电子信息类规划教材 ·『九五』电子部教材·

数据结构

周岳山 主编

华中科技大学出版社

数据结构

主编　周岳山

华中科技大学出版社

图书在版编目(CIP)数据

数据结构/周岳山 主编
武汉:华中科技大学出版社,2003年1月
ISBN 7-5609-2926-5

Ⅰ.数…
Ⅱ.周…
Ⅲ.数据结构-高等学校-教材
Ⅳ.TP311.12

数据结构 周岳山 主编

责任编辑:曾 光 封面设计:刘 卉
责任校对:封春英 责任监印:熊庆玉

出版发行:华中科技大学出版社 武昌喻家山 邮编:430074 电话:(027)87545012

录 排:华中科技大学出版社照排室
印 刷:武汉市新华印刷有限责任公司

开本:787×1092 1/16 印张:15.25 字数:330 000
版次:2003年1月第1版 印次:2003年1月第1次印刷 印数:1—4 000
ISBN 7-5609-2926-5/TP·505 定价:19.50元

内 容 提 要

本书介绍了各种常见的数据结构，对数据结构的基本概念、逻辑结构和存储结构作了深入浅出的介绍，对各种结构的算法设计做了详细、通俗的讲述。本书还详尽地说明了在软件工程中大量存在的查找和排序问题。书中以框图形式和C语言来描述算法，具有一定的实用性。各章都配有小结和习题，书后附有上机实习题目和上机要求，最后还提出了课程设计的目的和技术要求，便于教学和实践操作。

本书可作为高职高专院校计算机课程教材使用，也可供从事计算机工作的科技人员自学或参考。

出版说明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》的文件精神，我们组织有关高等学校、中等专业学校、出版社、各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21世纪教学内容和课程体系改革的要求，编制了《1996—2000年全国电子信息类专业教材编审出版规划》。

本轮规划教材的出版是先由个人申报，经各学校、出版社推荐，然后由各专业教学指导委员会评选，并由我部教材办协商各专业教学指导委员会、出版社审核确定的。在本轮教材规划的编制过程中，注意了将教学改革力度较大，有创新精神，有特色风格的教材和质量较高、教学适应性较好、需要修订的教材以及教学急需、尚无正式教材的选题优先列入规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科建设具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的教师、同学和广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

电子工业部教材办公室

前　言

本教材系按电子工业部《1996—2000年全国电子信息类专业教材编审出版规划》，由全国中专计算机教学指导委员会编审、推荐出版。本教材由上海电子技术学校周岳山担任主编，黄大胜担任主审，王安君担任责任编委。

本教材的参考学时数为75学时，其主要内容为数据的逻辑结构、物理结构以及对各种结构所定义的运算、算法及应用。其中包括线性表、链表、数组、栈、队列、串、树和图结构以及相应的存储表示。本课程还介绍了程序设计中大量存在的查找和排序问题，简单介绍了文件的结构和存储形式。本教材偏重于数据结构的实用算法和算法的具体应用，对于大多数的算法都配有算法的流程框图，便于阅读和学习。教材中出现的算法均用C语言描述，重要的算法都给出了C语言的程序。在算法的描述中，以非递归程序为主，同时也给出了算法的递归表示。为了便于学习并提高对数据结构的应用和程序的调试，动手能力，在每章的末尾都有小结、应用举例和C语言程序设计的描述。全书的末尾还附有学生上机实习和课程设计的要求。使用本教材时应注意各种数据结构的基本概念、形式和存储表示，着重注意各种数据结构的算法实现及算法的基本应用。本课程是实践性较强的一门课程，要注意算法及算法应用程序的上机调试。

参加本教材编写的还有胡瑾华。参加审阅工作的曾玉崑、凌林海等同志都为本书提出了许多宝贵意见，在此表示诚挚的感谢。由于编者水平有限，书中难免还存在缺点和错误，殷切希望广大读者批评指正。

