Computing zeta functions and L-functions
Lecture 2

Andrew V. Sutherland

June 26, 2019

CMI-HIMR Summer School in Computational Number Theory

Primer on fast finite field arithmetic

Represent $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$ by $[0, p-1] \subseteq \mathbb{Z}$ and $\mathbb{F}_{p^e} \simeq \mathbb{F}_p[x]/(f(x))$ by $\{g \in \mathbb{F}_p[x] : \deg g \le e\}$.
For simplicity, assume $q = p^e$ with $\log e = O(\log p)$ or $\log p = O(1)$.

operation	complexity $(n = \log q, m \ge \deg)$	key idea
$\alpha \pm \beta$	$O(n)$	none
$\alpha\beta$	$O(n \log n)$	HvdH FFT + Newton
α^{-1}	$O(n \log^2 n)$	fast xgcd
$\alpha_1^{-1}, \ldots, \alpha_m^{-1}$	$O(mn \log n)$ $(\log n \ll m)$	$(\alpha_1 \cdots \alpha_m)^{-1}$
α^m	$O(n \log n \log m)$	binary exp
gh	$O(mn \log(mn))$	HvdH FFT (conditional)
$g = qh + r$	$O(mn \log(mn))$	Newton
$d = sg + th$	$O(mn \log(mn) \log m)$	fast xgcd
$g(\alpha_1), \ldots, g(\alpha_m)$	$O(mn \log(mn) \log m)^*$	product tree
$g(\alpha_i) = \beta_i$	$O(mn \log(mn) \log m)$	stare at Lagrange
$\#\{\alpha : g(\alpha) = 0\}$	$O(mn^2 \log n)$ $(m \ll q)$	$\deg \gcd(g(x), x^q - x)$
$\alpha \in \{\alpha : g(\alpha) = 0\}$	$O(mn^2 \log n)$ **(expected)**	$\gcd(g(x), (x + \delta)^s - 1)$

Naïve point counting

To count points on a curve $X/\mathbb{F}_q$ defined by a (possibly singular) plane model $f(x,y,z)=0$:

1. Count non-singular affine points by counting roots of $g(y)=f(x_0,y,1)$ for each $x_0\in\mathbb{F}_q$.
2. Count singular or non-affine points (for suitable f there are $O(1)$ of these).

Example: $y^2=f(x)$. Count rational square roots of $g(y)=f(x_0,y)$ for $x_0\in\mathbb{F}_q$, then add 0,1,2 points at infinity depending on parity of $\deg f$ and squareness of $\mathrm{lc}(f)$.

Example: smooth $f(x,y,z)=0$. Count roots of $g(y)=f(x_0,y,1)$ for $x_0\in\mathbb{F}_q$, then count roots of $g(y)=f(1,y,0)$, and add 1 if $f(0,1,0)=0$.

Complexity is $O(q\log^2 q\log\log q)$.

Generic group algorithms

Assumptions:

1. Each group element is assigned a unique id in $\{0,1\}^n$ with $n = O(\log \#G)$.
2. We have a black box to compute group operations (ids go in, ids come out).
3. The black box has a "random" button that outputs the id of uniformly random $\alpha \in G$.

Examples of generic groups algorithms:

1. binary exponentiation (multiplication) via square-and-multiply (double-and-add)
2. fast order algorithms (compute $|\alpha|$ given a multiple N of $|\alpha|$ in quasi-linear time).
3. Pohlig-Hellman (reduce discrete log to prime order case)
4. Pollard-rho (low memory randomized algorithm for order and discrete log)
5. Structure computation in finite abelian p-groups
6. baby-steps giant-steps

Baby-steps giant-steps

BSGS: To compute $|\alpha|$ given an upper bound $|\alpha| \le M$, pick $r, s \approx \sqrt{M}$ such that $rs > M$.

