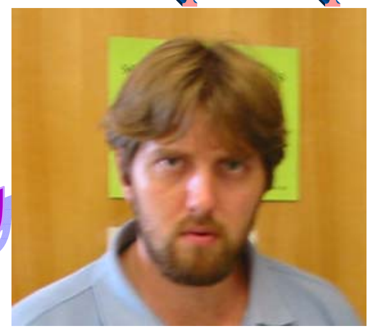
Running time predictions for square products,

Ernie Croot,



Robin Pemantle,

Prasad

Tetali,

& Andrew Granville,

and large prime variations

Running time predictions for square products, and large prime variations

by

Ernie Croot,

Andrew Granville,

Robin Pemantle,

& Prasad Tetali

Factoring algorithms

- Dixon's random squares algorithm
- The quadratic sieve
- Multiple polynomial quadratic sieve
- The number field sieve

. All have a common central idea

Factoring algorithms

- Dixon's random squares algorithm
- The quadratic sieve
- Multiple polynomial quadratic sieve
- The number field sieve

Idea: Generate a pseudo-random sequence of integers a_1, a_2, ..., with each

$$a_i \equiv b_i^2 \pmod{n},$$

until the product of a subseq of the a_i's is a square,

$$\text{say } Y^2 = a_{i_1} \cdots a_{i_k}.$$

Factoring algorithms

- Dixon's random squares algorithm
- The quadratic sieve
- Multiple polynomial quadratic sieve
- The number field sieve

Idea: Generate a pseudo-random sequence of integers a_1, a_2, ..., with each

$$a_i \equiv b_i^2 \pmod{n},$$

until the product of a subseq of the a_i's is a square,

$$\text{say } Y^2 = a_{i_1} \cdots a_{i_k}.$$

$$\text{Now, set } X^2 = (b_{i_1} \cdots b_{i_k})^2$$

and then

$$n \mid Y^2 - X^2 = (Y - X)(Y + X).$$

Factoring algorithms

- Dixon's random squares algorithm
- The quadratic sieve
- Multiple polynomial quadratic sieve
- The number field sieve

Idea: Generate a pseudo-random sequence of integers $a_1, a_2, \ldots$, with each

$$a_i \equiv b_i^2 \pmod{n},$$

until the product of a subseq of the a_i's is a square,

$$\text{say } Y^2 = a_{i_1} \cdots a_{i_k}.$$

$$\text{Now, set } X^2 = (b_{i_1} \cdots b_{i_k})^2$$

and then

$$n \mid Y^2 - X^2 = (Y - X)(Y + X).$$

$\geq 50\%$ CHANCE $\gcd(n, Y - X)$ IS A NON-TRIVIAL FACTOR OF n.

Analysis of running times

1994 ICM, Pomerance: In the (heuristic) *running time analysis* of such factoring algorithms one assumes that the pseudo-random sequence $a_1, a_2, \ldots$ is close enough to random, to make predictions based on this assumption.

. Why not study this as an abstract problem?

Analysis of running times

1994 ICM, Pomerance: In the (heuristic) *running time analysis* of such factoring algorithms one assumes that the pseudo-random sequence $a_1, a_2, \ldots$ is close enough to random, to make predictions based on this assumption.

Pomerance's problem

Select integers $a_1, a_2, \cdots \leq x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \leq m \leq x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = Y^2$$

What is expected stopping time?

Pomerance's problem

Select integers $a_1, a_2, \cdots \leq x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \leq m \leq x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = \square$$

Expected stopping time?

Pomerance's problem

Select integers $a_1, a_2, \cdots \leq x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \leq m \leq x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = \square$$

Expected stopping time?

Interesting because...

- If this expected stopping time is far less than what is obtained by the algorithms currently used, *maybe* we can speeding up factoring algorithms.

Pomerance's problem

Select integers $a_1, a_2, \cdots \leq x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \leq m \leq x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = \square$$

Expected stopping time?

Interesting because...

- If this expected stopping time is far less than what is obtained by the algorithms currently used, *maybe* we can speeding up factoring algorithms.

- Even if not, a good understanding might lead to better choice of parameters for most factoring algorithms.

Pomerance's problem

Select integers $a_1, a_2, \cdots \le x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \le m \le x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = \square$$

Expected stopping time?

Interesting because...

- If this expected stopping time is far less than what is obtained by the algorithms currently used, *maybe* we can speeding up factoring algorithms.

- Even if not, a good understanding might lead to better choice of parameters for most factoring algorithms.

- Possibility of proving something without assumption.

Previous best results

$\pi(y)$ = number of primes up to y.
n is y-*smooth* if $p|n \Rightarrow p \leq y$
$\Psi(x, y) = \#y$-smooths up to x.

Previous best results

$\pi(y)$ = number of primes up to y.
n is y-*smooth* if $p|n \Rightarrow p \le y$
$\Psi(x, y) = \#y$-smooths up to x.

Choose $y_0 = y_0(x)$ to maximize $\Psi(x, y)/y$, and let

$$J_0(x) \ := \ \frac{\pi(y_0)}{\Psi(x, y_0)} \cdot x.$$

$$\left(J_0(x) \approx e^{\sqrt{2 \log x \log \log x}} \right)$$

Previous best results

$\pi(y)$ = number of primes up to y.
n is y-*smooth* if $p|n \Rightarrow p \le y$
$\Psi(x, y)$ = #y-smooths up to x.

