Computational Number Theory and Algebra	May 14, 2012

Lecture 8

Lecturers: Markus Bläser, Chandan Saha — *Scribe: Chandan Saha*

In the previous lecture, we started our discussion on factoring polynomials over finite fields and showed that after the square-free factoring and the distinct-degree factoring steps, we are left with factoring a polynomial that splits into *equal-degree* irreducible factors. We will see how to handle this equal-degree factoring case, today. Today's topics of discussion are:

- Equal-degree factoring,
- Reducing polynomial factoring to root finding,
- Testing irreducibility of a polynomial.

1 Equal-degree factoring over finite fields

We are now left with the task of factoring a square-free polynomial $f = f_1 \ldots f_k$ over a finite field $\mathbb{F}_q$ such that all the irreducible factors $f_i \in \mathbb{F}_q[x]$ have the same degree, say, d (i.e., $dk = n = \deg(f)$). This step is called *equal-degree factoring*. The key to solving this case lies in understanding the structure of the ring $\mathcal{R} = \frac{\mathbb{F}_q[x]}{(f)}$. By the Chinese remaindering theorem, $\mathcal{R}$ has the following structure,

$$\mathcal{R} = \frac{\mathbb{F}_q[x]}{(f)} \cong \oplus_{i=1}^{k} \frac{\mathbb{F}_q[x]}{(f_i)} \cong \underbrace{\mathbb{F}_{q^d} \oplus \mathbb{F}_{q^d} \oplus \ldots \oplus \mathbb{F}_{q^d}}_{k \text{ times}}. \quad (1)$$

Since, f_i is irreducible of degree d, the ring $\frac{\mathbb{F}_q[x]}{(f_i)}$ is (isomorphic to) the finite field $\mathbb{F}_{q^d}$. So, basically $\mathcal{R}$ is a direct sum of k finite fields $\mathbb{F}_{q^d}$. We call these k fields as the k *components* of $\mathcal{R}$. At this point, we would need a few basic facts about finite fields.

Definition 1 *An element a in a finite field $\mathbb{F}_q$ is called a* quadratic residue *if there exists an element $b \in \mathbb{F}_q$ such that $b^2 = a$ in $\mathbb{F}_q$. Otherwise, a is called a* quadratic non-residue.

Lemma 2 *Let $\mathbb{F}_q$ be the finite field containing q elements. Then, $\frac{q-1}{2}$ elements of $\mathbb{F}_q^{\times} = \mathbb{F}_q \setminus \{0\}$ are quadratic residues, and the remaining $\frac{q-1}{2}$ elements of $\mathbb{F}_q^{\times}$ are quadratic non-residues.*

Lemma 3 *An element $a \in \mathbb{F}_q^{\times}$ is a quadratic residue if and only if $a^{\frac{q-1}{2}} = 1$, otherwise a is a quadratic non-residue in which case $a^{\frac{q-1}{2}} = -1$ in $\mathbb{F}_q$.*

We leave the proofs of the above lemmas as an exercise. Keeping in mind the correspondence between $\mathcal{R}$ and the k component fields $\mathbb{F}_{q^d}$ (in Equation (1)), we will use the notation $A = (a_1, \ldots, a_k) \in \mathcal{R}$ to mean that an element $A \in \mathcal{R}$ (which can be treated as a polynomial over $\mathbb{F}_q$ of degree less than n) has the *direct sum representation* $(a_1, \ldots, a_k)$, where $a_i = A \mod f_i$ belongs to the i^{th} component field $\mathbb{F}_{q^d}$. We say that $a_1, \ldots, a_k$ are the components of A. Now notice one thing: Picking an element in $\mathcal{R}$ uniformly at random, is like picking an element in each of the k component fields $\mathbb{F}_{q^d}$ *independently* and uniformly at random. Let A be an element of $\mathcal{R}$ chosen uniformly at random, and $A = (a_1, \ldots, a_k)$ be its direct sum representation. Suppose that none of the $a_i = 0$, $1 \leq i \leq k$. Since, $a_i \neq 0$ is a random element of $\mathbb{F}_{q^d}^{\times}$, by Lemma 2 with probability 1/2 it is a quadratic residue and with probability 1/2 it's a non-residue. Therefore, assuming that $a_i \neq 0$ for all $1 \leq i \leq k$, each of the k components of $B \stackrel{def}{=} A^{\frac{q^d-1}{2}} = (a_1^{\frac{q^d-1}{2}}, \ldots, a_k^{\frac{q^d-1}{2}}) \in \mathcal{R}$ is either 1 or -1 (by Lemma 3). Moreover, the probability that all the components are 1 or -1 is at most $1/2^{k-1} < 1/2$ (assuming that f is not irreducible) (why?). Therefore, with probability $1 - 1/2^{k-1}$, some but not all components of $B + 1$ are zero. What does this mean?

