Harvey's Average Polynomial Time Algorithms

Joshua E. Hill

Department of Mathematics, University of California, Irvine

Math 239B Arithmetic Geometry
January 6 and 8, 2014
`http://bit.ly/198j5HY`

v1.01, compiled March 18, 2014

Talk Outline

Introduction Outline

1. Introduction
   - GOOOOOOOOOOOOOOOOOOOOOOOOOOOOALS!
   - Time Complexity Notes
2. The Sieve of Eratosthenes
3. Searching for Wilson Primes
4. Computing Zeta Functions of Arithmetic Schemes Modulo Many Primes
5. Conclusion

Subsection 1

GOOOOOOOOOOOOOOOOOOOOOOOOOOOOALS!

A Tale of Two Complexities

- Calculating the number of elements can be hard.
- We often don't have general algorithms that run in polynomial time (with respect to p).
- We have two basic classes of responses:
  - Make the problem much smaller (extra hypotheses that impose some nice structure.)
  - Make the problem much larger.

Say What!?!

Make the problem much...

Say What!?!

Make the problem much...

Larger

Make the problem much...

Larger

(and then hope for a reasonable amortized runtime.)

So it begins...

We'll look at a few examples:

- The Sieve of Eratosthenes
- Searching for Wilson Primes
- Calculate the zeta function for reductions of an arithmetic scheme mod all primes less than some bound.

Papiere, Bitte

We'll be exploring the following papers (the third provides the basic algorithm used in the second):

- Edgar Costa, Robert Gerbicz, and David Harvey, *A Search for Wilson Primes*.
- David Harvey, *Computing Zeta Functions of Arithmetic Schemes*

Subsection 2

Time Complexity Notes

Big-O Notation (and Family)

- We have two eventually positive real valued functions $A, B : \mathbb{N}^k \to \mathbb{R}$. Take $\mathbf{x}$ as an n-tuple, with $\mathbf{x} = (x_1, \ldots, x_n)$
- We'll write $|\mathbf{x}|_{\min} = \min_i x_i$.

Definition

$A(\mathbf{x}) = O(B(\mathbf{x}))$ if there exists a positive real constant C and an integer N so that if $|\mathbf{x}|_{\min} > N$ then $A(\mathbf{x}) \leq CB(\mathbf{x})$. (*i.e.* A is bounded above by B asymptotically.)

Definition

$A(\mathbf{x}) = o(B(\mathbf{x}))$ if for all positive real constants C there is an integer N so that if $|\mathbf{x}|_{\min} > N$ then $A(\mathbf{x}) \leq CB(\mathbf{x})$. (*i.e.* A is dominated by B asymptotically.)

Definition

An algorithm is considered polynomial time if it is time complexity $O(x^k)$ where k is a fixed positive integer and x is the input length.

Definition

An algorithm is considered exponential time if it is time complexity $O(2^{x^k})$ where k is a fixed positive integer, and x is the input length.

The Algorithm

- Determine the largest number you want to test, N.
- Make a bit-array (initialized to all 0s) of length N.
- Mark 1 as *not prime* (set the first entry to 1).
- Let $k = 2$
- Until k is larger than $\left\lfloor \sqrt{N} \right\rfloor$, do the following:
  - Mark every positive integer multiple of k greater than k and less than or equal to N as not prime (set their corresponding bit array entries to 1).
  - Let k be the next entry marked as prime.
- The values marked as prime are the primes less than or equal to N.