1. Compute baby steps: $0\alpha, 1\alpha, 2\alpha, \ldots, (r-1)\alpha$ (store in lookup table).
2. Compute giant steps: $0\alpha, r\alpha, 2r\alpha, \ldots, (s-1)r\alpha$ (lookup as you go).
3. When we reach a collision $rj\alpha = i\alpha$ with $0 \le i < r$ and $0 \le j < s$, put $N = rj - i$.
4. Use a fast order algorithm to compute $|\alpha|$ given N (this involves factoring N).

Uses at most $2\sqrt{M}$ group operations.

- If you don't know M, start with $M = 4$ and double until you succeed, or go triangular.
- If inverses are cheap, negate baby steps and double giant step size.
- Batch group operations using baby/giant armies that march in parallel.
- Easily adapted to search an interval or arithmetic progression containing a multiple of $|\alpha|$.
- Can optimize for expected distribution of $|\alpha|$ in $[1, M]$, and you can start in the center.
- Complexity can be improved to $O(\sqrt{M}/\log\log M)$ using a primorial search.

Computing the group order

Let G be an abelian group of order $N \in [M_0, M_1]$ with $2M_0 > M_1$:

1. Compute a lower bound e on $\lambda(G) = \operatorname{lcm}\{|\alpha| : \alpha \in G\}$ by computing $|\alpha|$ for at least two random $\alpha \in G$ (with probability at least $1/\zeta(2) > 1/2$, two elements suffice).
2. If e has a unique multiple N in $[M_0, M_1]$ then return $N = |G|$.
3. Heuristically compute the structure of p-Sylow subgroups of G for $p|e$ less than $\sqrt{M_1}$.
4. Using e and the results of step 3, determine a known divisor M of $|G|$, and if M has a unique multiple N in $[M_0, M_1]$ return $N = |G|$, otherwise go to step 1.

The heuristic computation of the p-Sylow subgroup uses $\tilde{O}(\sqrt{p}\log|G|)$ GOPs and outputs the structure of a subgroup that is equal to the full p-Sylow subgroup probability greater than $1/2$.

We can exponentially amplify the probability of correctness in steps 1,3 via repetition so that with probability greater than $1/2$ at step 4 we have $M = |G|$ and the algorithm terminates.

Bottom line: we can rigorously compute $\#G$ using $\tilde{O}((M_1 - M_0)^{1/2} + (\#G)^{1/4})$ expected group operations.

Remark: For $X/\mathbb{F}_q$ of genus 1 with $q > 49$ one can avoid p-sylow structure computations completely using a generalization of Mestre's approach (via the quadratic twist). For $g > 1$ this is not possible.

Computing the order of $\mathrm{Jac}(X)(\mathbb{F}_q)$

Let $X/\mathbb{F}_q$ be a nice curve of genus g.

For $G = \mathrm{Jac}(X)(\mathbb{F}_q)$ the Hasse-Weil bounds imply $\#G \in [(\sqrt{q}-1)^{2g}, \sqrt{g}+1)^{2g}]$, so if

$$2(\sqrt{q}-1)^{2g} > (\sqrt{g}+1)^{2g},$$

we can rigorously compute $\#G$ in $\tilde{O}(q^{(g-1/2)/2} + q^{g/4}) = \tilde{O}(q^{(2g-1)/4})$ expected time.

[1]If X has no nontrivial quadratic twist we can instead compute $L_X(-1) = \#\ker(\pi+1)/\#\,\mathrm{Jac}(X)[2](\mathbb{F}_q)$.

Computing the order of $\mathrm{Jac}(X)(\mathbb{F}_q)$

Let $X/\mathbb{F}_q$ be a nice curve of genus g.

For $G = \mathrm{Jac}(X)(\mathbb{F}_q)$ the Hasse-Weil bounds imply $\#G \in [(\sqrt{q}-1)^{2g}, \sqrt{g}+1)^{2g}]$, so if

$$2(\sqrt{q}-1)^{2g} > (\sqrt{g}+1)^{2g},$$

we can rigorously compute $\#G$ in $\tilde{O}(q^{(g-1/2)/2} + q^{g/4}) = \tilde{O}(q^{(2g-1)/4})$ expected time.

