Finding small factors of integers

D. J. Bernstein
University of Illinois at Chicago

The $\mathbf{Q}$ sieve factors n
by combining enough
y-smooth congruences $i(n+i)$.

"Enough" $\approx$ "$> y/\log y$."
Plausible conjecture: if $y \in \exp\sqrt{\left(\frac{1}{2}+o(1)\right)\log n \log\log n}$
then $y^{2+o(1)}$ congruences
have enough smooth congruences.

Linear sieve, quadratic sieve,
random-squares method,
number-field sieve, etc.: similar.

Also combine congruences for
discrete logs, class groups, etc.

Finding small factors

Find smooth congruences
by finding small factors
of many congruences:

$$\begin{array}{c}
\text{Neverending supply} \\
\text{of congruences} \\
\Big\downarrow \text{select} \\
\text{Smallest congruences} \\
\Big\downarrow \text{find small factors} \\
\text{Partial factorizations} \\
\text{using primes} \le y \\
\Big\downarrow \text{abort non-smooth} \\
\text{Smooth congruences}
\end{array}$$

How to find small factors?

Could use trial division:
For each congruence,
remove factors of 2,
remove factors of 3,
remove factors of 5,
etc.; use all primes $p \le y$.

$y^{3+o(1)}$ bit operations:
$y^{1+o(1)}$ per congruence.

Want something faster!

Early aborts

$$\begin{array}{c}
\text{Neverending supply} \\
\text{of congruences} \\
\downarrow \text{select} \\
\text{Smallest congruences} \\
\downarrow \\
\text{Partial factorizations} \\
\text{using primes} \le y^{1/2} \\
\downarrow \text{early abort} \\
\text{Smallest unfactored parts} \\
\downarrow \\
\text{Partial factorizations} \\
\text{using primes} \le y \\
\downarrow \text{final abort} \\
\text{Smooth congruences}
\end{array}$$

Find small primes by trial division.
Cost $y^{1/2+o(1)}$ for primes $\le y^{1/2}$.
Cost $y^{1+o(1)}$ for primes $\le y$.

Say we choose "smallest"
so that each congruence
has chance $y^{1/2+o(1)}/y^{1+o(1)}$
of surviving early abort.
Have reduced trial-division
cost by factor $y^{1/2+o(1)}$.

Fact: A y-smooth congruence
has chance $y^{-1/4+o(1)}$
of surviving early abort.
Have reduced identify-a-smooth
cost by factor $y^{1/4+o(1)}$.

Example from Andrew Shallue:

A uniform random integer in $[1, 2^{64} - 1]$ has chance about $2^{-8.1}$ of being 2^{15}-smooth, chance about $2^{-3.5}$ of having 2^7-unfactored part below 2^{44}, and chance about $2^{-9.8}$ of satisfying both conditions.

Given congruence, find primes $\le 2^7$; abort if unfactored part is above 2^{44}; then find primes $\le 2^{15}$. Compared to skipping the abort: about $2^{3.5}$ times faster, about $2^{1.7}$ times less productive; gain $2^{1.8}$.

More generally, can abort at $y^{1/k}$, $y^{2/k}$, etc. Balance stages to reduce cost per congruence from $y^{1+o(1)}$ to $y^{1/k+o(1)}$.

Fact: A y-smooth congruence has relatively good chance of surviving early abort.
Have reduced identify-a-smooth cost by factor $y^{(1-1/k)/2+o(1)}$.

Increase k slowly with y.
Find enough smooth congruences using $y^{2.5+o(1)}$ bit operations.

Want something faster!

Sieving

Textbook answer: Sieving finds enough smooth congruences using only $y^{2+o(1)}$ bit operations.

To sieve: Generate in order of p, then sort in order of i, all pairs (i, p) with i in range and $i(n+i) \in p\mathbf{Z}$.

