*A course in computational algebraic number theory*, by Henri Cohen. Springer-Verlag, New York, 1993, xxi+534 pp., $49.00. ISBN 3-540-55640-0

Computational number theory has been in the news recently. On March 23, just before this review was finished, the *New York Times* printed an article on the anticipated completion of a project to factor RSA129, a 129-digit integer proposed as a challenge over fifteen years ago in connection with a cryptographic method for which cracking the code is equivalent to achieving such a factorization. The project may well be completed by the time this review is published. The calculation uses a sophisticated quadratic sieve method of Pomerance [P] (or see the final chapter of the book under review) that reduces the number of arithmetic operations required from the more than $10^{50}$ needed to check all possible prime factors by trial division to a mere $10^{17}$ or so.

This illustrates how computers have changed the nature of number-theoretic computation by making a whole new range of problems feasible and changing the criteria for judging the usefulness of algorithms. There is not much point in devising a method that reduces the work involved by even a factor like $10^{33}$ if it merely replaces one utterly impractical hand computation by another. Less dramatically, a method of cutting the cost of something like testing irreducibility of a polynomial over the rationals of moderate degree to a few million operations, which would be of little help in hand computation, allows a computer to check many cases in search of an example of some particular kind.

It is worth remarking that the increase in reliability has been as spectacular as the increase in speed and equally important. The combination of solid-state devices with internal redundancy and error checking makes machines capable not only of carrying out many billions of elementary operations in a reasonably small fraction of a human lifetime but, even more remarkably, of carrying them out with reasonably high probability of no machine error. (There remains ample opportunity for human error in programming, but producing correct programs for carrying out algorithms that are well understood is not much more difficult than producing detailed and rigorous proofs of theorems that are well understood.)

Computers have also influenced the subject by creating practical applications of number theory and algebra in areas such as cryptography, data compression, and error-correcting codes, leading to increased interest in efficient computation. The development of computer systems such as Reduce, Macsyma, Maple, Mathematica, Axiom, ... has also added impetus to the search for efficient algorithms in algebra and arithmetic, including some (for example, calculation of polynomial gcd's) that are highly relevant to algebraic number theory.

Of course, number-theoretic computation has a long history, and investigation of the efficiency of algorithms long antedates computers. Consider the Euclidean algorithm, which in various forms is an essential component of any computer algebra system. In Book 7 of *Euclid's Elements* it serves merely for proving that any two integers have a greatest common divisor, and its usefulness in calculation is not mentioned. Formal analysis of its properties as an algo-

rithm, however, dates back more than a century and a half to Binet's proof [Bi1] in 1841 that the number of division steps needed to calculate the gcd of two integers is less than 10/3 of the number of decimal digits in the smaller integer, when division with a least absolute remainder is used. A few years later, at the session of the Académie des Sciences on 28 October 1844, Lamé presented a note [L] in which (apparently unaware of Binet's work) he gave a similar result for division with a least positive remainder, with 5 in place of 10/3; at the November 4 session a week later, Binet presented a note [Bi2] mentioning his own previous work and giving a different derivation of Lamé's lemma on the rate of growth of the Fibonacci numbers.

These results of Binet and Lamé may be the first explicit demonstrations that a particular algorithm can be executed in polynomial time. An algorithm is said to be *polynomial-time* if the computation time it requires on a given input is bounded by a polynomial function of the size of the input, measured by, for example, the number of binary or decimal digits for integers or the degree plus the number of digits in all the coefficients in the case of polynomials. Lamé cites unnamed "traités d'Arithmétique" as satisfied with asserting that the number of steps in the Euclidean algorithm is bounded by half the smaller argument, which of course is exponential in the size of the input (and, as he points out, not even true for very small arguments).

The search for more efficient methods has led to discoveries that, while they would probably not have been found without the computational motivation, are interesting and sometimes elegant as mathematics in their own right. To illustrate this, we shall outline the development of methods for factoring polynomials over the rationals, something one obviously needs even to begin computing with algebraic numbers, since the most basic construction of an algebraic number field is as the ring of polynomials modulo an *irreducible* polynomial. (Advances in primality testing and factoring large integers involving sophisticated application of algebraic number theory have been even more striking, but that story is both harder to treat briefly and more widely known. The book under review has good accounts of both subjects.)

We focus here on how various ideas were brought together to attain the theoretical goal of a polynomial-time algorithm, without attempting to discuss questions of practical efficiency. For a thorough discussion of the latter, see Chapter 3 of the book under review, where it is noted that for problems within the range of practical computation today, the known polynomial-time algorithm appears to be less efficient in practice than other less elaborate methods.

