

BOOK REVIEWS

BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 18, Number 1, January 1988
©1988 American Mathematical Society
0273-0979/88 \$1.00 + \$.25 per page

Prime numbers and computer methods for factorization, by Hans Riesel. Progress in Mathematics, vol. 57, Birkhäuser, Boston, Basel and Stuttgart, 1985, xvi + 464 pp., \$44.95. ISBN 0-8176-3291-3

While as old as the hills, the Fundamental Theorem of Arithmetic also embodies two aspects of modern mathematics. The Fundamental Theorem states that every composite number is uniquely decomposable into primes. With such a theoretically satisfying description of the multiplicative structure of the integers as our template, we have sought out and often found analogous structure in almost every corner of mathematics. Endowing seemingly chaotic material with structure and discovering new truths as a result is unquestionably a major function of mathematics. But the Fundamental Theorem also represents another trend in mathematics, a trend that was out of favor in the middle decades of this century. Computation, algorithms, and issues of effectivity, mainstays of the last century, have enjoyed reawakened interest in this, the age of computers. The Fundamental Theorem of Arithmetic, in the active form of actually distinguishing between primes and composites and factoring the latter into primes, has played a fundamental and benchmark role in this reawakening.

It is perhaps not so widely known, but the two problems just mentioned, distinguishing primes from composites and factoring composites, are quite different. This seems paradoxical since if a factoring algorithm is applied to a prime input p , the failure of the algorithm to properly factor p should then give us the information that p is after all prime. In fact, the best known factoring algorithm of all, trial division, when exhaustively applied to every trial divisor up to \sqrt{p} , is also the best known primality test.

To see clearly the difference between the two problems, one need look no farther than Fermat's "little theorem": if p is prime, then $a^p \equiv a \pmod{p}$. Since it is a simple procedure to compute the residue $a^n \pmod{n}$ when given a and n (this can be done in $O(\log n)$ arithmetic steps in the ring \mathbf{Z}/n by the repeated squaring method), if the residue is not $a \pmod{n}$, then n has been revealed as composite. Yet we are no closer to factoring n .

Thus the computational side of the Fundamental Theorem of Arithmetic falls into two separate problems. The first, generally called primality testing, involves deciding if an input integer is prime or composite. Some algorithms under this general banner will prove all or most composite inputs are composite, but not say anything about prime inputs (using Fermat's little theorem

as described above is such a “compositeness” test). Other algorithms find proofs of primality for prime inputs (true primality tests) while others might do both. The other computational problem associated with the Fundamental Theorem is, of course, factoring composite integers.

How can a compositeness test prove all composite inputs are composite but not say anything about prime inputs? Such a feat is possible through the use of randomness. For example, it has been known since Euler that if p is an odd prime and a is not divisible by p , then $a^{(p-1)/2} \equiv (a/p) \pmod{p}$ where (a/p) , the Legendre symbol, is 1 or -1 depending on whether a is congruent to a square mod p or not. The symbol (a/p) is computable by viewing it as a Jacobi symbol (which does not require the “denominator” to be prime) and using the law of quadratic reciprocity for Jacobi symbols with a procedure similar to Euclid’s well-known algorithm for computing greatest common divisors. In fact, both sides of the Euler congruence can be computed for a pair a, n where a is not divisible by n and n is odd. Thus an odd n can be proved composite (as with Fermat’s little theorem) by exhibiting some a with $a \not\equiv 0 \pmod{n}$ and $a^{(n-1)/2} \not\equiv (a/n) \pmod{n}$. It can be shown (Lehmer [6], Solovay-Strassen [16]) that if n is odd and composite, then at least half of the integers a in the interval $[1, n]$ would work via Euler’s congruence to prove n composite. Thus in [16], the following *random* compositeness test is described. On input of an odd integer n , choose random integers a in $[1, n]$ and check the Euler congruence for the pair a, n . If n is composite, we expect to prove n composite after only finitely many trials. But if the input n is prime, the test proves nothing (unless one actually performs the test for more than $n/2$ different values of a).

This algorithm and a similar one by Rabin [13] show that compositeness testing is in the complexity class R (random polynomial time). While the idea of randomness was known before as a useful tool in algorithms, this test seemed to grab people’s imaginations and really opened the door for a flood of random algorithms in many areas.

True primality testing (proving prime inputs are prime) is harder, but here too there have been some important advances in recent years. The algorithm in [1] gets more information out of the Euler congruence than either pass or fail and finds a way to glue this information together with similar tests that involve the higher power reciprocity laws. This test is deterministic and in time $O((\log n)^{c \log \log \log n})$ will determine if the input n is prime or composite. It stands as the fastest known deterministic primality test (and compositeness test). A random version of this test [2] is computer practical and has been used to routinely prove numbers prime with more than 200 decimal digits.

The most recent developments in primality testing involve the theory of elliptic curves over finite fields. Using Hasse’s theorem, which gives an interval in which the order of an elliptic curve group over a finite field must fall, and an algorithm of Schoof [15] for actually computing this order exactly, Goldwasser and Kilian [3] found a random true primality test that is expected to prove most prime numbers prime in polynomial time. The possible exceptional set is probably empty—it could be proved empty if we had stronger tools from analytic number theory on the distribution of primes in short intervals. Very

recently, Adleman and Huang announced a complicated way of getting around this exceptional set by using arithmetic on higher-dimensional abelian varieties. If their algorithm holds up it will show true primality testing is also in the complexity class R . Although the above tests are not computer practical, Atkin has recently found an elliptic curve primality test that is and that even beats the times found in [2].