Pairs for one p are (p, p), $(2p, p)$, $(3p, p)$, etc. and $(p - (n \bmod p), p)$ etc.

e.g. $y = 10$, $n = 611$, $i \in \{1, 2, \ldots, 100\}$:

For $p = 2$ generate pairs $(2, 2), (4, 2), (6, 2), \ldots, (100, 2)$ and $(1, 2), (3, 2), (5, 2), \ldots, (99, 2)$.

For $p = 3$ generate pairs $(3, 3), (6, 3), \ldots, (99, 3)$ and $(1, 3), (4, 3), \ldots, (100, 3)$.

For $p = 5$ generate pairs $(5, 5), (10, 5), \ldots, (100, 5)$ and $(4, 5), (9, 5), \ldots, (99, 5)$.

For $p = 7$ generate pairs $(7, 7), (14, 7), \ldots, (98, 7)$ and $(5, 7), (12, 7), \ldots, (96, 7)$.

Sort pairs by first coordinate:

$(1, 2)$, $(1, 3)$, $(2, 2)$, $(3, 2)$, $(3, 3)$, $(4, 2)$, $(4, 3)$, $(4, 5)$, $\ldots$, $(98, 2)$, $(98, 7)$, $(99, 2)$, $(99, 3)$, $(99, 5)$, $(100, 2)$, $(100, 3)$, $(100, 5)$.

Sorted list shows that
the small primes in $i(n + i)$ are
$2, 3$ for $i = 1$;
2 for $i = 2$;
$\ldots$
$2, 7$ for $i = 98$;
$2, 3, 5$ for $i = 99$;
$2, 3, 5$ for $i = 100$.

In general, for $i \in \{1, \ldots, y^2\}$:

Prime p produces $\approx y^2/p$ pairs (p, p), $(2p, p)$, $(3p, p)$, etc. and produces $\approx y^2/p$ pairs $(p - (n \bmod p), p)$ etc.

Total number of pairs $\approx \sum_{p \le y} 2y^2/p \approx 2y^2 \log\log y$.

Easily generate pairs, sort, and finish checking smoothness, in $y^2(\lg y)^{O(1)}$ bit operations. Only $(\lg y)^{O(1)}$ bit operations per congruence.

Hidden costs

Is that what we do
in record-setting factorizations?
No!

Sieving has two big problems.

First problem:
Sieving needs large i range.
For speed, must use batch of
$\geq y^{1+o(1)}$ consecutive i's.
Limits number of sublattices,
so limits smoothness chance.

Can eliminate this problem
using "remainder trees."

Product trees

Given $c_1, c_2, \ldots, c_m$,
together having $y(\lg y)^{O(1)}$ bits:

Can compute $c_1 c_2 \cdots c_m$
with $y(\lg y)^{O(1)}$ operations.

Actually compute
"product tree" of $c_1, c_2, \ldots, c_m$.
Root: $c_1 c_2 \cdots c_m$.
Left subtree if $m \geq 2$:
product tree of $c_1, \ldots, c_{\lceil m/2 \rceil}$.
Right subtree if $m \geq 2$:
product tree of $c_{\lceil m/2 \rceil + 1}, \ldots, c_m$.

e.g. tree for $23, 29, 84, 15, 58, 19$:

