数据结构

SHU JU JIE GOU

王若梅　贺晓军　编

西安电子科技大学出版社

数据结构
王若梅　贺晓军　编
西安电子科技大学出版社

数 据 结 构

王若梅 贺晓军

西安电子科技大学出版社

1994

（陕）新登字 010 号

内 容 简 介

本书比较系统地介绍了数据结构的基本内容和查找、排序的各种方法，对于数据结构中的每一种类型，在进行了详细的基本内容、基本算法描述之外，还列举了具体实例加以练习；对查找和排序中的每一种方法，除给出了实现的算法之外，还进行了性能分析；最后还讨论了文件及其组织。

本书简明扼要，概念清楚，内容适中，并通过大量例题帮助理解，全部算法用类 PASCAL 描述。每章后面配有大量的习题，便于教学和学生自学。

本书可作为大专院校计算机专业的教科书，也可作为从事计算机应用的工程技术人员的自学参考书。

数 据 结 构

王若梅　贺晓军　编

责任编辑　汪海洋

西安电子科技大学出版社出版发行

陕西省富平县印刷厂　印刷

新华书店经销

开本 787×1092　1/16　印张 13 8/16　字数 318 千字

1994 年 5 月第 1 版　1994 年 5 月第 1 次印刷　印数 1—8 000

ISBN 7－5606－0330－0/TP・0118　　定价：8.80 元

前　言

随着计算机技术的飞速发展，计算机技术的应用也日益扩大，越来越多的人需要掌握计算机技术的基本知识。《数据结构》是计算机技术中一门专业基础课，在我国计算机各专业教学中，它是核心课程之一。

本书共分十章，第 1 章通过实例回答了什么是数据结构，解释了数据、数据类型及数据结构、算法等基本术语，并对算法、算法描述及算法分析作了简单说明；第 2 章到第 5 章介绍了线性结构中线性表、栈、队、串、数组及广义表的基本定义和基本算法；第 6 章、第 7 章介绍了非线性结构树和图；第 8 章、第 9 章讨论了查找和排序的基本算法并简单进行了性能分析；第 10 章讨论了文件及其组织。全书力求内容与实例相结合，总是先给出具体内容，然后建立实现的算法，此外再举出应用了具体内容的例题，有利于学生加深对内容的掌握。

本书为了便于自学，在文字描述上力求通俗易懂，实例列举恰当，并附有习题作为练习。作为大专教材，讲授学时为 54 学时，上机实习时数为 12～20 学时。希望学生通过学习本课程，学会分析研究计算机处理的数据对象的特性，来选择适当的数据结构和存贮结构建立实现的算法，并能对算法进行性能分析，因此要求学生除了学习数据结构的基本内容之外，还应多做练习。

本书第 1 章至第 8 章由王若梅编写，第 9 章和第 10 章由贺晓军编写。在本书编写过程中，得到了侯伯亨教授的热情帮助，并得到了计算机系以及出版社有关老师的大力支持，在此谨向他们表示衷心感谢。由于编者水平有限，加上时间匆促，错误和不当之处在所难免，敬请广大读者和讲授此课的老师们批评指正。

编　者

1994.4 于西安

目　录

第 1 章　绪　　论

计算机技术的飞速发展已远远地超出了人们对它的估计，它的应用范围也日益扩大，如今，计算机已不仅仅用于科学计算，而是更多地用于进行数据处理和实时控制。同时，计算机加工处理的对象，也从纯粹的数值发展到字符、声音、图像等各种复杂的且具有一定结构的数据。因此，在进行程序设计的过程中，除了应用一些程序设计的技巧和方法外，必须研究数据的特性和数据之间存在的内在关系，才能设计出优质的程序。《数据结构》这门学科就是在这个背景下形成和发展的。