编　者

1999年2月

目　录

第1章 绪 论

当今世界已经进入了信息时代，自第一台计算机问世至今，计算机的发展已远远超出了人们对它前景的估计。计算机的应用已逐步深入到社会的各个领域，其加工处理的对象也从简单的数值、字符发展到具有一定结构关系的数据。

在情报检索、企业管理、数据库、网络通信、人工智能等领域，对数据结构的处理显得尤为重要。若想使计算机能高效率、高可靠性、准确地处理大量数据，就需要对数据的组织、数据内部的逻辑关系加以深入研究，这就促使了数据结构这门学科的形成和发展。

1.1 数据及其逻辑结构

数据结构作为一门独立的课程是从1968年才开设的。但数据结构课程中的有关内容，如表处理、串处理、树结构等早已散见于计算机的其他课程，如操作系统、编译原理、表处理语言等课程中，但其内容都缺乏规范性和系统性。随着计算机科学的发展和计算机应用的普及，计算机加工处理的对象已从早期的数值、布尔值扩展到字符串、表格、图像甚至语音等等。这些能被计算机存储、加工的对象被称为数据。在计算机技术发展的初期，计算机主要用于数值的计算，处理的对象大多是数量较少，结构简单，但计算过程复杂的数值数据，例如方程求解等等。当时程序设计的重点放在了程序设计技巧上。随着计算机科学技术的发展，计算机的应用领域不断扩大，数据这个概念的外延被大大地扩展了，越来越多的非数值数据需要处理，如财务管理中的大量表格，人事档案管理中的文字处理等等，这不仅涉及到数值类型的变化，而且还关系到数据中结构的处理和对于这些结构在程序设计中的描述。也就是说，数据包含数值和非数值两种类型。

数据通常由数据元素组成，数据元素是数据的基本单位，具有完整确定的实际意义，在程序中作为一个整体而加以考虑和处理。如表1-1所示的学生成绩表中，每个学生的数据用学号、姓名和各门学科的成绩来表示，这些具体的数值和字符都是数据元素。这些数据元素准确地表示了某个学生的学习成绩。

表1-1 学生成绩表

学号	姓名	Pascal语言	电工	程序设计	数据结构	微机原理
1	张平	82	76	91	82	74
2	李园	90	85	86	88	79
⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮
35	赵晶	75	82	78	75	83

在很多情况下，数据元素又由数据项组成，但数据项通常不具有完整确定的实际意义或不被当作一个整体对待。如在学生成绩表中，学号、姓名、Pascal 语言等内容都是数据项，所有数据项中的数据组成了数据元素。因此，每个学生的成绩是一个数据元素，成绩中的每一项都是数据元素中的数据项。显然，每个项目是作为成绩的一个成分出现的，但单独的项目没有完整确定的实际意义。例如，将表格中第一行的各个数据项拆开分别地看，则“1”、“张平”、“82”……数据分别表示一个学号、一个学生名、Pascal 语言课程的成绩，它们各自在学生成绩表中是一个个离散的数据，不能完整地表示某个学生的学习档案，但这些数据项的组合就构成了一个具有完整意义的某学生的学习情况。所以，在由多个数据项所构造的数据元素中，数据项不是数据元素，而每一个项目的组合则是数据元素。

但是，当有些数据元素不能再分解为数据项的组合，即该数据元素仅由一个数据项组成时，这个数据项就是该数据元素自身。

通过以上的示例可以看到：现实的世界是信息的世界，所谓信息就是客观存在的反映，而数据就是信息的表现形式和描述，这种描述是为了能被计算机所识别、存储和处理。而这种描述一般可归结为数字、字符和各种符号的集合。如上例中的学生、成绩等。值得注意的是我们研究的数据不是孤立的，而表现出了相互关联的特性。如某学生的姓名以及该学生的学科成绩都作为数据存在，其数据与数据之间的关系为：学生姓名与班级名对应，成绩与学科相匹配。这种数据元素之间存在的相互关系就是数据的结构。这种由设计者建立的数据之间的结构也称为数据之间的逻辑结构。

