高职高专计算机教材精选

普通高等教育“十一五”国家级规划教材

数据结构

（第2版）

张世和 徐继延 编著

清华大学出版社

普通高等教育“十一五”国家级规划教材

高职高专计算机教材精选

数据结构（第2版）

张世和 徐继延 编著

清华大学出版社
北京

内 容 简 介

本书是《数据结构》的第 2 版。全书对常用的数据结构做了系统的介绍，力求概念清晰，注重实际应用。主要内容包括：数据结构的基本概念；算法描述和算法分析初步；线性表、堆栈、队列、串、数组、树、图等结构；排序和查找的各种方法；另外还用一章的篇幅详细介绍了链式存储结构以加深读者的理解。每一章后面均列举了典型应用实例，并配有算法和程序以供教学和实践使用。

本书作为“高职高专计算机教材精选”之一，主要面向高职高专院校计算机类专业的学生，也可以作为大学非计算机专业学生的选修课教材和计算机应用技术人员的自学参考书。

图书在版编目（CIP）数据

数据结构/张世和，徐继延编著. —2 版. —北京：清华大学出版社，2007.9
（高职高专计算机教材精选）
ISBN 978-7-302-15252-1

Ⅰ. 数…　Ⅱ. ①张…　②徐…　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(2007)第 073442 号

责任编辑：谢　琛
责任校对：梁　毅
责任印制：李红英

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969
印 装 者：北京市清华园胶印厂
经　　销：全国新华书店
开　　本：185×260　　**印　张**：12　　**字　数**：273 千字
版　　次：2007 年 9 月第 2 版　　**印　次**：2007 年 9 月第 1 次印刷
印　　数：1～5000
定　　价：18.00 元

前言

“数据结构”是计算机程序设计的重要理论基础，是计算机及其应用专业的一门重要基础课程和核心课程。它不仅是学习后继软件专业课程的先导，而且已成为其他工科类专业的热门选修课程。

本教材第1版列入“教育部高职高专规划教材”，第2版列入“普通高等教育‘十一五’国家级规划教材”，主要面向高职高专院校计算机类专业的学生，培养技术应用性人才。教材内容的构造力求体现“以应用为主体”，强调理论知识的理解和运用，实现专科教学以实践体系为主及以技术应用能力培养为主的目标。

本书共分9章。第1章阐述数据、数据结构和算法等基本概念。第2～7章分别讨论了线性表、栈、队列、串、数组、树和二叉树以及图等基本数据结构及其应用，其中，第3章专门总结了链式存储结构的基本概念和应用，为学好后面各类数据结构打好扎实的基础。第8～9章讨论查找和排序的各种实现方法和实用分析。

第2版教材对第1章至第8章中的“应用举例及分析”进行了大量实用例子的补充和调整，对每章的习题作了大量补充，并增加了实训题供学生独立完成。每章习题的参考答案汇集在配套的《数据结构习题解析与实训(第2版)》中。

本教材的特点有：

(1) 对基础理论知识的阐述由浅入深、通俗易懂。内容组织和编排以应用为主线，略去了一些理论推导和数学证明的过程，淡化算法的设计分析和复杂的时空分析。

(2) 各章(除第9章)都配有“应用举例及分析”一节，列举分析了很多实用的例子，这有助于学生加深对基础理论知识的理解和培养实际应用的能力。

(3) 考虑到此课程的先导课程是“C语言程序设计”，书中所有算法和程序的描述都采用可在计算机上调用运行的C语言函数和程序。这样，降低了算法设计的难度，使学生能更方便地在计算机上验证这些算法。书中算法所用的C语言编写函数和程序全部在PC机上用

Turbo C 调试通过。

(4) 配合本教材的教学,还编制了若干个多媒体课件,对加深理解基本概念起到更感性的效果。课件可在清华大学出版社的网站 http://www.tup.tsinghua.edu.cn 和上海应用技术学院计算机系网站 http://www.cs.sit.edu.cn 下载,或通过 E-mail 向徐继延老师索取:xjy@sit.edu.cn。

