Cours MPRI 2-12-2

Lecture 2/5: Sieving and other improvements

(lecturer for part 2/3): E. Thomé

Nov. 12th, 2012

Plan

QS

This lecture is mostly about QS, the quadratic sieve.

- QS is technology from the 1980's - 1990's.
- Superseded by NFS since circa 1995.
- Yet, QS is faster for factoring numbers below e.g. 120dd.

This not of merely historical value:

- QS embodies many of the state-of-the-art techniques still used nowadays.
- Stating these techniques in the QS context frees us from the mathematical clutter around NFS.

Our dummy example was not so stupid

The quadratic sieve (Pomerance, 1983) is a combination of two things:

- First idea: pick a simple «naturally small» function:
  - Consider $|f(i)| = \left|\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N\right|$.
  - For $|x| \leq S \ll \sqrt{N}$, we have $|f(i)| \leq 2S\sqrt{N} + \epsilon$
- Second idea: Factor residues completely differently.
  - The process used is known as sieving.
  - Sieving eliminates the per-relation factoring cost.

We will study more size improvements with the MPQS (multiple polynomial QS) algorithm.

Plan

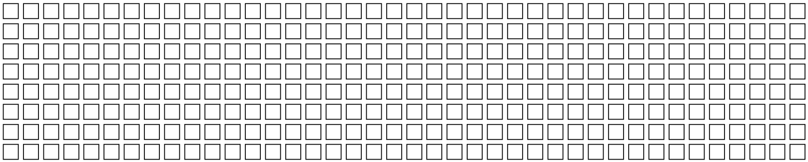
Plan

Sieving

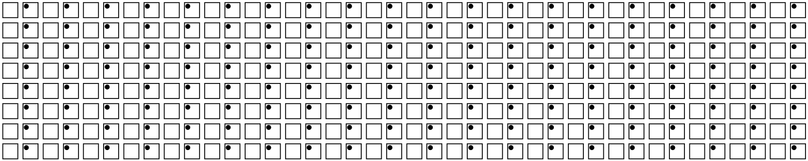
Key facts about sieving

- One decides beforehand of a sieving space: interval $[\![-S \dots S]\!]$.
- “for each i, for each p, do” becomes “for each p, for each i, do”.

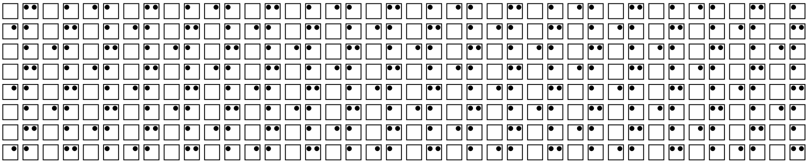
Sieving, visually



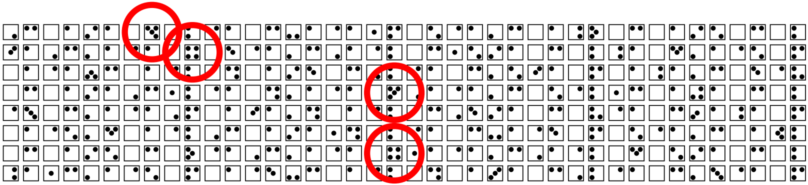
Sieving, visually



Sieving, visually



Sieving, visually



Sieving for the function in QS

Let $f(x) = (c+x)^2 - N$, with $c = \left\lceil \sqrt{N} \right\rceil$.
Given p, how does one describe the set:

$$\mathcal{S}_p = \{ i \in [\![-S \dots S]\!],\ f(i) \equiv 0 \mod p \}.$$

Sieving for the function in QS

Let $f(x) = (c + x)^2 - N$, with $c = \left\lceil \sqrt{N} \right\rceil$.
Given p, how does one describe the set:

$$\mathcal{S}_p = \{ i \in [\![-S \dots S]\!],\ f(i) \equiv 0 \mod p \} .$$

Answer:
- this depends on the roots mod p of the quadratic $f(x)$.
- 0, 1, or 2 roots depending on $\left(\frac{N}{p}\right)$.

