Scribe Notes for *Algorithmic Number Theory*
Class 18—June 11, 1998

Scribes: Nick Loehr, Lynn W. Jones, and Hussein Suleman

Abstract

In today's class, we continued our discussion of algorithms to find roots in finite fields. We completed discussion of Tonelli's algorithm with an example and then went on to discuss Cipolla's field theoretic method of finding square roots. Finally, we began looking at methods of finding generalized dth roots.

1 Tonelli's Algorithm

We first state the algorithm and then solve two problems using it.

Tonelli (a)

let $q-1 = 2^s t$, where t is odd.
choose $z \in \mathbb{F}_q^*$ at random.
$g \leftarrow z^t$.
if $g^{2^{s-1}} = 1$ then fail.
for $i \leftarrow 0$ to $s-1$ do
  if $(ag^{-e})^{(q-1)/2^{i+1}} \neq 1$ then $e \leftarrow e + 2^i$.
if e `mod` $2 = 1$ then fail.
$h \leftarrow ag^{-e}$.
$b \leftarrow g^{e/2} h^{(t+1)/2}$.
return b.

Example 1.1. Find the square root of $\overline{3}$ in the finite field $\mathbb{F}_{13}$.
Here $q = 13$ so $q-1 = 12 = 2^2 \cdot 3$. Thus we can set $s = 2$ and $t = 3$. We know that $a = \overline{3}$.
Choose $g = \overline{5}$ as a possible generator. Since $g^{2^{s-1}} = \overline{5}^2 \equiv \overline{-1} \bmod 13$, g is a suitable choice.
Now set $e = 0$. There are then two iterations of the `for` loop since $s = 2$.
Case $i = 0$: $(ag^{-0})^{12/2} = \overline{3}^6 = \overline{1} \Rightarrow e_0 = 0$. Thus e remains 0.
Case $i = 1$: $(ag^{-0})^{12/4} = \overline{3}^3 = \overline{1} \Rightarrow e_1 = 0$. Thus e remains 0.
$h = ag^{-0} = \overline{3}$.
$b = g^{e/2} h^{(t+1)/2} = \overline{1} \cdot \overline{3}^2 = \overline{9}$.
To verify the solution : $\overline{9}^2 = \overline{81} = \overline{3}$.

Example 1.2. Find the square root of $\overline{12}$ in the finite field $\mathbb{F}_{13}$.
Here $q = 13$ so $q-1 = 12 = 2^2 \cdot 3$. Thus we can set $s = 2$ and $t = 3$. We know that $a = \overline{12} = \overline{-1}$.
Choose $g = \overline{5}$ as a possible generator. Since $g^{2^{s-1}} = \overline{5}^2 \equiv \overline{-1} \bmod 13$, g is a suitable choice.
Now set $e = 0$. There are then two iterations of the `for` loop since $s = 2$.
Case $i = 0$: $(ag^{-0})^{12/2} = \overline{-1}^6 = \overline{1} \Rightarrow e_0 = 0$. Thus e remains 0.
Case $i = 1$: $(ag^{-0})^{12/4} = \overline{-1}^3 = \overline{-1} \Rightarrow e_1 = 1$. Thus $e = 0 + 2 = 2$.

$h = ag^{-e} = \overline{-1} \cdot \overline{5}^{-2} = \overline{-1} \cdot \overline{8}^2 = \overline{-1} \cdot \overline{-5}^2 = \overline{-1} \cdot \overline{25} = \overline{1}$.
$b = g^{e/2}h^2 = g = \overline{5}$.
To verify the solution : $\overline{5}^2 = \overline{25} = \overline{12}$.

Theorem 1.3. *(Theorem 7.1.3 in text)*
If a has a square root in $\mathbb{F}_q^*$ where q is an odd prime power then `Tonelli(a)` *returns a square root of a with probability $1/2$. Its time complexity is $O\left((\lg q)^4\right)$.*

This theorem is simply a formalization of Tonelli's algorithm. The time complexity is attributed to the exponentiation within the loop in the algorithm. The probability of success is $1/2$ because half of the elements are generators. The fact that it uses this nondeterministic method of finding a generator is the main drawback of the algorithm.

2 Square Roots: Field theoretic methods

2.1 Background

We can use the additional properties of fields to calculate square roots differently from the group theoretic methods. Cipolla's approach is to consider a degree 2 extension of $\mathbb{F}_q$ where q is odd:

$$\mathbb{F}_q \subseteq \mathbb{F}_{q^2}.$$

We need to find an irreducible polynomial in $\mathbb{F}_q$ to construct $\mathbb{F}_{q^2}$.

Now, suppose that $f(X) = X^2 + bX + c$ is a monic, irreducible polynomial over $\mathbb{F}_q$. Let x be a root of f in $\mathbb{F}_{q^2}$.

Then,

$$\begin{aligned}
x^2 + bx + c &= 0 \\
(x^2 + bx + c)^q &= 0^q \quad \texttt{(applying a Frobenius map)} \\
(x^2)^q + b^q x^q + c^q &= 0 \quad \texttt{(since the other terms equal 0 modulo } q\texttt{)} \\
(x^q)^2 + bx^q + c &= 0.
\end{aligned}$$

Thus, x^q is also a root of f.

