

Victor Shoup

A Computational  
Introduction

to  
and **Number Theory**  
**Algebra**

CAMBRIDGE

0156  
S559

A COMPUTATIONAL INTRODUCTION  
TO NUMBER THEORY AND ALGEBRA

VICTOR SHOUP



E200602414



**CAMBRIDGE**  
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS  
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press  
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

[www.cambridge.org](http://www.cambridge.org)  
Information on this title: [www.cambridge.org/9780521851541](http://www.cambridge.org/9780521851541)

© V. Shoup 2005

This book is in copyright. All rights reserved.

First published 2005

Printed in the United Kingdom at the University Press, Cambridge

*A catalog record for this book is available from the British Library*

ISBN-13 978-0-521-85154-1 hardback  
ISBN-10 0-521-85154-8 hardback

ISBN-13 978-0-521-61725-3 paperback  
ISBN-10 0-521-61725-1 paperback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party internet websites referred to in this book, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

A COMPUTATIONAL INTRODUCTION  
TO NUMBER THEORY AND ALGEBRA

# Preface

Number theory and algebra play an increasingly significant role in computing and communications, as evidenced by the striking applications of these subjects to such fields as cryptography and coding theory. My goal in writing this book was to provide an introduction to number theory and algebra, with an emphasis on algorithms and applications, that would be accessible to a broad audience. In particular, I wanted to write a book that would be accessible to typical students in computer science or mathematics who have a some amount of *general* mathematical experience, but without presuming too much *specific* mathematical knowledge.

**Prerequisites.** The mathematical prerequisites are minimal: no particular mathematical concepts beyond what is taught in a typical undergraduate calculus sequence are assumed.

The computer science prerequisites are also quite minimal: it is assumed that the reader is proficient in programming, and has had some exposure to the analysis of algorithms, essentially at the level of an undergraduate course on algorithms and data structures.

Even though it is mathematically quite self contained, the text does presuppose that the reader is comfortable with mathematical formalism and has some experience in reading and writing mathematical proofs. Readers may have gained such experience in computer science courses such as algorithms, automata or complexity theory, or some type of “discrete mathematics for computer science students” course. They also may have gained such experience in undergraduate mathematics courses, such as abstract or linear algebra—these courses overlap with some of the material presented here, but even if the reader already has had some exposure to this material, it nevertheless may be convenient to have all of the relevant material easily accessible in one place, and moreover, the emphasis and perspective here

will no doubt be different than in a typical mathematics course on these subjects.

**Structure of the text.** All of the mathematics required beyond basic calculus is developed “from scratch.” Moreover, the book generally alternates between “theory” and “applications”: one or two chapters on a particular set of purely mathematical concepts are followed by one or two chapters on algorithms and applications—the mathematics provides the theoretical underpinnings for the applications, while the applications both motivate and illustrate the mathematics. Of course, this dichotomy between theory and applications is not perfectly maintained: the chapters that focus mainly on applications include the development of some of the mathematics that is specific to a particular application, and very occasionally, some of the chapters that focus mainly on mathematics include a discussion of related algorithmic ideas as well.

In developing the mathematics needed to discuss certain applications, I tried to strike a reasonable balance between, on the one hand, presenting the absolute minimum required to understand and rigorously analyze the applications, and on the other hand, presenting a full-blown development of the relevant mathematics. In striking this balance, I wanted to be fairly economical and concise, while at the same time, I wanted to develop enough of the theory so as to present a fairly well-rounded account, giving the reader more of a feeling for the mathematical “big picture.”

The mathematical material covered includes the basics of number theory (including unique factorization, congruences, the distribution of primes, and quadratic reciprocity) and abstract algebra (including groups, rings, fields, and vector spaces). It also includes an introduction to discrete probability theory—this material is needed to properly treat the topics of probabilistic algorithms and cryptographic applications. The treatment of all these topics is more or less standard, except that the text only deals with commutative structures (i.e., abelian groups and commutative rings with unity)—this is all that is really needed for the purposes of this text, and the theory of these structures is much simpler and more transparent than that of more general, non-commutative structures.

