ALGORITHMS FOR FINITE FIELDS

H. W. Lenstra, Jr.*

In this paper, we survey the complexity status of some fundamental algorithmic problems concerning finite fields. In particular, we consider the following two questions: given a prime number p and a positive integer n, construct explicitly a field that is of degree n over the prime field of p elements; and given two such fields, construct an explicit field isomorphism between them. For both problems there exist good probabilistic algorithms. The situation is more complicated if deterministic algorithms are required.

1. Introduction.

Every finite field has cardinality p^n for some prime number p and some positive integer n. Conversely, if p is a prime number and n a positive integer, then there exists a field of cardinality p^n, and any two fields of cardinality p^n are isomorphic. These results are due to E. H. Moore (1893) [15]. In this paper, we discuss the complexity aspects of two algorithmic problems that are suggested by this theorem, and of two related problems.

Constructing finite fields. We say that a finite field is *explicitly given* if, for some basis of the field over its prime field, we know the product of any two basis elements, expressed in the same basis. Let, more precisely, p be a prime number and n a positive integer. Then by *explicit data* for a finite field of cardinality p^n we mean a system of n^3 elements $(a_{ijk})_{i,j,k=1}^n$ of the prime field $\mathbf{F}_p = \mathbf{Z}/p\mathbf{Z}$, such that $\mathbf{F}_p^n$ becomes a field with the ordinary addition and multiplication by elements of $\mathbf{F}_p$, and the multiplication determined by

$$e_i e_j = \sum_{k=1}^{n} a_{ijk} e_k,$$

where $e_1, e_2, \ldots, e_n$ denotes the standard basis of $\mathbf{F}_p^n$ over $\mathbf{F}_p$.

The problem of constructing finite fields is the following: given a prime number p and a positive integer n, find explicit data for a finite field of cardinality p^n. The elements of $\mathbf{F}_p$ are to be represented in the conventional way as integers modulo p. The current complexity status of this problem is discussed in Section 2.

Finding isomorphisms between finite fields. Given a prime number p, a positive integer n, and two sets of explicit data $(a_{ijk})_{i,j,k=1}^n$, $(a'_{ijk})_{i,j,k=1}^n$ for finite fields of cardinality p^n, find an isomorphism between these fields. The isomorphism is to be represented by means of its matrix on the given bases of the fields over the prime

* The author was supported by NSF contract DMS 87-06176.

field; more precisely, we ask for an invertible $n \times n$ matrix $(b_{ij})_{i,j=1}^{n}$ over $\mathbf{F}_p$ with the property that

$$\sum_{k=1}^{n} a_{ijk} b_{km} = \sum_{k,l=1}^{n} b_{ik} b_{jl} a'_{klm}$$

for all i, j, $m = 1, 2, \ldots, n$. This problem is discussed in Section 3.

Irreducibility testing. Given explicit data for a finite field E, and a non-zero polynomial $f \in E[X]$, the problem is to decide whether f is irreducible in $E[X]$. This problem is the analogue of the *primality testing* problem, with the ring of integers $\mathbf{Z}$ replaced by the ring $E[X]$. We refer to Section 4 for a brief discussion of the results that are known.

Factoring polynomials over finite fields. Given explicit data for a finite field E, and a non-zero polynomial $f \in E[X]$, the problem is to determine the decomposition of f into irreducible factors in $E[X]$. This is the analogue of the problem of factoring integers, with $\mathbf{Z}$ replaced by $E[X]$. In Section 5, we survey the main results that have been obtained on this problem.

Our main interest will be in the *running times* of the algorithms that have been proposed for the above four problems. In particular, we are interested in whether these algorithms run in *polynomial time*, i. e. in time $(n+\log p)^{O(1)}$ for the first two problems and in time $(\deg f + \log \#E)^{O(1)}$ for the last two. We will not be concerned with the problem of obtaining good values for the O-constants, and they will be left unspecified.

To do justice to the results that have been obtained, it is appropriate to distinguish three types of algorithms. The first are the deterministic algorithms for which the running time bounds have been proved rigorously. From a mathematical point of view, these are the most satisfactory algorithms.

