Computing L-series of low genus curves

Andrew V. Sutherland

Massachusetts Institute of Technology

SIAM Conference on Applied Algebraic Geometry

August 2, 2013

joint work with David Harvey

The problem

Given a smooth projective curve $X/\mathbb{Q}$ and a bound N, we wish to compute $L_p(T)$ for all primes $p \le N$ where X has good reduction.

Here $L_p(T)$ is the L-*polynomial* of the reduction $X_p/\mathbb{F}_p$ of X at p. It is an integer polynomial of degree $2g$ that satisfies:

- $L(X;s) = \prod_p \boldsymbol{L_p}(p^{-s})^{-1}$;
- $Z(X_p;T) = \exp\left(\sum_{n=1}^{\infty} \#X_p(\mathbb{F}_{p^n})T^n/n\right) = \frac{\boldsymbol{L_p}(T)}{(1-T)(1-pT)}$;
- $\chi(X_p;T) = T^{2g}\boldsymbol{L_p}(T^{-1})$.

Applications: computing L-functions and Sato-Tate distributions.

Some existing solutions

Four methods were analyzed in [Kedlaya-S, 2008]:

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p),\ldots,X_p(\mathbb{F}_{p^g})$	$p\log^{1+\epsilon}p$	$p^2\log^{1+\epsilon}p$	$p^3\log^{1+\epsilon}p$

Some existing solutions

Four methods were analyzed in [Kedlaya-S, 2008]:

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p),\ldots,X_p(\mathbb{F}_{p^g})$	$p\log^{1+\epsilon}p$	$p^2\log^{1+\epsilon}p$	$p^3\log^{1+\epsilon}p$
generic group algorithms[1]	$p^{1/4}\log^{1+\epsilon}p$	$p^{3/4}\log^{1+\epsilon}p$	$p^{5/4}\log^{1+\epsilon}$

[1] one uses $L_p(1) = \#\mathrm{Jac}(X_p)$ and $L_p(-1) = \#\mathrm{Jac}(\tilde{X}_p)$.

Some existing solutions

Four methods were analyzed in [Kedlaya-S, 2008]:

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p), \ldots, X_p(\mathbb{F}_{p^g})$	$p \log^{1+\epsilon} p$	$p^2 \log^{1+\epsilon} p$	$p^3 \log^{1+\epsilon} p$
generic group algorithms[1]	$p^{1/4} \log^{1+\epsilon} p$	$p^{3/4} \log^{1+\epsilon} p$	$p^{5/4} \log^{1+\epsilon}$
p-adic cohomology (Kedlaya-Harvey)	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$

[1] one uses $L_p(1) = \#\mathrm{Jac}(X_p)$ and $L_p(-1) = \#\mathrm{Jac}(\tilde{X}_p)$.

Some existing solutions

Four methods were analyzed in [Kedlaya-S, 2008]:

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p),\ldots,X_p(\mathbb{F}_{p^g})$	$p\log^{1+\epsilon} p$	$p^2\log^{1+\epsilon} p$	$p^3\log^{1+\epsilon} p$
generic group algorithms[1]	$p^{1/4}\log^{1+\epsilon} p$	$p^{3/4}\log^{1+\epsilon} p$	$p^{5/4}\log^{1+\epsilon}$
p-adic cohomology (Kedlaya-Harvey)	$p^{1/2}\log^{2+\epsilon} p$	$p^{1/2}\log^{2+\epsilon} p$	$p^{1/2}\log^{2+\epsilon} p$
ℓ-adic CRT (Schoof-Pila)[2]	$\log^{5+\epsilon} p$	$\log^{8+\epsilon} p$	$\log^{14?+\epsilon} p$

[1] one uses $L_p(1) = \#\mathrm{Jac}(X_p)$ and $L_p(-1) = \#\mathrm{Jac}(\tilde{X}_p)$.

[2] SEA has a heuristic complexity of $O(\log^{4+\epsilon} p)$ in genus 1, but is still not worth using.