An Example of the Sieve

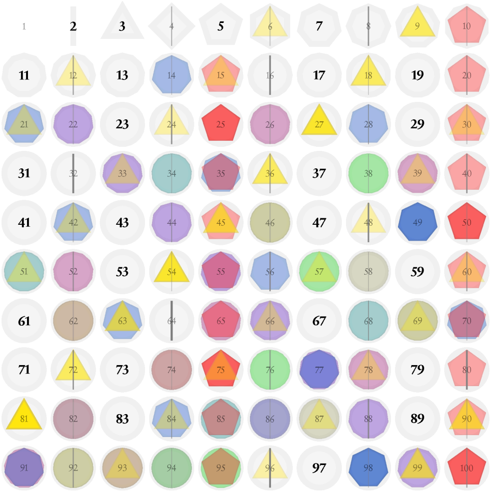

Figure : Sieve of Eratosthenes, $N = 100$

Computational Complexity

- k can only be a prime number.
- There are fewer than $\lfloor N/k \rfloor$ values excluded for each value of k.
- Each step is just an addition of a value of size $O(\log N)$.
- The total number of excluded values, d, is thus (by Mertens' theorem)

$$d \approx \sum_{\substack{k \leq \sqrt{N} \\ k \text{ prime}}} \lfloor N/k \rfloor = O(N \log \log N)$$

- We can then read out the primes by looking for 0 bits in the bitstring in $O(N)$.
- The runtime is thus $O(d \log N) = O(N \log N \log \log N)$.
- This algorithm requires $O(d \log N)$ storage.
- Even if we don't assume a RAM model (and instead use a Turing model) we can get a similar result using sorting.

A Note on $O(N)$

- $O(N) = O\left(e^{\log N}\right)$ is clearly exponential in $\log N$ (the size of N).
- $O(N \log N \log\log N)$ is exponential in the size of N.
- We got $\pi(N) \sim N/\log N$ primes from the algorithm.
- The amortized runtime per-prime is thus $O(\log^2 N \log\log N)$, which is polynomial in the input size.

A Note on $O(N)$

- $O(N) = O\left(e^{\log N}\right)$ is clearly exponential in $\log N$ (the size of N).
- $O(N \log N \log \log N)$ is exponential in the size of N.
- We got $\pi(N) \sim N / \log N$ primes from the algorithm.
- The amortized runtime per-prime is thus $O(\log^2 N \log \log N)$, which is polynomial in the input size.
- Jazz Hands!

Search for Wilson Primes Outline

Wilson Primes

- By Wilson's Theorem, we know that p is prime if and only if $(p-1)! \equiv -1 \pmod{p}$.
- For a prime p, define $w_p = ((p-1)! + 1)/p \pmod{p}$.

Definition

A Wilson Prime is a prime where $w_p \equiv 0 \pmod{p}$, or equivalently when $(p-1)! \equiv -1 \pmod{p^2}$.

- There are three known Wilson Primes: 5, 13, and 563.
- It is conjectured that there are an infinite number of Wilson Primes.

WILSON!!!

- In general, the tests to see if a prime p is a Wilson prime are exponential in the size of p.
- By using a dynamic programming technique called *memoization*, one can (in aggregate) make this calculation more efficient.
- Idea: As p varies, we repeat quite a lot of arithmetic in calculating $(p-1)!$.

I Wanted to be... A LUMBERJACK!

- We seek Wilson primes less than or equal to some fixed N.
- We first need to find all the primes up to N.
- We use the Sieve of Eratosthenes, which we have seen runs in $O(N \log N \log \log N)$.

Trees for the Forest: The Larch

- We'll break the interval $[1, N]$ into 2^i roughly equal intervals:

$$U_{i,j} = \left\{ k \in \mathbb{Z} \mid j\frac{N}{2^i} < k \leq (j+1)\frac{N}{2^i} \right\}$$

- The recurrence relation $U_{i,j} = U_{i+1,2j} \coprod U_{i+1,2j+1}$ provides a tree structure.
- Let $d = \lceil \log_2 N \rceil$. Note $\left| U_{d,j} \right|$ is either 0 or 1 for all j.