Computing all valuations at once

Fix p. Let (at most) r_0, r_1 be the roots mod p of f.

$$\{i \in [\![-S \dots S]\!],\ f(i) \equiv 0 \mod p\} = \{r_0, r_0 \pm p, \dots\} \cup \{r_1, r_1 \pm p, \dots\}.$$

Algorithm: We maintain an array $T[i]$ indexed by $i \in [\![-S \dots S]\!]$.

- For each $p \leq B$, do:
  - Compute r_0, r_1
  - $r := r_0$. While $r \leq S$ do:
    - $T[r] \leftarrow T[r] + \log p$,
    - $r \leftarrow r + p$.
  - idem for r_1 as well as $\{r_i - kp\}$.
- Do this also for prime powers
- For all i such that $T[i] = \log|f(i)|$, we know that $f(i)$ is smooth.

Sieving with powers

(harder)

Assume that $f(i) \equiv 0$ has 2 distinct roots mod p (so $p \nmid \mathrm{disc}(f)$.

- How many roots mod p^2 ?
- How many roots mod p^k ?
- Which log contribution should we add ?

$T[i] = \log|f(i)| \Leftrightarrow f(i)$ smooth

For each p^k (assuming we consider k up to ∞. In fact we don't):

- we have characterized the set $\mathcal{S}_{p^k} = \{i,\ \nu_p(f(i)) \geq k\}$.
- we have added $\log_2 p$ to $T[i]$ for each i in this set.

Thus eventually:

$$\begin{aligned}
T[i] &= \sum_{p\in\mathcal{P}_B}\left(\sum_{k \text{ s.t. } i\in\mathcal{S}_{p^k}} \log p\right),\\
&= \sum_{p\in\mathcal{P}_B}\left(\sum_{k,\ \nu_p(f(i))\geq k} \log p\right),\\
&= \sum_{p\in\mathcal{P}_B} \nu_p(f(i)) \log p,\\
&= \log\left(B\text{-smooth part of } f(i)\right).
\end{aligned}$$

Plan

QS: analysis

How many sieve updates per prime number p ?

QS: analysis

How many sieve updates per prime number p ? $\dfrac{\leq 4S}{p-1}$.

Total number of sieve updates:

QS: analysis

How many sieve updates per prime number p ? $\frac{\leq 4S}{p-1}$.

Total number of sieve updates: $O(S \log\log B)$.

Assuming S is large enough so that we have enough relations eventually, the relation collection cost is $\widetilde{O}(S) \stackrel{\text{def}}{=} O(S(\log S)^{O(1)})$.

Strategy for analysis:

- Size of residues.
- Smoothness probability.
- Number of relations obtained. Condition for having enough.
- Cost for re-factoring $f(i)$ when once has been identified as smooth.
- Linear system cost.

QS: analysis

Let $S = L_N[\sigma, s]$ and $B = L_N[\beta, b]$.

- $|f(i)| = |\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N| \leq$?

QS: analysis

Let $S = L_N[\sigma, s]$ and $B = L_N[\beta, b]$, with $\sigma < 1$.

- $|f(i)| = |\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N| \leq 2S\sqrt{N} + \epsilon = L_N[1, 1/2 + o(1)]$.
- Smoothness probability:

QS: analysis

Let $S = L_N[\sigma, s]$ and $B = L_N[\beta, b]$, with $\sigma < 1$.

- $|f(i)| = |\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N| \leq 2S\sqrt{N} + \epsilon = L_N[1, 1/2 + o(1)]$.
- Smoothness probability: $L_N[1-\beta, -\frac{1}{2b}(1-\beta)]$.
- Condition for having enough relations:

QS: analysis

Let $S = L_N[\sigma, s]$ and $B = L_N[\beta, b]$, with $\sigma < 1$.