Some existing solutions

Four methods were analyzed in [Kedlaya-S, 2008]:

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p), \ldots, X_p(\mathbb{F}_{p^g})$	$p \log^{1+\epsilon} p$	$p^2 \log^{1+\epsilon} p$	$p^3 \log^{1+\epsilon} p$
generic group algorithms[1]	$p^{1/4} \log^{1+\epsilon} p$	$p^{3/4} \log^{1+\epsilon} p$	$p^{5/4} \log^{1+\epsilon}$
p-adic cohomology (Kedlaya-Harvey)	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$
ℓ-adic CRT (Schoof-Pila)[2]	$\log^{5+\epsilon} p$	$\log^{8+\epsilon} p$	$\log^{14?+\epsilon} p$

Within the feasible range of $p \leq N$, it *never* makes sense to use the polynomial-time algorithm that is asymptotically the best choice.

For practical purposes, group algorithms work best in genus 1 and 2, and a combination of group algorithms and p-adic methods works best in genus 3.

At least this was the situation until the fall of last year. . .

[1] one uses $L_p(1) = \#\mathrm{Jac}(X_p)$ and $L_p(-1) = \#\mathrm{Jac}(\tilde{X}_p)$.

[2] SEA has a heuristic complexity of $O(\log^{4+\epsilon} p)$ in genus 1, but is still not worth using.

An average polynomial-time algorithm

All of the methods above perform separate computations for each prime p. But we want to compute $L_p(T)$ for all good $p \leq N$ using reductions of *the same curve* in each case.

Is their a way to take advantage of this fact?

An average polynomial-time algorithm

All of the methods above perform separate computations for each prime p. But we want to compute $L_p(T)$ for all good $p \le N$ using reductions of *the same curve* in each case.

Is their a way to take advantage of this fact?

Theorem (Harvey, 2012)

Let $y^2 = f(x)$ be a hyperelliptic curve over $\mathbb{Q}$, with $\deg f = 2g + 1$ odd. There is an algorithm to compute $L_p(T)$ for all good primes $p \le N$ in

$$O\big(g^{8+\epsilon} N \log^{3+\epsilon} N\big)$$

time, using $O(g^3 N \log^2 N)$ space (assuming g and $\|f\|$ are suitably bounded).

This yields an average time of $O\big(g^{8+\epsilon} \log^{4+\epsilon} p\big)$ per prime $p \le N$.

But how practical is it for feasible values of N?

The Hasse-Witt matrix

Harvey's algorithm uses the same basic approach as Kedlaya's algorithm: compute the action of Frobenius on the Monsky-Washnitser cohomology to sufficient p-adic precision. For a suitable choice of basis, this action can be described by a matrix $A_p \in \mathbb{Z}_p^{2g\times 2g}$ that satisfies

$$A_p \equiv \left(\begin{array}{c|c} W_p & 0 \\ \hline 0 & 0 \end{array}\right) \bmod p,$$

where $W_p \in (\mathbb{Z}/p\mathbb{Z})^{g\times g}$ is the *Hasse-Witt matrix* of X.

For hyperelliptic curves $y^2 = f(x)$, the matrix W_p is given by

$$W_p = \begin{pmatrix} c_{p-1} & c_{p-2} & \cdots & c_{p-g} \\ c_{2p-1} & c_{2p-2} & \cdots & c_{2p-g} \\ \vdots & \vdots & \vdots & \vdots \\ c_{gp-1} & c_{gp-2} & \cdots & c_{gp-g} \end{pmatrix},$$

where c_k is the coefficient of x^k in the expansion of $f(x)^{(p-1)/2}$ modulo p.

Computing the Hasse-Witt matrix

Theorem (Harvey-S, 2013)

Let $X/\mathbb{Q}$ be a hyperelliptic curve. The matrices W_p can be computed for all good primes $p \le N$ in $O(g^{2+\omega} N \log^{3+\epsilon} N)$ time using $O(gN)$ space.
(assuming g and $\|f\|$ are suitably bounded).

Computing the Hasse-Witt matrix

Theorem (Harvey-S, 2013)

Let $X/\mathbb{Q}$ be a hyperelliptic curve. The matrices W_p can be computed for all good primes $p \leq N$ in $O(g^{2+\omega} N \log^{3+\epsilon} N)$ time using $O(gN)$ space. (assuming g and $\|f\|$ are suitably bounded).

