Solving equations in $\mathbb{Z}_p$

CNT@SSHS #003

05/07/2019 (Tue)

4411 윤창기

Contents

- Tonelli - Shanks : Solving quadratic eqn.

- Cantor - Zassenhaus : Solving general eqns.

Tonelli - Shanks

Able to deal with classic NT tools

Invented by Alberto Tonelli, in 1891

Want to solve

- $x^2 \equiv n \pmod{p}$ in "polynomial" time. (For odd p, nonzero n)
- Finding one solution r is enough. The other one is $-r$.
- Hold on, does the solution exist?

Euler's criterion

- $x^2 \equiv n \pmod p$ has a solution, if and only if $n^{\frac{p-1}{2}} \equiv 1 \pmod p$.
  - $\pmod p$ notation will be omitted for convenience.

- If $x^2 \equiv n$ and $n^{\frac{p-1}{2}} \equiv -1$, $x^{p-1} \equiv n^{\frac{p-1}{2}} \equiv -1$, which contradicts with FLT. So we immediately reject in this case.

- If $n^{\frac{p-1}{2}} \equiv 1$, there is a primitive root g s.t. $n \equiv g^{2k}$ for some k.
- So g^k would be the soln. of the quadratic equation.

- Then should we solve DLP now?

Hell nah.

- DLP is way too expensive!
- Our algorithm works in $O(\log^2 p)$, in average case.
  - Still polytime assuming GR..H…

An ansatz

- Let $p-1 = Q \cdot 2^S$, with odd Q.
- We know that there is a minimal k s.t. $n^{p-1} \equiv n^{\frac{p-1}{2}} \equiv \cdots \equiv n^{Q \cdot 2^k} \equiv 1$.
- What if $n^Q \equiv 1$, (i.e. $k=0$) in the extremely lucky case?
  - We put $r \equiv n^{\frac{Q+1}{2}}$, which gives $r^2 \equiv n$ and the algorithm terminates.

An ansatz

- Needless to say, k is nonzero in general case.
  But we still use $r \equiv n^{\frac{Q+1}{2}}$ as a 'pseudo-solution'!

- The 'pseudo-equation' is $r^2 \equiv nt$ with $t \equiv n^Q$.

- Our goal is achieving $t = 1$, maintaining the relation $r^2 \equiv nt$ through the iteration.

Inspecting t

- t is a 2^{S-1}-th root of 1.
  - $t^{2^{S-1}} \equiv n^{Q \cdot 2^{S-1}} \equiv n^{\frac{p-1}{2}} \equiv 1$.

- In firm, $M = S$ is a kind of "strict upper bound" for "order of t"; there is minimal i s.t. $t^{2^i} \equiv 1$, guaranteed that $i < M$.

- Assume we can generate $t' \equiv tb^2$ with smaller M, for appropriate b.

  The pseudo-equation still holds for $r' = rb$.
  Then M will drop to 1 ($i = 0$), after $O(\log p)$ times of loop.

Reducing M

- We will show that there is a b which makes $M = i$.
- Let z be a quadratic irresidue modulo p.
- Then $b = z^{Q \cdot 2^{M-i-1}}$ goes well with $t' = tb^2$, because $t'^{2^{i-1}} \equiv t^{2^{i-1}} z^{Q2^{M-1}} \equiv (-1) \cdot (-1) \equiv 1$. (for $M = S$)

Reducing M

- In general, we need to prepare c s.t. $c^{2^{M-1}} \equiv -1.$ and $c = z^{Q2^{S-M}}$ is enough!
- Initially $M = S$, so $c = z^Q$.
- If we change M to $M' = i$, $c' = c^{2^{M-i}} = z^{Q2^{S-M}2^{M-i}} = z^{Q2^{S-i}}$ plays the same role.
- Note that $b = z^{Q2^{M-i-1}} = \sqrt{c}$.

Complete algorithm

1. Test if n is a quadratic residue, with Euler's criterion. (Accept $n = 0$ here)
2. Find a quadratic irresidue z.
3. Initially, $M = S$, $c = z^Q$, $t = n^Q$, and $r = n^{\frac{Q+1}{2}}$.

