Selected Lectures in Symbolic Computation

符号计算选讲

王东明 主编

杨 路 李志斌 侯晓荣
陈发来 夏壁灿 支丽红 编著

清华大学出版社

符号计算选讲

Selected Lectures in Symbolic Computation

王东明 主编

杨 路 李志斌 侯晓荣
陈发来 夏壁灿 支丽红 编著

清华大学出版社
北 京

内 容 简 介

本书介绍符号计算的基本概念、思想、理论、方法、软件和应用. 全书共分 6 章，概述符号计算的 6 个主要分支. 内容包括计算机代数，几何定理机器证明的代数方法，计算实代数几何，代数计算在计算机辅助几何设计中的应用，符号微分、符号积分和微分方程的符号解，以及符号与数值混合计算. 本书选材侧重基础知识和经典算法，兼收学科的前沿发展和最新研究成果.

本书可作为高等院校数学和计算机科学系高年级学生及研究生的教学参考书，也可供有关科研和工程技术人员参考.

图书在版编目(CIP)数据

符号计算选讲/王东明主编；杨路等编著. —北京：清华大学出版社，2003
ISBN 7-302-06839-9

Ⅰ. 符… Ⅱ. ①王…②杨… Ⅲ. 计算机应用—抽象代数—代数几何 Ⅳ. O187

中国版本图书馆 CIP 数据核字(2003)第 050894 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

责任编辑：刘　颖
封面设计：常雪影
印 刷 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×230　**印张**：18.5　**字数**：352 千字
版　　次：2003 年 8 月第 1 版　2003 年 8 月第 1 次印刷
书　　号：ISBN 7-302-06839-9/O・305
印　　数：1～3000
定　　价：36.00 元

前 言

数学以及许多相关学科所处理的对象都是具有含义的抽象符号. 这些符号包括数、多项式、有理函数、三角函数、几何图形、逻辑公式和计算机程序. 引进适当的符号, 建立它们之间的运算规则和推理机制, 并研究这些符号的性质、用途和相互关系便构筑了相应的学科. 有别于近似数值计算, 符号之间的运算和推理通常是形式的, 也是精确的, 因而没有误差. *符号计算* 研究如何在计算机上表示和处理有含义的抽象符号, 设计用于符号运算和推理的有效算法以及适合实施这些算法的程序语言和软件系统, 有效地实施所设计的系统和算法并将其用于各式各样的理论与实际问题.

作为数学与计算机科学的一门交叉学科, 符号计算始于 20 世纪 60 年代. 它的发展始终与代数计算和软件开发联系在一起, 并受到了物理计算的激励, 其主要分支包括计算机代数与分析、几何计算、自动推理与编程等. 符号计算软件已成为解决各种科学与工程问题的有力工具. 符号计算、自动推理与吴文俊院士开创并倡导的数学机械化密切相关, 而作为相近学科它们又各具特色、侧重不一. 符号计算强调构造性理论的建立与发展、有效算法的设计与实施、软件系统的研制与开发, 以及它们在科学工程中的应用.

为了向研究生和青年学者介绍符号计算这门学科, 让他们了解其中的基本概念、思想、方法和软件, 帮助他们掌握一些先进的理论、算法和技巧, 同时促进和加强有关科研人员之间的学术交流与合作, 北京大学数学科学学院、中国科学院数学与系统科学研究院和中国科学技术大学理学院定于 2003 年 7 月 13 日至 26 日在安徽黄山主办符号计算暑期讲习班. 该班将设置基础短课程, 向学员介绍符号计算的若干主要分支. 同时, 多位国内外著名专家学者, 包括国际符号计算杂志主编、美国北卡罗来纳州立大学 Hoon Hong 教授, 北京大学郑志明教授, 中国科学院杨路、刘卓军研究员等, 将在讲习班上作专题学术演讲, 报告符号计算领域的重要科研成果和最新进展. 他们还将向学员讲述符号计算对现代科研与教学的意义、作用和潜力, 鼓励研究生和青年学者加入符号计算及其相关领域特别是数学机械化的研究行列, 为发展壮大我国在该领域中的学术队伍, 提高其科研水平和学术地位而共同努力.