Secondly, we will consider deterministic algorithms for which the running time bounds have only been established on the assumption of the *generalized Riemann hypothesis*, which asserts that all non-trivial zeros of the zeta function of any algebraic number field (see [10], Chapter VIII) have real part $\frac{1}{2}$. In other contexts, such as primality testing, one also encounters algorithms of which the *correctness* depends on the truth of the generalized Riemann hypothesis. We shall not encounter such algorithms in this paper.

The third type of algorithms are the *probabilistic* ones. These algorithms employ a random number generator and the random numbers that are drawn influence the course of the algorithm. Both the outcome of the algorithm and its running time have therefore, for each given input, a *distribution.* When we speak about the running time of a probabilistic algorithm, we shall mean the time that is needed to let the algorithm terminate successfully with probability at least $\frac{1}{2}$; to increase this success probability one can perform the algorithm several times. The time that the random number generator may need is not counted. We shall not be concerned with the problem of minimizing the number of calls that are made to the random number generator (see [4]).

Again, in primality testing one also encounters algorithms that are probabilistic in a different sense; namely, not the running time of the algorithm but the correctness of the answer is subject to uncertainty. We shall not have occasion to deal with such algorithms in this paper: each algorithm that gives an answer gives provably a *correct* answer.

In practical circumstances, one is usually willing to use probabilistic algorithms instead of deterministic ones. As we shall see, there exists for each of our four problems a probabilistic algorithm that runs in polynomial time. This suggests that from a practical point of view all four problems may be considered to be well-solved, and this appears indeed to be the case.

Turning to deterministic algorithms, we shall see that there exist fully proved polynomial time algorithms for the second problem (*finding isomorphisms*) and the third problem (*irreducibility testing*). For the first problem (*constructing finite fields*) and the fourth problem (*factoring polynomials*) fully proved polynomial time algorithms are currently only known in special cases; e.g., the case that the characteristic p is fixed, or more generally the case that p is bounded by a fixed power of n or $\deg f$. If we accept the truth of the generalized Riemann hypothesis, then also for the first problem (*constructing finite fields*) there is a deterministic polynomial time algorithm. This is not the case for the fourth problem (*factoring polynomials*), although many special cases have been dealt with.

2. Constructing finite fields.

Let p be a prime number and n a positive integer. If we know an irreducible polynomial $f \in \mathbf{F}_p[X]$ of degree n, then explicit data for a field of cardinality p^n are readily calculated, since $\mathbf{F}_p[X]/f\mathbf{F}_p[X]$ is such a field. Conversely, given explicit data for a field of cardinality p^n, one can, in deterministic polynomial time, exhibit an element α in this field that has degree n over $\mathbf{F}_p$, and calculate its irreducible polynomial f over $\mathbf{F}_p$, which has degree n (see [12], Section 2). Thus the problem of constructing explicit data for a finite field of cardinality p^n is equivalent to the problem of constructing an irreducible polynomial of degree n in $\mathbf{F}_p[X]$.

For $n = 2$ and p odd, a clearly equivalent problem is to find, given p, an element a of $\mathbf{F}_p$ that is not a square. This can easily be done in polynomial time if a probabilistic algorithm is allowed: draw a at random, until one that satisfies $a^{(p-1)/2} = -1$ is found. The same applies if the generalized Riemann hypothesis is assumed: try all a up to $2(\log p)^2$ (see [3]). However, even for this special case, there is apparently at present no hope of finding a fully proved deterministic polynomial time algorithm.

The general case can be reduced to the case that n is prime in the following sense: if for each prime divisor r of n an irreducible polynomial of degree r in $\mathbf{F}_p[X]$ is given, then an irreducible polynomial of degree n can be constructed in deterministic polynomial time ([12], Theorem (1.1)).