Choose $y_0 = y_0(x)$ to maximize $\Psi(x, y)/y$, and let

$$J_0(x) \; := \; \frac{\pi(y_0)}{\Psi(x, y_0)} \cdot x.$$

1985 Schroeppel: As $x \to \infty$,

$$\boxed{\text{Prob}(T \; < \; (1+\epsilon)J_0(x)) \; \to \; 1.}$$

$$\left(J_0(x) \approx e^{\sqrt{2 \log x \log\log x}} \; \right)$$

Previous best results

$\pi(y)$ = number of primes up to y.
n is y-*smooth* if $p|n \Rightarrow p \leq y$
$\Psi(x, y)$ = #y-smooths up to x.

Choose $y_0 = y_0(x)$ to maximize $\Psi(x, y)/y$, and let

$$J_0(x) \ := \ \frac{\pi(y_0)}{\Psi(x, y_0)} \cdot x.$$

1985 Schroeppel: As $x \to \infty$,

$$\boxed{\text{Prob}(T \ < \ (1+\epsilon)J_0(x)) \ \to \ 1.}$$

1994 Pomerance: As $x \to \infty$,

$$\boxed{\text{Prob}(T \ > \ J_0(x)^{1-\epsilon}) \ \to \ 1.}$$

$$\left(J_0(x) \approx e^{\sqrt{2 \log x \log \log x}} \right)$$

Pomerance's problem

Select integers $a_1, a_2, \dots \leq x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \leq m \leq x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = \square$$

Expected stopping time, T?

Schroeppel-Pomerance: With probability going to 1, we have

$$J_0(x)^{1-\epsilon} < T \ < \ (1+\epsilon) J_0(x)$$

Pomerance's problem

Select integers $a_1, a_2, \cdots \leq x$ independently at random; i.e.

$$\text{Prob}(a_j = m) = \frac{1}{x} \quad \forall\, 1 \leq m \leq x;$$

Stop at a_T as soon as there exists

$$a_{i_1} \cdots a_{i_k} = \square$$

Expected stopping time, T?

Schroeppel-Pomerance: With probability going to 1, we have

$$J_0(x)^{1-\epsilon} < T \ < \ (1+\epsilon) J_0(x)$$

Recently, in probability theory, results showing "*sharp transitions*"; i.e. random algorithms tend to stop at a certain precise time with probability going to 1.

Pomerance's problem

Expected stopping time, T?

From Schroeppel and Pomerance:

$$J_0(x)^{1-\epsilon} < T \ < \ (1+\epsilon)J_0(x).$$

with probability going to 1.

Pomerance's problem

Expected stopping time, T?

From Schroeppel and Pomerance:

$$J_0(x)^{1-\epsilon} < T \ < \ (1+\epsilon)J_0(x).$$

with probability going to 1.

Guess: There exists $f(x)$, s.t.

$$(1-\epsilon)f(x) < T \ < \ (1+\epsilon)f(x),$$

with probability going to 1.

Pomerance's problem

Expected stopping time, T?

From Schroeppel and Pomerance:

$$J_0(x)^{1-\epsilon} < T \ < \ (1+\epsilon)J_0(x).$$

with probability going to 1.

Guess: There exists $f(x)$, s.t.

$$(1-\epsilon)f(x) < T \ < \ (1+\epsilon)f(x),$$

with probability going to 1.

Conjecture: $f(x) = e^{-\gamma}J_0(x)$, i.e.

$$(1-\epsilon)e^{-\gamma}J_0(x) < T \ < \ (1+\epsilon)e^{-\gamma}J_0(x),$$

with probability going to 1.

Pomerance's problem

Expected stopping time, T?

From Schroeppel and Pomerance:

$$J_0(x)^{1-\epsilon} < T \; < \; (1+\epsilon)J_0(x).$$

with probability going to 1.

Guess: There exists $f(x)$, s.t.

$$(1-\epsilon)f(x) < T \; < \; (1+\epsilon)f(x),$$

with probability going to 1.

Conjecture: $f(x) = e^{-\gamma}J_0(x)$, i.e.

$$(1-\epsilon)e^{-\gamma}J_0(x) < T \; < \; (1+\epsilon)e^{-\gamma}J_0(x),$$

with probability going to 1.

Theorem: Almost proved it!

$$\left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma}J_0(x) < T \; < \; (1+\epsilon)e^{-\gamma}J_0(x),$$

with probability going to 1.

Pomerance's problem

Theorems: With prob going to 1:

- $\frac{\pi}{4}\left(e^{-\gamma} - \epsilon\right) J_0(x) < T \; < \; (e^{-\gamma}+\epsilon)J_0(x).$

Pomerance's problem

Theorems: With prob going to 1:

- $\frac{\pi}{4}\left(e^{-\gamma}-\epsilon\right)J_0(x) < T < (e^{-\gamma}+\epsilon)J_0(x)$.

- All numbers in the square product $a_{i_1}\cdots a_{i_k}$ are $y_0^{2+\epsilon}$-smooth.

- There are $k = y_0^{1+o(1)}$ different a_i's in the square product

Pomerance's problem

Theorems: With prob going to 1:

- $\frac{\pi}{4}\left(e^{-\gamma}-\epsilon\right) J_0(x) < T < (e^{-\gamma}+\epsilon) J_0(x)$.

- All numbers in the square product $a_{i_1} \cdots a_{i_k}$ are $y_0^{2+\epsilon}$-smooth.

- There are $k = y_0^{1+o(1)}$ different a_i's in the square product

- If $T < \left(\frac{\pi}{4}-\epsilon\right) e^{-\gamma} J_0(x)$ then the square product is just one a_i, a square.

Schroeppel's 1985 approach

Study only the y-smooth a_i's.

Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \ldots p_k^{e_{i,k}}$

Schroeppel's 1985 approach

Study only the y-smooth a_i's.

Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \dots p_k^{e_{i,k}}$

Form the matrix with rows $(e_{i,1}, e_{i,2}, \dots, e_{i,k})$

Schroeppel's 1985 approach

Study only the y-smooth a_i's.
Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \dots p_k^{e_{i,k}}$
Form the matrix with rows $(e_{i,1}, e_{i,2}, \dots, e_{i,k})$

A subset of the y-smooth a_i's forms a square product *if and only if* the sum of those rows is $0 \pmod 2$.

Schroeppel's 1985 approach

Study only the y-smooth a_i's.
Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \ldots p_k^{e_{i,k}}$
Form the matrix with rows $(e_{i,1}, e_{i,2}, \ldots, e_{i,k})$

A subset of the y-smooth a_i's forms a square product *if and only if* the sum of those rows is $0 \pmod 2$.

So $\exists$ a square product if $> k$ rows.

Schroeppel's 1985 approach

Study only the y-smooth a_i's.
Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \dots p_k^{e_{i,k}}$
Form the matrix with rows $(e_{i,1}, e_{i,2}, \dots, e_{i,k})$

A subset of the y-smooth a_i's forms a square product *if and only if* the sum of those rows is $0 \pmod 2$.

So $\exists$ a square product if $> k$ rows.

$\text{Prob}(a_i \text{ is } y-\text{smooth}) = \psi(x, y)/x$.
$\therefore\ \mathbb{E}(\#\text{ of rows}) = T\psi(x, y)/x$,

Schroeppel's 1985 approach

Study only the y-smooth a_i's.
Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \dots p_k^{e_{i,k}}$
Form the matrix with rows $(e_{i,1}, e_{i,2}, \dots, e_{i,k})$

A subset of the y-smooth a_i's forms a square product *if and only if* the sum of those rows is $0 \pmod 2$.

So $\exists$ a square product if $> k$ rows.

Prob(a_i is $y-$smooth) $= \psi(x, y)/x$.
$\therefore\ \mathbb{E}(\#$ of rows$) = T\psi(x, y)/x$,

This is $> k = \pi(y)$ for $T > x\pi(y)/\Psi(x, y)$.
Minimized at $y = y_0$, so $T > J_0(x)$.

Schroeppel's 1985 approach

Study only the y-smooth a_i's.
Factor each as $a_i = 2^{e_{i,1}} 3^{e_{i,2}} \dots p_k^{e_{i,k}}$
Form the matrix with rows $(e_{i,1}, e_{i,2}, \dots, e_{i,k})$

A subset of the y-smooth a_i's forms a square product *if and only if* the sum of those rows is $0 \pmod 2$.

So $\exists$ a square product if $> k$ rows.

$\text{Prob}(a_i \text{ is } y\text{-smooth}) = \psi(x,y)/x$.
$\therefore\ \mathbb{E}(\#\text{ of rows}) = T\psi(x,y)/x$,

This is $> k = \pi(y)$ for $T > x\pi(y)/\Psi(x,y)$.
Minimized at $y = y_0$, so $T > J_0(x)$.

Technique works, with prob 1, for

$$T > (1+\epsilon)J_0(x)$$

Schroeppel's 1985 approach

Factor base: Primes up to y

Keep only y-smooth a_i's

Square product after $\geq k$ such a_i's

Schroeppel's 1985 approach

Factor base: Primes up to y

Keep only y-smooth a_i's

Square product after $\geq k$ such a_i's

Practical to implement

Schroeppel's 1985 approach

Factor base: Primes up to y

Keep only y-smooth a_i's

Square product after $\geq k$ such a_i's

Practical to implement

New result: By time $(1+\epsilon)J_0(x)$ one has not one square product, but *many*, about $\epsilon\pi(y_0)$, with prob $\to 1$.

Schroeppel's 1985 approach

Factor base: Primes up to y

Keep only y-smooth a_i's

Square product after $\geq k$ such a_i's

Practical to implement

New result: By time $(1+\epsilon)J_0(x)$ one has not one square product, but *many*, about $\epsilon\pi(y_0)$, with prob $\to 1$.

Key Variant: Large Prime Variation

Also keep $a_i = pb_i$ where b_i is y-smooth and p is a prime $> y$.

Schroeppel's 1985 approach

Factor base: Primes up to y

Keep only y-smooth a_i's

Square product after $\geq k$ such a_i's

Practical to implement

New result: By time $(1+\epsilon)J_0(x)$ one has not one square product, but *many*, about $\epsilon\pi(y_0)$, with prob $\to 1$.

Key Variant: Large Prime Variation

Also keep $a_i = pb_i$ where b_i is y-smooth and p is a prime $> y$.

With two, pb_i and pb_j, their product $p^2 b_i b_j$ is a *y-pseudosmooth* and can be used as a row in our matrix.

Schroeppel's 1985 approach

Factor base: Primes up to y

Keep only y-smooth a_i's

Square product after $\geq k$ such a_i's

Practical to implement

New result: By time $(1+\epsilon)J_0(x)$ one has not one square product, but *many*, about $\epsilon\pi(y_0)$, with prob $\to 1$.

Key Variant: Large Prime Variation

Also keep $a_i = pb_i$ where b_i is y-smooth and p is a prime $> y$.

With two, pb_i and pb_j, their product $p^2 b_i b_j$ is a *y-pseudosmooth* and can be used as a row in our matrix.

Theorem: Speed up by factor

$$.74997591747934498263\ldots \approx \frac{3}{4}$$

. Will prove this later

Multiple large prime variations

Can try two large primes: $a_i = pqb_i$,
b_i is y-smooth, p, q primes $> y$.

Multiple large prime variations

Can try two large primes: $a_i = pqb_i$, b_i is y-smooth, p, q primes $> y$.

And three large primes, etc.

Multiple large prime variations

Can try two large primes: $a_i = pqb_i$, b_i is y-smooth, p, q primes $> y$.

And three large primes, etc.

Practical issues: Upper bound on extra primes, say $y < p < My$.