1.1　什么是数据结构

让我们先对全书中使用的名词和术语赋以确定的含义。

数据是描述客观事物的数、字符，以及所有能够输入到计算机中并被计算机程序处理的符号的集合。它是信息的载体，计算机程序加工的“原料”。例如，在数值计算中所使用的数据是整数和实数；文本编辑程序中使用的数据是字符串。随着计算机技术的发展，计算机应用的普及，数据的含义也变得广泛了，如今，图像、声音等也可以经过一定的输入变换成为数字化的信息数据。利用计算机进行加工和处理，所以，它们也属于数据的范畴。

数据元素是数据的基本单位，表现为数据这个集合中的一个客体。一个数据元素通常的表现形式为由一个或若干个数据项组成。数据项是具有独立含义的最小标识单位。例如，在人事管理中的每一个人就是一个数据元素，组成它的数据项可以是姓名、年龄、性别、工资和简历等。数据元素有时也称为结点或记录。

数据对象是具有相同特性的数据元素的集合。例如，人事管理中的数据对象就是该部门的全体人员。

通常，我们还会谈到数据元素的逻辑关系，也称为数据的**逻辑结构**，它可以看作是从具体问题中抽象出来的数学模型，表现出数据元素之间体现的逻辑关系。我们将数据元素及其关系在计算机中的存贮表示称为数据的**存贮结构**，也称为物理映象。由于存贮方式的不同使得数据的存贮结构又分作顺序存贮和非顺序存贮，具体的内容将在后面章节中详细介绍。

为了给出数据结构的概念，下面先讨论两个例子。

例 1　学生成绩表如图 1.1 所示：

学　号	姓　名	性　别	课　名	成　绩
……				
389011	李　红	女	英　语	69
389012	赵　军	男	英　语	71
389013	张　刚	男	英　语	80
……				

图 1.1　学生成绩表

在这张表中，每一行中描述了一个人的有关信息，由学号、姓名、各科成绩等数据项组成，即每一行表示一个数据元素。表中数据元素之间的逻辑关系是顺序的，除了第一个元素和最后一个元素之外，其余元素有且仅有唯一的一个与它相邻且在它前面的元素(称为**直接前趋**)和唯一的一个与它相邻且在它后面的元素(称为**直接后继**)。例如表中“赵军”所在的这个数据元素的直接前趋为“李红”这个数据元素，直接后继为“张刚”这个数据元素。对于满足这种顺序关系的表在计算机中如何进行存贮表示则是存贮结构研究的内容，根据不同的方式可采用顺序存贮与非顺序存贮。另外，在这张表中可能要经常查阅某一学生的成绩，如有新生加入时要增加数据元素，或有学生退学时要删除相应元素。因此，进行查找、插入和删除就是数据的运算问题。把上表中数据的逻辑关系、存贮结构和运算这三个问题搞清楚，也就弄清了学生成绩表这个数据结构，从而可以有针对性地进行问题的求解。