We write $f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$ for the polynomial to be factored, assuming, to simplify the exposition, that it has integer coefficients and is monic (has leading coefficient 1) and we consider only monic factors. The whole endeavor depends fundamentally on Gauss's lemma, which implies that if $f$ is monic in $\mathbf{Z}[x]$, then any monic factor of it in $\mathbf{Q}[x]$ is actually in $\mathbf{Z}[x]$. The finite nature of the factorization problem follows almost immediately. First, one obtains a bound on the coefficients of a possible factor, say, $g(x) = x^k + b_{k-1}x^{k-1} + \cdots + b_0$. For a crude estimate, note that the roots of $g$ are roots of $f$; so if $M$ is any bound on the roots of $f$, such as $1 + |a_{n-1}| + \cdots + |a_0|$, then $|b_i| \leq \binom{k}{i}M^{k-i}$. (Better bounds are available. See [M] for an elementary proof of the less obvious inequalities $|b_i| \leq \binom{k}{i}\|f\|$, where $\|f\|^2 = 1 + |a_{n-1}|^2 + \cdots + |a_0|^2$.) The problem then reduces to testing the

finite number of $g$ with coefficients within these bounds as possible factors of $f$, but even for quite small $n$ the number of possibilities is too large for this to be a practical method of actually finding factors.

Nonetheless, we can use the complex roots to factor polynomials of moderate degree quite efficiently. There are good algorithms, for example [JT], for finding accurate approximations to the complex roots (and hence the irreducible real factors) of polynomials. Each subset of these approximate real factors can be checked to see if its product approximates some $g$ in $\mathbf{Z}[x]$ to within computational error; and if it does, then the exact computation can be done to check whether $g$ is a factor. Subsets of factors can often be rejected because the sum of the roots is not close enough to an integer; then the other coefficients in the product need not be computed, so the amount of computation per subset is usually small. Even so, the number of irreducible factors of $f$ over $\mathbf{R}$ is at least $n/2$ and the number of subsets grows exponentially with the degree of $f$, and this is not a polynomial-time algorithm.

Alternatively, one can restrict possible factors of $f$ by modular arguments. Kronecker's method uses the simple observation that if $f(x) = g(x)h(x)$ in $\mathbf{Z}[x]$, then $g(m)$ divides $f(m)$ for any integer $m$, so the values of $g(m)$ are restricted to a finite set. Since a polynomial of degree $k$ is determined by its value at $k + 1$ points, this leaves only a finite number of possible factors to check. Except for factors of very low degree, however, the large number of possibilities again makes the method impractical for actually finding them.

Working modulo primes and prime powers is much more effective. We describe later how to factor $f$ modulo a prime $p$. We may assume that $f$ has a nonzero discriminant, since otherwise a nontrivial common factor of $f$ and its derivative can be found and removed. Then if $p$ does not divide the discriminant, $f$ is the product of distinct irreducible factors $\bmod p$. As Zassenhaus showed [Z1, Z2], these can be efficiently refined to yield a factorization $\bmod p^k$, which can be viewed as an approximation to the factorization of $f$ into irreducibles in $\mathbf{Q}_p[x]$, where $\mathbf{Q}_p$ is the $p$-adic completion of $\mathbf{Q}$. Under our simplifying assumption that $f$ is monic and in $\mathbf{Z}[x]$, these factors will be monic with coefficients in the $p$-adic integers $\mathbf{Z}_p$. If the approximation is good enough (i.e., if $k$ is sufficiently large), we can proceed exactly as we did above with the factorization of $f$ in $\mathbf{R}[x]$, checking products of subsets of the approximate $p$-adic factors to see whether they are close (i.e., congruent $\bmod p^k$) to polynomials in $\mathbf{Z}[x]$ with coefficients small enough for them to be possible factors of $f$. The advantage of using $\mathbf{Q}_p$ instead of $\mathbf{R}$ is that the number of factors of $f$ over $\mathbf{Q}_p$ is typically of the order of $\log n$, so for "most" $f$ the number of subsets of factors is polynomial in $n$ and the algorithm requires only polynomial time on the average. (Collins has given a precise formulation and proof of this statement in [C].) This does not apply in all cases, however, because there are $f$ with many irreducible factors over $\mathbf{Q}_p$ for every $p$. For a simple family of examples, take $f$ to have roots equal to the $2^k$ quantities obtained by taking all possible choices of signs in $\pm\sqrt{q_1} \pm \cdots \pm \sqrt{q_k}$ where the $q_i$ are distinct primes of the form $4m + 1$. Then $f$ is irreducible over $\mathbf{Q}$, but all its roots lie in $\mathbf{Q}_p(\sqrt{p})$ if $p$ is one of the $q_i$ and in the unramified quadratic extension of $\mathbf{Q}_p$ otherwise; so it has no irreducible factor of degree higher than 2 over any $\mathbf{Q}_p$.

