Lecture 10: Deterministic Factorization Over Finite Fields

Algorithmic Number Theory (Fall 2014)
Rutgers University
Swastik Kopparty
Scribe: Ross Berkowitz

1 Deterministic Factoring of 1-Variable Polynomials

Let \( q = p^e \) and \( f \in \mathbb{F}_q[x] \) a polynomial of degree \( d \). Last class we saw a randomized algorithm of Berlekamp’s which factored \( f \) in expected time \( poly(\log q, d) \). This class we will show a deterministic algorithm also due to Berlekamp which runs in time \( poly(p, \log q, d) \).

First we will show how to factor in \( poly(q, d) \) time, and then modify our method slightly to get the desired run time. To do this we need an important lemma:

**Lemma 1.** Given \( f(x) \in \mathbb{F}_q[x] \) squarefree, \( f(x) \) is irreducible iff the only \( \alpha \) satisfying \( \alpha^q \equiv \alpha \mod f \) are the constants \( \alpha \in \mathbb{F}_q \).

**Proof.** If \( f \) is irreducible then \( \mathbb{F}_q \mathbb{x} f(x) \) is a field containing \( \mathbb{F}_q \) (in particular it is isomorphic to \( \mathbb{F}_{q^d} \)). So we have that there are at most \( q \) solutions to the polynomial \( x^q - x \), and these are already known to be the constants \( \alpha \in \mathbb{F}_q \).

For the reverse direction if \( f \) is not irreducible, say \( f(x) = \prod_{i=1}^n f_i(x) \) where the \( f_i \) are irreducible, then by the Chinese Remainder Theorem we have that \( \mathbb{F}_q \mathbb{x} f(x) \cong \bigoplus_{i=1}^n (\mathbb{F}_q \mathbb{x} f_i(x)) \). Therefore we have exactly \( q^n \) solutions of the form \((\alpha_1, \alpha_2, \ldots, \alpha_n)\) where for all \( i, \alpha_i \in \mathbb{F}_q \).

**Observation 2.** \( x^q - x \) is an \( \mathbb{F}_q \)-linear function on \( \mathbb{F}_q \mathbb{x} f(x) \), so its kernel forms a vector space, which the above proof shows to have dimension \( n \) where \( n \) is the number of irreducible factors of the polynomial \( f \).

We use this observation in making the following algorithm for factoring \( f \) in \( poly(q, d) \) time:

**Algorithm 1 for factoring polynomials over \( \mathbb{F}_q \):**

0. Make \( f \) squarefree

1. Find a basis for the space \( V = \{ a(x) \mid \deg(a(x)) < d, a(x)^q \equiv a(x) \mod f \} \). (Note we are solving an \( \mathbb{F}_q \)-linear system of equations in the coefficients of the polynomial \( a(x) \)).

2. Let \( a(x) \) be a basis vector of \( V \). We know that \( f(x) \) divides \( a(x)^q - a(x) = \prod_{\alpha \in \mathbb{F}_q} (a(x) - \alpha) \) by the definition of \( V \). Therefore for some \( \alpha \in \mathbb{F}_q \) we have \( \gcd(a(x) - \alpha, f(x)) \neq 1 \), and so after trying all \( q \) distinct possibilities for \( \alpha \) we are guaranteed to have found some factor of \( f \) of the form \( f_i = \gcd(a(x) - \alpha, f(x)) \neq 1 \).
We showed in the previous class how to perform step 0 in $\text{poly}(\log q, d)$ time. Step 1 can be done in $\text{poly}(\log q, d)$ by using fast modular exponentiation. In particular one can in $\text{poly}(\log q, d)$ time compute for each $i < d$ the remainder polynomial $r_i(x) := (x^i)^q \mod f(x)$ and so we have that if $a(x) = \sum_{i=0}^{d-1} a_i x^i$ then

$$a(x)^q - a(x) \equiv 0 \mod f \iff \sum_{i=0}^{d-1} a_i x^i \equiv \left( \sum_{i=0}^{d-1} a_i x^i \right)^q \equiv \sum_{i=0}^{d-1} a_i r_i(x) \mod f$$

Where note we have used the facts that the coefficients $a_i$ are in $\mathbb{F}_q$ and that $\deg(a) < d$.