例 2　研究城市交通网络问题。如图 1.2 所示。

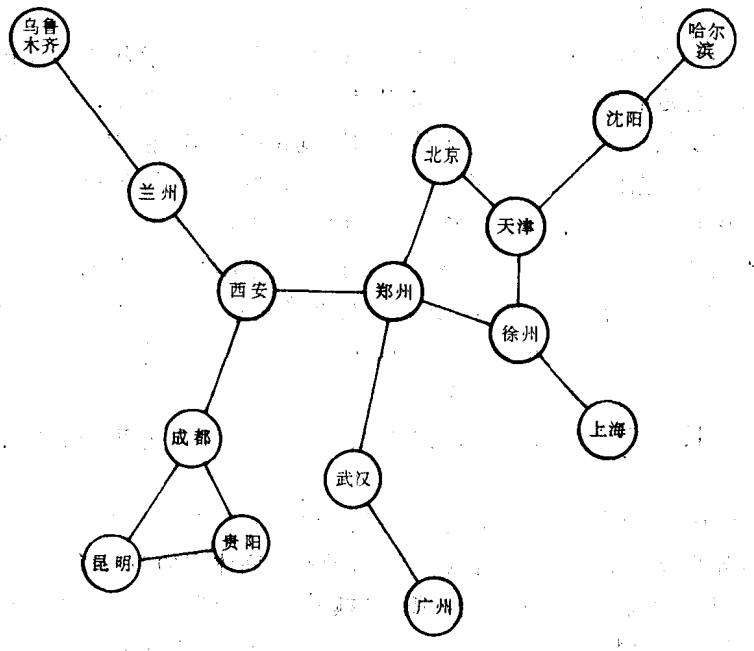

图 1.2　城市交通网例图

图中的圆圈表示城市，边表示城市间的交通。在这个问题中，每一个城市就是一个数据元素，它们之间的逻辑关系不再是顺序的，而是一种称为图的结构。这样一种复杂的结构如何在计算机中存贮表示也是有关存贮结构研究的问题。此外，仍然会有数据的运算，如求城市间的距离，求某两个城市间的最短路径等。可见，数据元素间的逻辑结构、存贮结构和运算这三个问题也是我们要在求解问题之前首先搞清楚的问题。这些问题弄清楚，问题也就易于解决了。

通过以上例题，我们发现在实际应用中，有些问题的数学模型不是数值方程，而是一些表、图、树等，因此可以通过下面的描述来理解数据结构的内涵。

数据结构指的是数据元素之间的相互关系。虽然至今没有一个关于数据结构的标准定

义，数据结构可以被理解为包含有以下三个方面的内容：即数据结构是指数据元素之间的逻辑结构、存贮结构及其数据的抽象运算。

把在数据上所施加的一系列操作称为抽象运算是指只考虑这些操作的功能，而不去考虑如何完成。只有在确定了存贮结构之后，才具体考虑实现这些操作。

我们常将数据的逻辑结构简称为数据结构。数据的逻辑结构又分为线性结构和非线性结构。线性结构的逻辑特征是有且仅有一个开始元素和一个终结元素，除第一个元素和最后一个元素之外，其余元素有且仅有唯一的一个直接前趋和直接后继。本书的第二章到第四章介绍的内容就是线性结构。非线性结构的逻辑特征是一个结点可能有多个直接前趋和直接后继，第五章到第七章的内容都是非线性结构的数据结构介绍。

由于存贮方式的不同，数据的存贮结构有顺序的和非顺序的两种。同一种逻辑结构可以选择不同的存贮结构来表示，而数据的运算在具体实现时也依赖于所选取的存贮结构，因此，数据的逻辑结构、存贮结构及其抽象运算三者是不可分割的一个整体，在学习过程中，要注意它们之间的联系，不可独立地去理解一个方面。

在数据结构中往往涉及数据类型的概念，虽然有些叫法与数据结构相同，如数据结构中的数组与数据类型中的数组，数据结构中的串与数据类型中的串等，但二者的意义不相同，可以说数据类型是数据结构在程序设计语言中的实现。

正是由于数据的逻辑结构、存贮结构和抽象运算是有密切关系的，所以我们也常常将同一种逻辑结构的不同存贮结构用不同的数据结构名称来标识，如线性表这种逻辑结构用顺序存贮方式存贮时，称该结构为顺序表，而用链表进行存贮表示时，则称该结构为链表。同样，在给定了数据的逻辑结构后，若定义的运算集合及运算性质不同时，也可导致不同的数据结构，如对线性表的插入和删除运算限制在表的一端进行，则产生了栈和队的结构。

1.2 学习数据结构的意义

数据结构是计算机软件和应用专业的核心课程，随着计算机技术的发展，计算机应用的普及，仅掌握计算机语言和程序设计的技巧与方法，而缺乏有关数据结构的知识，就难以应付众多复杂的课题，不能有效地使用计算机。