Lemma 4 *If $E = (e_1, \ldots, e_k) \in \mathcal{R}$ is such that some but not all $e_i \in \mathbb{F}_{q^d}$ are zero, then $\gcd(E, f)$ yields a proper factor of f.*

Proof An element $E \in \mathcal{R}$ is a polynomial over $\mathbb{F}_q$ of degree less than $n = \deg(f)$. If $e_i = E \mod f_i = 0$ then it means that the polynomial E is divisible by f_i. Hence, $\gcd(E, f)$ yields a proper factor of f since not all e_i's are zero. ∎

By Lemma 4, it follows that $\gcd(B+1, f)$ is a proper factor of f with probability at least $1 - 1/2^{k-1}$, under the assumption that none of the a_i's is zero. Let's get rid of this assumption that none of the a_i's is zero: If the random element $A \neq 0$ and some $a_i = 0$ then $\gcd(A, f)$ is a proper factor of f (again, by lemma 4). Therefore, if $A \neq 0$, we get a proper factor of f with probability $1 - 1/2^{k-1} > 1/2$ (assuming that f in not irreducible). This suggests the following algorithm, which is due to Cantor and Zassenhaus [CZ].

Algorithm 1 Equal-degree factoring

1. Pick an element $A \neq 0$ in $\mathcal{R}$ uniformly at random.
2. If $\gcd(A, f) \neq 1$, return this gcd.
3. Else, let $B = A^{\frac{q^d-1}{2}} \mod f$.
4. If $\gcd(B+1, f) = 1$ or f, return 'failure'.

Time complexity - We leave the task of finding the exact time complexity of this algorithm as an excercise, noting that it is polynomial in n and $\log q$.

Since, the failure probability of the algorithm is less than $1/2$, we can repeat this algorithm independently for m times, to bring down the failure probability to less that $1/2^m$.

2 Reducing equal-degree factoring to root finding

It follows from Algorithm 1 that the problem of polynomial factoring over finite fields admits a randomized polynomial time algorithm. What about a deterministic algorithm? It turns out that if the finite field $\mathbb{F}_q$ is small in size (say, $q = 5$ or 7) then it is indeed possible to factor f deterministically [1]. This is done by reducing the equal-degree factoring problem to a root finding problem over $\mathbb{F}_q$. We continue to use the same notations as before. We would need the following lemma, the proof of which is left as an exercise.

Lemma 5 *The roots of the polynomial $x^q - x$ over $\mathbb{F}_{q^d}$ are exactly the elements of the finite field $\mathbb{F}_q$, which is contained in $\mathbb{F}_{q^d}$.*

Let $g \in \mathcal{R}\backslash\mathbb{F}_q$ be such that $g^q = g$ in $\mathcal{R}$. First, we need to show that such a g exists in $\mathcal{R}$. By the isomorphism given in Equation (1) (and by Lemma 5), any g whose direct-sum representation belongs to $\oplus_{i=1}^{k}\mathbb{F}_q$ satisfies the condition $g^q = g \mod f$. Also, if f is not irreducible then such a $g \notin \mathbb{F}_q$ exists (why?). This means, there exists $c_i, c_j \in \mathbb{F}_q$ ($i \neq j$) such that $c_i = g \mod f_i$, $c_j = g \mod f_j$ and $c_i \neq c_j$. This also implies that there is a $c \in \mathbb{F}_q$ such that $\gcd(g - c, f)$ yields a non-trivial factor of f. For instance, for $c = c_i$, f_i divides $\gcd(g - c, f)$ but f_j does not.