The best fully proved deterministic algorithm for the problem of constructing finite fields is due to V. Shoup [22]. He proved that the problem can be reduced to the problem of factoring polynomials over finite fields. This is a "Turing reduction" in the sense that the construction of a single finite field requires the factorization of several polynomials, which are computed in the course of the algorithm. Shoup's reduction and the results of Section 5 lead to the running time bound $O\big(\sqrt{p} \cdot (n + \log p)^{O(1)}\big)$. In particular, there exists a fully proved deterministic polynomial time algorithm if p is fixed, e.g. $p = 2$, or if p is bounded by a fixed power of n.

If the generalized Riemann hypothesis is assumed, then the problem of constructing finite fields can be solved in polynomial time (see [1] and [7] (independently)). We briefly sketch the method of [1] in the case that n is prime, to which the general case can be reduced, as we just saw. To find an irreducible polynomial of prime degree n in $\mathbf{F}_p[X]$, one picks a small prime q with the properties

$$q \equiv 1 \bmod n, \qquad p^{(q-1)/n} \not\equiv 1 \bmod q;$$

the generalized Riemann hypothesis guarantees that the least such q is $(n + \log p)^{O(1)}$. Now let $H \subset \mathbf{F}_q^*$ be the subgroup of n-th powers and ζ_q a primitive q-th root of unity. Then the polynomial

$$f = \prod_{C \in \mathbf{F}_q^*/H} \Big(X - \sum_{x \in C} \zeta_q^x\Big)$$

lies in $\mathbf{Z}[X]$, and it can be proved that $(f \bmod p)$ is irreducible in $\mathbf{F}_p[X]$.

If probabilistic algorithms are allowed, the construction of finite fields is also possible in polynomial time. This follows, of course, from Shoup's result just mentioned, but it is easier to proceed as follows: pick $f \in \mathbf{F}_p[X]$, $\deg f = n$, at random, test f for irreducibility (see Section 4), and repeat until an irreducible f is found. This is efficient, because a random polynomial of degree n in $\mathbf{F}_p[X]$ is irreducible with probability $(1 + O(p^{-n/2}))/n$.

The problem of constructing finite fields has an interesting companion problem: given positive integers p and n with $p \geq 2$ and a system of n^3 elements $(a_{ijk})_{i,j,k=1}^n$ of $\mathbf{Z}/p\mathbf{Z}$, decide whether these form explicit data for a field of cardinality p^n. For $n = 1$ this problem is equivalent to the *primality testing* problem: given an integer $p \geq 2$, decide whether p is prime. For this problem no fully proved deterministic polynomial time algorithm is known. Using the techniques of [12, Section 2] one can show that primality testing is the *only* obstacle: there is a deterministic polynomial time algorithm that, given p, n, (a_{ijk}) as above, either proves that they do not form explicit data for a field of cardinality p^n, or proves that if p is prime they do.

3. Finding isomorphisms.

Although the problem of finding an isomorphism between two explicitly given finite fields of the same cardinality is from a theoretical point of view just as fundamental as the problem of constructing finite fields, I do not believe that the problem arises in many practical circumstances. If it ever would, one would probably solve it by means of the following probabilistic algorithm. Let E, E' be two explicitly given finite fields of cardinality p^n. As we mentioned in the previous section, one can find $\alpha \in E$ with $E = \mathbf{F}_p(\alpha)$ and determine the irreducible polynomial f of α over $\mathbf{F}_p$. Finding a field isomorphism $E \to E'$ is now equivalent to finding a zero of f in E'. Since finding a zero is equivalent to finding a linear factor, this problem can be solved by means of one of the algorithms discussed in Section 5.

The procedure just sketched, combined with the results of Section 5, shows that the problem of finding isomorphisms can be solved by means of a probabilistic algorithm in polynomial time. It was shown by S. A. Evdokimov [7] that it can be done by means of a deterministic polynomial time algorithm if the truth of the generalized

Riemann hypothesis is assumed. These two results were superseded recently, when it was proved that there is a fully proved deterministic polynomial time algorithm for finding isomorphisms between finite fields [12].