Trees for the Forest: The Pine

- Multiply together the elements in each set:

$$A_{i,j} = \prod_{k \in U_{i,j}} k$$

- By the recurrence relation for $U_{i,j}$ we get $A_{i,j} = A_{i+1,2j} \cdot A_{i+1,2j+1}$.
- $A_{d,j}$ is either 1 (when $U_{d,j} = \emptyset$) or k (when $U_{d,j} = \{k\}$).
- The $A_{i,j}$ product tree can be computed from bottom ($i = d$) to top ($i = 0$) using the above recurrence relation.
- Elements in the ith level are $O(2^{-i} N \log N)$ bits long.
- For fixed i, all the $A_{i,j}$ can be computed in work factor $2^i (2^{-i} N \log N) \log^{1+\epsilon} N$.
- There are $\log_2 N$ levels, so the cost for computing the $A_{i,j}$ tree is $N \log^{3+\epsilon} N$.

Trees for the Forest: The Redwood

- Multiply together the squares of the prime elements in each set:

$$S_{i,j} = \prod_{\substack{p \in U_{i,j} \\ p \text{ prime}}} p^2$$

- The characteristics of $S_{i,j}$ are similar to those of $A_{i,j}$.
- By the recurrence relation for $U_{i,j}$ we get $S_{i,j} = S_{i+1,2j} \cdot S_{i+1,2j+1}$.
- The $S_{i,j}$ product tree can be computed from bottom ($i = d$) to top ($i = 0$) using the above recurrence relation.
- Elements in the ith level are at most $O(2^{-i} N \log N)$ bits long.
- The cost for computing the $S_{i,j}$ tree is less than $N \log^{3+\epsilon} N$.

Trees for the Forest: The Sequoia

- Calculate factorial parts and reduce.

$$W_{i,j} = \prod_{0 \le r < j} A_{i,j} \pmod{S_{i,j}} = \left\lfloor j \frac{N}{2^i} \right\rfloor ! \pmod{S_{i,j}}$$

- By convention, $W_{0,0} = 1$.
- By definition, $W_{i+1,2j} = W_{i,j} \pmod{S_{i+1,2j+1}}$.
- We can also construct $W_{i+1,2j+1} = W_{i,j} \cdot A_{i+1,2j} \pmod{S_{i+1,2j+1}}$.
- We construct the $W_{i,j}$ tree from top to bottom, which also requires time $N \log^{3+\epsilon} N$.
- For prime $p \le N$, $j = \lceil 2^d p/N \rceil - 1$, then $U_{d,j} = \{p\}$, so $S_{d,j} = p^2$ and $W_{d,j} = (p-1)! \pmod{p^2} = w_p$.
- Wilson quotients are thus in the bottom of the tree.

Trouble at the Mill

- The algorithm runs in $N \log^{3+\epsilon} N$.
- It requires significant storage. There is a time/memory trade off that reduces the storage requirement.
- This algorithm is clearly exponential in the size of N.
- We got w_p for $\pi(N)$ total values, so the amortized cost, per prime, is asymptotically $\log^{4+\epsilon} N$, which is polynomial in N.

Trouble at the Mill

- The algorithm runs in $N \log^{3+\epsilon} N$.
- It requires significant storage. There is a time/memory trade off that reduces the storage requirement.
- This algorithm is clearly exponential in the size of N.
- We got w_p for $\pi(N)$ total values, so the amortized cost, per prime, is asymptotically $\log^{4+\epsilon} N$, which is polynomial in N.
- Jazz Hands!

Counting Points on Hyperelliptic Curves Outline

Phtttt! It's Fresh!

- Based on a paper that is currently in draft form (dated January 6th).
- Introduces a set of related algorithms:
  - Calculate the zeta function of the reduction of an arithmetic scheme, X, mod p (that is, $X_p = X \times_{\mathbb{Z}} \mathbb{Z}/p\mathbb{Z}$) in time complexity $p^{1/2} \log^{2+\epsilon} p$.
  - A somewhat slower variant with better space complexity.
  - An algorithm that finds all such X_p for all prime $p < N$ in time complexity $N \log^{3+\epsilon} N$.