For $g = 1, 2, 3, 4, \ldots$ we need $q \geq 32, 131, 293, 529, \ldots, O(g^2)$.

For $g > 2$ we can improve this by first computing $\#X(\mathbb{F}_q), \ldots, \#X(\mathbb{F}_{q^{\lfloor (g-1)/2 \rfloor}})$. This improves the complexity to $\tilde{O}(q^{(2g-1-\lceil g/2 \rceil)/4})$, which is $\tilde{O}(q)$ for $g = 3$.

For $g \leq 3$ and $q \geq 1600$ we can determine the L-polynomial $L_X(T)$ by computing $L_X(1) = \#\,\mathrm{Jac}(X)(\mathbb{F}_q)$ and $L_X(-1) = \#\,\mathrm{Jac}(\tilde{X})(\mathbb{F}_q)$.[1]

[1] If X has no nontrivial quadratic twist we can instead compute $L_X(-1) = \#\ker(\pi + 1)/\#\,\mathrm{Jac}(X)[2](\mathbb{F}_q)$.

Schoof's algorithm

Let $E/\mathbb{F}_q$ be an elliptic curve $y^2 = f(x)$ with $q = p^e$ odd.
Then $\#E(\mathbb{F}_q) = q + 1 - t$, where $|t| \leq 2\sqrt{q}$.
Let $n = \log q$ and fix $L \sim \frac{n}{2}$ such that $\prod_{\ell \leq L} > 4\sqrt{q}$ (for simplicity, let us exclude $\ell = p$)

Our plan is to compute $t_\ell := t \bmod \ell$ for primes $\ell \leq L$, then recover t via the CRT.
We will compute the characteristic polynomial of π_E acting on $E[\ell]$, which has the form

$$T^2 - t_\ell T + q_\ell \in (\mathbb{Z}/\ell\mathbb{Z})[T],$$

where $q_\ell = q \bmod \ell$. As elements of $\operatorname{End}(E[\ell])$, we have the identity

$$\pi_\ell^2 - t_\ell \pi_\ell + q_\ell = 0. \tag{1}$$

where π_ℓ is the restriction of π_E to $E[\ell]$. It suffices to compute $\pi_\ell, \pi_\ell^2, q_\ell \in \operatorname{End}(E[\ell])$ (which we know a priori) and then try every possibility for t_ℓ (or better, use a BSGS search).

In order to do this, we need to explicitly represent elements of $\operatorname{End}(E[\ell])$ and understand how to compute the ring operations in $\operatorname{End}(E[\ell])$.

Representing elements of $\mathrm{End}(E[\ell])$

We can compute $t_2 \equiv \#E[2](\mathbb{F}_q) \bmod 2$ by computing $\#\{\alpha \in \mathbb{F}_q : f(\alpha) = 0\}$, so let $\ell > 2$.

Let $h \in \mathbb{F}_q[x]$ be the ℓth division polynomial of E; so $P = (x_0, y_0) \in E[\ell] \Leftrightarrow h(x_0) = 0$.
We can compute $h \in \mathbb{F}_q[x]$ in $\tilde{O}(\ell^2 \log q)$ time using well known recurrence relations.

We represent elements of $\mathrm{End}(E[\ell])$ using rational maps defined by polynomials that we may view as elements of the ring

$$R_\ell := \frac{\mathbb{F}_q[x,y]}{(h(x), y^2 - f(x))}$$

where $y^2 = f(x) = x^3 + Ax + B$ is a Weierstrass equation for $E/\mathbb{F}_q$. We have

$$\begin{aligned}\pi_\ell &= \big(x^q \bmod h(x),\; y^q \bmod (h(x), y^2 - f(x))\big)\\ &= \Big(x^q \bmod h(x),\; \Big(f(x)^{(q-1)/2} \bmod h(x)\Big)\, y\Big).\end{aligned}$$

and we also note that

$$1_\ell = (x \bmod h(x),\; \big(1 \bmod h(x)\big)\, y)$$

Ring operations in $\mathrm{End}(E[\ell])$

Observe that π_ℓ and 1_ℓ both have the form $(a(x), b(x)y)$. Let $\alpha_1 = (a_1(x), b_1(x)y)$ and $\alpha_2 = (a_2(x), b_2(x)y)$ be nonzero elements of $\mathrm{End}(E[\ell])$.