- $|f(i)| = |\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N| \leq 2S\sqrt{N} + \epsilon = L_N[1, 1/2 + o(1)]$.
- Smoothness probability: $L_N[1-\beta, -\frac{1}{2b}(1-\beta)]$.
- Condition for having enough relations:

$$L_N[\sigma, s] \times L_N[1-\beta, -\frac{1}{2b}(1-\beta)] = L_N[\beta, b],$$

$$\sigma = \beta = 1/2, \qquad s - \frac{1}{4b} = b.$$

  (s bigger would just cost more). Relation collection: $L_N[1/2, b + \frac{1}{4b}]$.
- Refactoring ?

QS: analysis

Let $S = L_N[\sigma, s]$ and $B = L_N[\beta, b]$, with $\sigma < 1$.

- $|f(i)| = |\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N| \leq 2S\sqrt{N} + \epsilon = L_N[1, 1/2 + o(1)]$.
- Smoothness probability: $L_N[1-\beta, -\frac{1}{2b}(1-\beta)]$.
- Condition for having enough relations:

$$L_N[\sigma, s] \times L_N[1-\beta, -\frac{1}{2b}(1-\beta)] = L_N[\beta, b],$$

$$\sigma = \beta = 1/2, \qquad s - \frac{1}{4b} = b.$$

  (s bigger would just cost more). Relation collection: $L_N[1/2, b + \frac{1}{4b}]$.
- Refactoring ? $L_N[1/2, 2b]$.
- Linear system ?

QS: analysis

Let $S = L_N[\sigma, s]$ and $B = L_N[\beta, b]$, with $\sigma < 1$.

- $|f(i)| = |\left(\left\lceil\sqrt{N}\right\rceil + i\right)^2 - N| \leq 2S\sqrt{N} + \epsilon = L_N[1, 1/2 + o(1)]$.
- Smoothness probability: $L_N[1-\beta, -\frac{1}{2b}(1-\beta)]$.
- Condition for having enough relations:

$$L_N[\sigma, s] \times L_N[1-\beta, -\frac{1}{2b}(1-\beta)] = L_N[\beta, b],$$

$$\sigma = \beta = 1/2, \qquad s - \frac{1}{4b} = b.$$

  (s bigger would just cost more). Relation collection: $L_N[1/2, b + \frac{1}{4b}]$.
- Refactoring ? $L_N[1/2, 2b]$.
- Linear system ? $L_N[1/2, \omega b]$ for some ω ($\omega = 3$ for Gauss).

$$\text{Total:} \quad L_N[1/2, b + \frac{1}{4b}] + L_N[1/2, 2b] + L_N[1/2, \omega b].$$

QS: complexity

We need to optimize $L_N[1/2, b + \frac{1}{4b}] + L_N[1/2, \omega b]$ (since $\omega \geq 2$).
Unless summands are equal, one is $o()$ of the others.
Thus b_{opt} given by $b_{\text{opt}} + \frac{1}{4b_{\text{opt}}} = \omega b_{\text{opt}}$.

- Set $\omega = 3$. Then $b_{\text{opt}} = \frac{1}{2\sqrt{2}}$, and QS $= L_N[1/2, \frac{3}{2\sqrt{2}}]$.
- If we can do $\omega = 2$, $b_{\text{opt}} = \frac{1}{2}$, and QS $= L_N[1/2, 1]$.

Notice that the cost for factoring relations has vanished.

Therefore, the complexity of linear algebra plays a role.

Plan

MPQS (Montgomery)

Annoying feature of QS: $|f(x)|$ gets bigger as x grows.
$\max = 2S\sqrt{N}$.

Quest: find other functions playing the role of $f(x)$.

What happens if we look at $(ax+b)^2$ for some a, b ?

$$(ax+b)^2 = a^2x^2 + 2axb + b^2$$

MPQS (Montgomery)

Annoying feature of QS: $|f(x)|$ gets bigger as x grows.
$\max = 2S\sqrt{N}$.