(5) 最后的附录 A 汇总了本书各章中介绍各类数据结构时用到的数据结构类型说明,供学生在上机时参考使用。

本教材讲课时数可为 50～60 学时。上机时数可灵活安排。教师可根据学时数、专业和学生的实际情况选讲应用举例中一些较难的例子。

由于编写教材时间紧张,难免存在疏漏,敬请读者批评指正。

作　者

2007 年 4 月

目录

第1章

绪　论

1.1 引言

电子计算机是20世纪科学技术最卓越的成就之一。自1946年第一台电子计算机问世以来，计算机产业和应用的发展远远超出了人们对它的预料。如今，计算机的应用已不再局限于科学计算，而更多地用于数据处理、信息管理、实时控制等非数值计算的各个方面。用数字计算机解决任何问题都离不开程序设计。为了编制“好”的程序，必须分析程序处理的数据的特性及数据之间的关系，这就是“数据结构”这门学科形成和发展的背景。

数据结构主要研究非数值应用问题中数据之间的逻辑关系和对数据的操作，同时还研究如何将具有逻辑关系的数据按一定的存储方式存放在计算机内。分析数据之间的逻辑关系和确定数据在计算机内的存储结构是程序设计前两个必须完成的任务。

处理非数值计算问题和数值计算问题的解决方案不同。例如，求解梁架结构中应力的数学模型为线性方程组，预报人口增长情况的数学模型为微分方程。但还有更多的非数值计算问题是无法用数学方程加以描述的。

例1-1　某单位职工档案的管理。

表1.1中的职工档案表就是一个数据结构。计算机档案管理的主要功能包括查找、浏览、插入、修改、删除、统计等。如果把表中的一行看成一个记录并称为一个结点，则在此表中，结点和结点之间的关系是一种最简单的线性关系。

表1.1　某单位职工档案表

工号	姓名	性别	出生年月	婚否	学历	进厂日期
0001	张丽萍	女	08/21/1962	已	大专	09/01/1984
0005	李小明	男	04/06/1972	未	大学	04/10/1996
0006	王冠英	男	06/06/1942	已	高中	03/12/1961
⋮	⋮	⋮	⋮	⋮	⋮	⋮

例 1-2 某学校教师的名册。虽然可以用例 1-1 中的二维表格将全校教师的名单列出,但采用图 1.1 所示的结构更好。它像一棵根在上而倒挂的树,清晰地描述了教师所在的系和教研组,这样一来可以从树根沿着某系某教研组很快找到某个教师,查找的过程就是从树根沿分支到某个叶子的过程。类似于树这样的数据结构可以描述家族的家谱、企事业单位中的人事关系,甚至可用树来反映人机下棋的动态过程等。

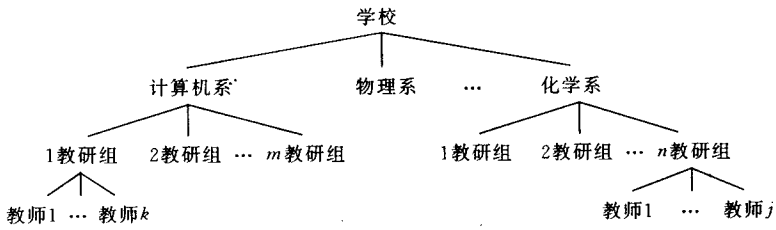
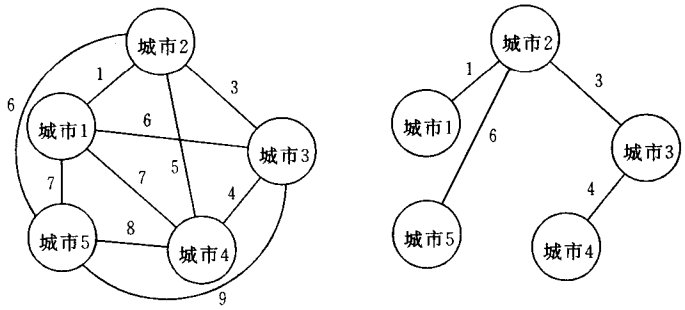


图 1.1 某学校教师名册

例 1-3 在 n 个城市之间建立通信网络,要求在其中任意两个城市之间都有直接的或间接的通信线路,在已知某些城市之间直接通信线路预算造价的情况下,使网络的造价最低。

当 n 很大时,这样的问题只能用计算机来求解。我们可以用图 1.2(a)中描述的关系来说明:图中的小圆圈表示一个城市,两个圆圈之间的连线表示对应城市之间的通信线路,连线上的数值表示该通信线路的造价。这一描述的结构为图状结构,利用计算机可以求出满足要求的最小造价通信网络,如图 1.2(b)所示。



