MIT/LCS/TR-213

PROBABILISTIC ALGORITHMS IN FINITE FIELDS

Michael O. Rabin

This blank page was inserted to preserve pagination.

MIT/LCS/TR-213

PROBABILISTIC ALGORITHMS IN FINITE FIELDS

Michael O. Rabin

January 1979

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

PROBABILISTIC ALGORITHMS IN FINITE FIELDS

by

Michael O. Rabin

Visiting Professor of Applied Mathematics, MIT
Professor of Mathematics, Hebrew University
Jerusalem, Israel

Abstract. We present probabilistic algorithms for the problems of finding an irreducible polynomial of degree n over a finite field, finding roots of a polynomial, and factoring a polynomial into its irreducible factors over a finite field. All of these problems are of importance in algebraic coding theory, algebraic symbol manipulation, and number theory. These algorithms have a very transparent, easy to program structure. For finite fields of large characteristic p, so that exhaustive search throng z_p is not feasible, our algorithms are of lower order in the degrees of the polynomial and fields in question, than previously published algorithms.

Research on probabilistic algorithms in finite fields was work conducted during 1976 while at MIT.

Key Words and Phrases. Computations in finite fields; root-finding; factorization of polynomials; probabilistic algorithms.

PROBABILISTIC ALGORITHMS IN FINITE FIELDS

Michael O. Rabin

In this paper we utilize the method of probabilistic algorithms to solve some important computational problems pertaining to finite fields. The questions we deal with are the following. Given a prime p and an integer n, how do we actually perform the arithmetical operations of $E = GF(p^n)$. Given a polynomial f(x) of degree m with coefficients in E, we wish to find a root $\alpha \in E$ of $f(x) = 0$, if such a root does exist. This is the root-finding problem. Finally, given a polynomial $f(x) \in E[x]$, we want to find the factorization $f = f_1 \cdot f_2 \cdot \ldots \cdot f_k$ of f into its irreducible factors $f_i(x) \in E[x]$. This is the factorization problem.

All of the above problems are of great significance in algebraic coding theory, see [2], in algebraic symbol manipulation, and in computational number theory.

Algorithms for the latter two problems are given in Berlekamp's [2] and more completely in the important paper [3] which culminates his own work on the subject and also incorporates important ideas of Collins, Knuth, Welch, Zassenhaus, and others.

Berlekamp solves the root-finding problem for $f \in GF(p^n)$, $\deg(f) = m$, by reducing it to the factorization problem of another polynomial $F(x) \in Z_p[x]$ ($Z_p = GF(p)$, is the field of residues mod p), where $\deg(F) = mn$. The problem of factoring $F(x) \in Z_p[x]$ is solved by reducing it to finding the roots in Z_p of another polynomial $G(x) \in Z_p[x]$. Thus everything is reduced to root-finding in Z_p. For root-finding in a large Z_p, a case in which search is not feasible, Berlekamp proposes a probabilistic algorithm involving a random choice of $d \in Z_p$. The article [3] does not contain a proof for the validity of this algorithm.

Our starting point is to solve directly the problem of root-finding in $GF(p^n) = E$ for polynomials $f \in E[x]$, by a probabilistic algorithm which generalizes to arbitrary finite fields Berlekamp's algorithm for Z_p. The validity of the algorithm is based on Theorem 4 which has a surprisingly simple proof.

We now base factorization of a polynomial $f(x) \in Z_p[x]$ on root-finding for the same f. Namely, if f(x) has irreducible factors of degree m, $h_i(x) \in Z_p[x]$, $1 \leq i \leq k$, then the product $D(x) = \Pi h_i(x)$ of these factors can be readily found by computations in $Z_p[x]$. The roots of D(x) are in $GF(p^m)$ and the above root-finding algorithm allows us to directly find such a root $\alpha \in GF(p^m)$. The minimal polynomial $h(x) \in Z_p[x]$ of α, which is of degree m, can be found by one of two methods given in Section 3. Now, α is also a root of some $h_i(x)$ of degree m, so that $h(x) = h_i(x)$, and we have found one irreducible factor of f(x). An iteration of this process finds all the irreducible factors. The same algorithm works for factorization of polynomials $f(x) \in E[x]$, where E is any finite field, by use of roots of the polynomial f(x) itself.

In terms of the number of Z_p-operations (additions and multiplications mod p, of numbers $0 \leq a, b < p$) used, our algorithms are of complexity proportional to log p. Thus they are feasible even for fields $GF(p^n)$ where p is so large that exhaustive search through Z_p is not possible.

Leaving out the factor log p and factors of order $\log n \cdot \log \log n$, the algorithms presented here have the following complexities. A root of $f(x) \in GF(p^n)$, deg f = m,

can be found in $0(n^2m)$ Z_p-operations. A polynomial $f(x) \in Z_p[x]$, $\deg(f) = n$, can be factored in $0(n^3)$ operations.