For primes p of good reduction the identity

$$L_p(T) \equiv \det(I - TW_p) \bmod p$$

determines $O(1)$ possibilities for $L_p(T)$ in genus 2, and $O(p^{1/2})$ in genus 3.

In the latter case, these can be distinguished in $O(p^{1/4} \log^{1+\epsilon} p)$ time, which is negligible for the feasible range of $p \leq N$.

In any case, W_p determines the trace of Frobenius for all sufficiently large p. When approximating $L(X; s)$, only the trace is needed for $p \geq N^{1/2}$.

The algorithm in genus 1

Let $X/\mathbb{Q}$ be the elliptic curve $y^2 = f(x) = x^3 + ax + b$.
Then $L_p(T) = pT^2 - t_pT + 1$, where $t_p \equiv c_{p-1} \bmod p$ is the trace of Frobenius.

We wish to compute the coefficient c_{p-1} of x^{p-1} in $f(x)^{(p-1)/2} \bmod p$ for $p \le N$.
Equivalently, the coefficient of x^{2n} in $f(x)^n \bmod 2n + 1$ for $n \le (N-1)/2$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$ and reduce the x^{2n} coefficient of $f(x)^n$ modulo $2n + 1$.

The algorithm in genus 1

Let $X/\mathbb{Q}$ be the elliptic curve $y^2 = f(x) = x^3 + ax + b$.
Then $L_p(T) = pT^2 - t_pT + 1$, where $t_p \equiv c_{p-1} \bmod p$ is the trace of Frobenius.

We wish to compute the coefficient c_{p-1} of x^{p-1} in $f(x)^{(p-1)/2} \bmod p$ for $p \leq N$.
Equivalently, the coefficient of x^{2n} in $f(x)^n \bmod 2n+1$ for $n \leq (N-1)/2$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$ and reduce the x^{2n} coefficient of $f(x)^n$ modulo $2n+1$.

But the polynomials f^n are huge, each has $\Omega(n^2)$ bits.
This approach would require $\Omega(N^3)$ time and $\Omega(N^2)$ space.

So this is a terrible idea...

The algorithm in genus 1

Let $X/\mathbb{Q}$ be the elliptic curve $y^2 = f(x) = x^3 + ax + b$.
Then $L_p(T) = pT^2 - t_pT + 1$, where $t_p \equiv c_{p-1} \bmod p$ is the trace of Frobenius.

We wish to compute the coefficient c_{p-1} of x^{p-1} in $f(x)^{(p-1)/2} \bmod p$ for $p \leq N$.
Equivalently, the coefficient of x^{2n} in $f(x)^n \bmod 2n+1$ for $n \leq (N-1)/2$.

Naïve approach: iteratively compute $f, f^2, f^3, \ldots, f^{(N-1)/2}$ in $\mathbb{Z}[x]$ and reduce the x^{2n} coefficient of $f(x)^n$ modulo $2n+1$.

But the polynomials f^n are huge, each has $\Omega(n^2)$ bits.
This approach would require $\Omega(N^3)$ time and $\Omega(N^2)$ space.

So this is a terrible idea...

But we don't need all the coefficients of f^n, we only need one,
and we only need to know its value modulo $2n+1$.

A better approach

Let f_k^n denote the coefficient of x^k in $f(x)^n$.
Using $f^n = ff^{n-1}$ and $(f^n)' = nf'f^{n-1}$, one obtains the relations

$$\begin{aligned}
(n+2)f_{2n-2}^n &= n\left(2af_{2n-3}^{n-1} + 3bf_{2n-2}^{n-1}\right),\\
(2n-1)f_{2n-1}^n &= n\left(3f_{2n-4}^{n-1} + af_{2n-2}^{n-1}\right),\\
2(2n-1)bf_{2n}^n &= (n+1)af_{2n-4}^{n-1} + 3(2n-1)bf_{2n-3}^{n-1} - (n-1)a^2f_{2n-2}^{n-1}.
\end{aligned}$$

Letting $D_n = 2(n+2)(2n-1)b$ and

$$M_n = \begin{pmatrix} 0 & 4n(2n-1)ab & 6n(2n-1)b^2 \\ 6n(n+2)b & 0 & 2n(n+2)ab \\ (n+1)(n+2)a & 3(n+2)(2n-1)b & (1-n)(n+2)a^2 \end{pmatrix},$$

we can compute $v_n = (f_{2n-2}^n, f_{2n-1}^n, f_{2n}^n)$ from v_{n-1} via $v_n = v_{n-1}M_n^{\text{tr}}/D_n$.