(a) 通信网络中的数据　　(b) 最小造价通信网络

图 1.2 用图描述通信网络问题

通过上面三个例子可以看出:数据结构中元素和元素之间存在着逻辑关系,而线性表,树,图是三种基本的逻辑结构,其他各类的数据结构都是由这三种基本结构派生的。数据结构就是解决如何分析数据元素之间的关系、如何确立合适的逻辑结构、如何存储这些数据,并对为完成数据操作所设计的算法作出时间和空间的分析。"数据结构"在计算机科学中是一门综合性的专业基础课,它不仅是一般程序设计(特别是非数值计算的程序设计)的基础,而且也是设计和实现编译程序、操作系统、数据库系统及大型应用程序的重要基础。

简单来说，数据结构是研究程序设计中非数值计算的数据以及它们之间的关系和操作等的一门课程，重点分析数据之间抽象的相互关系，不涉及数据的具体内容。

1.2 基本概念和术语

数据(data) 是指所有能输入到计算机中并被计算机程序处理的符号的总称。计算机输入和处理的数据除数值外，还有字符串、表格、图像甚至声音等，它们都是数字编码范畴。

数据元素(data element) 数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。一个数据元素可以由若干个数据项组成，也可以只由一个数据项组成。数据元素又被称为元素、结点(node)或记录(record)。

数据项(data item) 是指数据的不可分割的、含有独立意义的最小单位，数据项有时也称字段(field)或域。上面的职工档案表格是要存放到计算机中进行处理的数据，表中每一行记录了一个职工的档案信息，在数据操作中作为一个整体考虑，对应为一个数据元素，又称为一个记录。这个记录中包含有工号、姓名、学历等若干个数据项。操作的基本单位是记录，如职工的插入或删除一定是作用于一个职工的全部信息即一个记录，而不可能是作用于其中的某个数据项。设想删除操作只删除某个职工的姓名或工号，将引起数据的不完整等严重后果。每个数据项(如职工的姓名或工号)均有独立的含义，但在档案管理这个实际问题中并无完整的意义，而组合在一个记录中构成职工的档案，就具有了完整的实际意义。数据、数据元素、数据项实际上反映了数据组织的三个层次：数据可由若干个数据元素构成，而数据元素又可以由一个或若干个数据项组成。

数据逻辑结构(data logical structure) 数据结构主要是研究数据元素之间的关联方式。数据元素之间存在的一种或多种特定的关系被称为数据的逻辑结构。通常有集合、线性结构、树形结构和图状结构四类基本结构。见图 1.3。

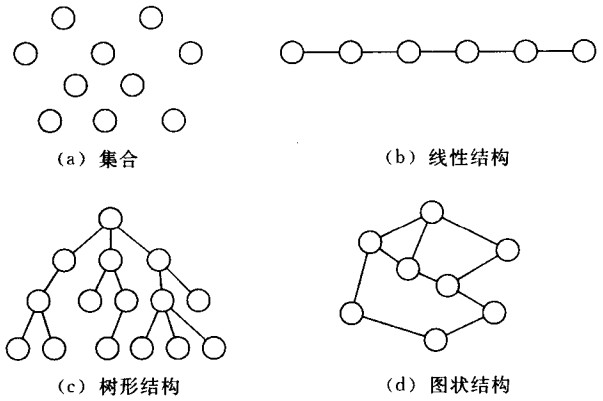


图 1.3 四类基本逻辑结构的示意图

数据物理结构(data physical structure) 数据在计算机中的存放方式称为数据的物理结构，又称存储结构。数据的存储结构是逻辑结构在计算机存储器中的实现。数据元

素在计算机中主要有两种不同的存储方法,即顺序存储结构和链式存储结构。顺序存储的特点是在内存中开辟一组连续的空间(高级语言中的数组)来存放数据,数据元素之间的逻辑关系通过元素在内存中存放的相对位置来确定,又称向量存储。链式存储的特点是通过指针反映数据元素之间的逻辑关系,又称动态存储。