Multiple large prime variations

Can try two large primes: $a_i = pqb_i$, b_i is y-smooth, p, q primes $> y$.

And three large primes, etc.

Practical issues: Upper bound on extra primes, say $y < p < My$.

How to find pseudosmooths?

Multiple large prime variations

Can try two large primes: $a_i = pqb_i$, b_i is y-smooth, p, q primes $> y$.

And three large primes, etc.

Practical issues: Upper bound on extra primes, say $y < p < My$.

How to find pseudosmooths?

PROVED SPEED-UPS

ℓ	$M = \infty$	$M = 100$	$M = 10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation, each large prime in (y, My).

. Speed-up for # of a_i's searched, not for factoring algorithm

Speed-ups

ℓ	$M = \infty$	$M = 100$	$M = 10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation, each large prime in (y, My).

SPEED-UPS

ℓ	$M = \infty$	$M = 100$	$M = 10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation, each large prime in (y, My).

As $\ell, M \to \infty$ (M slowly), our speed up factor $\to e^{-\gamma} = .5614594836\ldots$

Speed-ups

ℓ	$M = \infty$	$M = 100$	$M = 10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation, each large prime in (y, My).

As $\ell, M \to \infty$ (M slowly), our speed up factor $\to e^{-\gamma} = .5614594836\ldots$

How big are y_0 and J_0?

Speed-ups

ℓ	$M=\infty$	$M=100$	$M=10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation, each large prime in (y, My).

As $\ell, M \to \infty$ (M slowly), our speed up factor $\to e^{-\gamma} = .5614594836\ldots$

How big are y_0 and J_0?

For $L(x) = e^{\sqrt{\frac{1}{2}\log x \log\log x}}$,

$y_0(x) = L(x)^{1+o(1)}$ and $J_0(x) = L(x)^{2+o(1)}$

. These estimates can be made more precise but not to asymptotics

Such precise estimates?

If we cannot be precise about the value of J_0, how can we determine these speed-up constants?

Such precise estimates?

If we cannot be precise about the value of J_0, how can we determine these speed-up constants?

1986 Hildebrand and Tenenbaum: Even if we cannot give an asymptotic formula for $\Psi(x, y)$, we can do so for

$$\frac{\Psi(2x, y)}{\Psi(x, y)}$$

and similar qns where the variables involved are close in size.

Our results here make full use of their estimates,

Such precise estimates?

If we cannot be precise about the value of J_0, how can we determine these speed-up constants?

1986 Hildebrand and Tenenbaum: Even if we cannot give an asymptotic formula for $\Psi(x, y)$, we can do so for

$$\frac{\Psi(2x, y)}{\Psi(x, y)}$$

and similar qns where the variables involved are close in size.

Our results here make full use of their estimates,

But still cannot estimate J_0 accurately – maybe value can be computed (Bernstein) in any example.

Analysis; one large prime, I

$$\text{Prob}(a = pb \leq x : \; b \text{ is } y\text{−smooth}) = \frac{\Psi(x/p, y)}{x}$$

ANALYSIS; ONE LARGE PRIME, I

$$\text{Prob}(a = pb \leq x : \ b \text{ is } y - \text{smooth}) = \frac{\Psi(x/p, y)}{x}$$

$$= \frac{\Psi(x/p, y)}{\Psi(x, y)} \cdot \frac{\Psi(x, y)}{x} \sim \frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x}$$

for $y < p < y \log y$.

Analysis; one large prime, I

$$\text{Prob}(a = pb \le x : \ b \text{ is } y - \text{smooth}) = \frac{\Psi(x/p, y)}{x}$$

$$= \frac{\Psi(x/p, y)}{\Psi(x, y)} \cdot \frac{\Psi(x, y)}{x} \sim \frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x}$$

for $y < p < y \log y$.

$\therefore$ Expected # of k-tuples, from the $T = \eta J_0$ values of a_i, that are p times a y-smooth:

$$\sim \binom{T}{k} \left(\frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x} \right)^k$$

Analysis; one large prime, I

$$\text{Prob}(a = pb \le x : \ b \text{ is } y - \text{smooth}) = \frac{\Psi(x/p, y)}{x}$$

$$= \frac{\Psi(x/p, y)}{\Psi(x, y)} \cdot \frac{\Psi(x, y)}{x} \sim \frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x}$$

for $y < p < y \log y$.

$\therefore$ Expected # of k-tuples, from the $T = \eta J_0$ values of a_i, that are p times a y-smooth:

$$\sim \binom{T}{k} \left(\frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x} \right)^k \sim \frac{1}{k!} \left(\frac{\eta y}{p} \right)^k$$

as $J_0 \log y \Psi(x, y)/x \sim y$.

Analysis; one large prime, I

$$\text{Prob}(a = pb \le x : \; b \text{ is } y - \text{smooth}) = \frac{\Psi(x/p, y)}{x}$$

$$= \frac{\Psi(x/p, y)}{\Psi(x, y)} \cdot \frac{\Psi(x, y)}{x} \sim \frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x}$$

for $y < p < y \log y$.

$\therefore$ Expected # of k-tuples, from the $T = \eta J_0$ values of a_i, that are p times a y-smooth:

$$\sim \binom{T}{k} \left(\frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x} \right)^k \sim \frac{1}{k!} \left(\frac{\eta y}{p} \right)^k$$

as $J_0 \log y \Psi(x, y)/x \sim y$.

$\therefore$ Expected number of k-tuples, including all $p > y$, is

$$\sim \frac{(\eta y)^k}{k!} \sum_{p > y} \frac{1}{p^k}$$

ANALYSIS; ONE LARGE PRIME, I

$$\text{Prob}(a = pb \le x: \ b \text{ is } y - \text{smooth}) = \frac{\Psi(x/p, y)}{x}$$

$$= \frac{\Psi(x/p, y)}{\Psi(x, y)} \cdot \frac{\Psi(x, y)}{x} \sim \frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x}$$

for $y < p < y \log y$.

