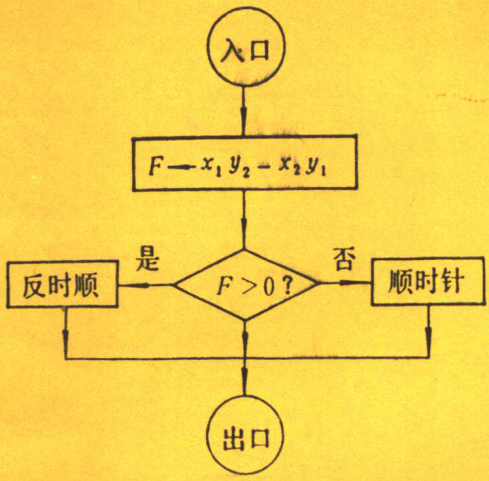

计算机算法与程序设计技术

顾小风　编

北京大学出版社

计算机算法与程序设计技术

顾小风 编

北京大学出版社

内 容 提 要

本书简略地介绍了计算机算法及程序设计的技术。内容包括：有关数学知识、算法概念、整数运算算法、查表算法、排序算法、以表格为工具的算法、建立算法工具的算法、算法分析、程序设计技术。另外还附有关于本书所用算法语言的说明及习题。

本书适用于高等院校计算数学和计算机专业的高年级学生、研究生阅读，并且可供计算机工作者参考。

计算机算法与程序
设计技术
顾小凤 编
责任编辑：王明舟
*
北京大学出版社出版
（北京大学校内）
北京大学印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
787×1092 毫米 32 开本 5.25 印张 150 千字
1988年12 月第一版 1988年12 月第一次印刷
印数：0001—8000册
ISBN 7-301-00305-6/PT·001
定价：1.80元

前　言

从事计算机软件工作不久的年青同志，他们大都具有思路敏捷、接受新知识的能力强等优点。但由于缺乏科学的训练和实际工作经验，因而也存在着诸如下列的一些弱点：

(1) 不懂得程序设计的技巧是什么含义，津津有味地去耍弄一些古怪的、别人难懂的、尽管可能提高几个毫秒的效率但却使程序的调试难度大大增加的小技巧。E.Yourdon 曾指出：好的程序设计者的标志之一是厌恶对雕虫小技的纵容。而不少程序设计者并不理解这一点。

(2) 不懂得什么叫做一个好的算法。经常片面地认为省指令、省存贮的算法就是好的算法。为此不惜为省一条指令、省一个存贮单元而耗费大量精力。

(3) 不懂得算法的正确性是什么意思。以为所有的测试数据都通过了的算法就是正确的算法。事实上，测试数据再多，它也是有限的。而使算法的某一隐蔽错误得以暴露的特殊测试数据，即使程序已经正式运行多少年也不一定碰到一次。因此，依靠测试数据来证明一个算法的正确，这是完全行不通的。

(4) 不熟悉某些经典的常用算法，放着一些成熟的、现成的算法不用而化很大力气去另外设计出非标准的甚至错误的算法。

(5) 在程序设计中以盲目性代替科学性。关于这一点，我们不妨举一个例子。

假如在某一软件系统中我们对“文件名”作如下定义：

“文件名是以字母开头的字母数字串，字母数字串后可以带有跟随着一至二位数字的‘——’号，文件名最后以空格结束。”

现在要求编一段程序来检查给定的某一串符号是否文件名。

这是一个典型的语法检查问题。对于这类问题，我们将在第六章中介绍用状态矩阵法建立形式化的严格的检查规则。按照这种规则，语法检查的工作将会变得十分简单而机械，并且能够毫无遗漏地检查出一切语法错误。

但是有的同志却不是这样做的。他们往往不是先去建立一套严格的检查规则，而是不加思索地匆匆忙忙就画出框图、编出程序，想到哪里就检查到哪里，其结果是丢三拉四、漏洞百出。实际上，他们对自己编出的程序也并无把握，而是抱着试试看的侥幸心理，这就是盲目性。

针对以上问题，这本书的目的是试图对跨过软件领域门槛不久的同志起一些指导和帮助的作用，把软件工作的水平从现有基础上再提高一步，使自己的工作步入科学的轨道，以适应软件发展的现代水平。