The choice of topics covered in this book was motivated primarily by their applicability to computing and communications, especially to the specific areas of cryptography and coding theory. For example, the book may be useful for reference or self-study by readers who want to learn about cryptography. The book could also be used as a textbook in a graduate

or upper-division undergraduate course on (computational) number theory and algebra, perhaps geared towards computer science students.

Since this is an introductory textbook, and not an encyclopedic reference for specialists, some topics simply could not be covered. One such topic whose exclusion will undoubtedly be lamented by some is the theory of lattices, along with algorithms for and applications of lattice basis reduction. Another such topic is that of fast algorithms for integer and polynomial arithmetic—although some of the basic ideas of this topic are developed in the exercises, the main body of the text deals only with classical, quadratic-time algorithms for integer and polynomial arithmetic. As an introductory text, some topics just had to go; moreover, there are more advanced texts that cover these topics perfectly well, and these texts should be readily accessible to students who have mastered the material in this book.

Note that while continued fractions are not discussed, the closely related problem of “rational reconstruction” is covered, along with a number of interesting applications (which could also be solved using continued fractions).

**Using the text.** Here are a few tips on using the text.

- There are a few sections that are marked with a “(\*),” indicating that the material covered in that section is a bit technical, and is not needed elsewhere.
- There are many examples in the text. These form an integral part of the text, and should not be skipped.
- There are a number of exercises in the text that serve to reinforce, as well as to develop important applications and generalizations of, the material presented in the text. In solving exercises, the reader is free to use any *previously* stated results in the text, including those in previous exercises. However, except where otherwise noted, any result in a section marked with a “(\*),” or in §5.5, need not and should not be used outside the section in which it appears.
- There is a very brief “Preliminaries” chapter, which fixes a bit of notation and recalls a few standard facts. This should be skimmed over by the reader.
- There is an appendix that contains a few useful facts; where such a fact is used in the text, there is a reference such as “see §An,” which refers to the item labeled “An” in the appendix.

**Feedback.** I welcome comments on the book (suggestions for improvement, error reports, etc.) from readers. Please send your comments to

victor@shoup.net.

There is also web site where further material and information relating to the book (including a list of errata and the latest electronic version of the book) may be found:

[www.shoup.net/ntb](http://www.shoup.net/ntb).

**Acknowledgments.** I would like to thank a number of people who volunteered their time and energy in reviewing one or more chapters: Sidhartha Annapureddy, John Black, Carl Bosley, Joshua Brody, Jan Camenisch, Ronald Cramer, Alex Dent, Nelly Fazio, Mark Giesbrecht, Stuart Haber, Alfred Menezes, Antonio Nicolosi, Roberto Oliveira, and Louis Salvail. Thanks to their efforts, the “bug count” has been significantly reduced, and the readability of the text much improved. I am also grateful to the National Science Foundation for their support provided under grant CCR-0310297. Thanks to David Tranah and his colleagues at Cambridge University Press for their progressive attitudes regarding intellectual property and open access.

*New York, January 2005*

*Victor Shoup*



# Preliminaries

We establish here a few notational conventions used throughout the text.

## *Arithmetic with $\infty$*

We shall sometimes use the symbols “ $\infty$ ” and “ $-\infty$ ” in simple arithmetic expressions involving real numbers. The interpretation given to such expressions is the usual, natural one; for example, for all real numbers  $x$ , we have  $-\infty < x < \infty$ ,  $x + \infty = \infty$ ,  $x - \infty = -\infty$ ,  $\infty + \infty = \infty$ , and  $(-\infty) + (-\infty) = -\infty$ . Some such expressions have no sensible interpretation (e.g.,  $\infty - \infty$ ).

## *Logarithms and exponentials*

We denote by  $\log x$  the natural logarithm of  $x$ . The logarithm of  $x$  to the base  $b$  is denoted  $\log_b x$ .