The “Hypersurface in an Affine Torus” Case

- Let $n \geq 1$, $q = p^a$ with $\mathbb{P}^n_{\mathbb{F}_q}$ denote projective n-space over $\mathbb{F}_q$.
- Coordinates $x_0, \ldots, x_n$.
- $\mathbb{T}^n_{\mathbb{F}_q} \subset \mathbb{P}^n_{\mathbb{F}_q}$ is an affine torus ($x_0 \cdot x_1 \cdots x_n \neq 0$).
- Let $\bar{F} \in \mathbb{F}_q[x_0, \ldots, x_n]$ be a homogeneous polynomial of degree $d \geq 1$, with $p \nmid d$.
- X is the hypersurface cut out by $\bar{F}$.
- Fixed p case runs in $2^{8n^2+16n} n^{4n+4+\epsilon} (d+1)^{4n^2+7n+\epsilon} a^{4n+4+\epsilon} p^{1/2} \log^{2+\epsilon} p$.
- Outputs

$$Z_X(T) = \exp\left(\sum_{r\geq 1} \frac{\left|X\left(\mathbb{F}_{q^r}\right)\right|}{r} T^r\right)$$

The General Case

- The general case of the algorithm applies to any arithmetic scheme, X (that is, X is a scheme of finite type over $\mathbb{Z}$).
- Think: the object is locally defined by polynomial equations in finitely many variables over $\mathbb{Z}$.
- X can be covered as a union of open affines.
- We can recursively reduce to the case where X is the disjoint finite union of finitely generated spectra of $\mathbb{Z}$-algebras, V_i.
- We use an inclusion/exclusion trick (due to Wan) to calculate the zeta function of X in terms of the zeta functions a set of hypersurfaces.

Basis of the Algorithm

- Follows the same general idea as Lauder-Wan (2008).
- Uses a "trace formula" that expresses $Z_X(T)$ in terms of an arbitrary p-adic lift of $\bar{F}$.
- Not the same trace formula as in Lauder-Wan.

Objects

- $\mathbb{Z}_q$ is the ring of Witt vectors over $\mathbb{F}_q$
- Note: $\mathbb{Z}_q/p\mathbb{Z}_q \cong \mathbb{F}_q$.
- We do arithmetic within the ring using finite approximations, $\mathbb{Z}_q/p^\lambda\mathbb{Z}_q$, with $\lambda \geq 1$.
- To represent these elements, take an arbitrary lift of $\bar{f} \in (\mathbb{Z}/p^\lambda\mathbb{Z})[t]$, a fixed monic irreducible polynomial of degree a. Call this lift f, .
- We then have $\mathbb{Z}_q/p^\lambda\mathbb{Z}_q \cong (\mathbb{Z}/p^\lambda\mathbb{Z})[t]/\bar{f}$.
- Each element in $\mathbb{Z}_q/p^\lambda\mathbb{Z}_q$ is thus represented as a polynomial of degree less than a.

Trace Formula Preliminaries

- Let $\phi : \mathbb{F}_q \to \mathbb{F}_q$ be the p-Frobenius map ($\alpha \mapsto \alpha^p$).
- This map uniquely lifts to $\mathbb{Z}_q$.
- Take $\psi : \mathbb{Z}_q[x] \to \mathbb{Z}_q[x]$ defined as $\psi(G) = \sum_u \phi^{-1}(G_{pu})x^u$.
- For $k \geq 1$ and $H \in \mathbb{Z}_q[x]_k$ define $T_H : \mathbb{Z}_q[x] \to \mathbb{Z}_q[x]$ as the multiplication-by-H operator, that is $T_H(G) = HG$
- Define $A_H = \psi \circ T_{H^{p-1}}$.
- We'll represent operators with respect to a basis of monomials.
- The submodule of degree k monomials is spanned by x^u with $u \in B_k$.