A polynomial-time algorithm was eventually obtained as a result of progress on a seemingly unrelated question. A classical problem in the geometry of numbers, going back to Minkowski, is to find nonzero vectors of minimal length in the lattice of integral linear combinations of a given basis in $\mathbf{R}^n$. The problem is difficult, and there is reason to believe that it has no polynomial-time solution. In [LLL] A. K. Lenstra, H. W. Lenstra, and L. Lovász exhibited a polynomial-time algorithm, now universally known as the LLL algorithm, for finding short vectors in such a lattice. (The vectors it finds are guaranteed only to be shorter than $2^{n/2}$ times the length of a minimal vector but are usually much shorter in practice.) They then derived from it a polynomial-time algorithm for polynomial factorization, as follows. Let $g$ be a fixed irreducible factor of $f$ in $\mathbf{Z}_p[x]$. If $f$ factors in $\mathbf{Z}[x]$, then one of its proper factors must be congruent mod $p^k$ to a multiple of $g$ for any $k$. For a fixed $k$, the polynomials in $\mathbf{Z}[x]$ congruent to such multiples form a lattice in the $n$-dimensional real vector space $\mathbf{R}[x]/f(x)$; it is shown in [LLL] that for suitably large $k$, the LLL algorithm will either find a polynomial in this lattice that has a nontrivial common factor with $f$ or prove $f$ irreducible. It is shown that when everything is taken into account, including the size required for $p^k$ and the cost of doing arithmetic on multiple precision integers of the size that may occur during the computation, the time required is $O(n^{12} + n^9(\log \|f\|)^3)$, where $\|f\|$ is defined as above.

Now we come to the problem of finding the factors of $f$ mod $p$ efficiently. All the known algorithms for doing so seek polynomials $g$ such that $\gcd(f, g)$ is a nontrivial factor of $f$. Only the remainder of $g$ mod $f$ is relevant, so we work in the algebra $A_p(f)$ defined as $\mathbf{F}_p[x]/f(x)$, where $\mathbf{F}_p$ is the $p$-element field. Note that the cost of arithmetic operations in $\mathbf{F}_p$ is polynomial in $\log p$, and of operations on polynomials of degree $n$, including operations in $A_p(f)$ and calculation of gcd's by the Euclidean algorithm, is polynomial in $n$ and $\log p$. Relatively small primes suffice for the application to factoring polynomials over the rationals; but for other problems such as factoring ideals in rings of algebraic integers, large primes need to be considered too, so we take the size of $p$ into account.

If $f$ is the product of distinct irreducible factors $h_1, \ldots, h_k$, then the map $\rho$ taking $g$ in $A_p(f)$ to $\rho(g) = (\rho_1(g), \ldots, \rho_k(g))$, where $\rho_i(g)$ is the remainder of $g$ mod $h_i$, is an isomorphism from $A_p(f)$ to the direct sum of the fields $A_p(h_i)$. Then $\gcd(g, f)$ is a nontrivial proper divisor of $f$ if and only if $g$ is a proper zero-divisor in $A_p(f)$, which happens if and only if some but not all of the $\rho_i(g)$ are $0$. Berlekamp [Be1, Be2] showed how to exploit this relation, even though we cannot compute $\rho_i$ without knowing $h_i$. Each $A_p(h_i)$ contains a unique copy of $\mathbf{F}_p$, consisting of the roots in $A_p(h_i)$ of $x^p - x$, from which we see that the $g$ in $A_p(f)$ such that $g^p - g = 0$ form a subalgebra $B$ isomorphic to the direct sum of $k$ copies of $\mathbf{F}_p$. Taking the $p$th power is a linear operation in characteristic $p$, and computing its matrix with respect to the basis $1, x, \ldots, x^{n-1}$ for $A_p(f)$ over $\mathbf{F}_p$ amounts to computing $x^{kp}$ in $A_p(f)$ for $k = 1, 2, \ldots, n-1$. Even for large $p$, this is polynomial in $n$ and $\log p$ because for any associative multiplication, $m$th powers can be computed in $O(\log m)$ multiplications, by using successive squaring to compute $y_i = x^{2^i}$ for $i = 1, 2, \ldots, [\log_2 m]$ and multiplying together the $y_i$ corresponding to the

nonzero bits in the binary expansion of $m$. Then it is a matter of linear algebra to find a basis for $B$ in $A_p(f)$. Its dimension is $k$, the number of factors. If $k = 1$, $B$ contains only the constant polynomials and $f$ is irreducible.