A better approach

If we set $D_0 = 1$, $M_0 = \left(\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$, and define

$$\mathcal{M}_n = \prod_{i=0}^{n} M_i^{\mathrm{tr}} \qquad \text{and} \qquad \mathcal{D}_n = \prod_{i=0}^{n} D_i,$$

then we can compute v_n as the bottom row of $\mathcal{M}_n/\mathcal{D}_n$. Thus it suffices to compute the partial products $\mathcal{M}_n$ and $\mathcal{D}_n$ modulo $2n+1$.

A better approach

If we set $D_0 = 1$, $M_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, and define

$$\mathcal{M}_n = \prod_{i=0}^{n} M_i^{\mathrm{tr}} \qquad \text{and} \qquad \mathcal{D}_n = \prod_{i=0}^{n} D_i,$$

then we can compute v_n as the bottom row of $\mathcal{M}_n/\mathcal{D}_n$. Thus it suffices to compute the partial products $\mathcal{M}_n$ and $\mathcal{D}_n$ modulo $2n+1$.

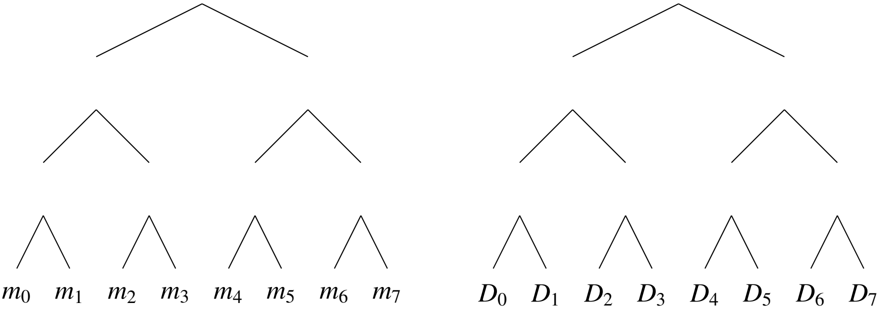
Considering just the $\mathcal{D}_n$, we want to compute

$$\begin{aligned} D_0D_1 &\bmod 3 \\ D_0D_1D_2 &\bmod 5 \\ D_0D_1D_2D_3 &\bmod 7 \\ &\vdots \\ D_0D_1D_2D_3\cdots D_{(N-1)/2} &\bmod N \end{aligned}$$

Doing this naïvely takes $O\left(N^{2+\epsilon}\right)$ time, but it can be done in $O\left(N^{1+\epsilon}\right)$ time.

Remainder trees

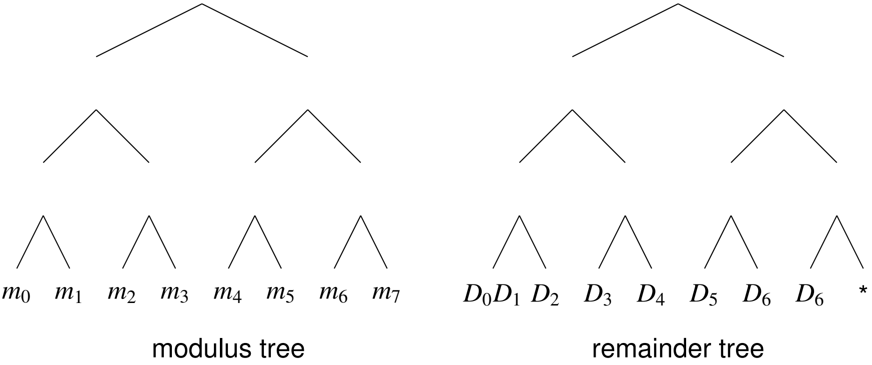
Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