Quest: find other functions playing the role of $f(x)$.

What happens if we look at $(ax+b)^2$ for some a, b ?

$$(ax+b)^2 = a^2x^2 + 2axb + b^2 - ac + ac \text{ for any } c,$$

If we have $b^2 - ac = N$:

$$\frac{1}{a}(ax+b)^2 \equiv ax^2 + 2bx + c \mod N.$$

Fix a s.t. $\left(\frac{N}{a}\right) = 1$. Choose $b \leq \frac{a}{2}$ s.t. $b^2 \equiv N \mod a$. Set $c < 0$ accordingly.

$$|ax^2 + 2bx + c| \mathrel{\widetilde{\in}} \left[\frac{N}{a}, S^2 a - \frac{N}{a}\right].$$

For a given S, smallest values for $a \approx \frac{2\sqrt{N}}{S}$. $\Rightarrow$ Bound $\frac{1}{\sqrt{2}}S\sqrt{N}$.

MPQS

- For a given sieve interval size, we have found a better polynomial.
- More important, we have many such polynomials.
- Provided a is a product of factor base primes, a large number of polynomials can be used (other option: $a = \square$).
- Shorter intervals per polynomial $\Rightarrow$ smaller residues.
- Initialization cost per polynomial: solving $b^2 \equiv N \mod a$. See SIQS for a way to amortize this (e.g. in CrPo).

MPQS (with improvements to be discussed) is the leading algorithm today for p below 100-120 decimal digits.

Plan

Sieving tricks

Sieving can be made less accurate but faster:

- For the array $T[]$, log values can be stored as 8-bit integers.
- One may skip some primes or prime powers (pays little).
  - very small primes (many sieve updates, small contribution, $\pm$ leveled).
  - large powers (contribution is only $\log p$ for one sieve value over p^k).
  - Unwise to skip large p: large contribution, important for accuracy.
- "qualification" test: $T[i] \geq \log f(i) - \kappa$, with e^κ = cofactor bound.
  - If $e^\kappa \leq B$, for each such i, cofactor $q \in \mathcal{P}_B$: we have a relation.
  - If $e^\kappa \leq B^2$, for each such i, q prime, possibly $\leq B$. We have a complete factorization, but not a relation. Too bad.

Large primes

Idea (dates back to CFRAC):

- Fix a "large prime bound" L.
- As long as the cofator is $\leq L$, keep the "partial relations" as well, since we get them for free (almost).
- The cofactor q is called a large prime.

Two partial relations with the same large prime q can be combined:

$$\begin{aligned} f(i) &= \text{smooth} \times q, \\ f(i') &= \text{smooth} \times q, \\ f(i)f(i') &= \text{smooth} \times \square. \end{aligned}$$

K partial relations $\Rightarrow$ how many recombined relations ? Birthday paradox.

Number of matches

Thm. K independent, uniformly random picks from a set of size L yield an expecte total number of $\frac{K^2}{2L}$ matches.

Proof: cheat a little, or use generating functions, or do otherwise.

Keeping partial relations seems a waste at first. Eventually this pays off.

Note: recombined relations are heavier.

Two large primes

Experimentally, sieving is efficient. This leads to PPMPQS:

- Not too harmful to loosen the qualification and allow cofactors $> B^2$.
- Such (not necessarily prime) cofactors need to be factored.
- Allowing two large primes, we obtain "partial-partial" relations.

Old terminology:
- "full" (FF) relation: no large prime ;
- "partial" (FP) relation: one large prime ;
- "partial-partial" (PP) relation: two large primes.

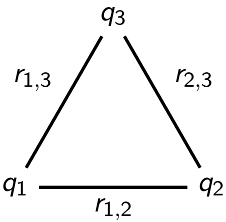
Modern statements of this method refer only to partial relations, and consider also more large primes.