We denote by  $e^x$  the usual exponential function, where  $e \approx 2.71828$  is the base of the natural logarithm. We may also write  $\exp[x]$  instead of  $e^x$ .

## *Sets and relations*

We use the symbol  $\emptyset$  to denote the empty set. For two sets  $A, B$ , we use the notation  $A \subseteq B$  to mean that  $A$  is a subset of  $B$  (with  $A$  possibly equal to  $B$ ), and the notation  $A \subsetneq B$  to mean that  $A$  is a proper subset of  $B$  (i.e.,  $A \subseteq B$  but  $A \neq B$ ); further,  $A \cup B$  denotes the union of  $A$  and  $B$ ,  $A \cap B$  the intersection of  $A$  and  $B$ , and  $A \setminus B$  the set of all elements of  $A$  that are not in  $B$ .

For sets  $S_1, \dots, S_n$ , we denote by  $S_1 \times \dots \times S_n$  the **Cartesian product**

of  $S_1, \dots, S_n$ , that is, the set of all  $n$ -tuples  $(a_1, \dots, a_n)$ , where  $a_i \in S_i$  for  $i = 1, \dots, n$ .

We use the notation  $S^{\times n}$  to denote the Cartesian product of  $n$  copies of a set  $S$ , and for  $x \in S$ , we denote by  $x^{\times n}$  the element of  $S^{\times n}$  consisting of  $n$  copies of  $x$ . (We shall reserve the notation  $S^n$  to denote the set of all  $n$ th powers of  $S$ , assuming a multiplication operation on  $S$  is defined.)

Two sets  $A$  and  $B$  are **disjoint** if  $A \cap B = \emptyset$ . A collection  $\{C_i\}$  of sets is called **pairwise disjoint** if  $C_i \cap C_j = \emptyset$  for all  $i, j$  with  $i \neq j$ .

A **partition** of a set  $S$  is a pairwise disjoint collection of non-empty subsets of  $S$  whose union is  $S$ . In other words, each element of  $S$  appears in exactly one subset.

A **binary relation** on a set  $S$  is a subset  $R$  of  $S \times S$ . Usually, one writes  $a \sim b$  to mean that  $(a, b) \in R$ , where  $\sim$  is some appropriate symbol, and rather than refer to the relation as  $R$ , one refers to it as  $\sim$ .

A binary relation  $\sim$  on a set  $S$  is called an **equivalence relation** if for all  $x, y, z \in S$ , we have

- $x \sim x$  (reflexive property),
- $x \sim y$  implies  $y \sim x$  (symmetric property), and
- $x \sim y$  and  $y \sim z$  implies  $x \sim z$  (transitive property).

If  $\sim$  is an equivalence relation on  $S$ , then for  $x \in S$  one defines the set  $[x] := \{y \in S : x \sim y\}$ . Such a set  $[x]$  is an **equivalence class**. It follows from the definition of an equivalence relation that for all  $x, y \in S$ , we have

- $x \in [x]$ , and
- either  $[x] \cap [y] = \emptyset$  or  $[x] = [y]$ .

In particular, the collection of all distinct equivalence classes partitions the set  $S$ . For any  $x \in S$ , the set  $[x]$  is called the **equivalence class containing**  $x$ , and  $x$  is called a **representative** of  $[x]$ .

## Functions

For any function  $f$  from a set  $A$  into a set  $B$ , if  $A' \subseteq A$ , then  $f(A') := \{f(a) \in B : a \in A'\}$  is the **image** of  $A'$  under  $f$ , and  $f(A)$  is simply referred to as the **image** of  $f$ ; if  $B' \subseteq B$ , then  $f^{-1}(B') := \{a \in A : f(a) \in B'\}$  is the **pre-image** of  $B'$  under  $f$ .

A function  $f : A \rightarrow B$  is called **one-to-one** or **injective** if  $f(a) = f(b)$  implies  $a = b$ . The function  $f$  is called **onto** or **surjective** if  $f(A) = B$ . The function  $f$  is called **bijective** if it is both injective and surjective; in this case,  $f$  is called a **bijection**. If  $f$  is bijective, then we may define the

**inverse function**  $f^{-1} : B \rightarrow A$ , where for  $b \in B$ ,  $f^{-1}(b)$  is defined to be the unique  $a \in A$  such that  $f(a) = b$ .