《数据结构》作为一门独立的课程是从 1968 年开始的。在此之前，数据结构的有关内容是分散在操作系统、编译原理和表处理语言等课程之中，直到 1968 年，美国的一些大学才在计算机科学系把数据结构规定为一门独立的课程，但也并没有对课程的范围作明确的规定。当时，图论，特别是表和树的理论与数据结构几乎是同义语。此后，数据结构这个概念被扩充到包括网络、代数、集合论和关系等内容中。由于数据必须在计算机中进行处理，所以不能只考虑数据本身的数学性质，还必须考虑数据的物理结构，这就进一步扩大了数据结构的内容。随着数据库系统的不断发展，文件管理的内容也被加入到数据结构课程中。美国研究计算机科学的著名教授 D. E. Knuth 所著的《计算机程序设计技巧》第 1 卷《基本算法》是一本较系统地阐述数据的逻辑结构、存贮结构及其运算的著作。从 70 年代中期到 80 年代初，各种版本的数据结构著作相继问世。作为一门学科，数据结构是新兴的，正值方兴未艾，蓬勃发展的阶段，其理论、应用范围和研究对象等内容正在不断扩充和深化。

数据结构在计算机科学中有着十分重要的地位，它与计算机软件、硬件以及数学有着密切的关系，是操作系统、编译原理、数据库和人工智能等课程的基础，广泛应用于信息科学、系统工程、应用数学以及各种工程技术领域。从计算机发展的初期，在数值计算问题中程序设计者的主要精力集中于程序设计技巧上，而不重视数据结构到当今非数值性问题中，数据元素之间由简单联系变为往往无法用数学方程进行描述的复杂结构，人们认识到程序设计的实质是对确定的问题选择一种好的结构从而设计出一种好的算法，数据结构受到了极大的重视。著名的瑞士计算机科学家 N. Wirth 教授指出：算法＋数据结构＝程序，这里算法指的是对数据运算的描述，数据结构指数据的逻辑结构和存贮结构。可见程序设计的实质是对实际问题选择一种好的数据结构，从而设计出一个好的算法有效地解决问题。而好的算法在很大程度上与所取的数据结构有关。下面我们通过实例来说明这个问题。

例 3　电话号码查询问题，登记表如图 1.3 所示。

问题的要求是查询任意的姓名，若该人有电话，则迅速找到其电话号码，否则指出该人没有电话。首先要构造一张电话号码登记表，表中由姓名和电话号码二个数据项构成。如图 1.3 所示。问题的关键是这张表如何组织。最简单的方法是将每个人的信息顺序地存贮表中，查找时从头开始依次查对姓名，直到找出正确的姓名或找遍整个表均没有找到为止。这种查找算法不适用于有成千上万电话的情况，不是一个好的算法。那么从数据结构的角度考虑，应该对表的组织和存贮加以考虑，可将表按姓氏排列，另外还可建立一张姓氏索引表，采用图 1.4 所示的存贮结构。

姓　名	电话号码
赵　力	69811
王　平	71101
丁　一	92911
李　华	31000
⋮	⋮

图 1.3　电话号码查询中的登记表

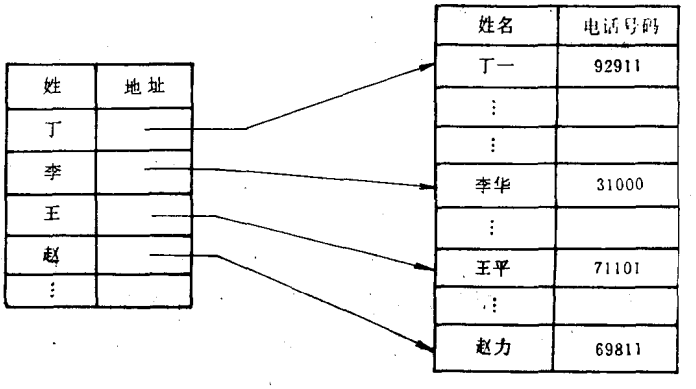

图 1.4　电话号码查询中的索引存贮

在这种存贮方式下进行查找，可先进行索引表中姓氏的查找，然后根据索引表中的地址到电话登记表中进行姓名的查找，而其它姓氏的姓名就可以不进行查找，显然这种查找算法更为有效。