Therefore, $f(X) = (X - x)(X - x^q)$, implying that $c = x^{q+1}$ and $b = -x - x^q$.

Cipolla noted that $\sqrt{c} = x^{(q+1)/2}$. In his approach, we want to find an irreducible polynomial where the constant is a and find the square root of this constant term in $\mathbb{F}_{q^2}$. This is then a square root of a in $\mathbb{F}_q$ as well, if it is in $\mathbb{F}_q$.

2.2 Cipolla's Algorithm

To use the Cipolla algorithm to find a square root for a given a, we need an irreducible polynomial of the form

$$f(X) = X^2 - tX + a.$$

We can randomly guess a $t \in \mathbb{F}_q$; a "good" guess for t will result in an irreducible f. We can use the quadratic formula,

$$X = \frac{t \pm \sqrt{t^2 - 4a}}{2},$$

and know that if $(t^2 - 4a)$ is not a square in $\mathbb{F}_q$, then f is irreducible.

Here is the algorithm:

Cipolla(a)

choose $t \in \mathbb{F}_q$, uniformly at random.
if $(t^2 - 4a)$ is a square in $\mathbb{F}_q$, then fail. `{f is not irreducible}`
$f \leftarrow X^2 - tX + a$.
$b \leftarrow X^{(q-1)/2} \bmod f$.
return b.

We may impose the additional test that if $b \notin \mathbb{F}_q$ then a does not have a square root in $\mathbb{F}_q$.

Since `Cipolla` does not have the for-loop that `Tonelli` has, it is faster and executes in $\mathrm{O}((\lg q)^3)$ bit operations.

Example 2.1. In $\mathbb{F}_{17}$, we have $q = 17$ and $q - 1 = 16 = 2^4$. Suppose we wish to find the square root of $a = \overline{8}$.
Choose $t = \overline{1}$. Then $t^2 - 4a = \overline{1}^2 - 4 * \overline{8} = \overline{1} - \overline{32} = \overline{-31} = \overline{3}$.
Use the Legendre symbol to determine if $\overline{3}$ a square in $\mathbb{F}_{17}$:

$$\left(\frac{3}{17}\right) = \left(\frac{17}{3}\right) = \left(\frac{2}{3}\right) = -1.$$

So $t^2 - 4a$ is not a square in $\mathbb{F}_{17}$.
$f = X^2 - X + \overline{8}$.
$b = X^9 \bmod (X^2 - X + \overline{8}) = \overline{12}$.
So, $\overline{12}$ is a square root of $\overline{8}$ in $\mathbb{F}_{17}$. We verify that $\overline{12} = \overline{-5}$ and $\overline{-5}^2 = \overline{25} = \overline{8}$.

Example 2.2. In $\mathbb{F}_{17}$, we have $q = 17$ and $q - 1 = 16 = 2^4$. Suppose we wish to find the square root of $a = \overline{3}$.
Choose $t = \overline{5}$. Then $t^2 - 4a = \overline{5}^2 - 4 * \overline{3} = \overline{25} - \overline{12} = \overline{13} = \overline{-4}$.
But $\overline{-4} = \overline{8}^2$, so fail.
Next, choose $t = \overline{7}$.
Then $t^2 - 4a = \overline{7}^2 - 4 * \overline{3} = \overline{49} - \overline{12} = \overline{3}$.
From the previous example, we know this t is good.
$f = X^2 - \overline{7}X + \overline{3}$.
$b = X^9 \bmod (X^2 - \overline{7}X + \overline{3}) = \overline{2}X + \overline{10}$.
So, b is a square root of $\overline{3}$ in $\mathbb{F}_{17^2}$. Unfortunately, this square root does not lie in $\mathbb{F}_{17}$.

2.3 Analysis of Cipolla's Algorithm

Consider the probability of randomly choosing an acceptable t in Cipolla's algorithm. The following lemma shows that this probability is roughly $1/2$ for large q.

Lemma 2.3. *(Lemma 7.2.1 in the text) Suppose the element a is a square in $\mathbb{F}_q^*$. If t is chosen at random from $\mathbb{F}_q$, then $t^2 - 4a$ is a non-square with probability $(q-1)/2q$.*

Proof. To determine how many such polynomials in $\mathbb{F}_q$ are irreducible, we can count the ones that are reducible and take the complement.

A reducible polynomial f in the form $X^2 - tX + a$ has a linear factor, $X - \alpha$ for some $\alpha \in \mathbb{F}_q$. Since $a \in \mathbb{F}_q^*$, α cannot be 0 (or a would be 0). Then $X^2 - tX + a = (X - \alpha)(X - a/\alpha)$. The elements in $\mathbb{F}_q$ for which $\alpha \neq \sqrt{a}$ are distinct factors of f.