If  $f : A \rightarrow B$  and  $g : B \rightarrow C$  are functions, we denote by  $g \circ f$  their composition, that is, the function that sends  $a \in A$  to  $g(f(a)) \in C$ . Function composition is associative; that is, for functions  $f : A \rightarrow B$ ,  $g : B \rightarrow C$ , and  $h : C \rightarrow D$ , we have  $(h \circ g) \circ f = h \circ (g \circ f)$ . Thus, we can simply write  $h \circ g \circ f$  without any ambiguity. More generally, if we have functions  $f_i : A_i \rightarrow A_{i+1}$  for  $i = 1, \dots, n$ , where  $n \geq 2$ , then we may write their composition as  $f_n \circ \dots \circ f_1$  without any ambiguity. As a special case of this, if  $A_i = A$  and  $f_i = f$  for  $i = 1, \dots, n$ , then we may write  $f_n \circ \dots \circ f_1$  as  $f^n$ . It is understood that  $f^1 = f$ , and that  $f^0$  is the identity function on  $A$ . If  $f$  is a bijection, then so is  $f^n$  for any non-negative integer  $n$ , the inverse function of  $f^n$  being  $(f^{-1})^n$ , which one may simply write as  $f^{-n}$ .

### *Binary operations*

A **binary operation**  $\star$  on a set  $S$  is a function from  $S \times S$  to  $S$ , where the value of the function at  $(a, b) \in S \times S$  is denoted  $a \star b$ .

A binary operation  $\star$  on  $S$  is called **associative** if for all  $a, b, c \in S$ , we have  $(a \star b) \star c = a \star (b \star c)$ . In this case, we can simply write  $a \star b \star c$  without any ambiguity. More generally, for  $a_1, \dots, a_n \in S$ , where  $n \geq 2$ , we can write  $a_1 \star \dots \star a_n$  without any ambiguity.

A binary operation  $\star$  on  $S$  is called **commutative** if for all  $a, b \in S$ , we have  $a \star b = b \star a$ . If the binary operation  $\star$  is both associative and commutative, then not only is the expression  $a_1 \star \dots \star a_n$  unambiguous, but its value remains unchanged even if we re-order the  $a_i$ .

# Contents

<i>Preface</i>	<i>page</i> x
<i>Preliminaries</i>	xiv
<b>1 Basic properties of the integers</b>	1
1.1 Divisibility and primality	1
1.2 Ideals and greatest common divisors	4
1.3 Some consequences of unique factorization	8
<b>2 Congruences</b>	13
2.1 Definitions and basic properties	13
2.2 Solving linear congruences	15
2.3 Residue classes	20
2.4 Euler's phi function	24
2.5 Fermat's little theorem	25
2.6 Arithmetic functions and Möbius inversion	28
<b>3 Computing with large integers</b>	33
3.1 Asymptotic notation	33
3.2 Machine models and complexity theory	36
3.3 Basic integer arithmetic	39
3.4 Computing in $\mathbb{Z}_n$	48
3.5 Faster integer arithmetic (*)	51
3.6 Notes	52
<b>4 Euclid's algorithm</b>	55
4.1 The basic Euclidean algorithm	55
4.2 The extended Euclidean algorithm	58
4.3 Computing modular inverses and Chinese remaindering	62
4.4 Speeding up algorithms via modular computation	63
4.5 Rational reconstruction and applications	66
4.6 Notes	73