本书不是常用算法的手册，因此不打算大量收集各种算法，特别是关于计算方法课程的内容并未包括在本书之中。书中介绍的一些算法只是举例性的，希望通过这些例子引起大家对算法研究的兴趣，了解各种算法的某些一般规律及实用中需要注意的问题。

由于本书偏重于算法的实用方面，故介绍每种算法时只作简单的分析，系统的算法分析将集中到第八章介绍。

关于如何阅读本书，提出以下建议：

(1) 对于数学基础较好的读者，可以跳过第一章不看。

(2) 第二章比较难懂，一时弄不懂可以先往下看，适当的时候回过头来再看，不要因为这一章卡住了而失去信心。

(3) 阅读第三章以前先看一下附录一，了解本书所用的算法语言的符号，因为从第三章开始要经常接触到这些符号。对于使用高级语言不熟练的同志，这一点尤为重要。

(4) 如果你学过“数据结构”这门课，则四、五两章的部分内容粗读一下就行了。

(5) 无论如何，六、八、九这三章应仔细阅读，特别是第九章，它可能会澄清你在程序设计方面的一些糊涂观念。

本书的内容曾作为北京大学计算机软件专业和数学系计算专业高年级学生的选修课讲义。对于从事计算机软件工作的科技人员也有一定的参考价值。

本书大部分内容取材于书末所列的前三本参考文献，一部分内容来源于作者近年来发表的几篇论文和实际工作经验。由于水平有限，错误在所难免，希望读者批评指正。

顾小凤

1986年于北京大学

目　录

第一章　有关的数学知识

§1　整数函数$\lfloor x\rfloor$,$\lceil x\rceil$

设x为任意实数，则$\lfloor x\rfloor$为小于等于x的最大整数。$\lceil x\rceil$为大于等于x的最小整数。

$\lceil x\rceil$和$\lfloor x\rfloor$有以下一些重要性质。这些性质的证明并不难，所以不妨留给大家自己去做。

（1）$\lfloor x\rfloor=\lceil x\rceil$当且仅当$x$是整数。

（2）$\lceil x\rceil=\lfloor x\rfloor+1$当且仅当$x$不是整数。

（3）$\lfloor -x\rfloor=-\lceil x\rceil$。

（4）$x-1<\lfloor x\rfloor\leqslant x\leqslant\lceil x\rceil<x+1$。

（5）若n为整数，x为实数，则

1）$\lfloor x\rfloor<n$　当且仅当 $x<n$。

2）$n\leqslant\lfloor x\rfloor$　当且仅当 $n\leqslant x$。

3）$\lceil x\rceil\leqslant n$　当且仅当 $x\leqslant n$。

4）$n<\lceil x\rceil$　当且仅当 $n<x$。

5）$\lfloor x\rfloor=n$　当且仅当 $x-1<n\leqslant x$

当且仅当 $n\leqslant x<n+1$。

6）$\lceil x\rceil=n$　当且仅当 $x\leqslant n<x+1$

当且仅当 $n-1<x\leqslant n$。

7）$\lfloor x+n\rfloor=\lfloor x\rfloor+n$；　$\lfloor x-n\rfloor=\lfloor x\rfloor-n$；

$\lceil x+n\rceil=\lceil x\rceil+n$；　$\lceil x-n\rceil=\lceil x\rceil-n$。

（6）$\left\lfloor\dfrac{\lfloor x\rfloor}{2}\right\rfloor=\left\lfloor\dfrac{x}{2}\right\rfloor$。

以 2 为底的对数 $\log_2 n$ 是算法分析中常用的函数。设 n 为正整数，则 $\log_2 n$ 有以下性质：

（1）$\lceil \log_2(n+1) \rceil = \lfloor \log_2 n \rfloor + 1$。

（2）$\lceil \log_2 n \rceil \leqslant 1 + \log_2 n$。

（3）$\lfloor \log_2 n \rfloor > \log_2 n - 1$。

（4）$n \leqslant 2^{\lfloor \log_2 n \rfloor} < 2n$。

（5）$\frac{n}{2} < 2^{\lfloor \log_2 n \rfloor} \leqslant n$。

（6）当 n 为奇数时，$\lfloor \log_2 n \rfloor = \lfloor \log_2(n-1) \rfloor$ $(n \neq 1)$。