从这个例中不难看出，解决问题的关键是选取适当的数据结构来表示问题，这样才能写出有效的算法。

1.3　算法的描述和算法分析

通过数据结构的定义以及在实际具体问题中数据结构知识的应用，我们发现算法的概

念始终起着重大的作用。的确，在实际应用中，为了求解一个问题，往往要在选取适当的数据结构之后，要给出一个具体的算法，经过计算机编程后，让计算机来帮助解决。下面我们先讨论一下算法的概念。

1.3.1 算法的概念

在数据结构的概念中，数据的运算是通过算法来描述的，因此，讨论算法是数据结构课程的重要内容之一。

算法是由若干条指令组成的有限序列，它必须满足以下性质：

(1) 输入性：具有零个或多个输入量，即算法开始前对算法的初始量。

(2) 输出性：至少产生一个输出。

(3) 有穷性：每一条指令的执行次数必须是有限的。

(4) 确定性：每条指令的含义必须明确，无二义性。

(5) 可行性：每条指令都应在有限的时间内完成。

例如，给定两个正整数 m 和 n，考虑求它们的最大公因子(m, n)的问题。

求解这个问题通常所用的方法为辗转相除法，西方称为欧几里得算法。我国数学家秦九韶在《数书九章》中也记载了这个方法。下面用三个计算步骤描述这个算法：

(1) 求余数：以 n 除 m，余数为 r，$0 \leqslant r < n$。

(2) 判余数是否等于零：若 $r=0$，输出 n 的当前值，算法结束；否则执行第三步。

(3) 更新被除数和除数：$n \rightarrow m$，$r \rightarrow n$，执行第一步。

也可以用图 1.5 所示的流程图来描述该算法。

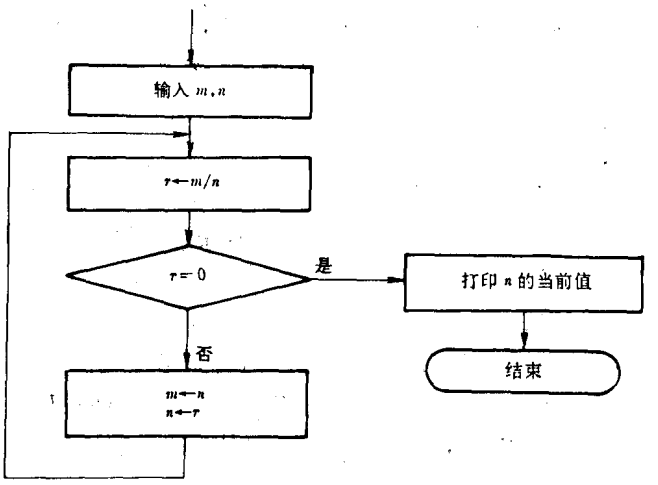

图 1.5 欧几里得算法流程图

上述计算过程给出了三个计算步骤，而且每一步骤意义明确，切实可行，虽然出现循环，但 m 和 n 都是给定的有限数，每次相除后得到的余数 r 若不为零，也总有 $r < \min(m, n)$，保证了循环经过有限次以后，必会终止。因此上述计算过程是一个算法。

可以想到许多能称为算法的计算过程，但一个算法并不等同于一个计算机程序，只是类似于方法、过程和程序等。算法与过程的主要区别在于算法经过有限步计算执行后必会

终止，而过程则可以是有穷的，也可以是无穷的。算法与程序的主要区别在于一个程序往往是指用某种机器语言书写的一个计算过程，但一个算法并不一定表现为一个计算机程序，它可以采用多种语言和方式描述。