modulus tree | remainder tree

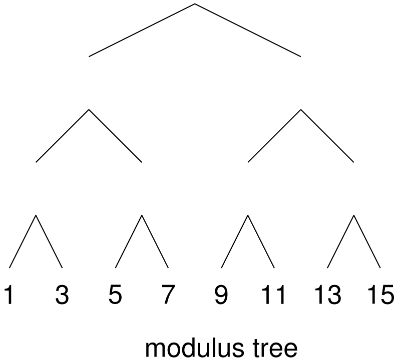
Remainder trees

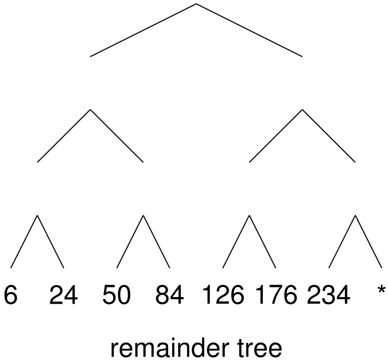
Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



Remainder trees

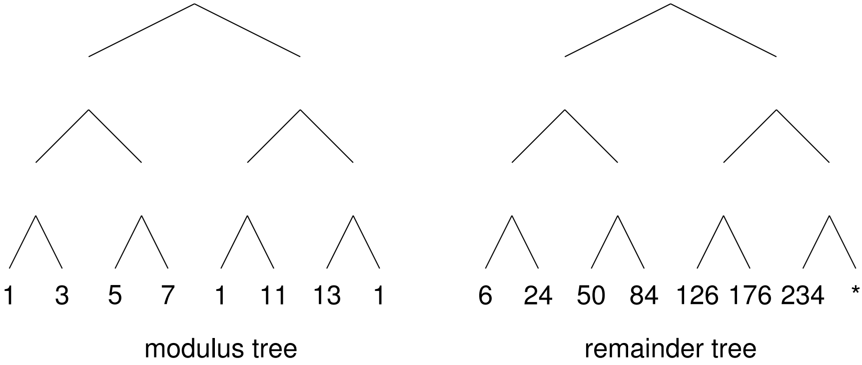
Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.





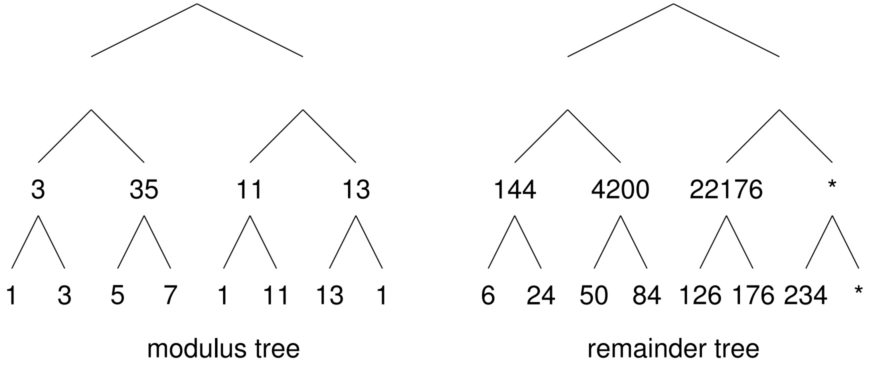
Remainder trees

Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



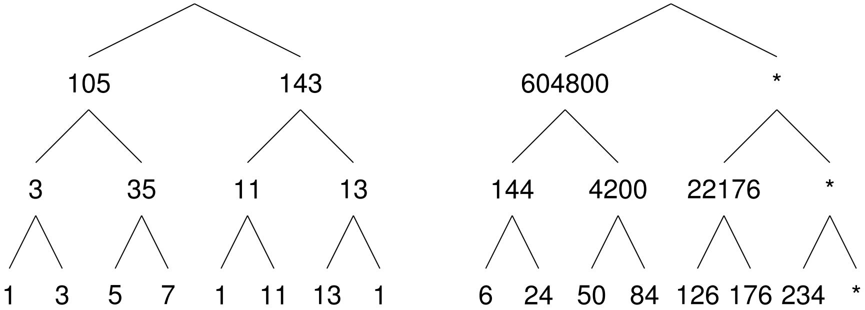
Remainder trees

Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



Remainder trees

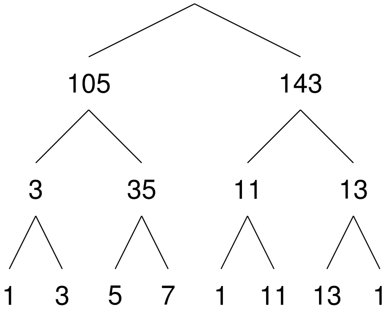
Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