Lastly for step 2 we needed to try taking a gcd over $\mathbb{F}_q[x]$ of possibly $q$ polynomials of degree at most $d - 1$. So this was done in $\text{poly}(q, d)$ time.

Step 2 was the slowest step, so we improve it by making the following modification to our algorithm:

**Algorithm 2 for factoring polynomials over $\mathbb{F}_q$:**

0. Make $f$ squarefree

1. Find a basis for the space $V = \{ a(x) \mid \deg(a(x)) < d, \ a(x)^p \equiv a(x) \mod f \}$. So if we represent $\mathbb{F}_q$ as an $\mathbb{F}_p$ vector space, then let $M$ be the matrix of the $\mathbb{F}_p$-linear transformation $x \mapsto x^p - x$ on $\mathbb{F}_q$. We can now express the elements of $V$ as the solutions of

$$a(x) \equiv a(x)^p \equiv \left( \sum_{i=0}^{d-1} a_i x^i \right)^p \equiv \sum_{i=0}^{d-1} a_i^p x^{ip} \equiv \sum_{i=0}^{d-1} (Ma_i) r_i(x) \mod f$$

2. Let $a(x)$ be a basis vector of $V$. We know that $f(x)$ divides $a(x)^p - a(x) = \prod_{\alpha \in \mathbb{F}_p} (a(x) - \alpha)$ by the definition of $V$. So for some $\alpha \in \mathbb{F}_p$ we have $\gcd(a(x) - \alpha, f(x)) \neq 1$, and so after trying all $p$ distinct possibilities for $\alpha$ we are guaranteed to have found some factor of $f$ of the form $f_i = \gcd(a(x) - \alpha, f(x)) \neq 1$.

We note that step 1 is similar to step 1 before, but now instead of solving an $\mathbb{F}_q$ linear system of equations, we solve an $\mathbb{F}_p$ linear system as both the right and left terms of the equation $a(x) \equiv \sum_{i=0}^{d-1} (Ma_i) r_i(x) \mod f$ are $\mathbb{F}_p$ linear. So this step is still solved in $\text{poly}(\log p, \log q, d) = \text{poly}(\log q, d)$ time. Step 2 is almost unchanged but done over $\mathbb{F}_p$ rather than $\mathbb{F}_q$ so it now takes only $\text{poly}(p, \log q, d)$ time. Therefore the runtime of the algorithm as a whole is $\text{poly}(p, \log q, d)$.

Lastly we note that for step 2 to work, in place of the polynomial $x^p - x$ any easily factorable sparse polynomial with few roots would have sufficed.

2 Factoring 2-Variable Polynomials

Recall the following "web of analogies" between integers/rationals and finite fields:

$$\mathbb{Z} \leftrightarrow \mathbb{F}_q[T] \quad \mathbb{Q} \leftrightarrow \mathbb{F}_q(T)$$
$$\mathbb{Z}[X] \leftrightarrow \mathbb{F}_q[T, X] \quad \mathbb{Q}[X] \leftrightarrow \mathbb{F}_q(T)[X]$$
The basic idea for factoring $F(T,X)$ will be to first fix $T = t_0$. We will find a root $x_0$ of the single variable polynomial $F(t_0,X)$ and then find a Taylor expansion of the curve of zeroes of $F(T,X)$ near $(t_0,x_0)$. After computing the power series of this curve to high precision we will identify the actual curve, and thereby find a factor of $F$.