例如计算机的操作系统，只要系统不遭破坏，它就永远不会停止，即使没有要处理的作业，它也处于一个等待循环中。可见算法的有穷性没有满足，它不是一个算法。

进行算法设计的步骤可概括为：

(1) 问题的陈述。为了设计某一问题的求解算法，首先必须了解问题的实质。即已知条件是什么，要求回答什么。

(2) 模型的选择。当问题陈述清楚之后，选择描述问题的数学模型是非常重要的。模型的适当与否直接影响算法的效率。模型的选择无公式可循，靠的是设计者的知识结构、工作经验等。

(3) 算法设计和正确性证明。一旦确定了数学模型，就可以进行算法设计。设计者要充分发挥主观能动性，运用已有的知识和抽象思维，从千头万绪之中，逐渐形成算法的基本思想，勾勒出一个算法的具体步骤。算法设计完之后，其正确性如何是十分重要的，必须加以证明。如何证明，在计算机的另一门课程《算法分析与设计》中有详细讨论。

(4) 算法的程序实现。将一个算法正确地编写成一个计算机程序，叫做算法的程序实现，这一转换并不是一件简单的工作。要求程序设计者有很好的程序设计基础，掌握多种程序设计方法和技巧。有关一个算法的程序实现，还包含有更多的方面，这里不一一讨论。

(5) 算法分析。算法设计和算法分析虽然有必然的联系，但仍是两个不同的任务。算法设计的中心任务是设计出一个求解的算法，不必过多地考虑优劣，而算法分析恰恰是要研究各种算法的特性和好坏。

算法设计始终是一种复杂而艰苦的劳动，只有多学习一些典型的算法，掌握一些基本算法的思想，才能得心应手地设计算法，而进行算法分析则需要一定的数学基础。

1.3.2　算法的描述和算法分析

一个算法可以采用人类的自然语言描述，也可以采用图表的方式，如果用各种计算机语言描述，就表现为一个程序。我们最感兴趣的还是用计算机语言来描述和在计算机上执行各种算法。

由于用各种程序设计语言书写一个正规的程序需作繁琐的变量说明，语句上的要求也比较严格，本书选用了一种类似于 PASCAL 的伪语言来描述算法(称为类 PASCAL 语言)。它的特点是借助于 PASCAL 语言的语法结构，附之以自然语言的叙述，从而使编写的算法具有良好的结构，又不拘泥于具体语言的限制。另外，这样描述的算法易读易写，而且也易于转换成某种语言的计算机程序。

下面对类 PASCAL 语言作一说明：

(1) 所有算法都以如下过程或函数的形式表示：

```
procedure　过程名（参数表）;
  begin
    语句组
  end;　　{过程名}
```

```
function  函数名(参数表)：类型名；
  begin
    语句组
  end；    {函数名}
```

其中的参数表可含有若干个值参和变参，值参和变参选择类似于 PASCAL 语言。语句组由一个或多个语句组成，两个语句之间用“;”进行分隔。正常情况下，过程结束于 end，也可以用 return 跳出过程。

(2) 除过程或函数中的参数表外，其它有关在过程中以局部变量出现的变量的说明均可以省略，假定该语言允许一切可能出现的变量种类。

(3) 基本语句有：

①赋值语句：

```
变量名 :=表达式；
```

②条件语句：

```
if  条件  then  语句组；
或  if  条件  then  语句组 1  else  语句组 2；
```

③循环语句：

```
while  条件  do  语句组
或  repeat  语句组  until  条件；
或  for  变量 :=初值  to  终值  do  语句组；
或  for  变量 :=初值  downto  终值  do  语句组；
```

前一个 for 语句的初值大于或等于终值，步长为 1，后一个 for 语句中终值小于或等于初值，步长为－1。

④情况语句

```
case
  条件 1：语句组 1；
  条件 2：语句组 2；
  ⋮
  条件 n：语句组 n；
  (else  语句组 n+1)
end；
或  case  表达式  of
    常量 1：语句组 1；
    常量 2：语句组 2；
    ⋮
    常量 n：语句组 n；
    (otherwise：语句组 n+1)
end；
```