$\therefore$ Expected # of k-tuples, from the $T = \eta J_0$ values of a_i, that are p times a y-smooth:

$$\sim \binom{T}{k} \left(\frac{\log y}{p} \cdot \frac{\Psi(x, y)}{x} \right)^k \sim \frac{1}{k!} \left(\frac{\eta y}{p} \right)^k$$

as $J_0 \log y \Psi(x, y)/x \sim y$.

$\therefore$ Expected number of k-tuples, including all $p > y$, is

$$\sim \frac{(\eta y)^k}{k!} \sum_{p > y} \frac{1}{p^k} \sim \frac{(\eta y)^k}{k!} \frac{1}{(k-1) y^{k-1} \log y}$$

Analysis; one large prime, I

$$\text{Prob}(a = pb \le x: \; b \text{ is } y-\text{smooth}) = \frac{\Psi(x/p,y)}{x}$$

$$= \frac{\Psi(x/p,y)}{\Psi(x,y)} \cdot \frac{\Psi(x,y)}{x} \sim \frac{\log y}{p} \cdot \frac{\Psi(x,y)}{x}$$

for $y < p < y \log y$.

$\therefore$ Expected # of k-tuples, from the $T = \eta J_0$ values of a_i, that are p times a y-smooth:

$$\sim \binom{T}{k} \left(\frac{\log y}{p} \cdot \frac{\Psi(x,y)}{x} \right)^k \sim \frac{1}{k!} \left(\frac{\eta y}{p} \right)^k$$

as $J_0 \log y \Psi(x,y)/x \sim y$.

$\therefore$ Expected number of k-tuples, including all $p > y$, is

$$\sim \frac{(\eta y)^k}{k!} \sum_{p>y} \frac{1}{p^k} \sim \frac{(\eta y)^k}{k!} \frac{1}{(k-1) y^{k-1} \log y}$$

$$\sim \frac{\eta^k}{(k-1)k!} \, \pi(y)$$

Analysis; one large prime, II

Sps $pb_1, pb_2, \ldots, pb_r$ amongst a_i's, each b_j is y-smooth.

Analysis; one large prime, II

Sps $pb_1, pb_2, \ldots, pb_r$ amongst a_i's, each b_j is y-smooth. These yield exactly $r-1$ mult indep pseudosmooths,

$$(pb_1)(pb_2), (pb_1)(pb_3), \ldots, (pb_1)(pb_r).$$

Analysis; one large prime, II

Sps $pb_1, pb_2, \ldots, pb_r$ amongst a_i's, each b_j is y-smooth. These yield exactly $r-1$ mult indep pseudosmooths,

$$(pb_1)(pb_2), (pb_1)(pb_3), \ldots, (pb_1)(pb_r).$$

Use identity

$$r - 1 = \sum_{\substack{I \subset \{1,\ldots,r\} \\ |I| \geq 2}} (-1)^{|I|};$$

Analysis; one large prime, II

Sps $pb_1, pb_2, \ldots, pb_r$ amongst a_i's, each b_j is y-smooth. These yield exactly $r-1$ mult indep pseudosmooths,

$$(pb_1)(pb_2), (pb_1)(pb_3), \ldots, (pb_1)(pb_r).$$

Use identity

$$r - 1 = \sum_{\substack{I \subset \{1,\ldots,r\} \\ |I| \geq 2}} (-1)^{|I|};$$

so total # smooths and pseudo-smooths:

$$\left(\eta + \sum_{k \geq 2} (-1)^k \frac{\eta^k}{(k-1)k!} \right) \pi(y)$$

ANALYSIS; ONE LARGE PRIME, II

Sps $pb_1, pb_2, \ldots, pb_r$ amongst a_i's, each b_j is y-smooth. These yield exactly $r-1$ mult indep pseudosmooths,

$$(pb_1)(pb_2), (pb_1)(pb_3), \ldots, (pb_1)(pb_r).$$

Use identity

$$r - 1 = \sum_{\substack{I \subset \{1,\ldots,r\} \\ |I| \geq 2}} (-1)^{|I|};$$

so total # smooths and pseudo-smooths:

$$\left(\eta + \sum_{k \geq 2} (-1)^k \frac{\eta^k}{(k-1)k!} \right) \pi(y)$$

Constant = 1 for $\eta = .74997591747934498263\ldots$

 And for two large primes?

Two large primes, I

Expected # pairs pqb_1 & pqb_2, and
Expected # triples pqb_1, prb_2, qrb_3,
bounded, so not useful!

Two large primes, I

Expected # pairs pqb_1 & pqb_2, and
Expected # triples pqb_1, prb_2, qrb_3,
bounded, so not useful!

Expected # triples pb_1, pqb_2, qb_3, is $\sim \eta^3\pi(y)$.

Two large primes, I

Expected # pairs pqb_1 & pqb_2, and
Expected # triples pqb_1, prb_2, qrb_3,
bounded, so not useful!

Expected # triples pb_1, pqb_2, qb_3, is $\sim \eta^3\pi(y)$.

How do you tell the difference?

Two large primes, I

Expected # pairs pqb_1 & pqb_2, and
Expected # triples pqb_1, prb_2, qrb_3,
bounded, so not useful!

Expected # triples pb_1, pqb_2, qb_3, is $\sim \eta^3\pi(y)$.

How do you tell the difference?

Construct matrix M for p, pq, p, with ≥ 2 ones in each col:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Two large primes, I

Expected # pairs pqb_1 & pqb_2, and Expected # triples pqb_1, prb_2, qrb_3, *bounded*, so not useful!