   1. Terminate if $t = 1$.
   2. Find minimal i s.t. $t^{2^i} \equiv 1$. Acquire $b = c^{Q2^{M-i-1}}$.
   3. Put $M' \leftarrow i$, $c \leftarrow b^2$, $t \leftarrow tb^2$, and $r \leftarrow rb$.

Complexity analysis

1. Test if n is a quadratic residue, with Euler's criterion. (Accept $n = 0$ here)
   - $O(\log p)$.
2. Find a quadratic irresidue z.
3. Initially, $M = S$, $c = z^Q$, $t = n^Q$, and $r = n^{\frac{Q+1}{2}}$.

   1. Terminate if $t = 1$.
   2. Find minimal i s.t. $t^{2^i} \equiv 1$. Acquire $b = c^{Q2^{M-i-1}}$.
   3. Put $M' \leftarrow i$, $c \leftarrow b^2$, $t \leftarrow tb^2$, and $r \leftarrow rb$.

Complexity analysis

1. Test if n is a quadratic residue, with Euler's criterion. (Accept $n = 0$ here)
2. Find a quadratic irresidue z.
   Possible with 2 attempts of Euler's criterion **in average**.
   Polytime in worst - case is guaranteed with **Generalized Riemann Hypothesis**. (Bach, 1990)

3. Initially, $M = S$, $c = z^Q$, $t = n^Q$, and $r = n^{\frac{Q+1}{2}}$.

   1. Terminate if $t = 1$.
   2. Find minimal i s.t. $t^{2^i} \equiv 1$. Acquire $b = c^{Q2^{M-i-1}}$.
   3. Put $M' \leftarrow i$, $c \leftarrow b^2$, $t \leftarrow tb^2$, and $r \leftarrow rb$.

Complexity analysis

1. Test if n is a quadratic residue, with Euler's criterion. (Accept $n = 0$ here)
2. Find a quadratic irresidue z.
3. Initially, $M = S$, $c = z^Q$, $t = n^Q$, and $r = n^{\frac{Q+1}{2}}$. $O(S)$ loops

   1. Terminate if $t = 1$.
   2. Find minimal i s.t. $t^{2^i} \equiv 1$. Acquire $b = c^{Q2^{M-i-1}}$. $O(S)$ multiplication
   3. Put $M' \leftarrow i$, $c \leftarrow b^2$, $t \leftarrow tb^2$, and $r \leftarrow rb$.

Complexity analysis

- Overall complexity : $O(\log p + S^2) = O(\log^2 p)$ in average.
- In many cases, S is way smaller than $\log p$.

- Assuming GRH, finding a quadratic irresidue is possible in $O(\log^3 p)$, which is not so practical compared to the random approach.

Hensel's lifting

- If we know the solution of $x^2 \equiv a \pmod{p^k}$, we can obtain the solution of $x^2 \equiv a\ (mod\ p^{k+1})$ as well.

$x^* = x + p^k t$ for unknown t.

$$(x^*)^2 \equiv x^2 + 2p^k xt + p^{2k}t^2 \equiv n + p^k u + 2p^k xt \pmod{p^{k+1}}$$

$$u + 2xt \equiv 0 \pmod{p} \Longrightarrow t \equiv -2^{-1}ux^{-1} \pmod{p}$$

Example

$x^2 \equiv 86 \pmod{97}$.

$86^{48} \equiv 1 \pmod{97}$, so 86 is a quadratic residue.

My 'random guess' to find z was 53 and 41, and I got 41 as z.

$97 = 3 \times 2^5 + 1 \Longrightarrow Q = 3,\ S = 5.$

Initial variables:

$M = 5$, $t = 86^3 \equiv 27$, $c = 41^3 \equiv 51$, $r = 86^{\frac{3+1}{2}} = 24$.

Example

$M = 5$, $t = 27$, $c = 51$, $r = 24$.

$27^{2^3} \equiv -1$, so $i = 4$.
$b = z^{Q2^{S-i}}$ is desired, and we know that $b = c^{2^{M-i-1}}$.
$M - i - 1 = 0$, so $b = c = 51$.

Now
$M' = i = 4$, $c' = b^2 = 79$, $t' = 27 \times 51^2 = 96$, $r' = rb = 24 \times 51 = 60$.