[提示：注意到

$\lfloor \log_2 2 \rfloor = \lfloor \log_2 3 \rfloor = 1$，

$\lfloor \log_2 4 \rfloor = \lfloor \log_2 5 \rfloor = \lfloor \log_2 6 \rfloor = \lfloor \log_2 7 \rfloor = 2$，

…，

一般地，

$\lfloor \log_2 2^n \rfloor = \lfloor \log_2(2^n+1) \rfloor = \cdots = \lfloor \log_2(2^{n+1}-1) \rfloor = n$。]

（7）$\sum_{i=1}^{n-1} \lfloor \log_2 i \rfloor = n \lfloor \log_2 n \rfloor - 2^{\lfloor \log_2 n \rfloor + 1} + 2$。

证　根据 $\lfloor \log_2 n \rfloor$ 的性质，画出函数图象后容易发现

$$\sum_{i=1}^{n-1} \lfloor \log_2 i \rfloor = \sum_{j=1}^{\lfloor \log_2 n \rfloor - 1} j2^j + \lfloor \log_2 n \rfloor (n - 2^{\lfloor \log_2 n \rfloor})。$$

注意到

$$\sum_{j=1}^{k-1} j2^j = \sum_{j=1}^{k-1} (j2^{j+1} - j2^j) = \sum_{j=2}^{k} (j-1)2^j - \sum_{j=1}^{k-1} j2^j$$

$$= \sum_{j=2}^{k} j2^j - \sum_{j=1}^{k-1} j2^j - \sum_{j=2}^{k} 2^j$$

$$= k2^k - 2 - 4(2^{k-1} - 1)$$
$$= k2^k - 2^{k+1} + 2.$$

因此

$$\sum_{i=1}^{n-1} \lfloor \log_2 i \rfloor = \lfloor \log_2 n \rfloor 2^{\lfloor \log_2 n \rfloor} - 2^{\lfloor \log_2 n \rfloor + 1} + 2$$
$$+ n \lfloor \log_2 n \rfloor - \lfloor \log_2 n \rfloor 2^{\lfloor \log_2 n \rfloor}$$
$$= n \lfloor \log_2 n \rfloor - 2^{\lfloor \log_2 n \rfloor + 1} + 2. \quad \blacksquare$$

整数函数在算法分析和软件设计中都极为常用。例如，常见的对小数点后第一位数进行四舍五入的运算就可以表示为 $\lfloor x + 0.5 \rfloor$。

§2 $x \bmod y$

定义 1 设 x，y 为任意实数，则

$$x \bmod y = x - y\left\lfloor \frac{x}{y} \right\rfloor, \text{ 当 } y \neq 0;$$