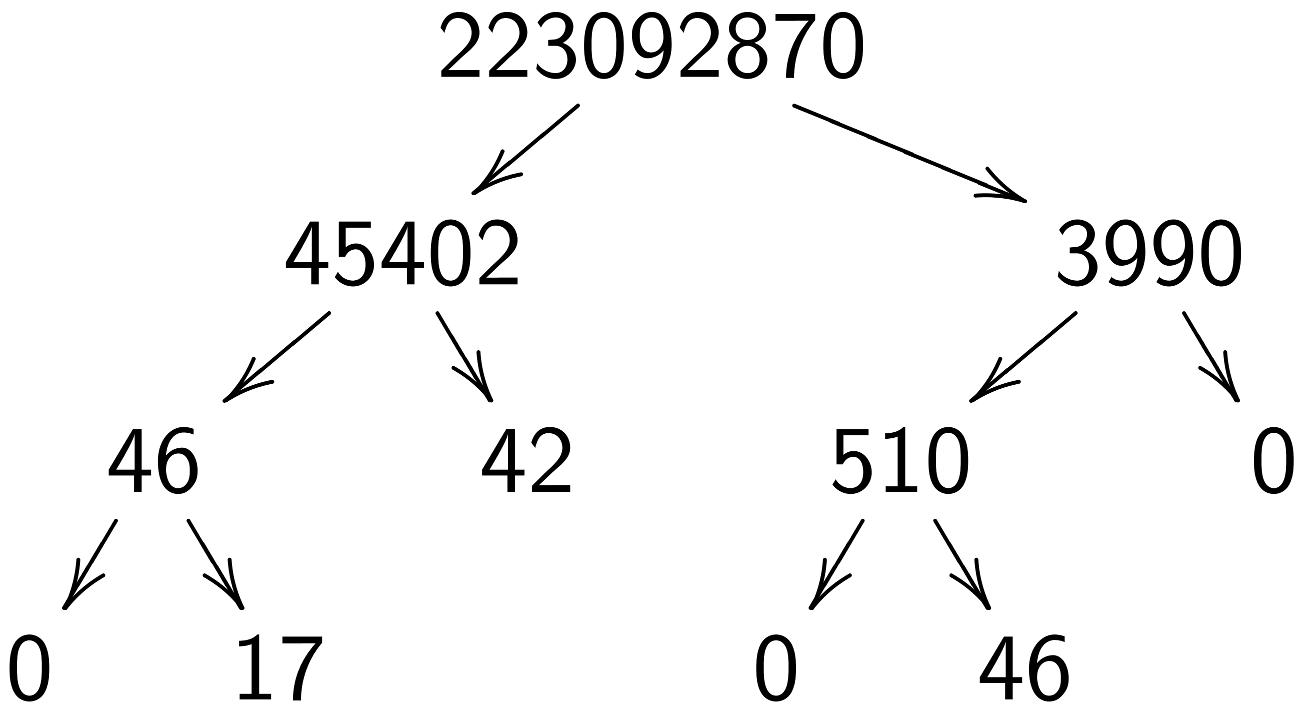
$$\begin{tikzcd}[column sep=small]
& & & & 926142840 & & & & \\
& & 56028 \arrow[urr] & & & & & 16530 \arrow[ulll] & \\
& 667 \arrow[ur] & & 84 \arrow[ul] & & & 870 \arrow[ur] & & 19 \arrow[ul] \\
23 \arrow[ur] & & 29 \arrow[ul] & & & 15 \arrow[ur] & & 58 \arrow[ul] &
\end{tikzcd}$$

Obtain each level of tree
with $y(\lg y)^{O(1)}$ operations
by multiplying lower-level pairs.
Use FFT-based multiplication.

Remainder trees

Remainder tree of $P, c_1, c_2, \ldots, c_m$ has one node $P \bmod C$ for each node C in product tree of $c_1, c_2, \ldots, c_m$.

e.g. remainder tree of $223092870, 23, 29, 84, 15, 58, 19$:



Use product tree to compute product P of primes $p \le y$.

Use remainder tree to compute $P \bmod c_1, P \bmod c_2, \ldots$.

Now c_1 is y-smooth iff $P^{2^k} \bmod c_1 = 0$ for minimal $k \ge 0$ with $2^{2^k} \ge c_1$. Similarly c_2 etc.

Total $y(\lg y)^{O(1)}$ operations if $c_1, c_2, \ldots$ together have $y(\lg y)^{O(1)}$ bits.

Hidden costs, continued

Second problem with sieving, not fixed by remainder trees: Need $y^{1+o(1)}$ bits of storage.