Example

$M = 4$, $t = 96$, $c = 79$, $r = 60$.

$96^{2^0} \equiv -1$, so $i = 1$.
$b = z^{Q2^{S-i}}$ is desired, and we know that $b = c^{2^{M-i-1}}$.
$M - i - 1 = 2$, so $b = c^4 = 22$.

Now
$M' = i = 1$, $c' = b^2 = 79$, $t' = 96 \times 22^2 = 1$, $r' = rb = 60 \times 22 = 59$.

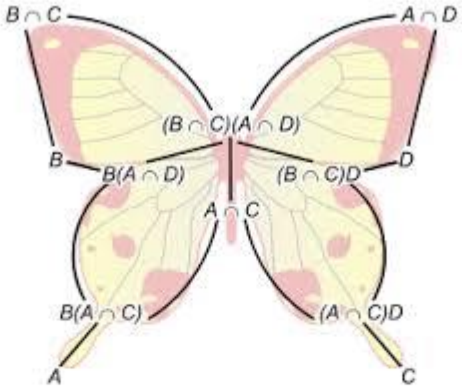
Example

Since $t = 1$, we stop the iteration and return $r = 59$.

$59^2 = 86$ holds, so algorithm was run successfully :)

Exercise

Project Euler #216, #437



Cantor - Zassenhaus

Sur-nonfiction ‘factoring’ algorithm 🤣

Invented in 1981

Our goal

- Input : $f \in \mathbb{Z}_p[x]$.
- Output : $f = g_1 g_2 g_3 \cdots g_m$
- Complexity : $O\left((n \log p)^c\right)$ in random

What can we do?

- Addition / Subtraction
- Multiplication / Division / Mod / GCD
- Everything is quadratic in naïve, linearithmic with FFT. (GCD requires $O(n \log^2 n)$
- They are all "polytime"s after all :)

The overall algorithm

1. Extract the square – free part of f (why?)
2. Run a **Distinct – Degree Factorization** algorithm, which returns the list of factors classified in degrees.
3. Factorize each degree-wise factor in randomized manner.

Step 1. Extracting the square-free part

- The simplest part!
- $g = \gcd(f, f')$
- f/g is the square - free part of f.

* For an irreducible polynomial h, $h^m \parallel f \implies h^{m-1} \parallel f'$.

Step 1. Extracting the square-free part

- Then, why should we treat the square-free polynomial?

- Irreducible polynomial ≈ prime
  - gcd≠1 ⇔ multiple

- For an irreducible h, $\mathbb{Z}_p[x]/h(x) \approx \mathbb{F}_{p^d}$.

Step 2. Distinct Degree Factorization (DDF)

- Given a square-free polynomial f, with degree n, return a list:

$$L = \{g_1, g_2, \cdots g_n\}$$

- g_i's are product of distinct irreducible factors with degree i.

Step 2. Distinct Degree Factorization (DDF)

An important lemma:

- Let $R_p(d)$ be a product of all irreducible polynomials in $\mathbb{Z}_p$ with degree d. Then

$$\prod_{d|m} R_p(d) = x^{p^m} - x.$$

Step 2. Distinct Degree Factorization (DDF)

DDF procedure

for $i = 1 \dots m$

$$g_i = \gcd(x^{p^i} - x, f)$$
$$f = f / g_i$$

Step 2. Distinct Degree Factorization (DDF)

DDF procedure

for $i = 1 \ldots m$

$$g_i = \gcd(x^{p^i} - x, f)$$

$$f = f / g_i$$

??? : Degree of x^{p^i} is exponential!!

Step 2. Distinct Degree Factorization (DDF)

Modified DDF procedure

for $i = 1 \dots m$

$$g_i = \gcd((x^{p^i} \bmod f) - x, f)$$
$$f = f / g_i$$

Step 3. Equal degree factorization

Interpreting the DDF result:

- If $g_n \neq 1$, f is irreducible.
- Else, factorize all g_is, with $\deg g_i > i$.

Factorizing a "chunk" consisted of **equal degree**, **distinct** polynomials?