图 1-1 所示的为四类基本逻辑结构的示意图。它们反映了四类基本的数据组织形式。图中的小圆圈称为结点。一个结点代表一个数据元素。结点之间的连线代表逻辑关系，即相应数据元素之间的邻接关系。在图 1-1 中，集合中的任何两个结点之间都没有逻辑关系，组织形式松散，其结构为离散型；线性结构中结点按逻辑关系依次排列，形成一条“锁链”；树形结构具有分支、层次的特征；图状结构比较复杂，其中的各个结点按逻辑关系互相连接，任意两个结点都可以邻接。

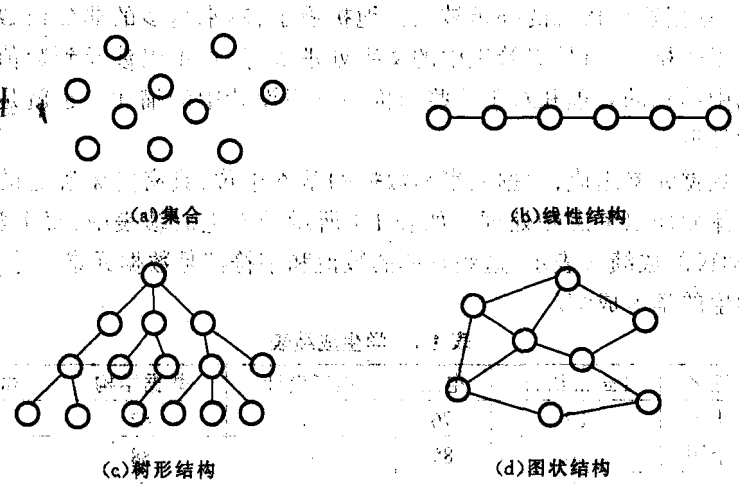

图 1-1 四类基本逻辑结构

以上的各种逻辑结构就是本书中将要详细介绍的数学模型。关于逻辑结构,有以下几点需特别注意。

① 逻辑结构与数据元素本身的形式、类型、内容无关。例如,在表 1-1 所示的学生成绩表中增加一个数据项"数据库原理及应用",就得到另一个数据。但由于所有的成绩表仍然是一个学生成绩接着一个学生成绩的线性关系,其数据与数据的逻辑关系依然未变。

② 逻辑结构与数据元素的相对位置无关。例如,学生成绩表中所有学生的数据是按照学号由小到大的顺序排列,如果现在按学生姓名的姓氏笔画重新排列,将得到另一个表格。由于新表格中的所有学生的排列仍是按"一个接一个"排列,其逻辑结构与原表格的逻辑结构相同,还是线性结构。

③ 逻辑结构与所含结点个数无关。如学生成绩表中增加或撤消若干个学生数据,经过适当的整理,所得表格仍为线性结构。

由此可见,不同的数据可以具有相同的逻辑结构。

把数据和表示数据之间的关系,即逻辑结构在计算机中的存储形式,称作数据的物理结构或存储结构。

1.2 数据结构和算法

数据结构(Data Structures)包括两方面的内容:数据和结构。数据结构就是一门研究数据的逻辑结构和物理结构,以及它们之间的相互关系和所定义的算法在计算机上运行的学科。认识数据结构在程序设计中的重要性是学习和研究计算机技术的一个重要基础。我们可以通过以下的例子加以简要说明。

最简单的数据结构的例子是一维数组。例如:要计算 100 个数 $a_1, a_2, \cdots, a_{100}$ 的累加和。

$$sum = a_1 + a_2 + \cdots + a_{100}$$

显然,用以上的赋值语句来描述求和的表达式是极不科学的。我们可以把这 100 个数以线性排列的结构组成一个一维数组 A[i]。在此基础上,计算这 100 个数累加和的程序段可描述为

```
sum = 0;
for(i = 1;i <= 100;i++ )sum = sum + A[i];
```