**High Level Overview of Algorithm for Bivariate Factoring:**

1. Find $t_0$ such that $F(t_0,X)$ (is squarefree) has no repeated roots in $X$.
2. Find a root $x_0$ of $F(t_0,X)$ (note: $x_0$ might not be in $\mathbb{F}_q$, but rather some extension)
3. Find a Taylor Series $X(T) = a_0 + a_1(T-t_0) + a_2(T-t_0)^2 \ldots$ such that $F(T,X(T)) = 0$ in $\mathbb{F}_q[[T-t_0]]$.
4. Use the Taylor Series $X(T)$ of a curve of zeroes of $F$ to find an associated factor of $F$.

Note that our Taylor series in step 3 will not "approximate" an actual function in any sense other than being correct up to, say, the first 100 terms, which is to say it will be correct modulo $T^{100}$.

Given $g(X) \in \mathbb{F}_q[X]$ how do we tell if $g$ is squarefree?

1. If $g'(X) = 0$ then $g$ is a perfect $p^{th}$ power, and we’ll do something else.
2. If $g'(X) \neq 0$ then $\gcd(g(x),g'(x))$ has degree $> 0$ iff $g(X)$ is not squarefree (and we have already found a factor).

To use this criterion effectively we will find a polynomial disc in the coefficients of the polynomial $g$ so that $\text{disc}(g) = 0$ if and only if $\gcd(g(x),g'(x)) = 1$ (and therefore $g$ squarefree) Note that for any polynomials $g = \sum_{i=0}^{d} g_i x^i$ and $h = \sum_{i=0}^{\ell} h_i x^i$ we have $\deg(\gcd(g,h)) > 0$ iff there exists $a(x), b(x)$ of degrees $< \ell, d$ respectively such that $a(x)g(x) + b(x)h(x) = 0$ (as the degree of $\text{lcm}(g,h)$ will have degree $\ell + d - \gcd(g,h)$).

**Definition 3.** Given $g, h \in \mathbb{F}_q[X]$ let $M$ be the $d + \ell \times d + \ell$ matrix

$$
M := \begin{pmatrix}
g_0 & g_1 & g_2 & \ldots & \ldots & g_d & 0 & \ldots & \ldots & 0 \\
0 & g_0 & g_1 & g_2 & \ldots & \ldots & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
0 & 0 & g_0 & \ldots & \ldots & g_{d-2} & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
\vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \ddots \\
0 & \ldots & \ldots & 0 & g_0 & \ldots & \ldots & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
h_0 & h_1 & h_2 & \ldots & \ldots & h_\ell & 0 & \ldots & \ldots & 0 \\
0 & h_0 & h_1 & \ldots & \ldots & h_{\ell-1} & h_\ell & 0 & \ldots & \ldots & 0 \\
0 & 0 & h_0 & \ldots & \ldots & h_{\ell-2} & h_{\ell-1} & h_\ell & 0 & \ldots & \ldots & 0 \\
\vdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & \ddots \\
0 & \ldots & \ldots & 0 & h_0 & \ldots & \ldots & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
0 & \ldots & \ldots & 0 & h_0 & \ldots & \ldots & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
0 & \ldots & \ldots & 0 & h_0 & \ldots & \ldots & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
0 & \ldots & \ldots & 0 & h_0 & \ldots & \ldots & g_{d-1} & g_d & 0 & \ldots & \ldots & 0 \\
\end{pmatrix}
$$
We define the Resultant of \( g \) and \( h \) to be \( \text{Resultant}(g, h) = \det(M) \) and define the discriminant of \( g \) to be \( \text{disc}(g) = \text{Resultant}(g, g') \).

**Observation 4.** Note that if \( g \) is a quadratic \( ax^2 + bx + c \) then \( \text{disc}(g) = b^2 - 4ac \). The discriminant isn't going to be a practical way to check if \( g \) is squarefree, but it will be a helpful theoretical tool. In general \( \text{disc}(g) \) is actually quicker to compute than determinants by finding the roots of \( g \) and using a special formula for the discriminant.

**Lemma 5.** \( \text{Resultant}(g, h) = 0 \iff \deg(\gcd(g, h)) > 0 \). Similarly \( \text{disc}(g) \neq 0 \iff g \) is squarefree.