Real machines don't have much fast memory: it's expensive.

Effect is not visible for small computations on single serial CPUs, but becomes critical in huge parallel computations.

How to quickly find primes above size of fast memory?

The rho method

Define $\rho_0 = 0$, $\rho_{k+1} = \rho_k^2 + 11$.

Every prime $\le 2^{20}$ divides $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)(\rho_3 - \rho_6) \cdots (\rho_{3575} - \rho_{7150})$.
Also many larger primes.

Can compute $\gcd\{c, S\}$ using $\approx 2^{14}$ multiplications mod c, very little memory.

Compare to $\approx 2^{16}$ divisions for trial division up to 2^{20}.

More generally: Choose z.
Compute $\gcd\{c, S\}$ where $S = (\rho_1 - \rho_2)(\rho_2 - \rho_4)\cdots(\rho_z - \rho_{2z})$.

How big does z have to be for all primes $\le y$ to divide S?

Plausible conjecture: $y^{1/2+o(1)}$; so $y^{1/2+o(1)}$ mults mod c.
Early-abort rho: $y^{1/4+o(1)}$ mults.

Reason: Consider first collision in $\rho_1 \bmod p, \rho_2 \bmod p, \ldots$.
If $\rho_i \bmod p = \rho_j \bmod p$ then $\rho_k \bmod p = \rho_{2k} \bmod p$ for $k \in (j-i)\mathbf{Z} \cap [i, \infty] \cap [j, \infty]$.

The $p - 1$ method

Have built an integer S
divisible by all primes $\leq y$.
Less costly way to do this?

First attempt: Choose z.
Define $S_1 = 2^{\text{lcm}\{1,2,3,\ldots,z\}} - 1$.

If $\text{lcm} \in (p-1)\mathbf{Z}$ then $S_1 \in p\mathbf{Z}$.

Can tweak to find more p's:
e.g., could instead use product
of $2^{\text{lcm}} - 1$ and $2^{\text{lcm}\cdot q} - 1$
for all primes $q \in [z+1, z \log z]$;
could replace lcm by lcm^2.

e.g. $z = 20$:

$$\begin{aligned} \mathrm{lcm} &= \mathrm{lcm}\{1, 2, 3, \ldots, 20\} \\ &= 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \\ &= 232792560. \end{aligned}$$

$S_1 = 2^{\mathrm{lcm}} - 1$ has prime divisors 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 53, 61, 67, 71, 73, 79, 89, 97, 103, 109, 113, 127, 131, 137, 151, 157, 181, 191, 199, etc.

Compute S_1 with 34 mults.

As $z \to \infty$: $(1.44\ldots + o(1))z$ multiplications to compute S_1.

Dividing $\mathrm{lcm}\{1,\ldots,z\}$ is stronger than z-smoothness but not much.

Plausible conjecture: if $z \in \exp\sqrt{\left(\frac{1}{2}+o(1)\right)\log y \log\log y}$ then $p-1$ divides $\mathrm{lcm}\{1,\ldots,z\}$ with chance $1/z^{1+o(1)}$ for uniform random prime $p \le y$.

So method finds some primes at surprisingly high speed.
What about the other primes?

The $p+1$ method

Second attempt:
Define $v_0 = 2$, $v_1 = 10$,
$v_{2i} = v_i^2 - 2$,
$v_{2i+1} = v_i v_{i+1} - v_1$.
Define $S_2 = v_{\mathrm{lcm}\{1,2,3,\ldots,z\}} - 2$.

Point of v_i formulas:
$v_i = \alpha^i + \alpha^{-i}$
in $\mathbf{Z}[\alpha]/(\alpha^2 - 10\alpha + 1)$.