There are $q - 3$ elements that are not square roots of a. For each one, we get a polynomial of the correct form, and each such polynomial is counted twice. So we get $(q-3)/2$ polynomials. We also have the two square roots of a, so there are $(q-3)/2 + 2 = (q+1)/2$ polynomials of the form $X^2 - tX + a$ that split over $\mathbb{F}_q$. There are q total polynomials. Taking the complement of our set shows that there are $(q-1)/2$ "good" choices for t. The probability that $t^2 - 4a$ is a non-square in $\mathbb{F}_q$ is $(q-1)/2q$. □

Theorem 2.4. *(Theorem 7.2.3 in the text) Cipolla's algorithm fails with probability* $1/2 + 1/2q$. *If it does not fail, then it returns a square root of its argument in a quadratic extension* $\mathbb{F}_{q^2}$ *of* $\mathbb{F}_q$. *Its time complexity is* $O((\lg q)^3)$.

3 Computing dth Roots in Finite Fields

Recall that Tonelli's algorithm used properties of the cyclic multiplicative group of $\mathbb{F}_q$ to compute square roots in $\mathbb{F}_q$. By generalizing the argument used to derive the algorithm for square roots, we obtain an algorithm that computes dth roots for any $d \geq 2$. As before, computing a dth root is quite easy if d is relatively prime to the order of the multiplicative group $\mathbb{F}_q^*$.

Theorem 3.1. *(Theorem 7.3.1) If d is relatively prime to $q - 1$, then we can compute dth roots in $\mathbb{F}_q^*$ in $O((\lg q)^3)$ bit operations.*

Proof. Using the extended Euclidean algorithm, we can find e such that

$$de \equiv 1 \pmod{q-1}$$

in $O((\lg q)^2)$ operations. Then a^e is the unique dth root of a in $\mathbb{F}_q$, since $(a^e)^d = a^{ed} = a^1 = a$. The exponentiation a^e requires $O((\lg q)^3)$ bit operations. □

Now suppose $\gcd(d, q-1) = k > 1$. To find a dth root of a, we can find a kth root of a, say b, and then find a (d/k)th root of b using Theorem 3.1. To find the kth root of a, we use one prime factor of k at a time. For example, to take a twelfth root of a, we first compute a square root of a, say a_2. Then take a square root of a_2, say a_4. Finally, take a cube root of a_4, which is a twelfth root of a. Abusing notation slightly, we can write

$$a^{1/12} = ((a^{1/2})^{1/2})^{1/3}.$$

Thus, it suffices to consider the problem of taking an rth root of a, where r is a prime dividing $q - 1$.

Adleman, Manders, and Miller generalized Tonelli's algorithm to cover this case. First, write $q - 1 = r^s t$, where $s \geq 1$ and r does not divide t. Since r is prime, r^s and t are relatively prime. By the Chinese Remainder Theorem,

$$\mathbb{F}_q^* \cong C_{r^s} \times C_t,$$

where C_{r^s} and C_t denote the cyclic subgroups of $\mathbb{F}_q^*$ of orders r^s and t, respectively. Let τ be the map that takes $x \in \mathbb{F}_q^*$ to $(x^t, x^{r^s}) \in C_{r^s} \times C_t$. It is easy to check that τ is a group isomorphism. Moreover, we can give an explicit formula for the inverse map τ^{-1}. First, use the extended Euclidean algorithm to obtain integers α and β such that

$$\alpha t + \beta r^s = 1.$$

Then we claim that, for $(x_r, x_t) \in C_{r^s} \times C_t$,

$$\tau^{-1}(x_r, x_t) = x_r^\alpha x_t^\beta \in \mathbb{F}_q^*.$$

To verify this claim, note that

$$\tau(x_r^\alpha x_t^\beta) = ((x_r^\alpha x_t^\beta)^t, (x_r^\alpha x_t^\beta)^{r^s}) = (x_r^{\alpha t}, x_t^{\beta r^s}) = (x_r^{1-\beta r^s}, x_t^{1-\alpha t}) = (x_r, x_t).$$

We can now outline the general strategy for computing an rth root of $x \in \mathbb{F}_q^*$. We first apply τ to x to obtain the pair (x_r, x_t). Next, find rth roots of x_r and x_t separately. Finding the root γ_t of x_t is easy, using Theorem 3.1. Finding the root γ_r of x_r is difficult but possible, as discussed below. Once we have these roots, we apply τ^{-1} to the pair (γ_t, γ_r) to obtain an rth root of x.

4 Next Time: Computing rth roots in C_{r^s}

Consider the problem of finding an rth root of $a \in C_{r^s}$, where a is a perfect rth power. Let g be any generator of C_{r^s}. Then $a = g^e$ for some integer e such that r divides e. Let the base r representation of e be

$$e = e_{s-1}r^{s-1} + e_{s-2}r^{s-2} + \cdots + e_1 r + e_0,$$

where $0 \le e_i < r$ for each i. Clearly, $e_0 = 0$, and $g^{e/r}$ is an rth root of a. Thus, if we can determine all of the e_i's, we can find the desired root. The method for determining the e_i's will be covered next time.

References

[1] E. Bach and J. Shallit, *Algorithmic Number Theory*, The MIT Press, Cambridge, Massachusetts, 1996.