数据的逻辑结构和物理结构是数据结构的两个密切相关的方面,同一逻辑结构可以对应不同的存储结构。算法的设计取决于数据的逻辑结构,而算法的实现依赖于指定的存储结构。

例如,10 以内的奇数 1,3,5,7,9 用顺序存储结构的方式依次存放在以 300 为起地址的内存向量中,并且两个字长存放一个奇数,如图 1.4(a)所示。在顺序存储结构中,如要读取第三个奇数,它的地址可以通过起地址和要读取奇数的位置序号计算得到。若本例中的奇数改用链式存储结构存放,那么,第一个奇数存放在地址为 300 的内存单元中,第二个奇数存放的地址和第一个奇数存放的地址无关,但第二个奇数所在的地址存放在第一个奇数相关的指针中,后面奇数的存放也按如此的规律。设两个字长存放一个奇数,一个字长存放一个指针,如图 1.4(b)所示。在链式存储结构中,如要读取第三个奇数,只能从第一个奇数所在的地址开始,通过第一个奇数关联的指针得到第二个奇数存放的地址 105,再找到第三个奇数存放的地址 400,才能读出。从以上的例子和分析看,数据结构主要就是研究数据的逻辑结构、相应的存储结构以及完成数据操作的算法设计。

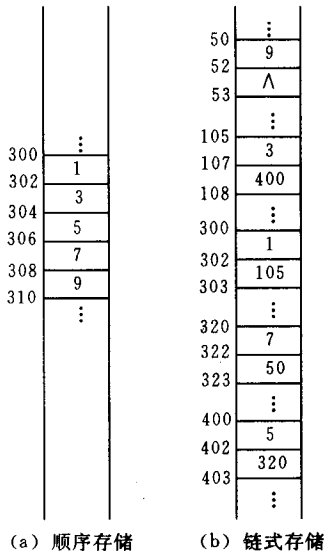


图 1.4 两种存储结构的示意图

数据类型(data type) 和数据结构密切相关的一个概念,在用高级程序设计语言编写的程序中,每个变量、常量或表达式都对应一个确定的数据类型。数据类型可分为两类:一类是非结构的原子类型,如 C 语言中的基本类型(整型、实型、字符型等)、指针类型和空类型;另一类是结构类型,它的成分可以由多个结构类型组成,并可以分解。结构类型的成分中可以是非结构的,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,也可以是数组等结构类型。

本书在讨论各种数据结构时,针对其逻辑结构和具体的存储结构给出对应的数据类型,进一步在确定的数据类型上实现各种操作。

算法(algorithm) 是指解决特定问题的一种方法或一种描述。

1968 年,美国唐·欧·克努特教授开创了数据结构的最初体系,他所著的《计算机程序设计技巧》第一卷《基本算法》是第一部较系统地阐述数据的逻辑结构、存储结构及算法的著作。20 世纪 70 年代初,大型程序出现,软件业飞速发展,结构化程序设计成为程序设计方法学的主要内容,人们越来越重视数据结构,认为程序设计的实质是确定数据的结构,加上设计一个好的算法,也就是人们所说的"程序=数据结构+算法"。

1.3 算法描述

程序设计人员需要对程序处理的问题准确理解，只有准确理解问题后才能研究出解决问题的方法。算法是指解决问题的一种方法或过程描述。如果将问题看作函数，那么算法能把输入转化为输出。解决一个问题可以有多种算法，但一个给定的算法只能解决一个特定的问题。例如对一组数据的排序，可给出5种甚至更多种的排序算法。可以用多种算法求解问题的优点在于：根据问题的具体限定条件，可以选用合适的算法求解。例如，有的排序算法适合于元素个数少的序列，有的算法适合于元素个数多的序列，有的算法则适合于定长数值型数据的排序。计算机程序就是用某种程序设计语言去具体地实现一个算法，或称为代真。本书中主要介绍各种算法，并给出一部分算法对应的C语言程序。当然使用其他的程序设计语言也可以实现算法的代真。综上所述，问题、算法、程序是三个互相关联的不同概念。

1.3.1 算法的重要特性

- 正确性　它必须解决具体的问题，完成所期望的功能，给出正确的输出。
- 确定性　算法执行的每一步和下一步必须确定，不能有二义性。
- 有限性　一个算法必须由有限步组成。无限步组成的算法无法用计算机程序来实现。因此算法必须可以终止，不能进入死循环。
- 输入　一个算法有零个或多个输入。
- 输出　一个算法有一个或多个输出。

1.3.2 数据结构上的基本操作

基本操作主要有以下几种：

- 查找　寻找满足特定条件的数据元素所在的位置。
- 读取　读出指定位置上数据元素的内容。
- 插入　在指定位置上添加新的数据元素。
- 删除　删去指定位置上对应的数据元素。
- 更新　修改某个数据元素的值。