Otherwise, for any nonconstant $g$ in $B$ the $\rho_i(g)$ are not all the same, so $g - \rho_i(g)$ is a proper zero-divisor in $A_p(f)$ and $\gcd(f, g - \rho_i(g))$ is a proper nontrivial factor of $f$, for any $i$. Of course, we do not know the $\rho_i(g)$, but because $g$ is in $B$, we know they are all in $\mathbf{F}_p$. If $p$ is very small, we can simply test $\gcd(f, g - s)$ for $s = 0, 1, \ldots$ until a factor is found. Otherwise it is a matter of linear algebra to compute the minimal polynomial $\varphi$ satisfied by $g$ as an element of $A_p(f)$; $\varphi$ is a product of linear factors over $\mathbf{F}_p$, and its roots are the distinct values among the $\rho_i(g)$.

The general factorization problem is thus reduced to the special case of factoring a product of linear factors. Every element of $\mathbf{F}_p$ is a root of $x^p - x$, which is equal to $(x-a)^p - (x-a) = (x-a)((x-a)^{(p-1)/2}+1)((x-a)^{(p-1)/2}-1)$ for any $a$ in $\mathbf{F}_p$. (We are assuming $p > 2$.) Unless all or none of the factors of $\varphi$ divide $((x - a)^{(p-1)/2} - 1)$, we get a nontrivial factor of $\varphi$ from $\gcd(\varphi, ((x - a)^{(p-1)/2} - 1))$. Using the algorithm mentioned above for computing powers, calculating such a gcd takes time polynomial in $\log p$ and the degree of $\varphi$. If $a$ is picked at random, the chance of breaking $\varphi$ into two factors is roughly $1/2$ if $k = 2$ and larger if $k > 2$; and it is not hard to show that the expected number of $a$'s needed for a complete factorization is $O(\log p)$, so in a probabilistic sense we have a polynomial-time algorithm. In practice it works well. The question of whether, for instance, simply taking $a = 0, 1, 2, \ldots$ until $\varphi$ is factored always succeeds in $O(\log p)$ steps is related to the question of whether the least quadratic nonresidue mod $p$ is $O(\log p)$ and does not seem likely to be resolved soon. (Another approach to the general factorization problem by Cantor and Zassenhaus [CZ] uses a variant of the idea of this paragraph. For $g$ chosen at random in $A_p(f)$, $\gcd(f, g^{(q-1)/2} - 1)$ where $q = p^k$ often contains some but not all of the irreducible factors of $f$ that have degrees dividing $k$, and this leads to a method that also works well in practice.)

The simplest way to give an idea of the scope of the book under review is to list the main topics by chapter. Chapter 1 gives a concise treatment of fundamental algorithms such as the powering method mentioned above and the Euclidean algorithm and touches lightly on computation with large integers. The second chapter deals with linear algebra over a field and over $\mathbf{Z}$ and includes a thorough treatment of the LLL algorithm. Chapter 3 covers factorization of polynomials over finite fields, over the rationals, and over algebraic number fields. (Not much is said about polynomials in more than one variable, a topic that belongs more to algebraic geometry than to algebraic number theory.)

The next three chapters deal with algebraic number theory. Chapter 4 covers the basic questions of how to represent and compute with algebraic numbers, modules, and ideals and introduces the problems of finding integral bases, ideal factorizations, units, and class numbers. It also discusses the problem of determining whether one algebraic number field is isomorphic to another (or, more generally, to a subfield of it) and presents an algorithm of the author and F. Diaz y Diaz for finding "simple" polynomials defining a given number field which, although it does not provide a canonical defining polynomial, does well enough

to be very useful. Chapter 5, the longest in the book, is devoted to quadratic fields and gives a complete picture of the extensive development that has led to methods making the computation of class groups and fundamental units feasible for quadratic fields with discriminants of the order of magnitude of $10^{25}$ or so. Chapter 6 gives state-of-the-art algorithms for finding integral bases, ideal factorizations, class groups, and units in general algebraic number fields, as well as for the determination of Galois groups for fields of degree up to seven.

The first six chapters constitute a thorough course in basic computational algebraic number theory. The remainder of the book is more specialized. Chapter 7 presents relevant aspects of elliptic curves, with algorithms, including computation of minimal models over $\mathbf{Q}$ by Tate's algorithm. Chapter 8 surveys pre-1980 methods for primality testing and factoring integers, while Chapters 9 and 10 describe the more recent advances in primality testing and factoring, respectively.