Expected # triples pb_1, pqb_2, qb_3, is $\sim \eta^3\pi(y)$.

How do you tell the difference?

Construct matrix M for p, pq, p, with ≥ 2 ones in each col:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Expect $c\pi(y)$ such triples *iff*

$$\boxed{\#\text{ones} = \#\text{rows} + \#\text{columns} - 1.}$$

Two large primes, I

Expected # pairs pqb_1 & pqb_2, and
Expected # triples pqb_1, prb_2, qrb_3,
bounded, so not useful!

Expected # triples pb_1, pqb_2, qb_3, is $\sim \eta^3\pi(y)$.

How do you tell the difference?

Construct matrix M for p, pq, p, with ≥ 2 ones in each col:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Expect $c\pi(y)$ such triples *iff*

$$\boxed{\#\text{ones} = \#\text{rows} + \#\text{columns} - 1.}$$

Yields exactly

$$\#\text{rows} - \text{rank}(M)$$

mult indep pseudosmooths

. Combinatorial identity like in one large prime case?

Many large primes, II

Before used

$$r - 1 = \sum_{\substack{I \subset \{1,\ldots,r\} \\ |I| \geq 2}} (-1)^{|I|}.$$

Many large primes, II

Before used

$$r-1=\sum_{\substack{I\subset\{1,\dots,r\}\\|I|\geq 2}}(-1)^{|I|}.$$

Generalization: $\mathcal{M}$:= matrices s.t.

$\#$ones $=$ $\#$rows $+$ $\#$columns -1.

M_R has rows R.

Many large primes, II

Before used

$$r - 1 = \sum_{\substack{I \subset \{1,\ldots,r\} \\ |I| \geq 2}} (-1)^{|I|}.$$

Generalization: $\mathcal{M}$:= matrices s.t.

$$\#\text{ones} = \#\text{rows} + \#\text{columns} - 1.$$

M_R has rows R. If $M_R \in \mathcal{M}$ then

$$\boxed{\#\text{rows} - \text{rank}(M_R) = \sum_{\substack{S \subset R \\ M_S \in \mathcal{M}}} (-1)^{\#\text{ones}(S)}}$$

Many large primes, II

Before used

$$r-1=\sum_{\substack{I\subset\{1,\dots,r\}\\ |I|\geq 2}}(-1)^{|I|}.$$

Generalization: $\mathcal{M}$:= matrices s.t.

#ones = #rows + #columns − 1.

M_R has rows R. If $M_R \in \mathcal{M}$ then

$$\#\text{rows} - \text{rank}(M_R) = \sum_{\substack{S\subset R\\ M_S\in\mathcal{M}}}(-1)^{\#\text{ones}(S)}$$

How do we count the different type of matrices that arise?

A graph theory approach

Construct graph $G = G(M)$:
For each a_i a vertex $v_i \in G$,
With $v_i \sim v_I$ of colour p_j if $p_j | (n_i, n_I)$
– $M_{i,j} = M_{I,j} = 1$.

A graph theory approach

Construct graph $G = G(M)$:
For each a_i a vertex $v_i \in G$,
With $v_i \sim v_I$ of colour p_j if $p_j | (n_i, n_I)$
– $M_{i,j} = M_{I,j} = 1$.

If $M \in \mathcal{M}$, $G(M)$ is simple, & only cycles in $G(M)$ are monochromatic

A graph theory approach

Construct graph $G = G(M)$:
For each a_i a vertex $v_i \in G$,
With $v_i \sim v_I$ of colour p_j if $p_j | (n_i, n_I)$
– $M_{i,j} = M_{I,j} = 1$.

If $M \in \mathcal{M}$, $G(M)$ is simple, & only cycles in $G(M)$ are monochromatic

So cycles are subsets of the complete graph of edges of that colour.

A graph theory approach

Construct graph $G = G(M)$:
For each a_i a vertex $v_i \in G$,
With $v_i \sim v_I$ of colour p_j if $p_j | (n_i, n_I)$
– $M_{i,j} = M_{I,j} = 1$.

If $M \in \mathcal{M}$, $G(M)$ is simple, & only cycles in $G(M)$ are monochromatic

So cycles are subsets of the complete graph of edges of that colour.

Hence any two-connected subgraph of $G(M)$ is a complete graph: This is known as a *Husimi graph*:

A GRAPH THEORY APPROACH

Construct graph $G = G(M)$:
For each a_i a vertex $v_i \in G$,
With $v_i \sim v_I$ of colour p_j if $p_j | (n_i, n_I)$
– $M_{i,j} = M_{I,j} = 1$.

If $M \in \mathcal{M}$, $G(M)$ is simple, & only cycles in $G(M)$ are monochromatic

So cycles are subsets of the complete graph of edges of that colour.

Hence any two-connected subgraph of $G(M)$ is a complete graph: This is known as a *Husimi graph*:

— Inspired combinatorics of 'species',

— Central to gas thermodynamics

The theory of species

Generating functions known for counting Husimi graphs with different parameters.

The theory of species

Generating functions known for counting Husimi graphs with different parameters. Leads to: For $T = \eta J_0$, the number of pseudosmooths is $\sim f(\eta)\pi(y_0)$ where $f(\eta)$ satisfies

$$\eta f'(\eta) = -\log(1 - f(\eta)).$$

The theory of species

Generating functions known for counting Husimi graphs with different parameters. Leads to: For $T = \eta J_0$, the number of pseudosmooths is $\sim f(\eta)\pi(y_0)$ where $f(\eta)$ satisfies

$$\eta f'(\eta) = -\log(1 - f(\eta)).$$

This implies

- $f(\eta)$ is monotone increasing,
- $f(e^{-\gamma}) = 1$, but
- $f(\eta)$ *diverges* for $\eta > e^{-\gamma}$.