How to find a $g \in \mathcal{R}$ that satisfies $g^q = g$? Solving a system of linear equations comes to our rescue again. To compute g, start with a generic element $g = \sum_{i=0}^{n-1} g_i x^i \in \mathcal{R}$, where $n = \deg(f)$ and g_i's are variables, and solve for $g_i \in \mathbb{F}_q$ such that $\sum_{i=0}^{n-1} g_i x^{qi} = \sum_{i=0}^{n-1} g_i x^i \mod f$. Solving this equation reduces to solving a system of linear equations in the g_i's. This reduction follows once we compute x^{qi} rem f for all i and equate the coefficients of x^j, for $0 \leq j \leq n-1$, from both sides of the equation. Now all we need to do, while solving the linear equations, is to choose a solution for the g_i's such that $\sum_{i=0}^{n-1} g_i x^i \notin \mathbb{F}_q$ (which just means that not all of $g_1, \ldots, g_{n-1}$ are zero). Take $g = \sum_{i=0}^{n-1} g_i x^i$ for that choices of the g_i's. Taking into account that

[1] in fact, it is possible to factor f in time $\mathsf{poly}(n, p)$, where $p = \mathrm{char}(\mathbb{F}_q)$.

x^{qi} rem f can be computed using repeated squaring, we conclude that g can be found in time polynomial in n and $\log q$.

The only task that remains is to find a $c \in \mathbb{F}_q$ such that $\gcd(g-c, f)$ gives a nontrivial factor of f. This is where the problem gets reduced to root finding. The fact that $\gcd(g-c, f) \neq 1$ means resultant of the polynomials $g - c = \sum_{i=1}^{n-1} g_i x^i + (g_0 - c)$ and f is zero over $\mathbb{F}_q$. This means, we need to solve for a $y \in \mathbb{F}_q$ such that $h(y) = \mathsf{Res}_x(\sum_{i=1}^{n-1} g_i x^i + (g_0 - y), f) = 0$, treating $g_0 - y$ as the constant term of the polynomial $g - y$. We can find h by computing the determinant of the Sylvester matrix, $S(\sum_{i=1}^{n-1} g_i x^i + (g_0 - y), f)$, of $g - y$ and f. Although there are entries in S containing variable y, we can find $\det(S) = h(y)$ in polynomial time using interpolation (how? see excercise 3). In this way, factoring polynomial $f(x)$ reduces to finding a root of the polynomial $h(y) \in \mathbb{F}_q[y]$.

Algorithm 2 Equal-degree factoring to root finding

1. Using linear algebra, find a $g \in \mathbb{F}_q[x]$ such that $g^q = g \mod f$ and $g \notin \mathbb{F}_q$.
2. If no such g exists then declare 'f is irreducible'.
3. Compute polynomial $h(y) = \mathsf{Res}_x(g - y, f)$ and find a root c of $h(y)$.
4. Find a nontrivial factor $\tilde{f} = \gcd(g - c, f)$ of f.
5. Repeat the above process to factor $\tilde{f}$ and $\frac{f}{\tilde{f}}$.

Time complexity - It follows from our discussion that, apart from the root finding task in step 3 of Algorithm 2 all the other steps run in polynomial time. To find a root of the polynomial $h(y)$, first compute the polynomial $\tilde{h} = \gcd(y^q - y, h)$ which splits completely over $\mathbb{F}_q$ into linear factors. Any root of $\tilde{h}$ is also a root of h and vice versa, and so the problem reduces to finding a root of $\tilde{h}$. (Once again, we would like to note that $\gcd(y^q - y, h)$ is computed by first computing $y^q \bmod h$ using repeated squaring.) Now, observe that we are back to the equal-degree factoring case where degree of each irreducible factor is $d = 1$. At this point, we can either invoke the randomized Algorithm 1, or we can try the obvious brute-force deterministic algorithm: Go over all elements of $\mathbb{F}_q$ to find a root of $\tilde{h}$. The latter strategy gives us a deterministic algorithm that runs in $\mathsf{poly}(n, q)$ time, which is fine if $q = n^{O(1)}$.