根据操作的结果可将操作分为两种基本类型：

- 加工型操作　其操作改变了原逻辑结构的“值”，如数据元素的个数、某数据元素的内容等（一般不考虑改变逻辑结构的类型）。上面基本操作中的后三种操作均为加工型操作。
- 引用型操作　其操作不改变原逻辑结构的“值”，只是查找或读取。

1.3.3 算法的描述方法

算法的描述方法有很多，根据描述算法语言的不同，可将算法分为以下四种：

- 框图算法描述　这种描述方法在算法研究的早期曾流行过。它的优点是直观、易

懂，但用来描述比较复杂的算法就显得不够方便，也不够清晰简洁。

- 非形式算法描述　用中文语言，同时还使用一些程序设计语言中的语句来描述算法，这称为非形式算法描述。
- 类C语言算法描述　类C语言算法又称为伪语言算法。这种算法不能直接在计算机上运行，但专业设计人员经常使用类C语言来描述算法，它容易编写、阅读和统一格式。
- C语言编写的程序或函数　这是可在计算机上运行并获得结果的算法，使给定问题能在有限时间内被求解，通常这种算法也称为程序。

下面以求两个整数 $m,n(m\geqslant n)$ 的最大公因子为例来看看不同的算法描述的方法。

(1) 该问题的框图描述如图 1.5 所示。

(2) 非形式算法描述：

a. [求余数] 以 n 除 m，并令 r 为余数 $(0\leqslant r<n)$；

b. [余数是零否] 若 $r=0$ 则结束算法，n 就是最大公因子；

c. [替换并返回 a] 若 $r\neq 0$ 则 $m\leftarrow n$，$n\leftarrow r$ 返回 a。

(3) C语言函数描述：

```
int max_common_factor (int m, int n)
{
   int r ;
   r = m % n;
   while (r != 0)
      {m = n ; n = r ; r = m % n;}
   return n ;
}
```

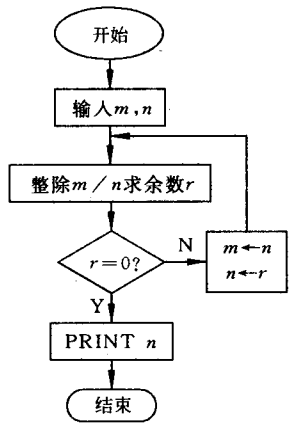


图 1.5　框图描述法

本书主要介绍算法的思路和实现过程，且尽可能地将算法对应的C语言函数或程序提供给读者阅读或上机运行，以便更好地理解算法。C语言函数或程序用到的数据类型说明的详细列表见附录A。

1.4　算法分析

1.4.1　算法设计的要求

设计一个“好”的算法应考虑以下几个方面：

- 易读性　算法应易于阅读和理解，以便于调试、修改和扩充。
- 健壮性　正确的输入能得到正确的输出这是算法必须具有的特性之一。但当遇到非法输入时，算法应能作出反应或处理(如提示信息等)，而不会产生不需要的或不正确的结果。
- 高效率　即达到所需的时空性能。一个算法的时空性能是指该算法的时间性能(时间效率)和空间性能(空间效率)。解决同一问题如果有多个算法，执行时间短

的算法时间效率高，而存储量和辅助空间量少的算法空间效率高。这两者都和问题的规模有关。

1.4.2 算法时间效率的度量分析

本节重点介绍算法的时间效率分析的基础知识。算法运行的时间分析和程序运行的时间分析有区别。同一算法由不同的编程员所编出来的程序有优劣之分，程序运行的时间也就有所不同；程序在不同的机器上运行的速度又和机器本身的速度有关。我们感兴趣的是对解决问题的算法作时间上的度量分析，或对解决同一问题的两种或两种以上的算法运行的时间加以比较。我们称这种度量分析为算法的时间复杂度分析。它可以估算出当问题的规模变大时，算法运行时间增长的速度。这种分析实际上是一种数学化了的估算方法。

估算算法运行时间的基本考虑是：确定问题的“规模”和确定算法执行“基本操作”的次数。一个算法的“规模”和“基本操作”要视具体算法而定。“规模”一般是指输入量的数目，比如在排序问题中，问题的规模可以是被排序的元素数目。“基本操作”一般是指在某个数据类型上的“标准操作”，比如两个整数相加、比较两个整数的大小等都可以视为是基本操作。