计算结果保存在 sum 中。这个程序段比 $sum = a_1 + a_2 + \cdots + a_{100}$ 这样的语句要简明得多,况且在程序设计中不允许书写省略号。这里实际上包括了两方面的内容:第一是数据结构 int A[100]表示的数组;第二是用数组作为数据结构的形式所描述的算法。从而达到使程序设计简明、易读的目的。

较复杂的一个例子是用数据结构中的树结构完成学校管理程序中的数据处理。图 1-2 给出了学校管理程序流程框图。

要用计算机实现对学校的管理,就要经课题调查、研究,按管理的模式确定程序的数据处理模式和流程,如图 1-2 所示,以及对具体部门确定数据的结构形式,如表 1-1 所示的学生成绩表。若在学生成绩管理中要查询某班、某学生的学习情况。其操作的逻辑关系为:从校部即根结点入口,选择查询的三个流向,确定"学生"这个结点的数据流;又通过"学生"这个结点选择某个年级,

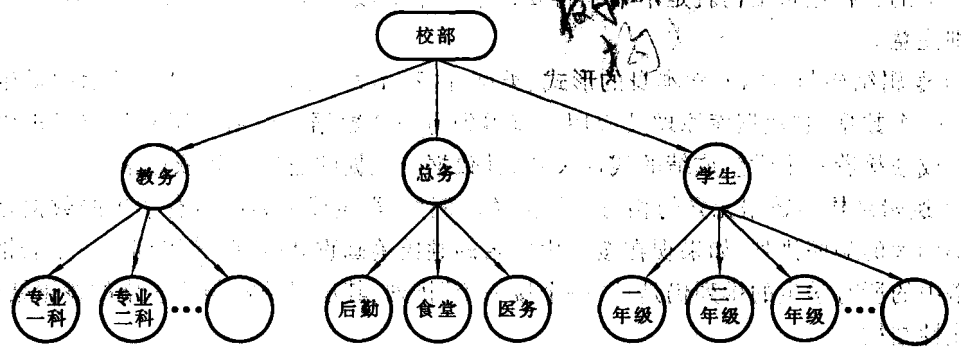

图 1-2 学校管理程序流程框图

再从某个年级这个结点最终确定需查询的某个班级的这个结点，该结点在数据结构中称作叶结点，而整个结构就称为树结构。这种以树的形式所表示的数据结构可以使整个管理的层次变得非常清晰，数据流向也一目了然。从程序设计角度来看，进行数据处理将很方便，程序的结构设计清楚，软件的维护和管理也容易。至于查询的最后一个结点，某班的学生成绩表就是数据结构中最常见的线性表。以线性表所表示的学生成绩很容易实现对某个学生的学习情况进行查询。总之用树和线性表来表示和实现学校管理应用软件是合适的，其数据的存储也是可行的。

至此可以认为：数据结构就是一门研究数据的逻辑结构和物理结构，以及它们之间的相互关系和所定义的算法如何在计算机上运行的学科。本书中将要讨论的逻辑结构有：线性关系的表、栈、队列、数组、串和链；非线性关系的包括树和图的结构。它们分别以顺序和非顺序的映像关系，在内存中表示不同的存储结构。本书还将讨论上述结构的算法和算法的基本应用，以及数据的查找和排序，同时附有相应的 C 语言所描述的程序。

1.3 算法语言的描述

数据结构既是一门理论性课程，更是一门实践性很强的课程。通过该课程的学习，应该掌握数据结构的基本概念和结构框架，以便更好地应用算法进行程序设计。在讨论各种数据结构的基本运算时，除了讨论和给出基本算法外，还给出相应的算法流程框图。流程框图不仅有文字说明，而且还配有对应的表达式，力图使读者在掌握和理解框图所表示的程序设计思想后，能较方便地用读者熟悉的计算机语言来编制源程序。