Matching multiple large primes – the old way

Consider a graph where:

- vertices are large primes ;
- edges are relations ;
- an edge (relation R) connects two vertices (q_1, q_2) iff R involves q_1, q_2.
- Add a special vertex 1 to which all FP relations are connected.

A cycle in this large prime graph yields:



$$\longrightarrow r_{1,2}r_{2,3}r_{1,3} = \text{smooth} \times \square.$$

Hunting cycles: union-find algorithm (easy).

Matching large primes – the modern way

Consider arbitrarily many large primes – no real distinction with $\mathcal{P}_B$ primes anyway.

We thus have a very large set of partial relations, which go through several passes.

- Duplicate removal.
- Singleton removal: when only one relation involves a given prime.
- Merges: when prime q appears in only k relations:
  - Replace k relations with q by $k-1$ without q.
  - Try to do this the smart way.
- Use hash tables and a lot of RAM everywhere.

Cofactorization

Because of large primes, we bring our interest on sieve locations which do not necessarily yield relations.

- Some sieve reports are promising.
- We need to factor them before we decide to keep or discard.

This is called the cofactorization step (more NFS terminology).

Several «strategies»:

Cofactorization

Because of large primes, we bring our interest on sieve locations which do not necessarily yield relations.

- Some sieve reports are promising.
- We need to factor them before we decide to keep or discard.

This is called the cofactorization step (more NFS terminology).

Several «strategies»:

- Trial-divide up to some bound.

Cofactorization

Because of large primes, we bring our interest on sieve locations which do not necessarily yield relations.

- Some sieve reports are promising.
- We need to factor them before we decide to keep or discard.

This is called the cofactorization step (more NFS terminology).

Several «strategies»:

- Trial-divide up to some bound.
- «Resieve» up to some bound, but record primes.

Cofactorization

Because of large primes, we bring our interest on sieve locations which do not necessarily yield relations.

- Some sieve reports are promising.
- We need to factor them before we decide to keep or discard.

This is called the cofactorization step (more NFS terminology).

Several «strategies»:

- Trial-divide up to some bound.
- «Resieve» up to some bound, but record primes.
- Use small-p-sensible algorithms: $p \pm 1$, ECM.

Cofactorization

Because of large primes, we bring our interest on sieve locations which do not necessarily yield relations.

- Some sieve reports are promising.
- We need to factor them before we decide to keep or discard.

This is called the cofactorization step (more NFS terminology).

Several «strategies»:

- Trial-divide up to some bound.
- «Resieve» up to some bound, but record primes.
- Use small-p-sensible algorithms: $p \pm 1$, ECM.
- Maybe even run (MP)QS recursively ?

On top of that, early abort, e.g.: if trial division yields too little, forget about this relation.

QS/MPQS: conclusion

((((P)P)MP)/SI)QS is a living algorithm for factoring.

- relatively easy to understand / implement.
- complexity for factoring N depends only on N.
- for N of moderate size, MPQS is the way to go today.
- publicly available implementation: `msieve`.

Note: none of the sieving tricks affect the complexity:
$L_N[1/2, 1 + o(1)]$ (assuming $\omega = 2$).

Plan

Factor $2N$, factor N

We often have some freedom

- factor N
- or factor kN for small k.

Knuth: «this is a rather curious way to proceed (if not downright stupid)» (TAOCP2, 4.5.4).

- CFRAC: CFE of $\sqrt{N}$ gives congruences $p_n^2 - Nq_n^2 = v_n$.
- (MP)QS: Values of a quadratic polynomial of discriminant N.

Key idea

If we use kN instead, maybe some small p divide more often ?

Plan

- Characterize p's which divide the residue.
- Get a heuristic measure to maximize $\#$ divisors.

Why focus on this ?

Yield optimization has been studied for CFRAC, MPQS, and NFS.

- For CFRAC and NFS: some technicalities.
- Easier for MPQS, useful to get the idea.