In fact, root finding over $\mathbb{F}_q$ reduces to root finding over $\mathbb{F}_p$, where p is the characteristic of the field $\mathbb{F}_q$, in time polynomial in n and $\log q$. This was shown by Berlekamp [Ber67, Ber70] (refer to chapter 4 of the book by Lidl and Neiderreiter [LN94]). Which means, if the characteristic of the field is small then even if q is large, we are still factor f deterministically. In particular, a degree-n polynomial over $\mathbb{F}_{2^m}$ can be factored deterministically in time polynomial in n and m. What if the characteristic p is also large? To find a deterministic root finding algorithm over $\mathbb{F}_p$ that runs in time polynomial in the degree n and $\log p$, is a major open problem in computational algebra.

3 Testing irreducibility of a polynomial over a finite field

Suppose, we want to design an algorithm to check if a polynomial f of degree n is irreducible over $\mathbb{F}_q$. If we could factor f then we could also check if f is irreducible. But, unfortunately, we still do not have a 'complete' factoring algorithm that runs in deterministic polynomial time. However, one might guess that checking if f is irreducible is perhaps an easier problem than actually finding the factors of f. This is indeed true - for irreducibility checking, we do have a deterministic polynomial time algorithm. The key to this lies in Lemma 5 of the previous lecture and the distinct-degree factoring step. For your convenience, we state the lemma here again.

Lemma 5 of Lecture 7: *For any $d > 1$, $x^{q^d} - x$ is the product of all monic irreducible polynomials in $\mathbb{F}_q[x]$ whose degree divides d.*

This immediately suggests the following irreducibility test.

Algorithm 3 Irreducibility test over a finite field

1. `Check if` $\gcd(x^{q^n} - x, f) = f$. `If not, declare '`f` is reducible'.`
2. `Find all prime factors of` $n = \deg(f)$.
3. `For every prime factor` t `of` n, `check if` $\gcd(x^{q^{n/t}} - x, f) \neq 1$.
4. `If there is a prime factor` t, `such that` $\gcd(x^{q^{n/t}} - x, f) \neq 1$, `declare '`$f$` is reducible'.`
5. `Otherwise, '`f` is irreducible'.`

Correctness and time complexity - We leave it as an exercise to check the correctness of this algorithm and show that it runs in time polynomial in n and $\log q$.

Exercises:

1. Prove Lemma 2, 3 and 5.
2. Show that the finite field $\mathbb{F}_{q^{d_1}}$ is contained in the finite field $\mathbb{F}_{q^{d_2}}$ if and only if d_1 divides d_2.
3. Let M be an $m \times m$ matrix whose entries are polynomials over $\mathbb{F}_q$ with degree bounded by d. Show that $\det(M)$, which is also a polynomial over $\mathbb{F}_q$, can be computed in time polynomial in m, d and $\log q$.
4. Work out the exact time complexity of Algorithm 1 and Algorithm 2 (barring step 3) in terms of n and $\log q$.
5. Prove the correctness of Algorithm 3 and find its time complexity in terms of n and $\log q$.

References

[Ber67] Elwyn Berlekamp. Factoring Polynomials Over Finite Fields. *Bell System Technical Journal*, 46:1853–1859, 1967.

[Ber70] E. R. Berlekamp. Factoring Polynomials Over Large Finite Fields. *Mathematics of Computation*, 24(111):713–735, 1970.

[CZ] David G. Cantor and Hans Zassenhaus. A New Algorithm for Factoring Polynomials Over Finite fields. *Mathematics of Computation*, 36(154):587–592.

[LN94] Rudolf Lidl and Harald Neiderreiter. *Introduction to finite fields and their applications*. Cambridge University Press, 1994.