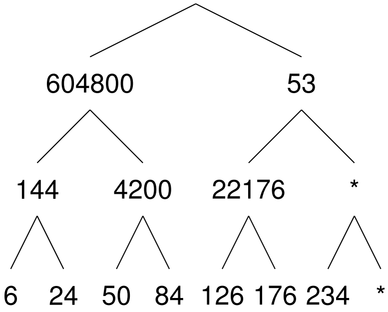
modulus tree | remainder tree

Remainder trees

Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0\leq i\leq n} D_n \bmod (2n+1)$ for $1 \leq n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



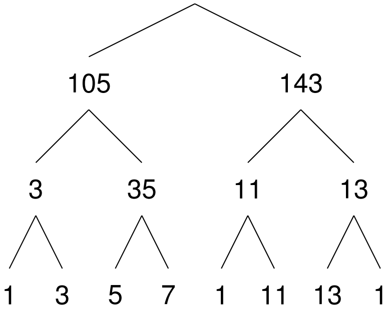
modulus tree



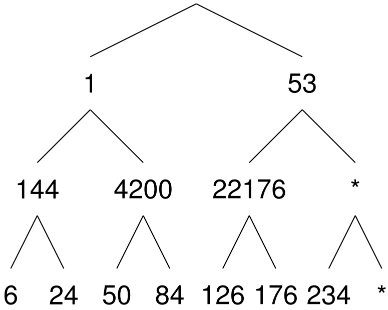
remainder tree

Remainder trees

Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



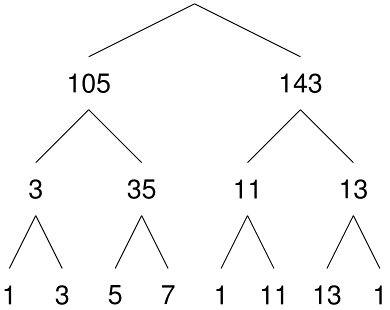
modulus tree



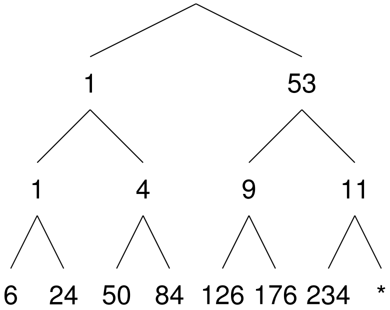
remainder tree

Remainder trees

Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



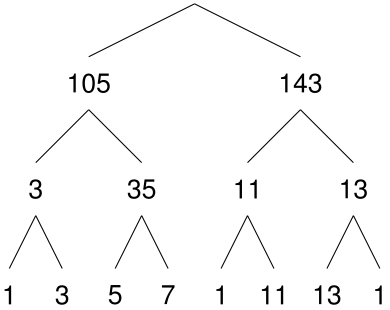
modulus tree



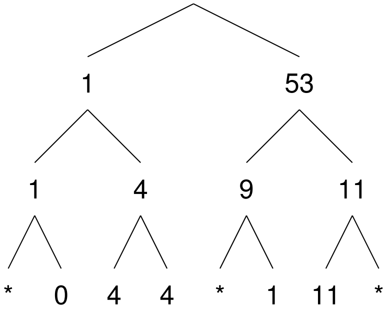
remainder tree

Remainder trees

Let X be the elliptic curve $y^2 = x^3 + x + 1$ (so $a = b = 1$).
Let us compute $\mathcal{D}_n = \prod_{0 \le i \le n} D_n \bmod (2n+1)$ for $1 \le n < 8$,
where $D_0 = 1$ and $D_i = 2(i+2)(2i-1)b$ for $i > 0$.



modulus tree



remainder tree

Remainder trees

1. Can be used over any ring (not necessarily commutative) to compute a sequence of partial products modulo a sequence of principal ideals (we use the ring $\mathbb{Z}^{g\times g}$ and work moduli p^g to avoid zero denominators).