为了便于理解和掌握算法的设计思想和实质，选择一种合适的算法语言来进行描述是十分重要的。以结构化程序设计思想为基础的 C 语言不仅具有丰富的数据类型，也是一种能较好地体现结构程序设计原则的语言，而且 C 语言在计算机界使用广泛，为计算机工作者所乐意使用。本书中，对大多数的算法略去了对变量的说明和约定，又假设某些数据的存在。这样，读者可以把注意力集中到算法上来，而不是把精力花费在了解某种高级语言的许多具体的约定和数据输入上。如果希望在计算机上实现本书所描述的算法，可以很容易地通过人工对某种计算机语言的翻译或补充说明，转换成具体的能在机器上执行的源程序。当然，如果有一

个进行预处理的翻译器,由机器实现自动翻译,那就更方便了。

以下给出一个文件操作中数据匹配的算法,使用 C 语言进行描述。假如把文件中每个记录的关键值(区别每个不同记录的数据)均存放于数组中,且又假设该数据已输入指定的数组。要求对文件中的记录作成批处理。我们把要处理文件中的数据,称为主文件数据,记为 m(j),其中 $1 \leqslant j \leqslant q$,它是有序的,把由成批的待处理记录组成的文件称作事务处理文件,其事务处理数据为 t(i),$1 \leqslant i \leqslant p$,它也是有序的。若 m(j)和 t(i)的值相等,则说明文件的记录和事务处理文件的记录相匹配。相同数据的 m(j)和 t(i)合并处理后产生的另一新文件的记录为 n(k),r(l)是 t 文件中的非处理文件,u(h)是 m 文件中的非活跃文件。图 1-3 所示的是成批数据匹配流程图,图 1-4 所示的是数据匹配图。

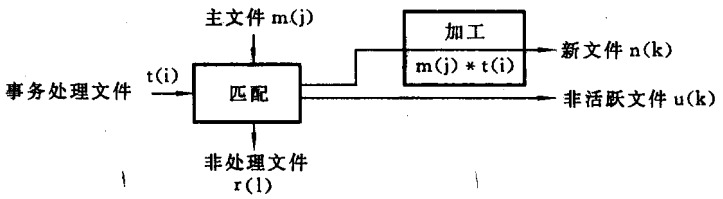

图 1-3 成批数据匹配流程图

这个成批数据匹配过程的算法可以按流程图和数据匹配的原则,按 C 语言的书写规则,编写成如下的算法,算法中文件记录内的数据都以数组表示。

若该算法中同时产生非活跃文件 u(h),则如何修改此算法,请读者在阅读该算法后自行修改和设计。

```
//数据匹配算法
PF(m[q],t[p])
//m 为主文件数组,t 为事务处理文件数组,r 为非处理文件数组,n 为新文件
{
    int k,j,i,l;
    k=l=0;j=i=1;
    while (i<=p)
    {    if (j<=q)
         if (t[i]==m[j])
         {          k++;
           n[k]=t[i]*m[j];
           j++;
           i++;
         }
         else if (t[i]<m[j])
            {          l++;
                 r[l]=t[i];
                 i++;
```