The theory of species

Generating functions known for counting Husimi graphs with different parameters. Leads to: For $T = \eta J_0$, the number of pseudosmooths is $\sim f(\eta)\pi(y_0)$ where $f(\eta)$ satisfies

$$\eta f'(\eta) = -\log(1 - f(\eta)).$$

This implies

- $f(\eta)$ is monotone increasing,
- $f(e^{-\gamma}) = 1$, but
- $f(\eta)$ *diverges* for $\eta > e^{-\gamma}$.

Sadly we re-organized a non-abs cvgent series in our proof, and we have been unable to fix this proof!

In fact we swapped order of summation in applying

$$\#\text{rows} - \text{rank}(M_R) = \sum_{\substack{S \subset R \\ M_S \in \mathcal{M}}} (-1)^{\#\text{ones}(S)}$$

Lower bound on T

Suppose $T < \left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma} J_0(x)$.

Lower bound on T

Suppose $T < \left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma} J_0(x)$.

Write each $a_i = b_i d_i$ where for each $p|a_i$ we have

$$p \leq y \Rightarrow p|b_i$$
$$p > y \Rightarrow p|d_i.$$

If $\prod_{i \in I} a_i = \square$ then $\prod_{i \in I} d_i = \square$.

Lower bound on T

Suppose $T < \left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma} J_0(x)$.

Write each $a_i = b_i d_i$ where for each $p | a_i$ we have

$$p \le y \Rightarrow p | b_i$$
$$p > y \Rightarrow p | d_i.$$

If $\prod_{i \in I} a_i = \square$ then $\prod_{i \in I} d_i = \square$.

Idea: For each k,

Expected # of I with $|I| = k$ & $\prod_{i \in I} a_i = \square$

$\le$ Expected # of I with $|I| = k$ & $\prod_{i \in I} d_i = \square$.

Lower bound on T

Suppose $T < \left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma} J_0(x)$.

Write each $a_i = b_i d_i$ where for each $p | a_i$ we have

$$p \leq y \Rightarrow p | b_i$$
$$p > y \Rightarrow p | d_i.$$

If $\prod_{i \in I} a_i = \square$ then $\prod_{i \in I} d_i = \square$.

Idea: For each k,

Expected # of I with $|I| = k$ & $\prod_{i \in I} a_i = \square$

$\leq$ Expected # of I with $|I| = k$ & $\prod_{i \in I} d_i = \square$.

Bound this using "Rankin's trick".

Lower bound on T

Suppose $T < \left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma} J_0(x)$.

Write each $a_i = b_i d_i$ where for each $p | a_i$ we have

$$p \le y \Rightarrow p | b_i$$
$$p > y \Rightarrow p | d_i.$$

If $\prod_{i \in I} a_i = \Box$ then $\prod_{i \in I} d_i = \Box$.

Idea: For each k,

Expected # of I with $|I| = k$ & $\prod_{i \in I} a_i = \Box$

$\le$ Expected # of I with $|I| = k$ & $\prod_{i \in I} d_i = \Box$.

Bound this using "Rankin's trick".

Picking $y = y(k)$ carefully we show the prob is $\ll T^2 (\log x)/x$ over all $k \ge 2$

Lower bound on T

Suppose $T < \left(\frac{\pi}{4} - \epsilon\right) e^{-\gamma} J_0(x)$.

Write each $a_i = b_i d_i$ where for each $p | a_i$ we have

$$p \le y \Rightarrow p | b_i$$
$$p > y \Rightarrow p | d_i.$$

If $\prod_{i \in I} a_i = \square$ then $\prod_{i \in I} d_i = \square$.

Idea: For each k,

Expected # of I with $|I| = k$ &

$$\prod_{i \in I} a_i = \square$$

$\le$ Expected # of I with $|I| = k$ &

$$\prod_{i \in I} d_i = \square.$$

Bound this using "Rankin's trick".

Picking $y = y(k)$ carefully we show the prob is $\ll T^2 (\log x)/x$ over all $k \ge 2$, *so most likely* is $|I| = 1$ (which has probability $\approx T/\sqrt{x}$).

Hypergraphs to the rescue

Vertices correspond to primes in (y, My), and *hyperedges* to the odd power prime factors of each a_i.

Hypergraphs to the rescue

Vertices correspond to primes in (y, My), and *hyperedges* to the odd power prime factors of each a_i.

Viewpoint: Construction of a random hypergraph, using only a_i that are "helpful".

Hypergraphs to the rescue

Vertices correspond to primes in (y, My), and *hyperedges* to the odd power prime factors of each a_i.

Viewpoint: Construction of a random hypergraph, using only a_i that are "helpful".

Use Poisson point processes. Translate our discrete problem to continuous setting – megatechnical!

Hypergraphs to the rescue

Vertices correspond to primes in (y, My), and *hyperedges* to the odd power prime factors of each a_i.

Viewpoint: Construction of a random hypergraph, using only a_i that are "helpful".

Use Poisson point processes. Translate our discrete problem to continuous setting – megatechnical!

This very different approach yields the same answer

$$e^{-\gamma} J_0(x).$$

Hypergraphs to the rescue

Vertices correspond to primes in (y, My), and *hyperedges* to the odd power prime factors of each a_i.

Viewpoint: Construction of a random hypergraph, using only a_i that are "helpful".

Use Poisson point processes. Translate our discrete problem to continuous setting – megatechnical!

This very different approach yields the same answer

$$e^{-\gamma} J_0(x).$$

And the same divergence now implies one gets many square products soon after getting the first!