$$x \bmod 0 = x.$$

根据上述定义，当 $y \neq 0$ 时，我们可以把 $x \bmod y$ 看作 x 除以 y 所得的余数。

$x \bmod y$ 当 x，y 都是整数时用得很广泛。以下的讨论凡未特别注明，均假定 x，y 为整数。

定义 2 若 $x \bmod z = y \bmod z$，则

$$x \equiv y \pmod z.$$

读作“x 与 y 对模 z 同余”。

根据定义 2 可知，$x \equiv y \pmod z$ 当且仅当 $x - y$ 是 z 的整数倍。

由于 $(x \bmod y) \bmod y = x \bmod y$，故根据定义 2 有

$$x \equiv x \bmod y (\bmod y).$$

例如：$5 \equiv 2 (\bmod 3)$，$8 \equiv 1 (\bmod 7)$。

关于同余，有以下重要性质（下面用到的变量都是整数）：

（1）若 $a \equiv b (\bmod m)$，$b \equiv c (\bmod m)$，则

$$a \equiv c (\bmod m).$$

证 根据定义 2，$a - b = pm$，$b - c = qm$，所以

$$a - c = (p + q)m，即$$

$$a \equiv c (\bmod m).$$ |

（2）若 $a \equiv b (\bmod m)$，$c \equiv d (\bmod m)$，则

$$a \pm c \equiv b \pm d (\bmod m).$$

（3）若 $a \equiv b (\bmod m)$，则

$$ac \equiv bc (\bmod m).$$

（4）若 $a \equiv b (\bmod m)$，$c \equiv d (\bmod m)$，则

$$ac = bd (\bmod m).$$

证 由 $a \equiv b (\bmod m)$及（3）得

$$ac \equiv bc (\bmod m).$$

由 $c \equiv d (\bmod m)$及（3）得

$$bc \equiv bd (\bmod m).$$

再由（1）得 $ac \equiv bd (\bmod m)$。|

（5） 若 a 与 m 互质，则存在 a' 使

$$aa' \equiv 1 (\bmod m) \quad (m \neq 1).$$

证 根据欧几里德定理（第八章有该定理的证明），若 a，m 互质，则存在整数 p，q 使

$$pa + qm = 1,$$

即

$$pa \equiv 1 (\bmod m).$$

所以

$$a' = p.$$

（6）若 $ac\equiv bd(\mathrm{mod}m)$，$a\equiv b(\mathrm{mod}m)$，且 a 与 m 互质，则

$$c\equiv d(\mathrm{mod}m).$$

证　$m=1$ 时显然正确．若 $m\neq 1$，则根据（5）存在 a' 使

$$aa'\equiv 1(\mathrm{mod}m).$$

又根据（4）

$$aa'\equiv a'b\equiv 1(\mathrm{mod}m),$$

$$a'ac\equiv a'bd(\mathrm{mod}m).$$

即
$$a'ac-a'bd=sm.$$

令　$a'a=pm+1$，$a'b=qm+1$．则

$$a'ac-a'bd=cpm+c-dqm-d=sm.$$

所以
$$c-d=(s-cp+dq)m.$$

即
$$c\equiv d(\mathrm{mod}m).$$

（7）$a\equiv b(\mathrm{mod}m)$ 当且仅当

$$an\equiv bn(\mathrm{mod}mn)(n\neq 0)$$

（8）若 r，s 互质，则

$a\equiv b(\mathrm{mod}rs)$ 当且仅当 $a\equiv b(\mathrm{mod}r)$ 和 $a\equiv b(\mathrm{mod}s)$．

利用以上性质，还可以推出一些有用的公式．

例如，设　$0\leqslant y<m$，则

$$((x+y)\mathrm{mod}m-x)\mathrm{mod}m=y.$$

证　$(x+y)\mathrm{mod}m\equiv x+y(\mathrm{mod}m)$

所以　$(x+y)\mathrm{mod}m-x\equiv x+y-x(\mathrm{mod}m)$．

即　$((x+y)\mathrm{mod}m-x)\mathrm{mod}m=y\,\mathrm{mod}\,m=y$．

“mod”运算在计算机中应用很广泛．以下我们举几个例子．

（1）字长为 n 的计算机中，负数 a 的补码表示就是

$$a \bmod 2^n.$$

例如，当 $n=8$，$a=-1$，则 a 的补码表示是

$$(-1) \bmod 2^8 = -1+256 = 255.$$

即机器中 8 位二进位数全是 1。

（2）字长为 n 的计算机中用二进位表示的两个机器字

$$x = x_n x_{n-1} \cdots x_1, \quad y = y_n y_{n-1} \cdots y_1$$

进行按位加（exclusive OR）的结果

$$z = z_n z_{n-1} \cdots z_1$$

可表示为

$$z_k = (x_k + y_k) \bmod 2 \quad (k = 1, 2, \cdots, n).$$

（3）循环计数器 a（即当计数达到给定的最大值 n 后又从零开始计数）的计数值应为 $a \bmod (n+1)$。

例如，x 为光标的横坐标（$0 \leqslant x \leqslant 31$），规定当光标移到最右端后再继续右移就应该移到最左端。设 x 的当前值为 30，要把光标再右移 3 次（$\Delta x = 3$），则 x 的新值应为

$$(x + \Delta x) \bmod 32 = 33 \bmod 32 = 1.$$

循环缓冲区（即缓冲区最末一个地址的下一个地址是缓冲区的第一个地址）的指针的计算也应该采用类似的方法。

第二章　算法的基本概念*

§1　算法的定义

关于算法的定义，我们先给出一个比较粗略的、直观的定义，然后再给出精确的数学定义。

算法是针对某一个特定问题而提供的解决此问题的步骤。一个算法必须具有以下五个重要特征：

(1) 有穷性。即一个算法必须经有限步结束。如果算法的其它特征都具备，唯独沒有有穷性，这就不能叫算法，只能称作“计算方法” (computational method)。例如，欧几里德曾提出求两个已知线段的最长公共度量线段的问题。这个“算法”与求最大公约数的算法类似，但是它不能经有限步结束。这是因为即使不考虑测量误差的因素，线段本身的长度也不一定是度量单位的整数倍。因此就可能出现不可通约的线段而造成计算过程沒完沒了。