Step 3. Equal degree factorization

Let g be the “chunk” defined in former slide. $n \coloneqq \deg g$.

Now we generate a polynomial $a(x)$ randomly, with $\deg a < n$.

We cross our finger and check $h = \gcd(a, g)$.

If h is nontrivial(neither 1, nor g),
Hooray! Factorize h and g/h recursively.

Step 3. Equal degree factorization

Assume $h = 1$. ($h \neq g$ since $\deg h \leq \deg a < \deg g$)

Then, $a(x)$ is an element of

$$\mathbb{Z}_p^*[x]/g(x).$$

Algebra alert!

It isn't difficult to imagine $\mathbb{Z}_p^*[x]/g(x)$.
It is a set of all polynomials $\bmod g(x)$ but coprime to $g(x)$.

Chinese remainder theorem is still alive and well:

If $g(x) = k_1(x)k_2(x)\cdots k_t(x)$, ($k_i(x)$ are pairwise coprime)

$$\mathbb{Z}_p[x]/g(x) \approx \mathbb{Z}_p[x]/k_1(x) \times \mathbb{Z}_p[x]/k_2(x) \times \cdots \times \mathbb{Z}_p[x]/k_t(x).$$

A GENUINE algebra alert!

Fact 1.

For an irreducible polynomial $f(x)$ with degree d,

$$\mathbb{Z}_p[x]/f(x) \approx \mathbb{F}_{p^d}.$$

Note that $\mathbb{F}_{p^d}$ is a field with p^d elements, should be distinguished with $\mathbb{Z}_{p^d}$.
Remark : $\mathbb{Z}_8$ is not a field!

A GENUINE algebra alert!

Fact 2. (Fermat's little theorem)

For a field $\mathbb{K}$ and $a \in \mathbb{K}$,

$$a^{|\mathbb{K}|-1} = 1.$$

Proof : Analogous to proof of the classic FLT.

i.e. $a(x)^{p^d-1} = 1$ for all $a(x) \in \mathbb{Z}_p[x]/f(x)$.

A GENUINE algebra alert!

Fact 3. (Lagrange's theorem)

For a field $\mathbb{K}$, and $p(x) \in \mathbb{K}[x]$, $p(x) = 0$ accepts at most $\deg p$ solutions in $\mathbb{K}$.

- Lemma (Fact 4.) For a field $\mathbb{K}$, $\mathbb{K}[x]$ is a Unique Factorization Domain.
  ($p(x) \in \mathbb{K}[x]$ accepts a unique factorization)

- The proof after the lemma is analogous to proof of the classic one.

Step 3. Equal degree factorization

To recall our problem, we focus on the fact that

"$a(x)$ is an element of $\mathbb{Z}_p^*[x]/g(x)$".

According to the Chinese Remaindering Theorem,

$a(x)$ can be written in the context of "CRT coordinate" –

$$a(x) = [a_1(x), a_2(x), \cdots, a_t(x)]$$

With $g(x) = h_1 h_2 \cdots h_t$, and $a_i(x) \in \mathbb{Z}_p^*[x]/h_i(x) \approx \mathbb{F}_{p^d}^*$.

Step 3. Equal degree factorization

Now we take the $\frac{p^d-1}{2}$-th power on $a(x)$. More precisely, we consider

$$b(x) = a(x)^{\frac{p^d-1}{2}} + 1.$$

Then each $b_i(x) \in \mathbb{F}_{p^d}$ becomes $a_i(x)^{\frac{p^d-1}{2}} + 1$ parallelly.

Step 3. Equal degree factorization

Then each $b_i(x) \in \mathbb{F}_{p^d}$ becomes $a_i(x)^{\frac{p^d-1}{2}} + 1$ parallelly.

Lemma. For all i, $a_i(x)^{\frac{p^d-1}{2}} = \pm 1$.

pf) $\forall a \in \mathbb{F}_{p^d}$, $a^{\frac{p^d-1}{2}}$ is a soln. of the eqn. $x^2 - 1 = 0$.
By Lag's theorem, there are no solutions of former equation except ± 1.

In the same manner, we know that $\mathbb{F}_{p^d}$ is **halved** by the value of its $\frac{p^d-1}{2}$-th power.