Pomerance's application

If one optimizes parameters in Pomerance's problem, then the running time of the factoring algorithm is dominated by finding the square product (i.e. *the matrix step*).

Matrix step: Wiedemann or Lanczos take time

$$\sim C \frac{y^2}{\log y \log \log y}$$

Pomerance's application

If one optimizes parameters in Pomerance's problem, then the running time of the factoring algorithm is dominated by finding the square product (i.e. *the matrix step*).

Matrix step: Wiedemann or Lanczos take time

$$\sim C\frac{y^2}{\log y \log\log y}$$

To optimize random squares need

$$y = y_1 = y_0^{1-(1+o(1))/\log\log x},$$

much smaller than y_0, so Pomerance's problem not so useful as had appeared!

Pomerance's application

If one optimizes parameters in Pomerance's problem, then the running time of the factoring algorithm is dominated by finding the square product (i.e. *the matrix step*).

Matrix step: Wiedemann or Lanczos take time

$$\sim C \frac{y^2}{\log y \log\log y}$$

To optimize random squares need

$$y = y_1 = y_0^{1-(1+o(1))/\log\log x},$$

much smaller than y_0, so Pomerance's problem not so useful as had appeared! Expected running time is:

$$J_1 := J_0 \; y_0^{(1+o(1))/(\log\log x)^2}$$

Practical speed-ups for random squares algorithm from large prime variations

Reduction in T obtained:

ℓ	$M = \infty$	$M = 100$	$M = 10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation,
each large prime in (y, My).

Practical speed-ups for random squares algorithm from large prime variations

Reduction in T obtained:

ℓ	$M = \infty$	$M = 100$	$M = 10$
0	1	1	1
1	.7499	.7517	.7677
2	.6415	.6448	.6745
3	.5962	.6011	.6422
4	.5764	.5823	.6324
5	.567	.575	.630

For ℓ-large primes variation, each large prime in (y, My).

If reduction in T here is a factor η, then the random squares algorithm is sped up by a factor

$$\approx \frac{1}{(\log x)^{\log(1/\eta)}}.$$

What practical people say

Practical considerations (not theory):

Design of the computer,

Language & implementation,

Memory issues (large arrays), swaps

How "reports" handled

Programmer's prejudices and prior experiences.

What practical people say

Practical considerations (not theory):

Design of the computer,

Language & implementation,

Memory issues (large arrays), swaps

How "reports" handled

Programmer's prejudices and prior experiences.

Recognizing y-smooths:

Quadratic sieve (& MPQS)

Early abort strategies

(for a_i's with small y_2-smooth part)

What practical people say

Practical considerations (not theory):

Design of the computer,

Language & implementation,

Memory issues (large arrays), swaps

How "reports" handled

Programmer's prejudices and prior experiences.

Recognizing y-smooths:

Quadratic sieve (& MPQS)

Early abort strategies (for a_i's with small y_2-smooth part)

Keep $a_i = b_i p_1 \dots p_j$, where b_i is y-smooth for $j \leq k$ and $p_\ell \in (y, My)$:

If M is large, then more a_j's, more pseudosmooths, but slows down sieving (as there are more primes to test).

What practical people say, II

Keep $a_i = b_i p_1 \ldots p_j$, where b_i is y-smooth for $j \leq k$ and $p_\ell \in (y, My)$:

If k is large, then more a_j's, but smaller proportion actually useful (lots of effort on useless a_j).

What practical people say, II

Keep $a_i = b_i p_1 \ldots p_j$, where b_i is y-smooth for $j \le k$ and $p_\ell \in (y, My)$:

If k is large, then more a_j's, but smaller proportion actually useful (lots of effort on useless a_j).

Fix: Discard if two larger prime factors

What practical people say, II

Keep $a_i = b_i p_1 \ldots p_j$, where b_i is y-smooth for $j \leq k$ and $p_\ell \in (y, My)$:

If k is large, then more a_j's, but smaller proportion actually useful (lots of effort on useless a_j).

Fix: Discard if two larger prime factors

Speed-ups in practice (Lenstra et al):

$.4$ to $.5$ for $k = 1$;
$.15$ to $.25$ for $k = 2$;
$.1$ and $.14$ for $k = 3$.

What practical people say, II

Keep $a_i = b_i p_1 \ldots p_j$, where b_i is y-smooth for $j \leq k$ and $p_\ell \in (y, My)$:

If k is large, then more a_j's, but smaller proportion actually useful (lots of effort on useless a_j).

Fix: Discard if two larger prime factors

Speed-ups in practice (Lenstra et al):

$.4$ to $.5$ for $k = 1$;

$.15$ to $.25$ for $k = 2$;

$.1$ and $.14$ for $k = 3$.

For more: Willemien Ekkelkamp's talk

What practical people say, II

Keep $a_i = b_i p_1 \ldots p_j$, where b_i is y-smooth for $j \le k$ and $p_\ell \in (y, My)$:

If k is large, then more a_j's, but smaller proportion actually useful (lots of effort on useless a_j).

Fix: Discard if two larger prime factors

Speed-ups in practice (Lenstra et al):

$.4$ to $.5$ for $k = 1$;

$.15$ to $.25$ for $k = 2$;

$.1$ and $.14$ for $k = 3$.

For more: Willemien Ekkelkamp's talk

For $x \approx 10^{180}$ our predictions: Speed-ups of $.41, 0.25, 0.20$ resp – not bad!

`¿Run experiments on Pomerance's problem directly?`

Methods used

First and second moment methods from probabilistic combinatorics.

Husimi graphs from statistical physics.

Lagrange inversion from algebraic combinatorics.

Analytic continuation of solutions to functional equations.

Comparative estimates on smooth numbers, via comparative estimates on saddle points.

Random graph theory, Poisson processes and conversion from continuous to discrete.