**Proof.** For a vector \( \mathbf{v} \in \mathbb{R}^{d+\ell} \) let \( a(x) = \sum_{i=1}^{\ell} v_i x^{i-1} \), and \( b(x) = \sum_{i=1}^{d} v_{i+\ell} x^{i-1} \). Note \( \mathbf{v} \) solves the system of equations \( M^T \mathbf{v} = 0 \) iff for all \( j \) we have

\[
0 = \sum_{i=1}^{d+\ell} M_{ij} v_i = \sum_{i=1}^{\ell} g_j - i v_i + \sum_{i=\ell+1}^{\ell+d} h_{j-i-\ell} v_{i+\ell} = \sum_{i=0}^{\ell-1} g_j - i a_{i-1} + \sum_{i=\ell}^{\ell+d} h_{j-i} b_{i-1}
\]

\[
\iff [a(x)g(x) + b(x)h(x)] |_{x=j-1} = 0
\]

So we see that the existence of a nonzero vector \( \mathbf{v} \) solving \( M^T \mathbf{v} = 0 \) is equivalent to finding \( a, b \) of degree less than \( \ell - 1, d - 1 \) respectively satisfying \( a(x)g(x) + b(x)h(x) = 0 \). But this property is equivalent to \( \deg(\gcd(g, h)) > 0 \). Therefore we have that \( \text{Resultant}(M) = 0 \iff \deg(\gcd(g, h)) > 0 \). Similarly \( \text{disc}(g) = 0 \iff \deg(\gcd(g, g')) > 0 \). But we also know that \( \deg(\gcd(g, g')) = 0 \iff g \) is squarefree so we are done. \( \square \)

So, considering \( F(T, X) \) as a univariate polynomial \( F_T(X) \in \mathbb{F}_q(T)[X] \), we have

\[
F(T, X) = \sum_{i=0}^{d} F_i(T) X^i
\]

\[
F'(T, X) = \sum_{i=1}^{d} i F_i(T) X^{i-1}
\]

Where \( F_i \) is some polynomial in \( T \) with degree less than \( d \). We compute the discriminant of \( F_T \) to be

\[
\text{disc}(F_T) =
\begin{bmatrix}
F_0 & F_1 & F_2 & \ldots & F_{d-1} & F_d & 0 & \ldots & 0 \\
0 & F_0 & F_1 & \ldots & F_{d-1} & F_d & 0 & \ldots & 0 \\
0 & 0 & F_0 & \ldots & F_{d-2} & F_{d-1} & F_d & 0 & \ldots & 0 \\
\vdots & 0 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 & \ldots & 0
\end{bmatrix}
\]

Where \( F_k \) is the \( k \)-th coefficient of \( F(T, X) \).
Observation 6. \( \text{disc}(F_T(X)) = \text{disc}(F)(T) \) is a univariate polynomial in \( T \) of degree at most \( d(2d - 1) \).

As long as \( \text{disc}(F)(T) \) is not the zero polynomial there exists some \( t \in \mathbb{F}_q \) so that \( \text{disc}(F)(t) \neq 0 \). In fact as \( \deg(\text{disc}(F)(T)) \leq (2d - 1)d \) we have to check at most \( (2d - 1)d + 1 \) values of \( t \) before we are guaranteed to find \( \text{disc}(F)(t) \neq 0 \).

We also know from the above lemma that \( \text{disc}(F)(T) \) is the zero polynomial in \( T \) iff \( F(T, X) \) is squarefree in \( \mathbb{F}_q(T)[X] \).

What have we shown so far? If \( F \) is squarefree then after less than \( (2d - 1)d + 1 \) tries there will be some appropriate \( t_0 \) so that \( F(t_0, X) \) is a squarefree univariate polynomial.

In the next class we will finish up bivariate polynomial factoring, and time permitting mention multivariate factoring and primality testing.