例 1-4 查找一维 n 元整数数组中最大元素的算法。该算法从数组中下标为 0 的元素开始，遍历数组中的所有元素，在遍历过程中，将当前最大元素保存在变量 currlarge 中。下面是对应的算法：

```
int largest (int * array, int n)
{
   int currlarge, i ;
   currlarge = array[ 0 ] ;
   for(i = 1; i < n ; i++)
      if (array[i] > currlarge) currlarge = array[i];
   return currlarge ;
}
```

其中，数组 array 中存放有 n 个整数，则问题的规模为 n。基本操作是“比较”，即将数组中的一个整数和现有的最大整数作比较。影响算法运行时间的最主要的因素就是输入规模 n，我们经常把运行算法所需要的时间 T 写成输入规模 n 的函数，记作 $T(n)$。

我们把 largest 函数中比较一个元素所需要的时间记作 c_1，“比较”这一基本操作是对数组中每一个元素都要做的工作。算法主要考虑这一基本操作所花的时间，忽略当找到一个新的最大元素时要做的工作的时间，也不考虑函数初始化时所需要的时间，这样就能得到运行该算法的一个合理的近似时间。因此，运行 largest 函数的总时间可近似地认为是 $c_1 n$。largest 函数运行的时间代价可以用下面的等式来表示：$T(n)=c_1 n$。这个等式表明了顺序检索数组中最大元素的算法的时间是随着 n 的增长而线性增长。

例 1-5 将一个整数数组的第一个元素值赋给另一个变量。完成这一功能所需要的时间是固定的。无论这个数组有多大，复制一个元素值的时间总是确定的，记作 c_2。因此该算法运行时间代价的等式就是 $T(n)=c_2$。输入规模即 n 的大小对运行时间不产生影

响。这个等式称为常数运行时间。

例 1-6 再看下面一个算法段：

```
sum = 0;
for (i = 1; i <= n; i++)
  for(j = 1; j <= n; j++)
sum++;
```

显然，随着 n 的增大，其运行时间也会增大。本例的基本操作是 sum 的累加，假设这个基本操作所需要的时间为 c_3，忽略了初始化 sum 的时间和循环变量 i 和 j 累加的时间，基本操作总次数为 n^2，因此，该算法运行时间代价可用下面的等式来表示：$T(n)=c_3 \cdot n^2$。

增长率的概念是非常重要的。它可以帮助我们比较算法的运行效率。图 1.6 给出了五个常见的运行时间函数的曲线，每一条曲线反映出某种算法的时间代价，显示了不同算法的增长率。标记为 $100n$ 的函数为一直线，增长率称为线性增长率。这说明，当 n 增大时，算法的运行时间线性增大。如果算法的运行时间函数中含有如 n^2 这样的高次项，则称为二次增长率，图中标有 $5n^2$ 的曲线就代表二次增长率。标有 2^n 的曲线属于指数增长率。

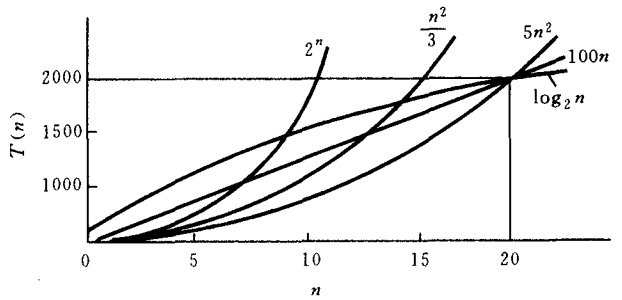


图 1.6 常见函数的增长率

由于算法的时间复杂度分析只考虑相对于问题规模 n 的增长率，因而，在难以精确计算基本操作执行次数的情况下，只要求出它关于 n 的增长率即可。我们可以在计算任何算法运行时间代价时，忽略所有的常数和低次项，用 O 表示法来表示算法的时间复杂度。例 1-4 中算法的时间复杂度为 $O(n)$，例 1-5 中算法的时间复杂度为 $O(1)$，例 1-6 为 $O(n^2)$，分别称为线性阶、常量阶和平方阶。如果某算法的运行时间代价为 $O(5n^2+n)$，可以忽略其中的低次项和常数，而视该算法的时间复杂度为 $O(n^2)$。算法的时间复杂度还有对数阶 $O(\log_2 n)$、指数阶 $O(2^n)$等。

1.5 应用举例及分析

例 1-7 用 C 语言描述下列算法，并给出算法的时间复杂度。

(1) 求一个 n 阶方阵的所有元素(正整数)之和。对应的算法如下：