Step 3. Equal degree factorization

Hence, for a randomly chosen polynomial $a(x)$, Each $b_i(x)$ is 0 in probability $\frac{1}{2}$.

Theorem*. For a randomly chosen polynomial $a(x)$,

$a(x)^{\frac{p^d-1}{2}}+1$ has a common factor with $g(x)$ w/ probability $1-\frac{1}{2^t}$.

Step 3. Equal degree factorization

Caution：

If all $b_i(x)$'s are -1, it means $b(x) = 0$, which gives a trivial factor $g(x)$.

Theorem. For a randomly chosen polynomial $a(x)$,

$a(x)^{\frac{p^d-1}{2}} + 1$ has a common **nontrivial** factor with $g(x)$,
w/ probability $1 - \frac{2}{2^t} = 1 - \frac{1}{2^{t-1}}$.

Summary

1. Extract the square – free part.
   - $O(n \log p + GCD)$
2. Run DDF.
   - $O(n \log p)$ multiplications / modular ops.

3. Randomly factorize the chunks.
   - About $O(n \log n \log p)$ multiplications / modular ops?
   - It's a poly after all :)

With optimized polynomial ops, the expected complexity is

$$O\left(n^2 \log n \log\log n\,(\log n + \log p)\right).$$

Apdx : Description on $\mathbb{F}_{p^d}$

Let $f(x) = x^d + f_{d-1}x^{d-1} + f_{d-2}x^{d-2} + \cdots + f_0$ be an irreducible.

Every element $p(x) \in \mathbb{Z}_p[x]/f(x)$ can be written as a vector –

$$
\begin{aligned}
&1 = (0, 0, \cdots, 0, 1) \\
&x = (0, 0, \cdots, 1, 0) \\
&x^{d-1} = (1, 0, \cdots, 0, 0), \\
&p(x) = (p_{d-1}, p_{d-2}, \cdots, p_1, p_0),
\end{aligned}
$$

And

$$x^d = -(f_{d-1}, f_{d-2}, \cdots, f_0).$$

In fact $\mathbb{F}_{p^d}$ **is** a good vector space,
with a multiplication operator between the elements.

Apdx. Generating an irreducible polynomial.

It is important to pick an irreducible polynomial,
because it's the only method to implement $\mathbb{F}_{p^d}$ in fact.

A **very** important lemma:

- Let $R_p(d)$ be a product of all irreducible polynomials in $\mathbb{Z}_p$ with degree d. Then

$$\prod_{d|m} R_p(d) = x^{p^m} - x.$$

Apdx. Generating an irreducible polynomial.

Inspection on degree

- $I(d) :=$ # of irreducible polynomials in $\mathbb{Z}_p$ with degree d.
- $\deg R_p(d) = dI(d)$

$$p^m = \sum_{d|m} dI(d)$$

$$dI(d) = \sum_{e|d} p^e \mu\left(\frac{d}{e}\right) \quad (\because \text{ Mobius inversion})$$

Apdx. Generating an irreducible polynomial.

$$dI(d) = \sum_{e|d} p^e \mu\left(\frac{d}{e}\right) \text{ (}\because\text{ Mobius inversion)}$$

$$\therefore I(d) = \frac{p^d}{d} + O(\sqrt{p^d})$$

If we randomly pick a polynomial with degree d,
it is irreducible with probability $\approx \frac{1}{d}$.

Apdx. Generating an irreducible polynomial.

If we randomly pick a polynomial with degree d,
it is irreducible with probability $\approx \frac{1}{d}$.

The irreducibility testing can be run deterministically with DDF.

If we repeat Generating – DDF procedure $5d$ times,
it gives an irreducible polynomial w/ probability $\approx 1 - \left(1 - \frac{1}{d}\right)^{5d} \approx 1 - \frac{1}{e^5} \approx 0.9933.$

Berlekamp’s algorithm

Slightly different to the Berlekamp - Massey algorithm

Invented in 1967

Berlekamp vs Cantor - Zassenhaus

- Both are randomized algorithm for factorization

- Time complexity：

$$T_B = O(n^\omega + n \log n \log\log n \log p)$$
$$T_{CZ} = O(n^2 \log n \log\log n \,(\log n + \log p))$$

- Space complexity：$O(n^2)$ in B, $O(n)$ in CZ.