The tone of the book is refreshingly practical throughout, as might be expected from an author who is the leader of a group that has implemented a powerful package of programs for algebraic number theory (the freely distributed pari/gp system; for a review see the AMS *Notices* for October 1991). Algorithms are presented in the style of Knuth [K] as sequences of numbered steps with operations described in mathematical terms but with assignments to variables and the flow of control given in explicit detail. A mathematician can read an algorithm, verify the correctness of the accompanying demonstration that it computes what is claimed, and (with some minimal understanding of what goes on in a computer) estimate its memory requirements and running time. A programmer need only understand the computer representation of the relevant data to be able to translate the algorithm into a programming language and produce a working implementation.

The book is recommended to anyone who wants to know about the theory or practice of computation in algebraic number theory, in general or for solving particular problems. The chances are good of finding an algorithm that does the job about as well as anyone currently knows how, along with a clear exposition of the theory behind it. References and comments provide an annotated guide to the literature for those who want further details.

The book is about as up-to-date as one can get, and the author has already put together a supplement containing some additions and revisions along with corrections for the very few misprints and minor errors that managed to escape detection in the first printing. It is available by anonymous ftp from megrez.ceremab.u-bordeaux.fr in the directory pub/cohenbook. (Your local Internet expert can get it for you if you are not familiar with the process.)

By presenting so much material so clearly and in such an accessible style, Professor Cohen has done a great service to both beginners and experts in the field.

## REFERENCES

[Be1] E. R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Tech. J. **46** (1967), 1853–1859.

[Be2] ———, *Factoring polynomials over large finite fields*, Math. Comp. **24** (1970), 713–735.

[Bi1]   J. Binet, *Recherches sur la théorie des nombres entiers et sur la résolution de l'équation indeterminée du premier degré qui n'admet que des solutions entières*, J. Math. Pures Appl. (1) **6** (1841), 449–494.

[Bi2]   _____, *Note sur le nombre des divisions à effectuer pour obtenir le plus grand diviseur commun de deux nombres entiers; suivie d'une remarque sur une classe de séries récurrentes*, Comptes Rendus **19** (1844), 937–941.

[C]     G. Collins, *Factoring univariate integral polynomials in polynomial average time*, Symbolic and Algebraic Computation, Lecture Notes in Comput. Sci., vol. 72, Springer, New York, 1969, pp. 317–329.

[CZ]    D. Cantor and H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. Comp. **36** (1981), 587–592.

[JT]    M. A. Jenkins and J. F. Traub, *A three-stage variable shift iteration for polynomial zeroes and its relation to generalized Rayleigh iteration*, Numer. Math. **14** (1970), 252–263.

[K]     D. E. Knuth, *The art of computer programming*, Vol. 2, Seminumerical algorithms, 2nd ed., Addison-Wesley, Reading, MA, 1981.

[L]     G. Lamé, *Note sur la limite du nombre des divisions dans la recherche du plus grand diviseur entre deux nombres entiers*, Comptes Rendus **19** (1844), 867–870.

[LLL]   A. K. Lenstra, H. W. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), 515–534.

[M]     M. Mignotte, *An inequality about factors of polynomials*, Math. Comp. **28** (1974), 1153–1157.

[P]     C. Pomerance, *Analysis and comparison of some integer factoring algorithms*, Computational Methods in Number Theory, Part I (H. W. Lenstra and R. Tijdeman, eds.), Math. Centre Tracts 154/155, Math. Centrum, Amsterdam, 1982.

[Z1]    H. Zassenhaus, *On Hensel factorization*. I, J. Number Theory **1** (1969), 291–311.

[Z2]    _____, *A remark on the Hensel factorization method*, Math. Comp. **32** (1978), 287–292.

HALE TROTTER
PRINCETON UNIVERSITY
*E-mail address*: hft@math.princeton.edu

*Nonlinear potential theory of degenerate elliptic equations*, by J. Heinonen, T. Kilpeläinen, and O. Martio. Oxford University Press, London, 1993, v+363 pp., $70.00. ISBN 0-19-853669-0

This is a remarkable book which covers a period of three decades starting with Serrin's 1964 fundamental paper [29] on the Harnack inequality and Hölder continuity of solutions to quasilinear equations with nonquadratic growth in the gradient. In that paper Serrin was able to extend the celebrated theorem of De Giorgi–Nash–Moser to a general class of nonlinear second-order equations. The prototype of such equations arises as the Euler equation of the energy functional

$$(1) \qquad D_p(u;\Omega) = \int_\Omega |\mathscr{D}u|^p \, dx, \qquad 1 < p < \infty.$$