To illustrate the idea we consider the case $n = 2$, $p > 2$. First, let explicit data for a finite field E of cardinality p^2 be given. It is not difficult to construct an element $\alpha \in E$ with $E = \mathbf{F}_p(\alpha)$ and $\alpha^2 = a \in \mathbf{F}_p$. Then a is not a square in $\mathbf{F}_p$, so $a^{(p-1)/2} = -1$. Writing $p - 1 = 2^t v$ with v odd, and replacing α, a by α^v, a^v, we may assume that $a^{2^{t-1}} = -1$.

Now let another finite field E' of cardinality p^2 be explicitly given. In a similar way we can write $E' = \mathbf{F}_p(\beta)$, where $\beta^2 = b \in \mathbf{F}_p$ and $b^{2^{t-1}} = -1$. To find an isomorphism $E \to E'$ it suffices to find $c \in \mathbf{F}_p$ with $a = bc^2$, since then we can map α to βc.

The element c can be found by an iterative procedure that is due to A. Tonelli (1891) ([6], page 215). The iteration starts with $c = 1$. In each iteration step, one first determines the least non-negative integer i for which $a^{2^i} = (bc^2)^{2^i}$. If $i = 0$ then $a = bc^2$, and the algorithm terminates. If $i > 1$, then we replace c by $cb^{2^{t-i-1}}$ and iterate.

To prove that the algorithm is correct and runs in polynomial time, it suffices to observe that at the beginning of the algorithm, when $c = 1$, one has $i \leq t-1$, and that i decreases by at least 1 in every iteration step. To prove the latter assertion, note that

$$a^{2^{i-1}} = -(bc^2)^{2^{i-1}} = \left(b(cb^{2^{t-i-1}})^2\right)^{2^{i-1}}.$$

The main obstacle in extending this algorithm to the case of general n is the impossibility of writing a general n-th degree extension of $\mathbf{F}_p$ in the form $\mathbf{F}_p(a^{1/n})$, with $a \in \mathbf{F}_p$. It turns out that it is sufficient to consider the case that n is prime, $n \neq p$. Evdokimov [7] deals with this problem by passing to the n-th cyclotomic extension of $\mathbf{F}_p$ and using Kummer theory. It is for the construction of this cyclotomic extension that the generalized Riemann hypothesis is needed. In [12], this problem is circumvented by using cyclotomic *ring* extensions, which can be obtained without any unproved hypotheses, and developing the required Kummer theory for ring extensions.

The problem of finding isomorphisms between finite fields can be generalized in several ways. For example, one may ask for an *embedding* of one explicitly given finite field into another; or for *all* such embeddings; and one may add the restriction that the embeddings are the identity on an explicitly given common subfield. All these variants can be dealt with in a straightforward way by means of the techniques of [12] (see in particular [12], Section 2).

4. Irreducibility testing.

Let E be an explicitly given finite field, and let $q = \#E$. The fact that irreducibility testing in $E[X]$ can be done by means of a deterministic polynomial time algorithm is an immediate consequence of the well known formula

$$X^{q^m} - X = \prod_g g,$$

where m is any positive integer and g ranges over the set of all monic irreducible polynomials in $E[X]$ of degree dividing m. It follows from this formula that a polynomial $f \in E[X]$ is irreducible if and only if

$$\gcd(X^{q^i} - X, f) = 1 \quad \text{for} \quad 1 \le i \le [(\deg f)/2],$$

and if and only if we have

$$X^{q^{\deg f}} \equiv X \bmod f$$

and

$$\gcd(X^{q^{(\deg f)/r}} - X, f) = 1$$

for each prime r dividing $\deg f$. To see that each of these irreducibility criteria gives rise to a deterministic polynomial time irreducibility test it suffices to show how to calculate $X^{q^i} \bmod f$ for $i \le \deg f$; the necessary greatest common divisors can then be calculated by means of the Euclidean algorithm. To calculate $X^{q^i} \bmod f$, one can use the well-known algorithm that depends on the binary expansion of q^i (see [9], Section 4.6.3). Alternatively, one does this only for $i = 1$ in order to obtain the $\deg f \times \deg f$ matrix that expresses the E-linear map

$$Q: E[X]/fE[X] \to E[X]/fE[X], \qquad Q(x) = x^q \quad \text{for all } x,$$

on the basis $1, X, \ldots, X^{(\deg f)-1}$ of $E[X]/fE[X]$ over E. The coefficients of the remainder of X^{q^i} modulo f can be read from the i-th power of this matrix.