If the arithmetical operations of the field $E = GF(p^n)$ are wired into circuitry so that an E-operation can be viewed as a unit, then the above root-finding algorithm uses $0(nm)$ operation. Under the same assumption for the fields $GF(p^i)$, $i \leq n$, the factorization of $f(x)$ uses $0(n^2)$ operations.

The root-finding and factorization algorithms for the case of large p, given in [3] are of higher order in n. Root-finding for $f(x) \in GF(p^m)$, $\deg(f) = n$, uses $0((n \cdot m)^3 \cdot m)$ Z_p-operations. Factorization of $f \in Z_p[x]$, $\deg(f) = n$, uses $0(n^4)$ Z_p-operations.

If p is small so that it is practicable to find a solution in Z_p of $f(x) = 0$ by search, then a more careful comparison between the algorithms given here and the non-probabilistic algorithms presented in [3] is necessary. The latter algorithm for factorization will run in time $0(n^3)$ but there is an $0(p)$ factor. Our algorithm will run in $0(n^3)$ (in the non-preprocessed case) with a factor of $0(\log p)$. Thus for very small p, exact comparisons will

depend on the numerical constants involved. However, the algorithms given here are sufficiently fast in all cases to justify their use even for small values of p.

The probabilistic nature of our algorithms does not detract from their practical applicability. The basic probabilistic step is a random choice of an element $\delta \in E$ which is then used in an attempt to split a polynomial $f(x)$ into two factors. We prove that for any fixed finite field E and any fixed $f(x)$, the probability of success by such a random choice is at least half. Thus the expected number of such steps leading to success is at most two. Furthermore, in an algorithm involving many such steps, the probability of a run of bad random choices leading to a significant deviation from the expected total number of steps is very small.

1. ARITHMETIC OF GF(p^n)

Let p be a prime, n an integer and $q = p^n$. As customary, denote by GF(q) = E the unique finite field of q elements. In particular GF(p) = Z_p is the field of residues mod p. We want to actually compute with elements of E. For $Z_p = \langle \{0,1,\ldots,p-1\}, +, \cdot \rangle$, the operations are simply addition and multiplication mod p. If

(1) $g(x) = x^n + a_{n-1}x^{n-1} + \ldots + a_0 \in Z_p[x]$,

is an <u>irreducible</u> polynomial of degree n , then

$GF(p^n) \cong Z_p[x]/(g(x))$

where (g) is the ideal generated by g. Given such a g(x), E can be represented as the set of n-tuples of elements of Z_p. Let $\beta = (b_{n-1},\ldots,b_0)$, $\gamma = (c_{n-1},\ldots,c_0)$. Addition is component-wise. To multiply, form

$$d(x) = (b_{n-1}x^{n-1}+\ldots+b_0)(c_{n-1}x^{n-1}+\ldots+c_0)$$

and find the residue $\delta(x) = d_{n-1}x^{n-1}+\ldots+d_0$ of $d(x)$ when divided by $g(x)$. Then $\beta\cdot\gamma = (d_{n-1},\ldots,d_0)$.

Thus we need a method for finding an irreducible polynomial (1). To test for irreducibility we use the following.

LEMMA 1. Let $\ell_1,\ldots,\ell_k$ be all the prime divisors of n and denote $n/\ell_i = m_i$. A polynomial $g(x) \in Z_p[x]$ of degree n is irreducible in $Z_p[x]$ if and only if

$$(2) \quad g(x) \mid (x^{p^n}-x),$$

$$(3) \quad (g(x), x^{p^{m_i}}-x) = 1, \quad 1 \le i \le k,$$

where (a,b) denotes the greatest common divisor of a and b.

Proof. Assume that $g(x)$ is irreducible, then every root α of $g(x) = 0$ lies in $E = GF(p^n)$. Hence $\alpha^{p^n} - \alpha = 0$, and $(x-\alpha) \mid (x^{p^n}-x)$. Since $g(x)$ has no multiple roots, (2) follows.

Since $g(x)$ is irreducible of degree n, it has no roots in any field $GF(p^m)$, $m<n$. This directly implies (3).

Assume conversely that (2) and (3) hold. From (2) it follows that all roots of $g(x) = 0$ are in $E = GF(p^n)$.

Assume that g has an irreducible factor $g_1(x)$ of degree $m<n$. The roots of $g_1(x)$ lie in $GF(p^m)$ which is generated over Z_p by any one of these roots. Hence $GF(p^m) \subseteq E$ and $m|n$. Consequently $m|m_i$ for one of the maximal divisors m_i of n, and all roots of $g_1(x)$ lie in $GF(p^{m_i})$. But then $(g(x), x^{p^{m_i}}-x)$ is divisible by $g_1(x)$ contradicting (3). Thus g(x) must be irreducible.