Nowadays, the grandchild of the CFRAC « choice of multiplier » is part of NFS's « polynomial selection » step.

ℓ-valuation of a random integer

If X is a random integer:

- $\text{Prob}\,(\nu_\ell(X) \geq 1) = \frac{1}{\ell}$;
- $\text{Prob}\,(\nu_\ell(X) \geq 2) = \frac{1}{\ell^2}$;
- $\text{Prob}\,(\nu_\ell(X) \geq 3) = \frac{1}{\ell^3}$.
- etc.

Total:

$$E[\nu_\ell(X)] = \frac{1}{\ell} \cdot \frac{1}{1 - \frac{1}{\ell}} = \frac{1}{\ell - 1}.$$

Residues for MPQS

Residues for MPQS are:

$$f(x) = ax^2+2bx+c = \frac{1}{a}\left[(ax+b)^2 - (b^2-ac)\right] = \frac{1}{a}\left[(ax+b)^2 - N\right].$$

Assume $\ell \nmid a$ ($\ell \mid a$ more boring).

$$\#\{\text{Classes of sieve updates mod } \ell\} = \#\{\sqrt{N} \mod \ell\}.$$

Residues for MPQS

Residues for MPQS are:

$$f(x) = ax^2 + 2bx + c = \frac{1}{a}\left[(ax+b)^2 - (b^2 - ac)\right] = \frac{1}{a}\left[(ax+b)^2 - N\right].$$

Assume $\ell \nmid a$ ($\ell \mid a$ more boring).

$$\#\{\text{Classes of sieve updates mod } \ell\} = \#\{\sqrt{N} \mod \ell\}.$$

Let $r_\ell = \#$ square roots of N mod ℓ.

On average, after sieving: $\log|f(x)| - T[x] = \log|f(x)| - \sum r_\ell \frac{\log \ell}{\ell - 1}$.

For a random integer y of the same size: $\log|y| - \sum \frac{\log \ell}{\ell - 1}$.

Discrepancy function: $\sum (1 - r_\ell) \frac{\log \ell}{\ell - 1}$.

Residues for MPQS

Residues for MPQS are:

$$f(x) = ax^2 + 2bx + c = \frac{1}{a}\left[(ax+b)^2 - (b^2 - ac)\right] = \frac{1}{a}\left[(ax+b)^2 - kN\right].$$

Assume $\ell \nmid a$ ($\ell \mid a$ more boring).

$$\#\{\text{Classes of sieve updates mod } \ell\} = \#\{\sqrt{kN} \mod \ell\}.$$

Let $r_\ell = \#$ square roots of kN mod ℓ.

On average, after sieving: $\log|f(x)| - T[x] = \log|f(x)| - \sum r_\ell \frac{\log \ell}{\ell - 1}$.

For a random integer y of the same size: $\log|y| - \sum \frac{\log \ell}{\ell - 1}$.

Discrepancy function: $\alpha(k) = \sum (1 - r_\ell) \frac{\log \ell}{\ell - 1}$.

Choice of the multiplier

Choosing an adequate multiplier k:

- may increase the amount of sieve contributions from small primes.
- drawback: $f(x)$ grows with k.

The key idea remains: having many roots modulo small primes is good.

Yield optimization (Pomerance-Wagstaff)

(harder)

CFRAC looks at $Q_n = p_n^2 - kNq_n^2$, with k a square-free integer.

Which necessary condition should an odd prime ℓ satisfy, in order to have $\ell \mid Q_n$?

Yield optimization (Pomerance-Wagstaff)

(harder)

CFRAC looks at $Q_n = p_n^2 - kNq_n^2$, with k a square-free integer.

Which necessary condition should an odd prime ℓ satisfy, in order to have $\ell \mid Q_n$? Answer: $\left(\frac{kN}{\ell}\right) = 1$.

Can choose k $\pm$ freely (as long as $k = L_N[1/2, o(1)]$).