Remainder trees

1. Can be used over any ring (not necessarily commutative) to compute a sequence of partial products modulo a sequence of principal ideals (we use the ring $\mathbb{Z}^{g \times g}$ and work moduli p^g to avoid zero denominators).

2. Provided multiplication and reduction of ring elements take quasi-linear time, the entire algorithm runs in quasi-linear time.

Remainder trees

1. Can be used over any ring (not necessarily commutative) to compute a sequence of partial products modulo a sequence of principal ideals (we use the ring $\mathbb{Z}^{g\times g}$ and work moduli p^g to avoid zero denominators).
2. Provided multiplication and reduction of ring elements take quasi-linear time, the entire algorithm runs in quasi-linear time.
3. One can reduce the space by a log factor (*without increasing the time*) using a forest of remainder trees and propagating results at the roots.

Remainder trees

1. Can be used over any ring (not necessarily commutative) to compute a sequence of partial products modulo a sequence of principal ideals (we use the ring $\mathbb{Z}^{g\times g}$ and work moduli p^g to avoid zero denominators).
2. Provided multiplication and reduction of ring elements take quasi-linear time, the entire algorithm runs in quasi-linear time.
3. One can reduce the space by a log factor (*without increasing the time*) using a forest of remainder trees and propagating results at the roots.
4. When many moduli are trivial, space can be further reduced (by another log factor in our setting).

Remainder trees

1. Can be used over any ring (not necessarily commutative) to compute a sequence of partial products modulo a sequence of principal ideals (we use the ring $\mathbb{Z}^{g\times g}$ and work moduli p^g to avoid zero denominators).
2. Provided multiplication and reduction of ring elements take quasi-linear time, the entire algorithm runs in quasi-linear time.
3. One can reduce the space by a log factor (*without increasing the time*) using a forest of remainder trees and propagating results at the roots.
4. When many moduli are trivial, space can be further reduced (by another log factor in our setting).
5. A time-space trade-off can be used to reduce space even more, but we do not need to do this.

Comparison

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p), \ldots, X_p(\mathbb{F}_{p^g})$	$p \log^{1+\epsilon} p$	$p^2 \log^{1+\epsilon} p$	$p^3 \log^{1+\epsilon} p$

Comparison

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p), \ldots, X_p(\mathbb{F}_{p^g})$	$p \log^{1+\epsilon} p$	$p^2 \log^{1+\epsilon} p$	$p^3 \log^{1+\epsilon} p$
generic group algorithms	$p^{1/4} \log^{1+\epsilon} p$	$p^{3/4} \log^{1+\epsilon} p$	$p^{5/4} \log^{1+\epsilon}$

Comparison

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p), \ldots, X_p(\mathbb{F}_{p^g})$	$p \log^{1+\epsilon} p$	$p^2 \log^{1+\epsilon} p$	$p^3 \log^{1+\epsilon} p$
generic group algorithms	$p^{1/4} \log^{1+\epsilon} p$	$p^{3/4} \log^{1+\epsilon} p$	$p^{5/4} \log^{1+\epsilon}$
p-adic cohomology (Kedlaya-Harvey)	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$

Comparison

	genus 1	genus 2	genus 3
enumerate $X_p(\mathbb{F}_p), \ldots, X_p(\mathbb{F}_{p^g})$	$p \log^{1+\epsilon} p$	$p^2 \log^{1+\epsilon} p$	$p^3 \log^{1+\epsilon} p$
generic group algorithms	$p^{1/4} \log^{1+\epsilon} p$	$p^{3/4} \log^{1+\epsilon} p$	$p^{5/4} \log^{1+\epsilon}$
p-adic cohomology (Kedlaya-Harvey)	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$	$p^{1/2} \log^{2+\epsilon} p$
ℓ-adic CRT (Schoof-Pila)	$\log^{5+\epsilon} p$	$\log^{8+\epsilon} p$	$\log^{14?+\epsilon} p$
Hasse-Witt matrices	$\log^{4+\epsilon} p$	$\log^{4+\epsilon} p$	$\log^{4+\epsilon} p +$ $p^{1/4} \log^{1+\epsilon} p$

In genus 2 the new algorithm already outperforms smalljac when $N > 2^{21}$.
The prospects in genus 3 look even better (work in progress).

Next steps: generalize to non-hyperelliptic curves of genus 3.