不能同时举行,否则丁一无法参赛),由此得到如图 1.2 所示的结构。

表 1.2

姓名	项目 1	项目 2	项目 3
丁一	A	B	E
马二	C	D	
张三	C	E	F
李四	D	F	A
王五	B	F	

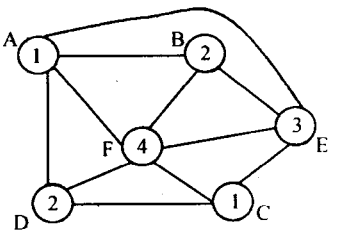


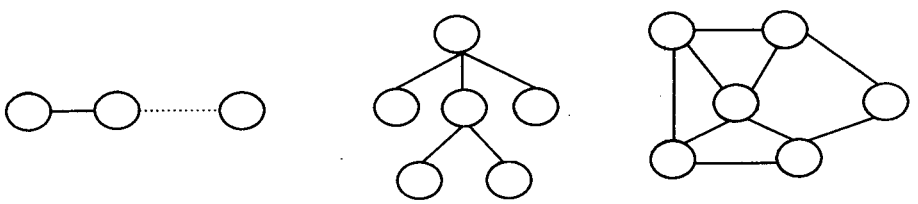
图 1.2

图 1.2 为解决此问题的数学模型,或数据结构的模型,

数据元素之间呈现的是一种网状关系,表示项目相对时间安排的一种约束。下面要解决的是对此图着色。设一种颜色表示一个比赛时间片,显然,同一直线上的两个元素不能同色(否则出现冲突)。若用 1、2、3、4 表示四种颜色,一种着色方法如图 1.2 所示,即:时间片 1 内比赛 A、C 项目,时间片 2 内比赛 B、D 项目,时间片 3 内比赛 E 项目,时间片 4 内比赛 F 项目。当然,这种着色方法是否最优还待进一步研究。

2.数据元素之间存在某种关系

数据元素并不是孤立存在的,它们之间存在着某种关系(或联系、结构)。关于关系的形式,对于例 1 - 1 ,数据元素之间呈现的是一种线性关系,或称线性(linear)结构;对于例 1 - 2,呈现的是一种层次关系,或称树(Tree)结构;对于例 1.3,呈现的是一种网状关系,或称图(Graph)结构,如图 1.3 所示。



(a)线性关系　　(b)层次关系　　(c)网状关系

图 1.3

对数据元素与关系而言,人们更多注意的是元素之间的关系。就像制作一扇窗户,原材料可以是木料、钢材、铝合金等,但我们更注重的是窗户的结构。如何设计,才能使窗户的采光性好、美观大方、节省原料,是真正需要研究的问题。

3.如何在数据和数据之间的关系上定义运算(或操作)

对数据结构最基本的运算为查找、插入、删除、排序等。

综上所述,数据结构主要是研究非数值性程序设计中计算机操作的对象(数据)及其相互间关系和运算的学科。

1.2 一些基本概念

1. 数据(Data)

数据即信息的载体,是能输入到计算机中并且能被计算机识别、存储和处理的符号总称。如方程中的整数、实数,源程序中的字符串,以及文字、图像和声音信号等等,都可作为计算机中的数据。

2. 数据元素(Data Element)

数据元素是数据的基本单位,又称之为记录(Record)。简单的数据元素可以是整数、字符串等形式。一般,数据元素由若干基本项(或称字段、域、属性等)组成。例如产品的记录表,如表 1.3 所列。

表 1.3

产品编号	名　称	规　格	出厂日期	……
0001	TV	29′	1999/09/09	……
0002	VCD	K98	1999/09/20	……
……	……	……	……	……

表的每一行为一数据元素,分别由“产品编号”、“名称”等基本项组成。在向计算机存取数据时,基本项是不可分割的最小存取单位。

3. 数据类型(Data Type)

数据类型是对数据元素取值范围与运算的限定。如整数是一个数据类型,它对应一个整数集合和施加在该集合上的运算(或操作)。在 C 语言中,若说明:int x;则变量 x 可以存放一个整数,取值范围一般为:[- 32768, + 32767],限定的操作为:[+ , - , * ,/,% (取模)]。