In computing the number of operations required to test a given polynomial for primality we count, here and elsewhere in this article, in terms of arithmetical operations of Z_p. To obtain a bit-operations count, we should multiply our results by B(p) - the number of bit operations required to multiply or divide two numbers of log p bits. As is well known, B(p) can be taken to be $O(\log p \log\log p)$.

In order to shorten subsequent formulas we introduce the following

Notation: $L(n) = \log n \cdot \log\log n$

The computation of $(g(x), x^{p^n}-x)$ is executed by computing x^{p^n} modulo g(x). As is well known, x^{p^n} can be calculated by

at most $2 \cdot \log p^n$ multiplications mod $g(x)$. Since we compute mod $g(x)$ we never deal with polynomials of degree greater than $2n$.

It is shown in [4] that multiplying two n-degree polynomials with coefficients in any finite field can be done by $0(n \log n \log \log n) = 0(n\, L(n))$ field operations. Consequently division and finding remainder can be done in $0(nL(n))$ operations, see [1 ,p.288]. Thus the basic step of computing $r(x) \cdot s(x) \bmod g(x)$, where $\deg(r), \deg(s) \leq n-1$, uses $0(nL(n))$ operations. The computation of x^{p^n} uses $0(n^2 L(n) \log p)$ operations.

To test (3) we need $k \leq \log n$ computations of the above type so that the total number of operations is $0(n^2 \log n L(n) \log p)$.

The search for an irreducible polynomial of degree n is based on the following result which is a weaker form, sufficient for our purposes, of Theorem 3.3.6 [2]. We give a proof not utilizing generating functions.

LEMMA 2. Denote by $m(n)$ the number of different monic polynomials in $Z_p[x]$ degree n which are irreducible. Then

$$(4) \qquad \frac{p^n - p^{n/2} \log n}{n} \leq m(n) \leq \frac{p^n}{n}$$

(5) $$\frac{1}{2n} \leq \frac{m(n)}{p^n} \sim \frac{1}{n} \quad .$$

Note that p^n is the number of all monic polynomials of degree n.

Proof. Let $g_1(x),\ldots,g_\ell(x)$, $\ell = m(n)$, be all the pairwise different irreducible monic polynomials of degree n. Any element $\alpha \in E = GF(p^n)$ which is of degree n over Z_p satisfies exactly one equation $g_i(x) = 0$ and each such equation has exactly n such roots. If $H \subseteq E$ is the set of elements of degree n over Z_p, then $c(H)/n = m(n)$.

An element $\alpha \in E$ is in H if it is not in any proper maximal subfield $GF(p^{m_i}) \subset E$, where m_i is a maximal divisor of n (see the notation in Lemma 1). The cardinality of such a subfield is at most $p^{n/2}$ and the number of these maximal subfields is smaller than log n. Thus $p^n - p^{n/2} \log n \leq c(H)$ from which (4) and (5) follow.

In [2] Berlekamp remarks that Theorem 3.36 means that a randomly chosen polynomial of degree n will be irreducible with probability nearly 1/n, without suggesting to base an algorithm on this fact. In the general spirit of the present paper, we solve the problem of finding an irreducible polynomial by randomization.

The algorithm for finding an irreducible polynomial proceeds as follows. Choose a polynomial (1) randomly and test for irreducibility; continue until an irreducible polynomial of degree n is found. Lemma 2 ensures that the expected number of polynomials to be tried before an irreducible one is found is n. Thus the expected number of operations (in Z_p) for finding an irreducible polynomial of degree n is $0(n^3 \log n L(n) \cdot \log p)$.

The root-finding algorithm for GF(q) assumes that the arithmetic of this field is given, so that the question of finding an irreducible polynomial actually does not arise. In the factorization of a polynomial of degree n we may need computations in fields $GF(p^{n_i})$, $1 \le i \le \ell$, such that $\Sigma n_i \le n$. The count of all operations, including the precomputation of the $g_{n_i}(x)$, will use the following.

LEMMA 3. Let n_i, $1 \le i \le \ell$, satisfy $\Sigma n_i \le n$. The expected number of operations used for finding irreducible polynomials $h_i(x)$, $\deg(h_i) = n_i$, $1 \le i \le \ell$, is $0(n^3 \log n L(n) \log p)$.

Proof.

$$\Sigma n_i^3 \log n_i L(n_i) \log p \le n^2 \log nL(n) \log p \Sigma n_i \le$$

$$\le n^3 \log n L(n) \log p.$$

2. ROOT-FINDING IN GF(p^n)

Let E = GF(q) be a fixed finite field, and f(x) ε E[x] be a polynomial of degree m. We wish to find one (or all) of the roots $\alpha \varepsilon E$ of f(x) = 0. We give a probabilistic algorithm for this problem, which is a generalization of the algorithm given in Berlekamp [3] for prime fields Z_p, to arbitrary finite fields E. Our proof for the validity of the general algorithm of course applies also to the special case of Z_p, which is given essentially without proof in [3].

