Math 261C: Randomized Algorithms

Lecture topic: Factoring Polynomials over Finite Fields

Lecturer: Sam Buss
Scribe notes by: Christian Woods
Date: April 30, 2014

1. Factoring Quadratic Polynomials over $\mathbb{Z}_p$

For this problem, we will be given a degree two polynomial $f(x)$ over $\mathbb{Z}_p$.

The goal is to factor $f(x)$ and find its roots, if it has any.

Suppose, without loss of generality, that f is monic so that $f(x) = x^2 + ax + b$ for some $a, b \in \mathbb{Z}_p$ (otherwise, we can multiply by the inverse of the leading coefficient without changing the roots).

If f has a double root then it must be $\frac{a}{2}$, i.e. $2^{-1}a$. Therefore, to ensure that 2 is invertible we assume that p is an odd prime.

We are able to prove the following.

Theorem. *There is a randomized, expected polynomial time algorithm that, given $f(x)$ as above, finds the roots of $f(x)$ if there are any.*

Proof. First we present the algorithm: (next page)

Input: p an odd prime, $f(x)$ a monic polynomial of degree 2
Output: the roots of f, if they exist
Step (1)
if *constant term of* $f(x)$ *is 0* **then**
  return 0 and root of $\frac{f(x)}{x}$
Compute $f_1(x) = \gcd\left(f(x), x^{p-1} - 1\right)$ using the Euclidean algorithm
if $f_1(x) = 1$ **then**
  return "irreducible"
if *deg*$(f_1) = 1$ **then**
  return the root of $f_1(x)$ as a double root of f
Step (2)
while *unsuccessful* **do**
  Choose an element $d \in \mathbb{Z}_p$ at random
  Let $g(x) = \gcd\left(f(x), (x-d)^{\frac{p-1}{2}} - 1\right)$
  if *deg*$(g) = 1$ **then**
    return its root and the root of $\frac{f(x)}{g(x)}$
end

Algorithm 1: The factoring algorithm for quadratic polynomials over $\mathbb{Z}_p$

We now analyze the algorithm to show that it indeed finds the roots of f.

First, consider step (1). The polynomial $x^{p-1} - 1$ has $p-1$ roots in $GF(p) = (\mathbb{Z}_p, +_p, \cdot_p)$. Therefore it is a product of linear factors with distinct roots. So there are three cases. If $f(x)$ has no roots, then $f_1(x) = 1$. If $f(x)$ has the form $(x-\alpha)^2$ for some $\alpha \in \mathbb{Z}_p - \{0\}$, then $f_1(x) = (x-\alpha)$. Otherwise, if $f(x) = (x-\alpha_1)(x-\alpha_2)$ for distinct $\alpha_1, \alpha_2 \in \mathbb{Z}_p - \{0\}$, then $f_1(x) = f(x)$. Therefore, for all but the last case step (1) successfully returns the roots of f.

To explain step (2), let $y = x - d$ and $r = \frac{p-1}{2}$. The roots of $y^r - 1$ are simply the quadratic residues modulo p, so x will be a quadratic residue shifted by d.

In step (2), we look at $\gcd\left(f(y+d), y^r - 1\right)$. The step will succeed if $\alpha_1 + d$ is a quadratic residue but $\alpha_2 + d$ is not, or vice versa. We know that

$$(\alpha_1 + d)^r = \pm 1 \text{ and } (\alpha_2 + d)^r = \pm 1.$$

So for $d \neq -\alpha_2$ step (2) will succeed when

$$\left(\frac{\alpha_1 + d}{\alpha_2 + d}\right)^r \equiv -1 \mod p.$$

The mapping $d \mapsto \frac{\alpha_1+d}{\alpha_2+d}$ is a one-to-one mapping from $\mathbb{Z}_p^* - \{-\alpha_2\}$ to $\mathbb{Z}_p^* - \{1\}$. So $\left(\frac{\alpha_1+d}{\alpha_2+d}\right)^r \equiv -1 \mod p$ with probability approximately $\frac{1}{2}$ (since exactly half of the elements of $\mathbb{Z}_p$ are not quadratic residues).