本书是为符号计算暑期讲习班基础课编写的教材, 书中每章对应于一门短课程. 笔者计划讲授的“消去法”将以科学出版社 2002 年出版的《消去法及其应用》一书作为教材, 因而这门课的内容不在本书之列. 这本书中的材料自

然难以囊括符号计算的大部分内容，因而不可能反映符号计算的全貌，但书中六章确实介绍了符号计算的六个主要领域. 值得一提的是，我国学者在这些领域都有突出的研究成果. 证明几何定理的吴方法更是我国的独创. 我们希望这本教材有助于符号计算在我国的传播和发展，能为后来者攀登学科顶峰铺路，为各界学者提供参考，也希望读者能从中领略到符号计算的美妙和效用.

本书的编写时间非常仓促，错误在所难免，欢迎读者指正.

王东明

2003 年 5 月于巴黎

2003 年符号计算黄山暑期讲习班由北京大学数学科学学院、中国科学院数学与系统科学研究院和中国科学技术大学理学院联合主办，中国科学技术大学数学系承办.

学术协调委员会

王东明 (中国科学技术大学 / 法国科学研究中心)，主任
郑志明 (北京大学)
刘卓军 (中国科学院)
陈发来 (中国科学技术大学)

组织委员会

陈发来 (中国科学技术大学)，主任
胡　森 (中国科学技术大学)
夏壁灿 (北京大学)

http://www.math.pku.edu.cn/sssc2003

目　录

第一章　计算机代数

(王东明　夏壁灿)

符号计算的一个主要分支是计算机代数，它所研究的对象是抽象的数学符号与概念，如整数、有理数、多项式、理想等. 设计处理这些代数对象的算法，将其在计算机上有效地实施，并用以解决各种代数计算和推理问题是计算机代数学的中心课题. 本章介绍计算机代数的一些基本内容，包括大整数和多项式的表示及运算、模方法、多项式的最大公因子和因式分解.

1.1 引　论

1.1.1 什么是计算机代数

计算机代数是以现代计算机为工具，研究代数对象的一门新兴学科. 这里代数对象是指抽象的数学符号与概念，如整数、有理数、多项式、理想等，而如何处理这些代数对象则是该学科研究的主题. 计算机代数的特征是符号与代数计算，它有别于通常的数值计算. 代数算法的设计、分析、实现及应用构成了计算机代数的研究内容.

代数计算是冗长繁复的，常常让人望而生畏. 传统的纸笔演算耗时、费力又易出错，因而不可能用于大规模的计算. 现代计算技术为大型符号计算提供了条件. 于是如何将基本代数理论算法化、精确化、效率化，如何将有效的算法在计算机上有效地实施，建立完整易用的软件系统，并用来处理形形色色的代数计算都是需要研究的问题. 对这些问题的研究便形成了计算机代数这门学科.

计算机代数的发展始于 20 世纪 60 年代初期. 其标志是美国 J. Slagle 在 1961 年用表处理语言 Lisp 所写的第一个自动符号积分程序 SAINT. 随后，几个基于 Fortran 和 Lisp 的符号计算系统，如 FORMAC, ALPAK, PM, MATHLAB 等，相继出现. 这些早期的系统主要是在美国的麻省理工学院、贝尔实验室和 IBM 公司研制开发的. 不难想象，计算机代数软件系统的开发始终刺激和左右着代数算法的研究. 我们将在 1.1.3 节中简单介绍一些主要的计算机代数系统.

1.1.2 理论、算法与实施

计算机代数的基础是 *构造性* 代数，加之逻辑与算法理论. 这里重点是代数对象的构造而非存在性证明，并且数学的理论和方法需要更加严密，而逻辑学与算法理论则有助于将这些数学理论和方法形式化和算法化.

计算机代数中最基本的问题是 *算法设计*, 即依据已有的或者发展新的数学理论，提出有效的方法，将这些方法描述为适合实施的 *算法*, 并证明所设计算法的 *正确性* 和 *终止性*. 将这些算法优化并研究其效率 (所需要的计算时间和计算机存储) 便是算法分析的内容. 算法分析的主要方式包括算法的复杂性分析和实验测试.

算法实施 是指将具体的算法翻译为某种特定计算机语言中的程序，以便这些算法能在计算机上运行. 这里的翻译需要正确、有效.

例 1.1.1　计算非负整数阶乘的算法可以描述如下.

算法 Factorial: $m := \text{Factorial}(n)$. 任给整数 n, 本算法计算 n 的阶乘 m (即 $m = n!$).

F1. 若 $n < 0$, 则指出 "非法输入" 且程序终止.

F2. 若 $n = 0$, 则 $m := 1$; 否则，计算 $m := n \cdot \text{Factorial}(n-1)$.