算法语言中一般都提供了基本数据类型的说明符,但用户编程时要用到的数据元素类型,需要自行定义。例如用 C 语言描述一个学生记录,一般要用结构类型,可按如下方式定义:

```
Typedef struct node
    {int sno;              //存放学生学号//
     char name[20];        //存放学生姓名//
     int age;              //存放学生年龄//
     ……
     ……
    } student;
```

此时若说明:student A; 则 A 为存放一个学生信息的结构变量。其中 A.sno 为该学生的学号,A.name[0] ~ A.name[19]存放该学生的姓名,A.age 存放该学生的年龄等等。在以后的章节中,以说明符 datatype 泛指数据元素的类型。

4. 数据结构(Data Structure)

数据结构简称 DS,指的是数据元素及数据元素之间的相互关系,或数据元素的组织

形式。有人认为:按照某种逻辑关系组织起来的一批数据,运用计算机语言,按照一定的存取方式把它们存储到计算机存储器中,并为这些数据定义一个运算集合,就称为一个数据结构。

数据结构(DS)可用形式化语言描述,即 DS 是一个二元组:

$$DS = (D, R)$$

其中,D 为数据元素的集合;R 为 D 上关系的集合。

例 1-4 将向量 $v(a_1, a_2, \cdots, a_n)$定义为一个 DS,则 $v = (D,R)$,其中:

$$D = \{a_i \mid a_i \in \text{datatype},\ i = 1,2,\cdots\cdots,n,\ n \geqslant 0\}$$

$$R = \{ < a_i,\ a_{i+1} > \mid a_i, a_{i+1} \in D,\ 1 \leqslant i \leqslant n-1 \}$$

符号"$< a_i,\ a_{i+1} >$"表示向量 v 中任意两相邻元素的先后次序关系,在这里称为有序对。如向量 $v = (1,3,5,7,9)$,对应 D,R 可分别表示为:

$$D = \{2 * i + 1 \mid i \in \text{整数},\ i = 0,1,2,3,4\}$$

$$R = \{ < 1,3 >, < 3,5 >, < 5,7 >, < 7,9 > \}$$

5. 数据的逻辑结构(Logical Structure)

如果只是描述数据结构中数据元素之间的联系规律,即数据元素之间的逻辑关系,则称此时的数据结构为数据的逻辑结构。它是从具体问题中抽象出来的数学模型,是独立于计算机存储器的(与计算机无关)。数据的逻辑结构类型一般可划分为线性结构和非线性结构,非线性结构又可分为层次结构(或树结构)和网状结构(或图结构),其表现形式如图 1.3 所示。在以后的章节中我们会详细谈到这些结构。

6. 数据的存储结构(或物理结构)(Physical Structure)

数据的存储结构指的是数据的逻辑结构在计算机存储器中的映象(或表示)。存储结构是通过计算机语言所编制的程序来实现的,因而是依赖于具体计算机语言的。目前,对一个数据结构较常用的有四种存储表示:

(1)顺序存储(Sequential Storage):将数据结构中各元素按照其逻辑顺序存放于存储器一片连续的存储空间中(如 C 语言的一维数组),此时逻辑上相邻的元素在物理位置上也是相邻的,由此得到的存储结构为顺序存储结构。每种类型的逻辑结构都可以顺序存储。如:表 $L = (a_1, a_2, \cdots, a_n)$的顺序结构如图 1.4 所示。

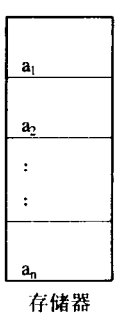


图 1.4

(2)链式存储(Linked Storage):将数据结构中各元素分布到存储器的不同点,用地址(或链指针)方式建立它们之间的联系,由此得到的存储结构为链式存储结构。如:表 $L = (a_1, a_2 \cdots, a_n)$的链式存储结构如图 1.5 所示。