<b>5</b>	<b>The distribution of primes</b>	74
5.1	Chebyshev's theorem on the density of primes	74
5.2	Bertrand's postulate	78
5.3	Mertens' theorem	81
5.4	The sieve of Eratosthenes	85
5.5	The prime number theorem . . . and beyond	86
5.6	Notes	94
<b>6</b>	<b>Finite and discrete probability distributions</b>	96
6.1	Finite probability distributions: basic definitions	96
6.2	Conditional probability and independence	99
6.3	Random variables	104
6.4	Expectation and variance	111
6.5	Some useful bounds	117
6.6	The birthday paradox	121
6.7	Hash functions	125
6.8	Statistical distance	130
6.9	Measures of randomness and the leftover hash lemma (*)	136
6.10	Discrete probability distributions	141
6.11	Notes	147
<b>7</b>	<b>Probabilistic algorithms</b>	148
7.1	Basic definitions	148
7.2	Approximation of functions	155
7.3	Flipping a coin until a head appears	158
7.4	Generating a random number from a given interval	159
7.5	Generating a random prime	162
7.6	Generating a random non-increasing sequence	167
7.7	Generating a random factored number	170
7.8	The RSA cryptosystem	174
7.9	Notes	179
<b>8</b>	<b>Abelian groups</b>	180
8.1	Definitions, basic properties, and examples	180
8.2	Subgroups	185
8.3	Cosets and quotient groups	190
8.4	Group homomorphisms and isomorphisms	194
8.5	Cyclic groups	202
8.6	The structure of finite abelian groups (*)	208
<b>9</b>	<b>Rings</b>	211
9.1	Definitions, basic properties, and examples	211
9.2	Polynomial rings	220

9.3	Ideals and quotient rings	231
9.4	Ring homomorphisms and isomorphisms	236
<b>10</b>	<b>Probabilistic primality testing</b>	<b>244</b>
10.1	Trial division	244
10.2	The structure of $\mathbb{Z}_n^*$	245
10.3	The Miller–Rabin test	247
10.4	Generating random primes using the Miller–Rabin test	252
10.5	Perfect power testing and prime power factoring	261
10.6	Factoring and computing Euler’s phi function	262
10.7	Notes	266
<b>11</b>	<b>Finding generators and discrete logarithms in <math>\mathbb{Z}_p^*</math></b>	<b>268</b>
11.1	Finding a generator for $\mathbb{Z}_p^*$	268
11.2	Computing discrete logarithms $\mathbb{Z}_p^*$	270
11.3	The Diffie–Hellman key establishment protocol	275
11.4	Notes	281
<b>12</b>	<b>Quadratic residues and quadratic reciprocity</b>	<b>283</b>
12.1	Quadratic residues	283
12.2	The Legendre symbol	285
12.3	The Jacobi symbol	287
12.4	Notes	289
<b>13</b>	<b>Computational problems related to quadratic residues</b>	<b>290</b>
13.1	Computing the Jacobi symbol	290
13.2	Testing quadratic residuosity	291
13.3	Computing modular square roots	292
13.4	The quadratic residuosity assumption	297
13.5	Notes	298
<b>14</b>	<b>Modules and vector spaces</b>	<b>299</b>
14.1	Definitions, basic properties, and examples	299
14.2	Submodules and quotient modules	301
14.3	Module homomorphisms and isomorphisms	303
14.4	Linear independence and bases	306
14.5	Vector spaces and dimension	309
<b>15</b>	<b>Matrices</b>	<b>316</b>
15.1	Basic definitions and properties	316
15.2	Matrices and linear maps	320
15.3	The inverse of a matrix	323
15.4	Gaussian elimination	324
15.5	Applications of Gaussian elimination	328