Once the matrix that describes the map Q has been calculated, one can use it in a different way to test f for irreducibility. Namely, f is irreducible if and only if

$$\gcd(f, f') = 1 \quad \text{and} \quad \operatorname{rank}(Q - \mathrm{id}) = (\deg f) - 1,$$

where id denotes the identity function from $E[X]/fE[X]$ to itself. For a proof of this fact, and a comparison of the different irreducibility tests that we discussed, see [11], Sections 4 and 5.

The generalized Riemann hypothesis or random number generators do not enter into any efficient algorithm for irreducibility testing in $E[X]$ that I am aware of.

5. Factoring polynomials.

Let E be an explicitly given finite field, p its characteristic, and $f \in E[X]$ a nonzero polynomial. In this section, we discuss algorithms for factoring f into irreducible factors. Although our interest is mainly theoretical, some of the ideas that will be discussed do have practical value. For a discussion of these aspects we refer to [11].

It is a fundamental consequence of Berlekamp's factoring algorithm ([11], Section 4), that there is a deterministic polynomial time algorithm that reduces the problem of factoring f in $E[X]$ to the problem of factoring a polynomial $g \in \mathbf{F}_p[X]$ into irreducible factors in $\mathbf{F}_p[X]$, in the special case that it is known that all those factors are *linear* and *distinct*; i.e., g divides $X^p - X$. This reduction is of a simpler sort than the "Turing" reduction of Shoup that we mentioned in Section 2: given E and f, the reduction produces in polynomial time a single polynomial g as above, such that knowledge of the linear factors of g in $\mathbf{F}_p[X]$ enables one to find the full factorization of f in $E[X]$ in polynomial time. For a description of this reduction we refer to [11], Sections 3 and 4.

For the remainder of this section, we assume that $g \in \mathbf{F}_p[X]$ is a polynomial with $X^p \equiv X \bmod g$, and we put $n = \deg g$. We are interested in algorithms to find all linear factors of g. Equivalently, we may ask for all zeros of g in $\mathbf{F}_p$. The problem is trivial if $n = 1$. We will often tacitly assume that $n > 1$, and in that case we may also be satisfied with obtaining a *splitting* of g, i.e. a decomposition $g = g_1 g_2$ into polynomials of lower degrees; one can then proceed recursively with g_1 and g_2.

For small p, an obvious approach is to try all elements of $\mathbf{F}_p$ as possible zeros of g. This works in time $O\big(p(n + \log p)^{O(1)}\big)$, and it shows that there is a deterministic algorithm that factors f in $E[X]$ in time $O\big(p(\deg f + \log \#E)^{O(1)}\big)$.

In these results, one can replace the factor p by $\sqrt{p}$ if one uses a faster method to check all elements of $\mathbf{F}_p$ as possible zeros of g. This can be done by means of a device that was used by Strassen in the context of factoring integers ([23], Section 6). Let s be the least integer $> \sqrt{p}$, and let $h \in \mathbf{F}_p[X]$ be the polynomial

$$h = \prod_{i=1}^{s} (X - i).$$

For each integer j we have

$$h(X - js) = \prod_{i=js+1}^{(j+1)s} (X - i).$$

Therefore the zeros of $\gcd(h(X - js), g)$ are precisely the zeros of g among $js + 1$, $js + 2$, $\ldots$, $js + s$. Hence, if we could calculate all $h(X - js) \bmod g$ for $0 \le j < s$ in time $\sqrt{p} \cdot (n + \log p)^{O(1)}$, then we could calculate all greatest common divisors $\gcd(h(X - js), g)$ and, for those values of j for which the gcd is non-trivial, check the elements $js + 1$, $js + 2$, $\ldots$, $js + s$ one by one. This would give us all zeros of g in time $\sqrt{p} \cdot (n + \log p)^{O(1)}$.