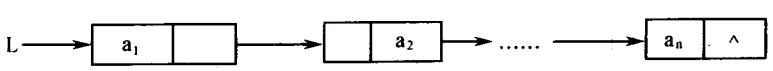


图 1.5

链式存储结构是数据结构的一个重点,因为数据结构元素之间的关系在计算机内部很大程度上是通过地址或指针来建立的。

(3)索引存储(Indexed Storage):在存储数据的同时,建立一个附加的索引表,即索引

存储结构 = 数据文件 + 索引表。

例 1-5 电话号码的查询问题。

为便于提高查询的速度，在存储用户数据文件的同时，建立一张姓氏索引表，如图 1.6 所示。

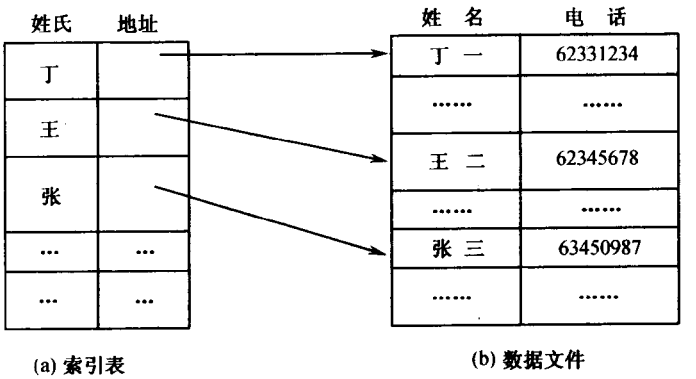


图 1.6

这样，查找一个电话就可以先查找索引表，再查找相应的数据文件，这无疑加快了查询的速度。

(4)散列存储(Hash Structure)：根据数据元素的特殊字段(称为关键字 key)，计算数据元素的存放地址，然后数据元素按地址存放，所得到的存储结构为散列存储结构(或 Hash 结构)。

以后章节中讨论较多的是(1)、(2)两种存储结构；至于建立索引和散列结构，其目的之一是提高对数据文件的查询速度，留在第八章和第十章讨论。

7. 数据结构的三个方面

数据结构的三个方面指的是数据的逻辑结构、数据的存储结构和对数据结构施加的运算。在对数据结构的详细讨论中，都是从这三个方面展开的，读者应掌握其中的规律，以便于对以后知识的学习。

1.3 学习数据结构的目的

1. 数据结构的发展简史及在计算机科学中的地位

数据结构的内容来源于图论、操作系统、编译系统、编码理论和检索与分类技术的相关领域。20 世纪 60 年代末，美国一些大学把上述领域中的技术归纳为《数据结构》课程。美国人 D.E. 克劳特著的《计算机程序设计技巧》(国防工业出版社引进出版)一书，对数据的逻辑结构、存储结构及算法进行了系统的阐述。我国从 70 年代末在各大专院校陆续开设了数据结构课程，目前该课程已经是计算机专业的核心课程之一。

关于计算机科学的概念，计算机界的权威人士认为：其一，计算机科学是信息结构转换的科学，构造有关信息结构的转换模型，并对其进行研究是计算机科学中最根本性的问题。而构造出现实问题中的数据结构模型，并合理地将其映像到计算机存储装置中，是这

种观点的具体体现;其二,计算机科学是算法的学问,研究的是对数据进行处理的方法或规则。要用计算机完成具体任务,离不开对算法的研究。因而,"数据结构"+"算法"应该是计算机科学研究中的基础课题。它的理论基础是离散数学中的图论、集合论和关系理论等等,实践基础是程序设计技术。

2. 学习数据结构的目的

(1) 在分析和开发计算机系统与应用软件中要用到数据结构知识。

如操作系统、编译系统、数据库技术和人工智能中普遍涉及到以下内容:栈和队列、存储管理表、目录树、语法树、索引树、搜索树、广义表、散列表、有向图等等。因而数据结构知识既是操作系统、编译系统等后续课程的基础,也是开发软件所必须具备的知识。

(2) 学习数据结构是为了提高程序设计水平。

我们知道,计算机的应用。一般是由计算机运行程序来实现的,而任何一个程序都是建立和运行在相应的数据结构基础上的,这就要求我们在做程序设计时,一方面要描述好对应的数据结构,另一方面要设计出正确、精确和快速处理数据的算法(或程序)。