Assume for the time being that $q = p^n$ is odd. We shall indicate later how to treat the important case $q = 2^n$.

Form the g.c.d.

$$f_1(x) = (f(x), x^{q-1}-1).$$

If $f_1(x) = 1$ then f(x) has no roots in E. In general

$$f_1(x) = (x-\alpha_1)\ldots(x-\alpha_k), \qquad k \leq m,$$

where the α_i are all the pairwise different roots in E of f(x) = 0.

Now

(6) $$x^{q-1}-1 = (x^d-1)(x^d+1) \quad , \; d = \frac{q-1}{2} \; .$$

The next natural step is to try $(f_1(x), x^d-1)$. If some of the α_i satisfy $\alpha_i^d-1 = 0$ while others satisfy $\alpha_j^d+1 = 0$, then this g.c.d. will be a true divisor of $f_1(x)$, and we will have further advanced towards the goal of finding a linear factor $x-\alpha$, i.e. a root, of $f(x)$. In general we are not guaranteed that the g.c.d will be different from 1 or $f_1(x)$. However, this advantageous situation can be created by randomization.

Call $\alpha,\beta \in E$, $\alpha \neq 0$, $\beta \neq 0$, of <u>different type</u> if $\alpha^d \neq \beta^d$, where $d = \frac{q-1}{2}$.

THEOREM 4. Let $\alpha_1,\alpha_2 \in E$, $\alpha_1 \neq \alpha_2$.

(7) $$\frac{q-1}{2} = c(\{\delta \mid \delta \in E, \; \alpha_1+\delta \text{ and } \alpha_2+\delta \text{ are of different type}\})$$

<u>Proof</u>. The elements $\alpha_1+\delta$ and $\alpha_2+\delta$ are of different type if and only if neither is zero and

$$\left(\frac{\alpha_1+\delta}{\alpha_2+\delta}\right)^d \neq 1; \quad \text{hence} \quad \left(\frac{\alpha_1+\delta}{\alpha_2+\delta}\right)^d = -1.$$

The equation $x^d = -1$ has exactly $d = \frac{q-1}{2}$ solutions in E. Consider the 1-1 mapping $\phi(\delta) = \frac{\alpha_1+\delta}{\alpha_2+\delta}$. As δ ranges over $E - \{-\alpha_2\}$, $\phi(\delta)$ ranges over $E - \{1\}$. Thus for exactly $\frac{q-1}{2}$ values of δ, $\phi(\delta)^d = -1$. This implies (7).

COROLLARY 5. Consider for $\delta \in E$ the g.c.d $f_\delta(x) = (f_1(x), (x+\delta)^d - 1)$. We have

$$(8) \quad \frac{1}{2} \leq \Pr(\delta \mid 0 < \deg f_\delta(x) < \deg f_1)$$

Proof. The common roots of $f_1(x)$ and $(x+\delta)^d - 1$ are those α_i $(f_1(\alpha_i) = 0)$ for which $(\alpha_i+\delta)^d - 1 = 0$. By Theorem 4, with probability 1/2, $\alpha_1+\delta$ has this property while $\alpha_2+\delta$ does not, or vice-versa. This entails (8). Actually the probability is nearly $1-1/2^k$, where $\deg f_1 = k$, but we cannot prove this.

Root-finding algorithm. Given f(x) of degree m, compute $f_1(x)$. Choose $\delta \in E$ randomly and compute $f_\delta(x)$. If $0 < \deg f_\delta < \deg f_1$ then let $f_2(x) = f_\delta(x)$ or $f_2(x) = f_1/f_\delta$, according as to whether $\deg f_\delta \leq 1/2 \deg f_1$ or not. If $f_\delta = 1$ or $f_\delta = f_1$ choose another δ and repeat the previous step. By Corollary 5, the expected number of choices of $\delta \in E$ until we find $f_2(x)$ is less than 2.

Since the degree is at least halved in each step, after at most log m steps we find a linear factor $x-\alpha_i$ of f(x), i.e. a root.

The number of (field -E) arithmetical operations required for finding $f_1(x)$ and $f_2(x)$ is $0(n \cdot m\, L(m) \log p)$, where $E = GF(p^n)$. Since deg $f_2 \leq \frac{1}{2} m$, it follows that the number of operations for finding $f_3(x)$ is at most half the number of operations for finding f_2; and similarly for f_4 etc. Thus the total number of E-operations used for finding a root of f(x) is still just $0(n \cdot mL(m) \log p)$.

In terms of operations in Z_p, each E-operation re-