Although it still remains to be seen if there is a deterministic polynomial time primality test, the subject of primality testing is in a far more satisfactory state than that of factoring, which is, relatively speaking, still in the dark ages. This disparity between the relative ease of primality testing and difficulty of factoring has been made the basis of the “one-way function” in the RSA public key cryptosystem.

The fastest rigorous deterministic factoring algorithm that we know appears in two similar forms [9, 17] and will produce a nontrivial factorization of a composite number n in time about $n^{1/4}$. The fastest rigorous random factoring algorithm [12] takes expected time about $L(n)^{\sqrt{2}}$ where $L(n) = \exp(\sqrt{\log n \log \log n})$. (This method is based on ideas of Dixon, Lenstra, and Wiedemann.)

Since the output of a factoring algorithm can easily be checked, it may also profitably employ heuristics. In fact, all practical factoring algorithms beyond the most basic ones (such as trial division) use heuristic methods. The two most practical algorithms for factoring both have a heuristic worst-case running time of about $L(n)$. These are the quadratic sieve [10, 11] and the elliptic curve method [8]. The quadratic sieve, however, works better than the elliptic curve method when n is the product of two roughly equal primes (the numbers of cryptographic interest) while the elliptic curve method works better on all other composite numbers. The quadratic sieve has factored numbers in the 80s of decimal digits while the elliptic curve method has factored numbers whose least prime has about 30 decimal digits. However, these calculations are far from routine, requiring some heavy-duty computing.

Because of the fundamental role played by the prime numbers, it is natural to consider their distribution among the integers. Probably the most important single theorem in number theory, the Prime Number Theorem gives an approximate formula for $\pi(x)$, the number of primes up to x . It states that $\pi(x) = (1 + o(1))x/\log x$. The efforts to prove this elegant equation during the last century played a large role in the development of complex analysis.

As with the Fundamental Theorem of Arithmetic, the Prime Number Theorem also has its computational side. In one direction one can try to prove inequalities for $\pi(x)$ valid in explicit ranges. A very satisfying result of this type [14] is that $\pi(x) > x/\log x$ for $x \geq 17$. The computations that go into a result such as this involve the so-called explicit formula which relates $\pi(x)$ to the zeros of Riemann’s zeta function.

Another kind of computational problem is to compute $\pi(x)$ exactly for a particular x . The sieve of Eratosthenes can be used in this regard in time $O(x \log \log x)$ and space $O(x)$. Various clever combinatorial counting arguments have been applied over the years to lower the time and space complexity. A recent algorithm of Lagarias and Odlyzko [4] uses analytic methods and

computes $\pi(x)$ in time $O(x^{3/5+\epsilon})$ and space $O(x^\epsilon)$, but is not computer practical. Another method by these two and Miller [5] is somewhat slower asymptotically but has been used for the phenomenal calculation $\pi(4 \cdot 10^{16}) = 1,075,292,778,753,150$.

As the first major book on these fascinating subjects in a long time, the book under review is quite welcome. The author is a practical and long time artisan of the myriad techniques involved with large scale integer problems on a computer. The book contains the wealth of a lifetime of experience and is brimming with tricks of the trade for those who themselves want to begin crunching their own big numbers.

In mild criticism, the book is uneven in places, going into obscure detail on points that are not too interesting, while skimming over other points that cry out for development. In any event, there are many references provided, so the reader with piqued curiosity will not be left high and dry. Riesel has been a bit skimpy on the theoretical development of the subject. The few complexity analyses of algorithms that are presented are not satisfying and seem to have been added as an afterthought. It seems odd that many of the intriguing unsolved problems connected with the subject, such as, are there infinitely many Carmichael numbers? are not even mentioned.

The level of Riesel's book makes it quite accessible to non-experts. It is an excellent choice for someone who actually wants to do elementary number theory on a computer. There are very few books like this around where the author's unabashed love of computation comes shining through so clearly.

With any book written on a fast-changing subject, there are going to be late-breaking developments that are missing. Here, the book appeared just as Lenstra and others were demonstrating the wonderful connection to the theory of elliptic curves over finite fields. An up-to-date survey [7] of mostly the theoretical side of factorization and primality algorithms will appear soon.

REFERENCES

1. L. M. Adleman, C. Pomerance and R. S. Rumely, *On distinguishing prime numbers from composite numbers*, Ann. of Math. **117** (1983), 173–206.
2. H. Cohen and A. K. Lenstra, *Implementation of a new primality test*, Math. Comp. **48** (1987), 103–121.
3. S. Goldwasser and J. Kilian, *Almost all primes can be quickly certified*, Proc. 18th Annual ACM Symp. on Theory of Computing (1986), 316–329.
4. J. C. Lagarias and A. M. Odlyzko, *Computing $\pi(x)$: an analytic method*, J. Algorithms **8** (1987), 173–191.
5. J. C. Lagarias, V. S. Miller and A. M. Odlyzko, *Computing $\pi(x)$: the Meissel-Lehmer method*, Math. Comp. **44** (1985), 537–560.
6. D. H. Lehmer, *Strong Carmichael numbers*, J. Austral. Math. Soc. Ser. A **21** (1976), 508–510.
7. A. K. Lenstra and H. W. Lenstra, Jr., *Algorithms in number theory*, Handbook of Theoretical Computer Science (to appear).
8. H. W. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math. (to appear).
9. J. M. Pollard, *Theorems on factorization and primality testing*, Proc. Cambridge Philos. Soc. **76** (1974), 521–528.
10. C. Pomerance, *Analysis and comparison of some integer factoring algorithms*, Computational Methods in Number Theory (H. W. Lenstra, Jr. and R. Tijdeman, eds.), Math. Centre Tracts 154/155, Mathematisch Centrum, Amsterdam, 1982, pp. 89–139.