有穷性并不是很苛刻的限制。实际上我们往往不但要求有穷，而且要求“十分”有穷。例如有的算法尽管具有有穷性，但却需要化费人们难以做到的长时间才能结束，这种算

---

* 算法的研究有两个不同的分支。一个分支是可计算理论 (computability theory) 或算法理论 (the theory of algorithms)，它研究解决某一特定问题的有效算法是否存在的问题。另一个分支被称为计算复杂性 (computational complexity) 或算法分析 (analysis of algorithms)，它研究一个算法所需的时间（效率问题）和空间（存贮量问题）。本书不涉及第一个分支的内容，而只涉及第二个分支的部分内容。

法就沒有什么实用价值。

(2) 确定性。即算法的每一步必须有明确的定义，告诉人们应该做什么。例如，当我们说“求m除以n的余数”时，如果事先沒有规定m,n是正整数，那么当$m<0$时余数的定义是什么？当$n=0$时怎么办？这些都必须明确规定好。

为了达到算法的确定性，我们常常需要有一种精确的语言来表达算法的每一步。通常我们用形式化定义的算法语言（高级程序设计语言或与其类似的语言）来描述算法。这种语言必须沒有二义性。

本书所举的算法例子中大多数采用了Sara Baase 在《计算机算法》一书中所使用的语言，它与通常流行的高级语言十分相似。有关该语言的定义请参阅本书附录一。

(3) 一个算法有零个或多个输入。

(4) 一个算法有一个或多个输出。

(5) 有效性。即算法中的全部运算都可以由人们用纸和笔在有限的时间内完成。换句话说，算法的每一种运算都应该是可以做到的。例如，如果我们对算法的某一步作如下规定：“若对正整数x,y,z，2是方程

$$x^n+y^n=z^n$$

的最大整数解，则去执行第四步。”那么，在我们有办法确定2是否$x^n+y^n=z^n$的最大整数解以前，这一步还不能认为是有效的运算。

以上五点并未包括“正确性”。也就是说，我们沒有规定一个算法的输出与输入之间必须满足某种规律。因此，我们仅仅定义了“算法”而并沒有定义“正确的算法”。一个算法的正确性的证明属于算法分析的范畴。

以下，我们从数学上给出算法的严格定义。

定义 1 计算方法 (computational method) 是一个四元组 (Q, I, Ω, f)，其中Q是包含子集I, Ω的集合，f是集合Q上的一个函数，且当$q \in \Omega$时$f(q) = q$。

在此四元组中定义了计算序列，即对每一个$x \in I$存在序列

$$x_0, x_1, \cdots;$$

其中

$$x_0 = x, \quad x_{k+1} = f(x_k) \quad (k \geqslant 0).$$

若存在一个使$x_k \in \Omega$的最小整数k，我们就说此计算序列经k步结束。

定义 2 一个计算方法，若对任一个$x \in I$其计算序列都经有限步结束，则称此计算方法为算法。

在以上定义中，我们可以把Q看作计算状态，I为输入，Ω为输出，f为运算规则。

例 求最大公约数 (g.c.d.) 的欧几里德算法。

算法2.1 给出正整数m, n，求它们的最大公约数。

1. m除以n得到余数r $(0 \leqslant r < n)$。
2. 若$r = 0$则算法结束，n为最大公约数。否则作 3。
3. $m \leftarrow n$，$n \leftarrow r$，回到 1。

利用刚才对算法所下的定义，我们可以对以上步骤重新描述如下：

设Q是所有一元组(n)，所有有序对(m, n)，所有有序四元组$(m, n, r, 1)$，$(m, n, r, 2)$，$(m, n, p, 3)$的集合。其中m, n, p是正整数，r是非负整数。

设I是包含所有有序对(m, n)的Q的子集，Ω是包含所有一元组(n)的Q的子集。

设f的定义是：