The Trace Formula

- Let r, λ and τ be positive integers satisfying

$$\tau \geq \frac{\lambda}{(p-1)ar}$$

- Let $F \in \mathbb{Z}_q[x]_d$ be any lift of $\bar{F}$.
- We then have

$$\left|X(\mathbb{F}_{q^r})\right| = (q^r-1)^n \sum_{s=0}^{\lambda+\tau-1} \alpha_s \mathrm{tr}(A_{F^s}^{ar}) \pmod{p^\lambda}$$

$$\alpha_s = (-1)^s \sum_{t=0}^{\tau-1} \binom{-\lambda}{t} \binom{\lambda}{s-t}$$

- Note that for sufficiently large p, $\tau = 1$ works.

Insert Tab A into Slot B

- The problem then reduces to finding $A_{F^s}^a$.
- Let $u, v \in B_{ds}$.
- For $F \in \mathbb{Z}_q[x]_d$, the matrix of $A_{F^s}^a$ on $\mathbb{Z}_q[x]_{ds}$ with respect to the basis B_{ds} is
$$\phi^{a-1}(M_{s,p}) \cdots \phi(M_{s,p}) M_{s,p}$$
- $(M_{s,p})_{v,u} = (F^{(p-1)s})_{pv-u}$
- ϕ is taken to act componentwise on matrices.

The Many-Prime Case

- First, we enumerate all the primes less than N (again using the Sieve of Eratosthenes)
- Uses a tree structure (an accumulating remainder tree) as with the Wilson Prime algorithm.
- Applies this idea to a recurrence formula that defines a set of matrices used in the trace calculation.
- The many-prime case has time-complexity $2^{8n^2+16n} n^{4n+6+\epsilon} (d+1)^{4n^2+7n+\epsilon} N \log^2 N \log^{1+\epsilon}(N\|F\|)$.

There Sure Are Quite A Few "Slot B"s

- We want to perform this calculation across many p's.
- The basic mechanism is reasonably general:

Theorem

Let $m \geq 1$, $\beta \geq 1$, $\mu \geq 1$, $N \geq 2$, and $\rho \in \mathbb{R}$ with $\rho > 1$. Given $E_1, \cdots, E_{N-1}$ (with entries bounded suitably by ρ), a set of $m \times m$ matrices with entries in $\mathbb{Z}[k]/k^\beta$. We can then compute $\prod_{i=1}^{p-1} E_i \pmod{p^\mu}$ for all primes $p < N$ in time $m^3\beta(\mu + \rho)N \log N \log^{1+\epsilon}(\beta\mu\rho N)$.

Same Old Story, Not Much to Say

- Happily, this is very similar to the instance where we searched for Wilson Primes.
- $\ell = \lceil \log_2 N \rceil$
- These all form binary trees in exactly the same way as before.
- $S_{i,t}$ (loosely) partitions the integers $0, \ldots, N-1$ into 2^i sets of roughly equal size.
- $P_{i,t}$ contains the primes of $S_{i,t}$.
- The modulus tree is defined $M_{i,t} = \prod_{p \in P_{i,t}} p^{\mu}$.
- The value tree $V_{i,t} = \prod_{j \in S_{i,t}} E_j$.
- The accumulating remainder tree is $A_{i,t} = V_{i,t-1} V_{i,t-2} \cdots V_{i,0} \pmod{M_{i,t}}$.
- This last tree is constructed using the recurrence relations $A_{i+1,2t} = A_{i,t} \pmod{M_{i+1,2t}}$ and $A_{i+1,2t+1} = V_{i+1,2t} A_{i,t} \pmod{M_{i+1,2t+1}}$.

Hearts are Broken, Everyday

- If we average this cost over all the primes less than N, we get a polynomial algorithm.
- This algorithm (not just the many-p case) is the fastest known algorithm of this type.

Section 5

Conclusion

Jazz Hands!

- This "amortized cost" style algorithm allows for use of a broader class of tools.
- For some styles of problem, we really do mainly care about the cost-per-result, rather than the cost of the entire operation.
- These algorithms are still "slow" with respect to input size.

Thank You!

Bibliography

- Edgar Costa, Robert Gerbicz, and David Harvey, *A Search for Wilson Primes*.
- David Harvey, *Computing Zeta Functions of Arithmetic Schemes*