这一算法可以在不同的程序语言中实施. 例如用 Maple 语言，我们可将其翻译为下列程序.

```
Factorial := proc(n)
     if n < 0 then RETURN(‘Invalid input‘)
     elif n = 0 then 1
     else n*Factorial(n-1)
     fi
end:
```

1.1.3 计算机代数系统

设计和实施一个理想的计算机代数系统是长期复杂的高技术项目，牵涉到计算机软件工程. 设计者需要了解计算机硬件和软件的最新发展，预测其未来走向，因而选取适当的基础编程语言，确定软件的特征和开发步骤，需要平衡软件的效率、界面、实用性、易用性、兼容性、可扩展性、开发时间和人力资源等诸多因素. 实施者需要有一定的计算机代数知识和编程技巧，顾及程序的结构、可读性和易改性，提供程序的说明文本，并能与其他实施者协调合作.

早期 (1980 年之前) 出现的计算机代数系统基本上都是基于 Lisp 和 Fortran 两种程序语言. 这些系统的主要功能是处理多项式和有理函数, 用于有关符号和物理计算, 其中大多数系统都是在美国研制开发的. 我们不再介绍那些已经过时了的软件系统, 而只将部分延续至今的计算机代数系统图示如下:

PM $\longrightarrow$ SAC-1 $\longrightarrow$ SAC-2 $\longrightarrow$ SAC/ALDES $\longrightarrow$ SACLIB
Reduce $\longrightarrow$ Reduce 2 $\longrightarrow$ Reduce 3
Scratchpad $\longrightarrow$ Scratchpad II $\longrightarrow$ Axiom
muMATH $\longrightarrow$ Derive

除 Axiom 和 Derive 外, 现行的计算机代数系统大多基于 C 语言. 这些系统的功能都极为丰富, 其中有些是为特殊目的开发的, 而那些一般性系统的功能已远远超出了计算机代数. 它们可以从事各种符号、数值和图形计算. 我们列出部分流行的计算机代数系统及其网址如下. 有兴趣的读者可据此获取更多的资讯.

- Aldor (http://www.aldor.org/)
- Axiom (http://home.earthlink.net/~jgg964/axiom.html)
- CoCoA (http://cocoa.dima.unige.it/)
- Derive (http://www.derive.com/)
- Macaulay 2 (http://www.math.uiuc.edu/Macaulay2/)
- Macsyma (http://www.scientek.com/macsyma/main.htm
  http://www.gosw.com/gosw/MacsymaInc/Macsyma422UxLx.html)
- Magma (http://magma.maths.usyd.edu.au/magma/)
- Maple (http://www.maplesoft.com/)
- Mathematica (http://www.wolfram.com/products/mathematica/)
- Maxima (http://maxima.sourceforge.net/)
- MuPAD (http://www.mupad.de/)
- Reduce (http://www.uni-koeln.de/REDUCE/)
- Risa/Asir (http://www.asir.org/)
- Singular (http://www.singular.uni-kl.de/)

1.1.4 问题及应用举例

在这一小节里，我们用几个简单的例子来说明计算机代数中的一些基本问题. 也许这能让读者对计算机代数到底能做什么有个粗略的了解.

1. *大整数运算*. 例如

$$20!/2^{20} = 9280784638125/4.$$

2. *整数因子分解*. 如何有效地将任给正整数分解为素数的乘积. 例如

$$20020408 = 2^3 \cdot 2502551,$$

其中 2502551 为素数.

3. *多项式因子分解*. 如何有效地将任给多项式在某一给定的数域上分解为不可约因子的乘积. 例如，在有理数域上有

$$x^8 - y^8 = (x-y)(x+y)(x^2+y^2)(x^4+y^4).$$

4. *多项式理想的准素分解*. 如何有效地将一组任给多项式生成的理想分解为准素理想的交.

5. *多项式的正定性*. 判定任给多项式是否正定. 例如

$$(\forall x, y)\ x^6 - x^4y^2 - x^2y^4 + y^6 - x^4 + 3x^2y^2 - y^4 - x^2 - y^2 + 1 \geqslant 0.$$

6. *解代数方程组*. 这是计算机代数，也是数学中的基本问题.