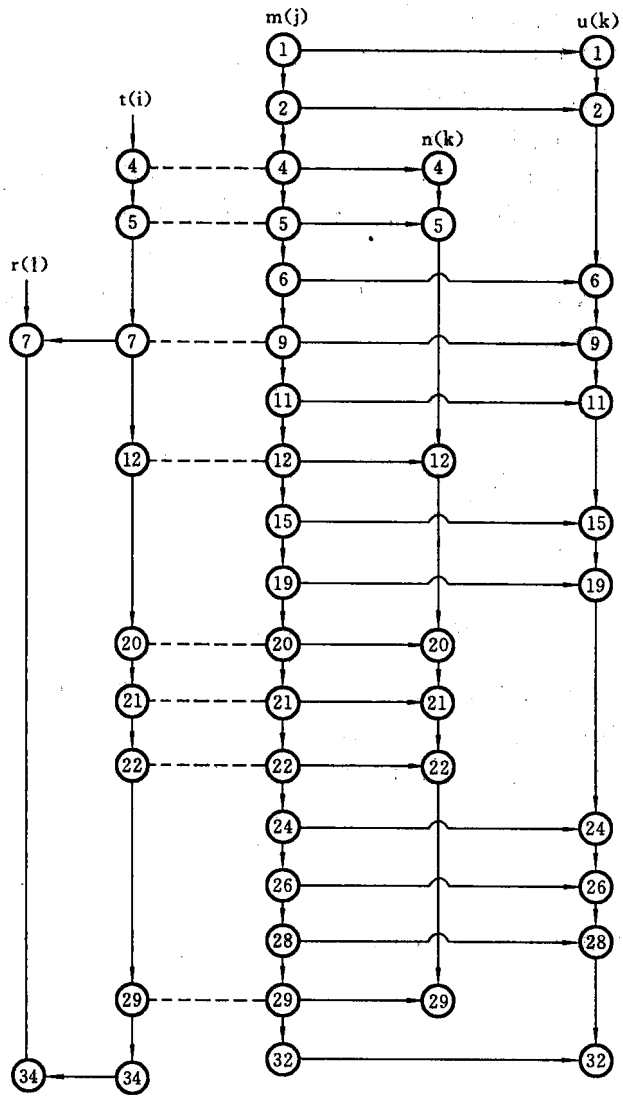

图 1-4　数据匹配图

```
        }
        else
            j++;
    else
    {     l++;
        r[l] = t[i];
        i++;
    }
}
```

1.4 存储实现和算法分析

在程序设计中,一旦确定了数据结构的逻辑组织形式,则数据在存储器中存储的实现就显得特别重要。存储实现的基本目标是建立数据的机内表示。根据已经建立的逻辑结构所选定的数据组织形式,其存储实现所建立的机内表示也必须遵循选定的逻辑结构。同时,由于逻辑结构不包括每个结点内的内容,即数据元素本身的表示,因此存储实现的另一方面是建立数据元素的机内表示,即数据的类型表示。所建立的数据的机内表示称为数据的存储表示。在通常情况下,所选定的存储结构包括以下三种主要的表现形式:

① 存储结点,每个存储结点存放一个数据元素;

② 数据元素之间关联方式的表示,其方式应符合逻辑结构的组织形式;

③ 附加结点,为便于算法设计、运算而设置的辅助结点或"哑结点"等。

在以上三种形式中,前两种的表示是所有存储结构都必须具备的,第三种形式则根据算法设计中的需要来决定是否设置。为了便于对数据结构具体类型的认识和研究,我们假定数据元素内部的组织形式比较简单,存储结点本身的组织也比较简单。比如说,在学生名单表中,其结构是线性表结构,但每个结点的数据元素只有一个数据项——姓名,即数据项的内容就是数据元素,本书主要选用这种方法来描述结点内容。由于每个存储结点存放一个数据元素,数据元素之间的逻辑结构即关联方式便由存储结点之间的关联方式间接地表达。一般情况下,存储结点之间有以下四种关联方式。

1) 顺序存储结构方式:设每个存储结点只存放一个数据元素。所有存储结点的内容相继存放在一个连续的存储区域内。用存储结点间的位置关系(顺序连续)表示数据元素之间的逻辑关系。按这种方式表示逻辑关系的存储结构称为顺序存储结构。

2) 链式存储方式:每个存储结点不仅含有一个数据元素,还包含一组指针指向一个与本结点有逻辑关系结点,用于表示数据之间的逻辑关系。按这种方式组织结点之间关联方式的存储结构称为链式存储结构。

3) 索引存储方式:设每个存储结点只包含一个数据元素,所有存储结点连续存放。同时增设一个结点位置索引表,索引表中的索引表示各存储结点的存储位置或某一存储区间的位置(区间的始点或终止位置)。按这种关联方式组织起来的存储结构称为索引存储结构。