1.4 算法的定义及其特性

1.算法的定义

算法(Algorithm)是一个有穷规则(或语句、指令)的有序集合。它确定了解决某一问题的一个运算序列。对于问题的初始输入,通过算法有限步的运行,产生一个或多个输出。

例 1-6 求两正整数 m、n 的最大公因子的算法(欧几里德算法),其步骤如下:

①输入 m,n;

②m/n(整除),余数→r (0≤r≤n);

③若 r=0,则当前 n=结果,输出 n,算法停止;否则,执行④;

④n→m,r→n; 转到②;

如初始输入 m=10,n=4,则 m,n,r 在算法中的变化如下:

m	n	r
10	4	2
4	2	0(停止)

即 10 和 4 的最大公因子为 2。

2. 算法的特性

(1)有穷性——算法执行的步骤是有限的;

(2)确定性——每个计算步骤无二义性;

(3)可行性——每个计算步骤能够在有限的时间内完成;

(4)输入——算法有一个或多个外部输入;

(5)输出——算法有一个或多个输出。

这里要说明的是,算法与程序既有联系又有区别。算法和程序都是为完成某个任务,或解决某个问题而编制的规则(或语句)的有序集合,这是它们的共同点。区别在于:其一,算法可以与计算机无关,但程序依赖具体的计算机语言;其二,算法必须是有穷尽的,

但程序可能是无穷尽的，例如控制卫星运行的程序，一旦启动，它就会一直运行下去，直至卫星毁灭为止；其三，算法可忽略一些语法细节问题，重点放在解决问题的思路上，但程序必须严格遵循相应语言工具的语法，算法转换成程序后才能在计算机上运行。另外，在设计算法时，一定要考虑它的确定性，即算法的每个步骤都是无二义性的（即一条规则不能有两种以上的解释）。如："一个没有来"，就是一个二义性的例子，到底是"一个都没有来"，还是"只有一个没有来"？这样的句子在算法中应杜绝。

下面将例 1-6 中的算法转换成 C 语言程序，程序如下：

```
# include "stdio.h"
  main()
    {int m,n, j;
     char flag = 'Y';
     while (flag = = 'y' || flag = = 'Y')
       {printf(" \ n");
        scanf("input = %d%d",&m,&n);          //输入两个整数 m,n//
        if(m > 0 && n > 0 )
          {j = maxog(m,n);                    //求 m,n 的最大公因子//
            printf ("output = %d \ n",j);     //输出结果//
          }
        printf ("continue? (y \ n)");
        flag = getchar();                     //输入'y'或'Y'继续,否则停止//
       }
    }
  int maxog(int m,n)
  //求 m,n 的最大公因子的算法(或函数)//
    {int r;
     r = m%n;
     while(r! =0)
      {m = n;
       n = r;
       r = m%n;
      }
    return (n);
    }
```

当然，这只是一个很简单的例子，实际应用中的算法及其所涉及的数据结构是比较复杂的。为方便问题讨论及节省篇幅，以后章节中的算法均以 C 语言的函数形式描述，而具体的 main()函数和数据 I/O 就不描述了。另外，由于某种原因（如参数错，存储分配失败等）导致算法无法继续执行下去时，约定调用函数 ERROR()进行出错处理。

讨论了数据结构与算法的基本概念后，有必要提到瑞士科学家沃思（N.Wirth）的著名公式：

数据结构 + 算法 = 程序

其中数据结构指的是对数据及其相互关系的表示，包括数据的逻辑结构和存储结构，实际上是研究从具体问题中抽象出来的数学模型如何在计算机存储器中表示出来的问题；而算法研究的是数据处理的方法，即如何在相应的数据结构上施加一些运算来完成所要求的任务。如果对问题的数据表示和数据处理都讨论清楚，并做出了具体的设计，那就相当于完成了相应的程序。