以上所见的语句组或语句组 1、语句组 2 等，当其中的语句多于一个语句时，则用方括号将这些语句括起来。

⑤过程或函数调用

call　过程名(参数表)；(有时也可直接写过程名)

变量 :=函数名(参数表)；

允许嵌套和递归调用。

⑥错误处理

error(字符串)；

⑦跳出循环

exit；

该语句终止循环体的语句执行过程，直接跳到循环体之后的第一个语句处。

⑧读写语句

read(变量表)；

write(变量表)；

变量表中的变量之间用逗号隔开。

⑨ 注释形式

为了便于对于一个算法附加相应的注释，可在算法的适当处表示为

　　{字符串}

例如将上述描述的辗转相除法用类 PASCAL 描述如下：

```
procedure EUCLID(n, m : integer; var r : integer);
  begin
    r :=0
    while n≠0 do
      [
        r :=m mod n;
        m :=n;
        n :=r   ]
  end;  {EUCLID}
```

今后我们所遇到的问题通常是首先选择恰当的数据结构和设计一个算法，然后再进行编程。那么如何衡量一个算法的好坏呢？

前题之一是所设计的算法是"正确的"，其二还必须考虑执行算法所耗费的时间；执行算法所耗费的空间，主要指辅助空间；算法的易读、易编码以及易于调试等性质。

显然我们希望所选的算法占用空间小，运行时间短，其它性能也好，但实际上是很困难的，计算机的时间和空间这两大资源往往相互抵触，缩短算法的运行时间往往以牺牲较多的空间为代价，而为了节省空间又可能以耗费更多的时间为条件。因此，到底如何选取应根据具体算法的应用而定。

对于反复使用的算法应选择运行时间短的算法，而使用次数少的算法可力求简明、易于编写和调试为主，对于那种数据量较大的算法可从如何节省空间的角度考虑。

本书主要考虑时间的特性。

一个算法转换为程序后所耗费的时间除了与所用计算机的软、硬件有关外，还取决于程序中指令重复执行的次数，也就是语句的频度。例如，两个 $n\times n$ 的矩阵相乘，其算法可

描述为

```
for i := 1 to n do                                  {n+1}
  for j := 1 to n do                                {n(n+1)}
    [ c[i, j] := 0;                                 {n²}
      for k := 1 to n do                            {n²(n+1)}
        c[i, j] := c[i, j] + a[i, k] * b[k, j]]     {n³}
```

其中每一条的频度说明在注释中。我们把算法所耗费的时间定义为该算法中每条语句频度之和，则上述算法的时间耗费为

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

显然它是矩阵的阶 n 的函数，并且当 $n \to \infty$ 时，$T(n)/n^3 \to 2$，这表示当 n 充分大时，$T(n)$ 和 n^3 是同阶的，引入“O”记号(读作“大 O”)，可记 $T(n)=O(n^3)$。称为算法的**时间复杂度**。

程序分析法则：

(1) 执行一条读写或赋值语句用 $O(1)$时间；

(2) 依次执行一系列语句所用时间用求和准则；

(3) if 语句的耗时主要是执行语句所用的时间，检验条件还需用 $O(1)$；

(4) 循环语句的运行时间为多次迭代中执行循环体以及检验循环条件耗时，常用乘法计算。

对于较复杂的算法，我们可以将它分成几个容易估算的部分，然后利用“O”的求和原则和乘法原则计算整个算法的时间复杂度。

大“O”下的求和准则为：若算法的两个部分的时间复杂度为 $T_1(n)=O(f(n))$和 $T_2(n)=O(g(n))$，则 $T_1+T_2=O(\max(f(n), g(n)))$。又若 $T_1(m)=O(f(m))$，$T_2(n)=O(g(n))$，则 $T_1+T_2=O(f(m)+g(n))$。