Berlekamp's algorithm : Overview

1. Extract the square - free part.

2. Find the Berlekamp - sub*algebra*.

3. Run randomized factorization several times.

No DDF!

Step 2. Finding the Berlekamp subalg.

- Step 1 is identical to CZ.

According to the CRT,

$$R = \mathbb{Z}_p[x]/f(x) \approx \mathbb{Z}_p[x]/f_1(x) \times \mathbb{Z}_p[x]/f_2(x) \times \cdots \times \mathbb{Z}_p[x]/f_t(x).$$

In the **ring** R, we define the Frobenius map:

$$T : x \mapsto x^p.$$

Note that T is an identity map in $\mathbb{Z}_p$.

Step 2. Finding the Berlekamp subalg.

Frobenius map is linear:

$$(a+b)^p = a^p + b^p.$$
$$(ca)^p = c^p a^p = ca^p. \, (c \in \mathbb{Z}_p)$$

So, for $w(x) = w_0 + w_1 x^1 + \cdots + w_{n-1} x^{n-1} \in R$, $(n \coloneqq deg\, w)$

$$T\big(w(x)\big) = w_0 + w_1 T(x) + w_2 T(x^2) + \cdots + w_{n-1} T(x^{n-1}).$$

Step 2. Finding the Berlekamp subalg.

Now, we find the polynomials satisfying $T(w) = w$.

Why?

Let $w = [w_1, w_2, \cdots, w_t]$. (CRT coordinate).

$T(w) = w$ is equivalent to the condition $T(w_i) = w_i$ for all i's:

But w_i is an element of the field $\mathbb{Z}_p[x]/f_i(x)$, so $T(w_i) = w_i$ **means** $\boldsymbol{w_i \in \mathbb{Z}_p}$**.**

So if we can generate w "**randomly**", then CRT-coordinate of $w^{\frac{p-1}{2}}$; will be confined in ± 1, which meets the idea of CZ.

Step 2. Finding the Berlekamp subalg.

The Berlekamp Subalgebra B

$B = \ker(T - I) = \{w : Tw = w\}$ forms a vector space as well.
And $w \in B$ can be constructed with the w_i's in $\mathbb{Z}_p$; which indicates

$$B \approx \mathbb{Z}_p \times \mathbb{Z}_p \times \cdots \mathbb{Z}_p = \mathbb{Z}_p^t.$$

So if we find the 'basis' of $B = \{b_1, b_2, \cdots, b_t\}$,
we can generate $w \in B$ in a faithful random sense:

$$w = c_1 b_1 + c_2 b_2 + \cdots c_t b_t.$$

Step 2. Finding the Berlekamp subalg.

We know that $\{1, x, \cdots, x^{n-1}\}$ spans R. So T accepts a matrix representation:

$$T(x^i) = \sum_j t_{ji} x^j \Longrightarrow T = (t_{ij}).$$

$(1, x, x^2, \cdots, x^{n-1})$ is a row vector:

$$r(x) = \begin{pmatrix} 1 & x & \cdots & x^{n-1} \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n-1} \end{pmatrix}.$$

$$T\big(r(x)\big) = \begin{pmatrix} T(1) & T(x) & \cdots & T(x^{n-1}) \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & x & \cdots & x^{n-1} \end{pmatrix} \cdot T \cdot \begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n-1} \end{pmatrix}.$$

Step 3. Run randomized factorization

After obtaining $B = \{b_i\}$,
a random linear combi. of b_i's will give a factor of f in high probability.

We can reuse B without re-computation:
the more iterations will give the more "different" factors.

We can decompose f into **"near"** the primitive factors by increasing the iterations.
Of course, re-applying Berlekamp's algorithm to the "small chunks" is good as well.

Further

Von zur Gathen & Shoup's algorithm (1992):

- Time : $T_{GS} = O(n^2 \log^2 n \log\log n + n \log n \log\log n \log p)$.
- Space : $O(n^{1.5})$
- Intermediate complexity between CZ and Ber.
- Better actual performance in large n, p than both.