Now we want to show that this algorithm indeed runs in expected polynomial time (i.e., its runtime is $(\log(p))^{O(1)}$). However, if we carry out the division of $x^{p-1}-1$ by $f(x)$ in the Euclidean algorithm by long division in step (1), this algorithm will take time $p^{O(1)}$. Recall that to apply the Euclidean algorithm to two polynomials g_1, g_2 with $\deg(g_1) \geq \deg(g_2)$, we iteratively compute

$$g_i(x) = g_{i-2}(x) \mod g_{i-1}(x),$$

i.e. the remainder after we divide g_{i-2} by g_{i-1}. So in reality we need only compute $x^{p-1}-1 \mod f(x)$, not $\frac{x^{p-1}-1}{f(x)}$. It suffices to compute $x^{p-1} \mod f(x)$.

We will use iterated powering to compute each of $x^{\lfloor \frac{p-1}{2^i} \rfloor}$ for $i = \log(p-1), \ldots, 2, 1, 0$, since

$$x^{\lfloor \frac{p-1}{2^{i-1}} \rfloor} \mod f(x) = \left(x^{\lfloor \frac{p-1}{2^i} \rfloor}\right)^2 \mod f(x)$$

or

$$x^{\lfloor \frac{p-1}{2^{i-1}} \rfloor} \mod f(x) = \left(x^{\lfloor \frac{p-1}{2^i} \rfloor}\right)^2 x \mod f(x)$$

depending on whether $\lfloor \frac{p-1}{2^{i-1}} \rfloor = 2\lfloor \frac{p-1}{2^i} \rfloor$ or $2\lfloor \frac{p-1}{2^i} \rfloor + 1$. All of the intermediate results are degree 1 polynomials, and therefore these steps can be computed in polynomial time. □

2. II. Factoring general degree polynomials over $\mathbb{Z}_p$

Theorem. *There is a randomized, expected polynomial time algorithm that, given prime power $q = p^n$ and a degree m polynomial $f(x)$ over $\mathbb{Z}_p$, produces all roots of $f(x)$ in $GF(q)$ if any exist and also a factorization of $f(x)$ into irreducible factors.*

Sketch of proof: The algorithm will be polynomial in $\log(q)$ and m. (i.e., has runtime $(\log(q)m)^{O(1)}$). We can manipulate members of $GF(p^m)$ by representing them as m-tuples of coefficients of powers of a root of unity of degree m.

For the moment, assume $m = n$. Then $f(x)$ factors into the product of linear factors over $GF(p^m)$.

It suffices to give an algorithm that outputs either a nontrivial factor of $f(x)$ or a root of $f(x)$. We present such an algorithm below.

Input: p an odd prime, $f(x)$ a monic polynomial
Output: the roots of f, if they exist
Step (1)
if *constant term of* $f(x)$ *is 0* **then**
| return 0 as a root and x as a factor
Compute $f_1(x) = \gcd\left(f(x), x^{p^m-1} - 1\right)$ using the Euclidean algorithm
if *deg*(f_1) <*deg*(f) **then**
| return $f_1(x)$ as a factor
Step (2)
while *unsuccessful* **do**
| Choose an element $d \in GF(p^m)$ at random
| Let $g(x) = \gcd\left(f(x), (x-d)^{\frac{p^m-1}{2}} - 1\right)$
| **if** $1 <$ *deg*$(g) <$ *deg*(f) **then**
| | return g as a factor of f
| **if** g *is linear* **then**
| | return its root as a root of f
end

Algorithm 2: The general factoring algorithm for polynomials over $\mathbb{Z}_p$

Let $r = \frac{p^m-1}{2}$. Now in step (2)

$$g(x) = \prod_{\alpha_i \text{ s.t. } (\alpha_i+d)^r \equiv 1 \mod p^m} (x - \alpha_i).$$

So it succeeds if $\left(\frac{\alpha_i+d}{\alpha_j+d}\right)^r \equiv -1 \mod p$ for any $i \neq j$. This happens with probability $\gtrsim \frac{1}{2}$.

Iterating the algorithm will give us all roots of $f(x)$ in $GF(p^m)$. By determining their minimal polynomials, we can find the irreducible factors of $f(x)$ over $\mathbb{Z}_p[x]$. □

References

[1] E. R. Berlekamp, "Factoring Polynomials Over Large Finite Fields", Mathematics of Computation 24, 111 (1970) 713-735.

[2] Rajeev Motwani and Prabhakar Raghavan. 1995. Randomized Algorithms. Cambridge University Press, New York, NY, USA.

[3] M.O. Rabin, "Probabilistic Algorithms in Finite Fields", SIAM J. Computing 9, 2 (1980) 273-280.