例 1.1.2　求代数曲线

$$F = xy(1-x-y)^2 = 0$$

的临界点. 为此，计算

$$F_1 = \frac{\partial F}{\partial x} = y(1-x-y)^2 - 2xy(1-x-y),$$
$$F_2 = \frac{\partial F}{\partial y} = x(1-x-y)^2 - 2xy(1-x-y).$$

用计算机代数系统解多项式方程组 $F_1 = 0, F_2 = 0$ 可得三组解：

$$\{x = 0, y = 0\},\quad \{x = 1-y, y = y\},\quad \{x = 1/4, y = 1/4\}.$$

计算机代数方法和系统还可以用于几何定理的*机器证明*，代数曲线和曲面的*参数化*，微分方程求解等诸多几何和分析中的计算问题. 读者将在以后的有关章节中看到这些应用. 本章的编写主要参照著作 [49, 115, 109, 105, 79].

1.2 数据表示及基本运算

计算机代数研究诸如整数、多项式、理想等具有基本代数结构的对象. 因此，一个首要的问题是如何在计算机上表示这些抽象的对象. 本章以大整数和多项式的表示与运算为例，说明数据表示的基本原理而不涉及特定的技巧.

1.2.1 大整数的表示

整数是最基本的代数对象，而整数的加减乘除则是最基本的运算. 在计算机代数中，我们常常会遇到 (超过计算机字长所能表示的) 大整数. 例如求多项式

$$F = 7x^7 + 2x^6 - 3x^5 - 3x^3 + x + 5,$$
$$G = 9x^5 - 3x^4 - 4x^2 + 7x + 7$$

的最大公因子. 使用熟知的 Euclid 算法 并在计算过程中将有理数化为整数，我们得到如下的多项式余式序列：

$$1890x^4 - 4752x^3 - 6930x^2 - 846x + 4527,$$
$$294168996x^3 + 257191200x^2 - 20614662x - 142937946,$$
$$-10368527836984130520 0x^2 - 325760542331156100 00x + 122453167842311670000,$$
$$29567908335036495467893420575652070982917635200 00x + 55532526180624799696603478407402529168762016 0000,$$
$$10920746857330312192010416027912598626591699661845938035186024187771406828843347696470604035436077376984268800000000000.$$

最后一个整数有 118 位，远远超过了计算机字长所能表示的最大整数. 当然，对于本例中的问题使用其他算法 (如模方法) 可以将系数控制在很小的范围. 像本例这种输入 (F 和 G) 和输出 (1 — 表示互素) 都不大，但中间数据很大的问题在计算机代数中被称为中间表示膨胀 (intermidiate expression swell)，它是计算机代数所面临的挑战性问题.

大整数的表示在原理上其实很简单. 设 $B > 1$ 为一整数，那么正整数 a 在 B 进制下可惟一地表示为

$$a = \sum_{i=0}^{n} a_i B^i, \quad 0 \leqslant a_i < B,\ a_n \neq 0.$$

称 a 是 $n+1$ 位的 (B 进制) 数. 显然，B 越大，相同的位数可表示的数就越大. 因此，在计算机上可以用一个字来表示 $B-1$，而用一列字来分别表示 B

进制下的每一位. 这样, 如果在一个 32 位的计算机上取 $B=2^{32}$, 并且用一列字的第一个存储数的符号及数位的大小, 那么可表示的整数集合为

$$\left\{(-1)^s\sum_{i=0}^{n}a_iB^i \;\middle|\; s\in\{0,1\},\ 0\leqslant n<2^{31}-1,\ 0\leqslant a_i\leqslant 2^{32}-1\right\}.$$

也就是说, $n+1$ 位的 B 进制数可用长为 $n+2$ 的阵列表示为

$$s\cdot 2^{32}+n+1, a_0,\cdots,a_n.$$

在现行的计算机代数系统 (如 Maple) 中, 阵列的第一个字通常要存储关于数据类型等其他信息, 因此不可能用 31 位来表示 $n+1$. 从效率和实际需要出发, 也不取 $B=2^{32}$. 以 Maple 5.2 为例, 用第一个字中的 17 位表示 $n+1$ 同时取 $B=10^4$ (这里 4 是满足 $B^2=(10^k)^2<2^{32}$ 的最大的 k), 这样所能表示的最大整数是 $(10^4)^{(2^{17}-1-1)}-1$, 长度为 $2^{19}-9$.

1.2.2 整数运算

我们所关心的是整数的运算. 加减是相对简单的, 所以我们首先考虑的是整数的乘法. 除非特别声明, 我们说的数都是指 B 进制的.