If $\mathrm{lcm}\{1, 2, 3, \ldots, z\} \in (p+1)\mathbf{Z}$
and $10^2 - 4$ non-square in $\mathbf{F}_p$
then $\mathbf{F}_p[\alpha]/(\alpha^2 - 10\alpha + 1)$
is a field so $S_2 \in p\mathbf{Z}$.

e.g. $z = 20$, lcm $= 232792560$:

$S_2 = v_{\text{lcm}} - 2$ has prime divisors 3, 5, 7, 11, 13, 17, 19, 23, 29, 37, 41, 43, 53, 59, 67, 71, 73, 79, 83, 89, 97, 103, 109, 113, 131, 151, 179, 181, 191, 211, 227, 233, 239, 241, 251, 271, 307, 313, 331, 337, 373, 409, 419, 439, 457, 467, 547, 569, 571, 587, 593, 647, 659, 673, 677, 683, 727, 857, 859, 881, 911, 937, 967, 971, etc.

The elliptic-curve method

Fix $a \in \{6, 10, 14, 18, \ldots\}$.

Define $x_1 = 2$, $d_1 = 1$,
$x_{2i} = (x_i^2 - d_i^2)^2$,
$d_{2i} = 4x_i d_i(x_i^2 + a x_i d_i + d_i^2)$,
$x_{2i+1} = 4(x_i x_{i+1} - d_i d_{i+1})^2$,
$d_{2i+1} = 8(x_i d_{i+1} - d_i x_{i+1})^2$.

Define $S_a = d_{\mathrm{lcm}\{1,2,3,\ldots,z\}}$.

Have now supplemented S_1, S_2
with S_6, S_{10}, S_{14}, etc.
Variability of a is important.

Point of x_i, d_i formulas:

If $d_i(a^2-4)(4a+10) \notin p\mathbf{Z}$ then ith multiple of $(2,1)$ on the elliptic curve $(4a+10)y^2 = x^3 + ax^2 + x$ over $\mathbf{F}_p$ is $(x_i/d_i, \ldots)$.

If $(a^2-4)(4a+10) \notin p\mathbf{Z}$ and lcm $\in$ (order of $(2,1)$)$\mathbf{Z}$ then $S_a \in p\mathbf{Z}$.

Order of elliptic-curve group depends on a but is always in $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$.

e.g. $z = 20$, $a = 10$, $p = 105239$:

p divides S_{10}.

Have $232792560(2,1) = \infty$
on the elliptic curve
$50y^2 = x^3 + 10x^2 + x$ over $\mathbf{F}_p$.

In fact, $(2,1)$ has order
$13167 = 3^2 \cdot 7 \cdot 11 \cdot 19$
on this curve.

Number of $\mathbf{F}_p$-points of curve
is $105336 = 2^3 \cdot 3^2 \cdot 7 \cdot 11 \cdot 19$.

Consider smallest z
such that product of S_a
for first z choices of a
is divisible by every $p \le y$.

Plausible conjecture: $z \in$
$\exp\sqrt{\left(\frac{1}{2}+o(1)\right)\log y \log\log y}$.

Computing this product
takes $\approx 12z^2$ mults; i.e.
$\exp\sqrt{(2+o(1))\log y \log\log y}$.

Early-abort ECM:
$\exp\sqrt{(8/9+o(1))\log y \log\log y}$
after careful optimization.

Are all primes small?

Instead of using these methods to find smooth congruences c, can apply them directly to n.

Worst case: n is product of two primes $\approx \sqrt{n}$.

Take $y \approx \sqrt{n}$.
Number of mults mod n in elliptic-curve method:
$\exp \sqrt{(2+o(1)) \log y \log \log y} =$
$\exp \sqrt{(1+o(1)) \log n \log \log n}$.

Faster than $\mathbf{Q}$ sieve.
Comparable to quadratic sieve, using much less memory.

Slower than number-field sieve for sufficiently large n.

One elliptic-curve computation found a prime $\approx 2^{219}$ in $\approx 3 \cdot 10^{12}$ Opteron cycles.
Fairly lucky in retrospect.