To calculate all $h(X - js) \bmod g$ efficiently we make use of methods that depend on the fast Fourier transform. The coefficients of h can be computed in time $\sqrt{p} \cdot (\log p)^{O(1)}$ (see [2]). If x denotes the image of X in $\mathbf{F}_p[X]/g\mathbf{F}_p[X]$, then

$$h(x - js) = \big(h(X - js) \bmod g\big),$$

so it suffices to calculate the s values

$$h(x), \quad h(x - s), \quad h(x - 2s), \quad \ldots, \quad h(x - (s-1)s) \in \mathbf{F}_p[X]/g\mathbf{F}_p[X]$$

of the s-th degree polynomial h. This can be done in time $\sqrt{p} \cdot (n + \log p)^{O(1)}$, as required, by the results in [2].

The storage requirement of the algorithm just sketched is at least of the order $\sqrt{p}$. Below we shall mention a deterministic algorithm that achieves the same time bound $O(\sqrt{p}\cdot(n+\log p)^{O(1)})$, but with storage requirement only $(n+\log p)^{O(1)}$.

If p is large, and in particular odd, then the following probabilistic algorithm splits g in polynomial time. Pick $a \in \mathbf{F}_p$ at random. The polynomial g divides

$$X^p - X = (X+a)^p - (X+a) = (X+a)\cdot\big((X+a)^{(p-1)/2}-1\big)\cdot\big((X+a)^{(p-1)/2}+1\big),$$

so we can hope to split g by

$$g = \gcd(X+a,\, g)\cdot\gcd\big((X+a)^{(p-1)/2}-1,\, g\big)\cdot\gcd\big((X+a)^{(p-1)/2}+1,\, g\big).$$

The first gcd is usually trivial, unless $g(-a)=0$, which can be checked directly. Before calculating the other gcd's, one should replace $(X+a)^{(p-1)/2}$ by its remainder upon division by g, which can be calculated by repeated squarings and multiplications modulo g.

If $n>1$ then the above splitting is non-trivial for at least half of all $a\in\mathbf{F}_p$ (see [11], Lemma 3.3). This implies that the probabilistic algorithm just described runs in polynomial time.

This probabilistic algorithm can be turned into a deterministic one that runs in time $O(\sqrt{p})\cdot(n+\log p)^{O(1)}$ by trying $a=0, 1, 2, \ldots$, in succession, and showing that the least successful a is at most $\sqrt{p}\cdot\log p$. This observation is due to Shoup [19, 20].

If we allow the generalized Riemann hypothesis in the running time analysis then the results do not become much better. Shoup [21] proved that in this case the factor $O(\sqrt{p})$ can be replaced by $\sqrt{S(p-1)}$, where $S(p-1)$ denotes the largest prime divisor of $p-1$. Similar results involving $S(p-1)$ have been proved by various authors [13, 14, 17, 24]. The number $p-1$ occurs in these results because it is the order of the multiplicative group $\mathbf{F}_p^*$. Other groups can also be used [5, 18]. As a byproduct, Bach and Von zur Gathen [5] obtain, without any unproved hypotheses, the amusing result that g can be factored by a deterministic algorithm in time $(n+\log p)^{O(1)}$ if p is a Mersenne prime.

Rónyai [16] proved the following result, assuming the truth of the generalized Riemann hypothesis: if r is a prime number dividing n, then a non-trivial factor of g can be found in time $(n^r+\log p)^{O(1)}$. The same result can be proved without assuming the generalized Riemann hypothesis if an irreducible polynomial of degree r over $\mathbf{F}_p$ is known; as we saw in Section 2, such a polynomial can be found in polynomial time if the generalized Riemann hypothesis is true.

It follows from Rónyai's result, under assumption of the generalized Riemann hypothesis, that polynomials $f\in E[X]$ with a bounded number of irreducible factors can be factored by a deterministic algorithm in polynomial time.