下面仅讨论两个正整数的乘法算法. 两个整数相乘的经典算法是将第二个数的每一位与第一个数的每一位相乘, 然后再将所得的结果适当对位相加. 如果两个数的位数都是 n, 那么所需一位数乘法的总次数为 n^2. 因此, 经典乘法的计算复杂性与 n^2 成正比. 设

$$a=\sum_{i=0}^{m-1}a_iB^i,\quad b=\sum_{i=0}^{n-1}b_iB^i,$$

记为 $a=[a_{m-1},\cdots,a_0]$, $b=[b_{n-1},\cdots,b_0]$. 不妨设 $m\geqslant n$, 并记

$$c=ab=[c_{m+n-1},\cdots,c_0].$$

按经典乘法得表如下:

$$\begin{array}{ccccccccc}
B^{m+n-2} & \cdots & B^m & B^{m-1} & \cdots & B^{n-1} & \cdots & B^1 & B^0\\
 & & & a_{m-1}b_0 & \cdots & a_{n-1}b_0 & \cdots & a_1b_0 & a_0b_0\\
 & & a_{m-1}b_1 & a_{m-2}b_1 & \cdots & a_{n-2}b_1 & \cdots & a_0b_1 & \\
 & & \vdots & \vdots & & \vdots & & & \\
a_{m-1}b_{n-1} & \cdots & a_{m-n+1}b_{n-1} & a_{m-n}b_{n-1} & \cdots & a_0b_{n-1} & & &
\end{array}$$

显然，直接对位相加是不足取的. 实际算法的核心部分可用 Maple 语言描述如下.

输入： $a=[a_{m-1},\cdots,a_0]$, $b=[b_{n-1},\cdots,b_0]$.

输出： $c=ab=[c_{m+n-1},\cdots,c_0]$.

```
for i from 0 to m-1 do c[i] := 0 od;
for j from 0 to n-1 do
    r := 0;
    for k from 0 to m-1 do
        t := a[k]*b[j]+c[k+j]+r;
        c[k+j] := irem(t,B);
        r := iquo(t,B)
    od;
    c[j+m] := r
od;
```

这里 iquo(t,B) 和 irem(t,B) 分别表示 t 除以 B 的商和余数.

命题 1.2.1 对上述算法，不等式 $r<B$ 和 $t<B^2$ 成立.

证 我们使用归纳法. 首先注意，对任意 i,j,k 都有 $a_i,b_j,c_k<B$. 在 $k=0$ 时， $r_0=0<B$, 从而

$$t_0=a_0b_j+c_j+r_0<(B-1)^2+B<B^2.$$

设结论对 $k<l$ 成立，于是 $r_l=\mathtt{iquo}(t_{l-1},B)<B$, 因而

$$t_l=a_lb_j+c_{l+j}+r_l<(B-1)^2+(B-1)+B=B^2. \qquad \square$$

下面介绍 A. Karatsuba 提出的快速算法，它的复杂性与 $n^{1.58}$ 成正比.

设 x 和 y 为两个位数 $\leqslant n$ 的正整数. 将 x,y 各分成两段，其位数 $\leqslant n/2$, 使得

$$x=aB^{n/2}+b,\quad y=cB^{n/2}+d. \tag{1.2.1}$$

那么

$$\begin{aligned} x\cdot y &= acB^n+(ad+bc)B^{n/2}+bd \\ &= acB^n+[(a+b)(c+d)-ac-bd]B^{n/2}+bd. \end{aligned} \tag{1.2.2}$$

因此，我们只需作三次位数 $\leqslant n/2$ 的整数乘法，以及一些对位和加法. 下面我们证明这种乘法的计算复杂性为 $O(n^{\log_2 3})$.

首先假定 n 是 2 的某次幂，比如说 $n=2^m$，而 x 和 y 是位数不超过 n 的正整数. 又设 a,b,c,d 如 (1.2.1) 所示. 按照 (1.2.2)，我们需要计算乘积 $(a+b)(c+d), ac, bd$. 其他运算都是对位和相加，它们的复杂性与 n 成正比.

乘积 ac 和 bd 中的因子都不超过 $n/2$ 位，但 $a+b$ 和 $c+d$ 有可能是 $n/2+1$ 位. 现将 $a+b$ 和 $c+d$ 用 B 进制来表示：

$$a+b=a_1B^{n/2}+b_1, \quad c+d=c_1B^{n/2}+d_1.$$

那么

$$(a+b)(c+d)=a_1c_1B^n+(a_1d_1+b_1c_1)B^{n/2}+b_1d_1.$$

上式中 b_1 和 d_1 的位数都不超过 $n/2$，而其他运算为对位、相加和含一位数的乘积，其复杂性的总和与 n 成正比.