大“O”下的乘法准则：若算法的二个部分的时间复杂度为 $T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$，则 $T_1 \cdot T_2=O(f(n) \cdot g(n))$。

看下面的程序段：

```
(1)     s := 0;
(2)     for i = 1 to n do
(3)       for j = 1 to n do
(4)         s := s + 1;
```

执行一条赋值语句与 n 无关，时间复杂度为 $O(1)$，$T_4(n)=O(1)$，对于第(3) 条语句，$T_3(n)=O(n)$，因此 $T_4(n)+T_3(n)=O(n)$，第(2)条语句 $T_2(n)=O(n)$，而(2)与(3)、(4)是循环嵌套，故有 $T_2(n) * (T_3(n)+T_4(n))=O(n^2)$，对第(1)条语句 $T_1(n)=O(1)$，它与其它语句之间关系是顺序执行，$T_1(n)+T_2(n) * (T_3(n)+T_4(n))=O(n^2)$。

实际上程序运行的时间不仅依赖于问题的规模，还与它处理的数据状态有关。一般在不做任何说明的情况下，指的是其最坏的情况。此外，衡量一个算法的好坏的另一个因素是程序运行时所占用的存贮量。

常见的时间复杂度有常数阶 $O(1)$、对数阶 $O(\log_2 n)$、线性阶 $O(n)$、线性对数阶 $O(n \log_2 n)$、平方阶 $O(n^2)$、立方阶 $O(n^3)$、指数阶 $O(2^n)$等等。图 1.6 展示了不同数量级的

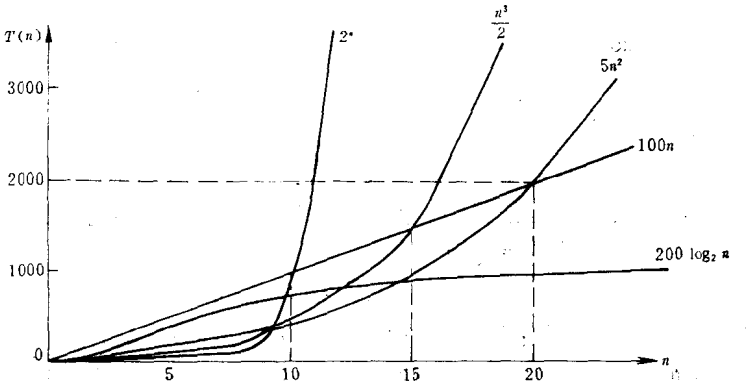

图 1.6　几种常见的函数增长率

时间复杂度，可以看到，指数阶的时间复杂度在 n 值稍大时就无法使用。

习　题　一

1. 简述下列名词术语：数据、数据元素、数据结构、逻辑结构、存贮结构、线性结构、非线性结构、算法。

2. 试比较类 PASCAL 语言和标准 PASCAL 语言的差异。

3. 试写一个算法，输出读入的三个整数 X、Y 和 Z 中的最大值。

4. 试编写一个算法，统计输入的 10 个整数中奇数和偶数的个数。

5. 设 n 为正整数，写出下列程序段的时间复杂度。

(1)
```
for i=1 to n do
  [s :=0;
    for j=1 to n do
      s :=s+j * i;]
```

(2)
```
for i=1 to n do
   for j=1 to n do
     for k=1 to n do
       x :=x+i+j+k;
```

(3)
```
k :=0; s :=0;
  repeat
    s :=s+k;
    k :=k+1;
  until (k=n);
```

(4)
```
k :=1;
  s :=0;
  while s<=n-1 do
  [ k :=k+s * 6;
   s :=s+1; ]
```

```
    write (s, k);
(5) for i=1 to n do
    for j=1 to i do
      for k=1 to j do
        s=s+1;
```