Theorem: average ℓ-valuation in CFRAC

The average ℓ-valuation of X (for ℓ odd prime) is

- $\frac{1}{\ell-1}$ if X is a random integer ;
- If $X = Q_n$: $\begin{cases} 0 & \text{if } \left(\frac{kN}{\ell}\right) = -1, \\ \frac{1}{\ell+1} & \text{if } \left(\frac{kN}{\ell}\right) = 0, \\ \frac{2}{\ell+1} \cdot \frac{\ell}{\ell-1} & \text{if } \left(\frac{kN}{\ell}\right) = 1 \quad \text{(assuming } p_n, q_n \text{ random co} \end{cases}$

ℓ-valuation: proof

(harder)

If $X = Q_n = p_n^2 - kNq_n^2$ and $\left(\frac{k}{\ell}\right) = 0$.

Assumption: (p_n, q_n) are random coprime integers.

Modulo ℓ: (p_n, q_n) maps to something uniformly random in ...

ℓ-valuation: proof

(harder)

If $X = Q_n = p_n^2 - kNq_n^2$ and $\left(\frac{k}{\ell}\right) = 0$.

Assumption: (p_n, q_n) are random coprime integers.

Modulo ℓ: (p_n, q_n) maps to something uniformly random in $\mathbb{P}^1(\mathbb{F}_\ell)$.

- $\text{Prob}\,(\nu_\ell(X) \geq 1) = \frac{1}{\ell+1}$;
- $\text{Prob}\,(\nu_\ell(X) \geq 2) = 0$ (because $\gcd(p_n, q_n) = 1$).

Total:

$$E[\nu_\ell(X)] = \frac{1}{\ell + 1}.$$

ℓ-valuation: proof

(harder)

If $X = Q_n = p_n^2 - kNq_n^2$ and $\left(\frac{k}{\ell}\right) = 1$.

Amongst $\ell + 1$ choices for $(p_n : q_n) \in \mathbb{P}^1(\mathbb{F}_\ell)$, exactly two lead to $\ell \mid Q_n$.

- $\mathrm{Prob}\,(\nu_\ell(X) \geq 1) = \frac{2}{\ell+1}$;
- $\mathrm{Prob}\,(\nu_\ell(X) \geq 2) = \frac{2}{\ell^2+\ell}$ (two roots in $\mathbb{P}^1(\mathbb{Z}/\ell^2\mathbb{Z})$) ;
- $\mathrm{Prob}\,(\nu_\ell(X) \geq 3) = \frac{2}{\ell^3+\ell^2}$;
- etc

Total:

$$E[\nu_\ell(X)] = \frac{2}{\ell+1} \cdot \frac{1}{1-\frac{1}{\ell}} = \frac{2\ell}{\ell^2-1}.$$

CFRAC: yield optimization

(harder)

Let $f(k,\ell)$ be the expected average ℓ-valuation of $p_n^2 - kNq_n^2$ as above.

We choose values of k for which $F(k)$ is large, where:

$$F(k) = \sum_{\ell<B} \left(f(k,\ell) - \frac{1}{\ell-1} \right) \log_2 \ell.$$

Idea: when e.g. $F(k) \approx 3$, we expect $Q_n \approx X$ to be smooth almost as often as a random integer $\approx \frac{X}{2^3}$.

Yield optimization is **important in practice**, and also important today with NFS.

Plan for next time

- CFRAC/QS/MPQS/NFS all build relations.
- We are faced with a linear system to be solved.
- The system is always sparse

Next lecture: sparse linear algebra algorithms.

Goal: solve a sparse $n \times n$ system in $\widetilde{O}(n^2)$.

(sparse: at most $(\log n)^{O(1)}$ non-zero coefficients per row).

Exercises

Exercise 1

Give the space complexity for sieving over an interval of length S with primes up to B, if one keeps track of all sieved primes for each location.