用 $T(n)$ 表示两个位数为 n 的正整数相乘的复杂性. 于是

$$T(1)=1, \qquad T(n)=3T(n/2)+Cn, \quad n>1,$$

其中 C 为某一常数. 由此可知

$$\begin{aligned} T(n)=T(2^m) &= 3[3T(2^{m-2})+C2^{m-1}]+C2^m \\ &= 3^2T(2^{m-2})+C2^m(1+3/2)=\cdots \\ &= 3^mT(1)+C2^m[1+3/2+\cdots+(3/2)^{m-1}] \\ &= 3^m+C2^m\frac{(3/2)^m-1}{(3/2)-1} \\ &= (2C+1)3^m-2Cn=(2C+1)n^{\log_2 3}-2Cn. \end{aligned}$$

这就证明了在 $n=2^m$ 时，$T(n)=O(n^{\log_2 3})\doteq O(n^{1.58})$.

对任意正整数 n，我们可以在 x 和 y 的前面加 0 使其位数变成 2 的 (最可能小的) 次幂. 这一过程至多将 x 和 y 的位数加倍. 所以此时有

$$T(n)=O\left((2n)^{\log_2 3}\right)=O(n^{\log_2 3}).$$

证毕.

现将基于上述结果的 Karatsuba 算法描述如下.

算法 Mult: $z:=\mathrm{Mult}(x,y)$. 任给正整数 x 和 y，本算法计算 x 和 y 的乘积 z (即 $z=x\cdot y$).

M1. 设 x 和 y 的位数分别为 n_x 和 n_y, 且命 $n := \max\{n_x, n_y\}$. 若 n 为奇数, 则命 $n := (n+1)/2$; 否则命 $n := n/2$. 如果 $n = 1$, 则命 $z := xy$, 且算法终止.

M2. 若 $n_x \leqslant n$, 则命 $a := 0$, $b := x$; 否则设 a 为 x 的前 $n_x - n$ 位, 而 b 为 x 的后 n 位.

若 $n_y \leqslant n$, 则命 $c := 0$, $d := y$; 否则设 c 为 y 的前 $n_y - n$ 位, 而 d 为 y 的后 n 位.

M3. 计算 $u := \mathrm{Mult}(a+b, c+d)$, $v := \mathrm{Mult}(a, c)$, $w := \mathrm{Mult}(b, d)$.

M4. 计算 $z := vB^{2n} + (u - v - w)B^n + w$.

由于常数 C 可能很大, 以上算法只在 n 很大时适用. 该算法的终止性是明显的, 而上面的论证保证了它的正确性.

下面我们讨论除法和最大公因子. 考虑两个整数 a 和 b, 其中 $b \neq 0$. 我们可以求出整数 q 和 r 使得

$$a = q \cdot b + r, \quad \text{且} \quad \begin{cases} 0 \leqslant r < |b|, & \text{若 } a \geqslant 0, \\ -|b| < r \leqslant 0, & \text{若 } a < 0. \end{cases} \tag{1.2.3}$$

整数 q 和 r 分别称为 a 对 b 的 **商** 和 **余数**, 用 $\mathrm{quo}(a, b)$ 和 $\mathrm{rem}(a, b)$ 来表示. 它们由 a 和 b 惟一确定. 如果 $r = 0$, 则说 b **整除** a, 记作 $b \mid a$. 这时也称 b 为 a 的因子, 而 a 为 b 的 **倍数**.

作除法最主要的问题就是试商. 在通常的纸笔演算中, 试商过程包含猜测和直觉, 因而不是算法或者说需要算法化. 设 a 和 b 分别为 $n+1$ 位和 n 位的 B 进制正整数:

$$a = \sum_{i=0}^{n} a_i B^{n-i}, \quad b = \sum_{i=1}^{n} b_i B^{n-i},$$

并且假设 $\dfrac{a}{b} < B$. 依带余除法, 我们有余式公式 (1.2.3). 下面的定理给出了试商的基本准则.

定理 1.2.2 令 $\hat{q} = \min\Big(\Big[\dfrac{a_0 B + a_1}{b_1}\Big], B - 1\Big)$, 则 $q \leqslant \hat{q}$.