15.6	Notes	334
<b>16</b>	<b>Subexponential-time discrete logarithms and factoring</b>	<b>336</b>
16.1	Smooth numbers	336
16.2	An algorithm for discrete logarithms	337
16.3	An algorithm for factoring integers	344
16.4	Practical improvements	352
16.5	Notes	356
<b>17</b>	<b>More rings</b>	<b>359</b>
17.1	Algebras	359
17.2	The field of fractions of an integral domain	363
17.3	Unique factorization of polynomials	366
17.4	Polynomial congruences	371
17.5	Polynomial quotient algebras	374
17.6	General properties of extension fields	376
17.7	Formal power series and Laurent series	378
17.8	Unique factorization domains (*)	383
17.9	Notes	397
<b>18</b>	<b>Polynomial arithmetic and applications</b>	<b>398</b>
18.1	Basic arithmetic	398
18.2	Computing minimal polynomials in $F[X]/(f)$ (I)	401
18.3	Euclid's algorithm	402
18.4	Computing modular inverses and Chinese remaindering	405
18.5	Rational function reconstruction and applications	410
18.6	Faster polynomial arithmetic (*)	415
18.7	Notes	421
<b>19</b>	<b>Linearly generated sequences and applications</b>	<b>423</b>
19.1	Basic definitions and properties	423
19.2	Computing minimal polynomials: a special case	428
19.3	Computing minimal polynomials: a more general case	429
19.4	Solving sparse linear systems	435
19.5	Computing minimal polynomials in $F[X]/(f)$ (II)	438
19.6	The algebra of linear transformations (*)	440
19.7	Notes	447
<b>20</b>	<b>Finite fields</b>	<b>448</b>
20.1	Preliminaries	448
20.2	The existence of finite fields	450
20.3	The subfield structure and uniqueness of finite fields	454
20.4	Conjugates, norms and traces	456

<b>21</b>	<b>Algorithms for finite fields</b>	462
21.1	Testing and constructing irreducible polynomials	462
21.2	Computing minimal polynomials in $F[\mathbf{x}]/(f)$ (III)	465
21.3	Factoring polynomials: the Cantor–Zassenhaus algorithm	467
21.4	Factoring polynomials: Berlekamp’s algorithm	475
21.5	Deterministic factorization algorithms (*)	483
21.6	Faster square-free decomposition (*)	485
21.7	Notes	487
<b>22</b>	<b>Deterministic primality testing</b>	489
22.1	The basic idea	489
22.2	The algorithm and its analysis	490
22.3	Notes	500
	<i>Appendix: Some useful facts</i>	501
	<i>Bibliography</i>	504
	<i>Index of notation</i>	510
	<i>Index</i>	512



# 1

## Basic properties of the integers

This chapter discusses some of the basic properties of the integers, including the notions of divisibility and primality, unique factorization into primes, greatest common divisors, and least common multiples.

### 1.1 Divisibility and primality

Consider the integers  $\mathbb{Z} := \{\dots, -2, -1, 0, 1, 2, \dots\}$ . For  $a, b \in \mathbb{Z}$ , we say that  $b$  **divides**  $a$ , or alternatively, that  $a$  is **divisible by**  $b$ , if there exists  $c \in \mathbb{Z}$  such that  $a = bc$ . If  $b$  divides  $a$ , then  $b$  is called a **divisor** of  $a$ , and we write  $b \mid a$ . If  $b$  does not divide  $a$ , then we write  $b \nmid a$ .

We first state some simple facts:

**Theorem 1.1.** *For all  $a, b, c \in \mathbb{Z}$ , we have*

- (i)  $a \mid a$ ,  $1 \mid a$ , and  $a \mid 0$ ;
- (ii)  $0 \mid a$  if and only if  $a = 0$ ;
- (iii)  $a \mid b$  and  $a \mid c$  implies  $a \mid (b + c)$ ;
- (iv)  $a \mid b$  implies  $a \mid -b$ ;
- (v)  $a \mid b$  and  $b \mid c$  implies  $a \mid c$ .

*Proof.* These properties can be easily derived from the definition using elementary facts about the integers. For example,  $a \mid a$  because we can write  $a = a \cdot 1$ ;  $1 \mid a$  because we can write  $a = 1 \cdot a$ ;  $a \mid 0$  because we can write  $0 = a \cdot 0$ . We leave it as an easy exercise for the reader to verify the remaining properties.  $\square$

Another simple but useful fact is the following:

**Theorem 1.2.** *For all  $a, b \in \mathbb{Z}$ , we have  $a \mid b$  and  $b \mid a$  if and only if  $a = \pm b$ .*