4) 哈希(散列)存储方式:设每个结点存放一个数据元素,各个结点比较均匀地分布在存储区域中,一般各个存储结点之间不连续,用哈希函数来表示各结点的存储位置或存储区间的位置。按这种关联方式组织的存储结构称为哈希存储结构。

值得指出的是,本书着重介绍的存储实现和运算实现是数据结构实现、应用的主要任务,但并不是全部任务。在实际应用中,即使一个任务得以实现,即按某种逻辑结构实现了程序,也还需要对其进行算法或程序的测试和修改。

在计算机的程序设计中,算法分析是十分重要的,同时也是相当复杂的。本书对此内容仅作一般介绍。对于某算法如何取得算法分析的结果不作进一步、深入的推导。重要的是讲解

对已经获得的算法的结果如何在应用程序中加以使用。

如何衡量和评价一个算法的优劣呢？假如所设计的算法在逻辑上是可行的，那么，尽管评价算法的标准很多，通常还是从以下三个方面来考虑：

① 算法在计算机中运行所消耗的时间，即所需的机器时间，也称作时间因素或时间复杂性；

② 执行算法时，在计算机中所占存储容量的大小，即所需的存储空间，也称作空间因素或空间复杂性；

③ 所设计的算法是否易读、易懂，是否容易转换成其他可运行的程序语言。

从理想的角度讲，我们希望所设计的算法是一个既简单易懂，又占内存量少，运行时间短的程序。然而，在实际应用中，往往一个看来很简便的程序，其运行时间却比一个形式上复杂的程序要多一些，譬如科学计算中多阶方程的求解；运行一个时间较短的程序，往往占用的存储空间较大。因此，不同的应用对算法有不同的选择。若该程序只使用一次或若干次，则力求算法简明易读，容易转换成执行的语言，同时便于上机调试。若该程序需要反复多次运行或多次调用，则应选用执行时间尽可能短的算法。若待处理的数据量甚大，而所使用的计算机存储空间相对较小，则选择的算法应主要考虑如何节省存储空间。考虑到算法实际分析的多重性和复杂性，本书主要讨论算法的时间因素，以及如何简要计算该算法或程序的具体执行时间和估算得出该算法在执行时间上的量级。

一个算法的执行时间等于算法所有语句执行时间的总和。而任一语句的执行时间是该语句的总执行次数与执行一次所需时间的乘积。由于精确计算出该算法的总执行时间是相当复杂的，而且是很困难的。因此，只能根据给定的计算模式，粗略地估算该算法在执行时间上的量级。这里，我们用大写的字符“O”来表示量级的概念。

设有程序段表示两个 $n \times n$ 的矩阵相乘，其算法描述如下：

```
for (i = 1;i <= n;i ++ )                                  /* n + 1 */
  {for (j = 1;j <= n;j ++ )                               /* n(n + 1) */
    { c[i,j] = 0;                                         /* n^2 */
    for(k = 1;k <= n;k ++ )                               /* n^2(n + 1) */
      {c[i,j] = c[i,j] + a[i,k] * b[k,j];                 /* n^3 */
      }                                                   /* n^3 */
    }                                                     /* n^2 */
  }                                                       /* n */
```

以上程序段的语句总执行次数 X_n 表示为

$$X_n = (n+1) + n(n+1) + n^2 + n^2(n+1) + n^3 + n^3 + n^2 + n = 3n^3 + 4n^2 + 3n + 1$$

用衡量执行时间的量级来表示为

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

其含义是：执行次数 X_n 将不超过 n^3 的某个常数倍或 X_n 的量级与 n^3 成正比。

那么如何分析不同算法的执行时间量级呢？$O(1)$表示算法的执行时间为常量，称该算法为常量阶的；$O(n)$表示算法的执行时间为线性的，称为线性阶；$O(n^2)$为平方阶，$O(2^n)$为指数阶。若两个算法分别为 $O(1)$和 $O(n)$，当 n 充分大时，显然 $O(1)$的执行时间要少，整个算法的