Multiplication: $\alpha_1 \circ \alpha_2 = (a_1(a_2(x)),\ b_1(a_2(x))b_2(x)\, y)$.

Addition: $\alpha_3 = \alpha_1 + \alpha_2$ has the form $(a_3(x), b_3(x)y)$ with

$$\begin{aligned} a_3 &= r^2 f - a_1 - a_2, \\ b_3 &= r(a_1 - a_3) - b_1, \end{aligned}$$

where $r = (b_1 - b_2)/(a_2 - a_1)$ when $a_1 \neq a_2$ and $r = (3a_1 + A)/(2b_1 f)$ when $\alpha_1 = \alpha_2$.

What happens if we hit a zero divisor in the computation of the denominator of r?

Ring operations in $\mathrm{End}(E[\ell])$

Observe that π_ℓ and 1_ℓ both have the form $(a(x), b(x)y)$. Let $\alpha_1 = (a_1(x), b_1(x)y)$ and $\alpha_2 = (a_2(x), b_2(x)y)$ be nonzero elements of $\mathrm{End}(E[\ell])$.

Multiplication: $\alpha_1 \circ \alpha_2 = (a_1(a_2(x)),\ b_1(a_2(x))b_2(x)\, y)$.

Addition: $\alpha_3 = \alpha_1 + \alpha_2$ has the form $(a_3(x), b_3(x)y)$ with

$$\begin{aligned} a_3 &= r^2 f - a_1 - a_2, \\ b_3 &= r(a_1 - a_3) - b_1, \end{aligned}$$

where $r = (b_1 - b_2)/(a_2 - a_1)$ when $a_1 \neq a_2$ and $r = (3a_1 + A)/(2b_1 f)$ when $\alpha_1 = \alpha_2$.

What happens if we hit a zero divisor in the computation of the denominator of r?
Life is good! We can use this zero divisor to obtain a nontrivial factor g of $h \in \mathbb{F}_q[x]$.
We then replace h by g or h/g and restart the computation working in a smaller ring.

CoCalc Worksheet

Computational complexity

With the implementation described above, Schoof's algorithm runs in $O(n^5 \log\log n)$ time, where $n = \log q$.

Elkies: If E admits an ℓ-isogeny φ, we can replace the division polynomial with the polynomial whose roots are the x-coordinates of the affine $\overline{\mathbb{F}}_q$-points in the kernel of φ. This reduces the degree of h from $(\ell^2-1)/2$ to $(\ell-1)/2$.

Heuristically this gives a complexity to $\tilde{O}(n^4)$, but we cannot prove this, not even under GRH. But we can prove it is true on average.

Pila showed how to generalize Schoof's algorithm to higher dimension abelian varieties to obtain a complexity that is polynomial in $\log q$ but doubly exponential in g.

Abelard, building on work of Gaudry, Harley, and Schöst improves Pila's result to $(\log q)^{O(g)}$ for hyperelliptic curves. Practical implementations exist for $g = 2$ with complexity $\tilde{O}((\log q)^8)$

For RM hyperelliptic curves we can do better: $\tilde{O}((\log q)^5)$ genus 2 and $\tilde{O}((\log q)^6)$ genus 3.

Lemma

Suppose $\pi_\ell^2(P) - c\pi_\ell(P) + q_\ell P = 0$ *for a nonzero* $P \in E[\ell](\overline{\mathbb{F}}_q)$ *and* $c \in \mathbb{Z}$*. Then* $t_\ell \equiv c \bmod \ell$*.*

Proof: Subtracting the characteristic polynomial of π_ℓ yields $(t_\ell - c)\pi_\ell(P) = 0$, and $\pi_\ell(E)$ has prime order ℓ, so $t_\ell \equiv c \bmod \ell$.