Several authors obtained results about factoring ($h \bmod p$) in $\mathbf{F}_p[X]$ for polynomials $h\in\mathbf{Z}[X]$ of which the Galois group of h over $\mathbf{Q}$ is subjected to various restrictions [7, 8, 18].

References

1. L.M. Adleman, H.W. Lenstra, Jr., 'Finding irreducible polynomials over finite fields', *Proc. 18th Annual ACM Symp. on Theory of Computing (STOC)*, (1986), 350–355.
2. A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The design and analysis of computer algorithms.* (Addison-Wesley, Reading, 1974.)
3. E. Bach, 'Explicit bounds for primality testing and related problems', *Math. Comp.*, to appear.
4. E. Bach, V. Shoup, 'Factoring polynomials using fewer random bits', Computer Sciences Department, University of Wisconsin, Madison, 1988.
5. E. Bach, J. von zur Gathen, 'Deterministic factorization of polynomials over special finite fields', Technical Report #799, Computer Sciences Department, University of Wisconsin, Madison, 1988.
6. L. E. Dickson, *History of the theory of numbers*, vol. I. (Carnegie Institute, Washington, 1919; Chelsea, New York, 1971.)
7. S.A. Evdokimov, 'Efficient factorization of polynomials over finite fields and generalized Riemann hypothesis', preprint, Leningrad Institute for Informatics and Automatization, 1986.
8. M.-D. Huang, 'Riemann hypothesis and finding roots over finite fields', *Proc. 17th Annual ACM Symp. on Theory of Computing (STOC)*, (1985), 121–130.
9. D. E. Knuth, *The art of computer programming*, vol. 2. (Second edition, Addison-Wesley, Reading, 1981.)
10. S. Lang, *Algebraic number theory.* (Addison-Wesley, Reading, 1970.)
11. A.K. Lenstra, 'Factorization of polynomials', *Computational methods in number theory*, H.W. Lenstra, Jr., and R. Tijdeman (eds), Mathematical Centre Tracts **154/155** (Mathematisch Centrum, Amsterdam, 1982), 169–198.
12. H.W. Lenstra, Jr., 'Finding isomorphisms between finite fields', to appear.
13. M. Mignotte, C.P. Schnorr, 'Calcul déterministe des racines des polynômes dans un corps fini', *C. R. Acad. Sci. Paris Sér. I Math.* **306** (1988), 467–472.
14. R.T. Moenck, 'On the efficiency of algorithms for polynomial factoring', *Math. Comp.* **31** (1977), 235–250.
15. E.H. Moore, 'A doubly-infinite system of simple groups', *Bull. New York Math. Soc.* **3** (1893), 73–78; Mathematical Papers read at the Congress of Mathematics (Chicago, 1893), 208–242, Chicago, 1896.
16. L. Rónyai, 'Factoring polynomials over finite fields', *Proc. 28th IEEE Symp. on Foundations of Computer Science (FOCS)*, (1987), 132–137.
17. L. Rónyai, 'Factoring polynomials modulo special primes', *Combinatorica*, to appear.
18. L. Rónyai, 'Galois groups and factoring polynomials over finite fields', to appear.
19. V. Shoup, 'On the deterministic complexity of factoring polynomials over finite fields', Computer Sciences Technical Report #782, University of Wisconsin, Madison, 1988.

20. V. Shoup, *Removing randomness from computational number theory*, Ph. D. thesis, Computer Sciences Technical Report #865, University of Wisconsin, Madison, 1989.

21. V. Shoup, 'A theorem on factoring polynomials over finite fields', Computer Sciences Technical Report #866, University of Wisconsin, Madison, 1989.

22. V. Shoup, 'New algorithms for finding irreducible polynomials over finite fields', *Math. Comp.*, to appear.

23. V. Strassen, 'Einige Resultate über Berechnungskomplexität', *Jahresber. Deutsch. Math.-Verein.* **78** (1976), 1–8.

24. J. von zur Gathen, 'Factoring polynomials and primitive elements for special primes', *Theoret. Comput. Sci.* **52** (1987), 77–89.

Department of Mathematics, University of California, Berkeley, California 94720, USA.