1.5 算法分析初步

算法分析是指在算法正确的情况下，对其优劣的分析。一个好的算法通常是指：

(1)算法对应的程序所耗时间少；

(2)算法对应的程序所耗存储空间少；

(3)算法结构性好、易读、易移植和易调试等。

但实际上“时间”和“空间”是矛盾的。要想算法执行得快，就得忍让一些存储空间；反之，要想节省存储空间，算法就会复杂一些，所耗时间可能也会长些。

本书中算法一般较短，对其分析主要是考虑算法的时间效率。算法对应程序的时耗，有下面诸多因素：

(1)程序中输入数据的时间；

(2)对源程序编译所需的时间；

(3)执行每条语句的时间；

(4)程序中可执行语句的执行次数。

对(1)，若采用脱机输入技术，则输入数据时间可与程序运行时间相覆盖；对(2)、(3)，依赖编译系统和计算机本身的速度；所以在编程中，我们主要考虑第(4)点。为此，引入下面几个概念：

1. 语句的频度(Frequency Count)

语句频度指在算法(或程序)中可执行语句重复执行的次数。若某语句执行一次的时间为 t，执行次数为 f，则该语句所耗的时间估计为 t·f。

例 1-7 求两个 n 阶方阵乘积。

$$C_{n.n} = A_{n.n} * B_{n.n} = \begin{bmatrix} c[0][0] & c[0][1] & \cdots\cdots & c[0][j] & \cdots\cdots & c[0][n-1] \\ c[1][0] & c[1][1] & \cdots\cdots & c[1][j] & \cdots\cdots & c[1][n-1] \\ \cdots\cdots & & & & & \\ \cdots\cdots & & & & & \\ c[i][0] & c[i][1] & \cdots\cdots & c[i][j] & \cdots\cdots & c[i][n-1] \\ \cdots\cdots & & & & & \\ c[n-1][0] & c[n-1][1] & \cdots\cdots & c[n-1][j] & \cdots\cdots & c[n-1][n-1] \end{bmatrix}$$

其中：

$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] \times B[k][j]$$

算法描述：

```
#define  n  10
MATRIXM(A,B,C)
  float A[n][n], B[n][n], c[n][n];
   {int i,j,k;
    for (i=0; i<n; i++)
      for (j=0;j<n; j++)
        {c[i][j] = 0;
          for (k=0;k<n;k++)
            c[i][j] = c[i][j] + A[i][k]*B[k][j];
        }
   }
```

for 循环语句的执行次数实际上是循环变量的变化次数，所以例 1－7 中第一个 for 语句的执行次数（即频度）应为 $n+1$，其它各语句的频度依次为 $n(n+1)$、n^2、$n^2(n+1)$ 和 n^3。

2. 算法的时间复杂度（Time Complexity）

算法的时间复杂度定义为算法中可执行语句的频度之和，记为 T(n)。T(n)是算法所需时间的一种估计，其中 n 为问题的规模（或大小、体积），它有时为算法的输入量，有时为算法的计算量。如例 1－7 中，问题的规模 n 为矩阵的阶，该算法的时间复杂度为：

$$T(n) = (n+1) + n(n+1) + n^2 + n^2(n+1) + n^3$$
$$= 2n^3 + 3n^2 + 2n + 1$$

当 $n \to \infty$ 时，$\lim_{n\to\infty}(T(n)/n^3) = 2$，故 T(n)与 n^3 为同阶无穷大，或说 T(n)与 n^3 成正比、T(n)的量级为 n^3，记为：

$$T(n) = O(n^3)$$

对不同的场合，有时以 n 取不同值时算法平均所耗时间作为 T(n)（如在查找场合），有时又以算法最长所需时间作为 T(n)（如在排序场合）。当然，算法设计时，应使 T(n)的量级越小越好，以提高算法的执行速度。另外，在算法分析中关键要抓住一些循环语句的执行次数，而对一些循环之外语句的执行次数有时可忽略不计。

例 1－8 在数组(A[0],A[1],A[2],…,A[n－1])中查找第一个与给定值 k 相等的元素的序号。

算法描述如下：

```
int search(datatype A[n],k)
//在 A[n]中查找 k 的算法//
  {int i;
   i=0;
   while (i<n && A[i]! =k)
     i++;
   if (i<n) return i;
     else return(-1);
     }
```

本例应以平均查找次数（即查找时 k 与 A 中元素比较次数的平均值）作为算法的T(n)。