How to find smooth parts of integers

D. J. Bernstein

Thanks to:
University of Illinois at Chicago
NSF DMS–0140542
Alfred P. Sloan Foundation

Prototypical factorization algorithm: continued-fraction method.
(1931 Lehmer Powers,
1975 Morrison Brillhart)

Given $n = 31415926535897932 3$:
Compute good approximations
$\sqrt{n} \approx 560499122/1$,
$\sqrt{n} \approx 1120998243/2$,
$\sqrt{n} \approx 1681497365/3$,
$\sqrt{n} \approx 6165490338/11$,
$\sqrt{n} \approx 14012478041/25$,
etc. via Euclid's algorithm.

$p^2 \equiv$ small $\pmod{n}$ if $\sqrt{n} \approx p/q$:

$560499122^2 \equiv 403791561$,

$1120998243^2 \equiv -626830243$,

$1681497365^2 \equiv 271129318$,

$6165490338^2 \equiv -465143839$,

$14012478041^2 \equiv 145120806$, etc.

Find nonempty subsequence of

$403791561, -626830243, \ldots$

with square product.

The p^2's also have square product.

Hope $1 < \gcd\{\sqrt{} - \sqrt{}, n\} < n$.

How to find square product
among first few thousand numbers?

Numbers with large prime factors
are useless.

But many numbers are **smooth**:
$145120806 = 2 \cdot 3^2 \cdot 17 \cdot 647 \cdot 733$;
$-521969851 = -13^3 \cdot 193 \cdot 1231$;
etc.

Recognize these smooth numbers;
find their exponent vectors;
do linear algebra on vectors mod 2 to
find nonempty subset with even sum.

$$
\begin{aligned}
-5421351 &= -3 \cdot 13 \cdot 13 \cdot 17 \cdot 17 \cdot 37, \\
454304721 &= 3 \cdot 13 \cdot 31 \cdot 613 \cdot 613, \\
401224998 &= 2 \cdot 3 \cdot 193 \cdot 317 \cdot 1093, \\
-362966643 &= -3 \cdot 3 \cdot 3 \cdot 13 \cdot 17 \cdot 59 \cdot 1031, \\
-461281298 &= -2 \cdot 17 \cdot 83 \cdot 223 \cdot 733, \\
68104737 &= 3 \cdot 3 \cdot 17 \cdot 31 \cdot 83 \cdot 173, \\
278236113 &= 3 \cdot 101 \cdot 859 \cdot 1069, \\
-443339082 &= -2 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 89 \cdot 97 \cdot 317, \\
258865542 &= 2 \cdot 3 \cdot 3 \cdot 13 \cdot 29 \cdot 37 \cdot 1031, \\
13005213 &= 3 \cdot 13 \cdot 31 \cdot 31 \cdot 347, \\
-185619402 &= -2 \cdot 3 \cdot 3 \cdot 131 \cdot 223 \cdot 353, \\
-308945194 &= -2 \cdot 31 \cdot 47 \cdot 97 \cdot 1093, \\
88949286 &= 2 \cdot 3 \cdot 3 \cdot 3 \cdot 47 \cdot 101 \cdot 347, \\
733202886 &= 2 \cdot 3 \cdot 13 \cdot 31 \cdot 353 \cdot 859, \\
162594973 &= 59 \cdot 109 \cdot 131 \cdot 193, \\
143972541 &= 3 \cdot 3 \cdot 17 \cdot 89 \cdot 97 \cdot 109, \\
312539253 &= 3 \cdot 13 \cdot 89 \cdot 127 \cdot 709, \\
96382078 &= 2 \cdot 13 \cdot 17 \cdot 17 \cdot 101 \cdot 127, \\
-70194923 &= -47 \cdot 89 \cdot 97 \cdot 173, \\
244225878 &= 2 \cdot 3 \cdot 13 \cdot 29 \cdot 101 \cdot 1069, \\
-219831831 &= -3 \cdot 3 \cdot 47 \cdot 709 \cdot 733.
\end{aligned}
$$

These have square product.

Obtain divisor 990371647 of n.

Given set P of primes
and sequence S of numbers,
can factor S over P
in time $b(\lg b)^{3+o(1)}$
where b is number of input bits.
(2000 Bernstein)

Much faster than handling
each element of S separately
by trial division,
Pollard's rho method,
Pollard-Williams-Lenstra
smooth-sized-group methods, etc.

Batch factorization algorithm:

1. Compute $y = \prod_{x \in S} x$.
(Product tree; standard.)

2. Compute $y \bmod p$ for each $p \in P$.
(1972 Moenck Borodin.)

3. Discard p's not dividing y.

4. If $\#S \leq 1$, done.
(For exponents: 1995 Bernstein.)

5. Recursively handle halves of S.

Use fast multiplication everywhere.
(1971 Pollard, 1971 Nicholson,
1971 Schönhage Strassen)

This is not the best way to recognize P-smooth numbers! Can usually achieve $b(\lg b)^{2+o(1)}$.

(2004 Franke Kleinjung Morain Wirth; buried inside paper on ECPP; no recognition of speedup; no serious analysis; grrr)

Then use previous algorithm to factor the smooth numbers. Usually not many smooth numbers, so this is fast.

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\text{Positive integer } x \arrow[d, "\substack{\text{batch time}\\ b(\lg b)^{3+o(1)}\\ \text{(2000 Bernstein)}}"] \arrow[dr, bend left, "\substack{\text{batch time usually } b(\lg b)^{2+o(1)}\\ \text{(2004 Franke Kleinjung Morain Wirth);}\\ \text{batch time } b(\lg b)^{2+o(1)}\\ \text{(2004 Bernstein)}}"] & \\
\text{Small factors of } x \arrow[r, "\substack{\text{time } b(\lg b)^{2+o(1)}\\ \text{(standard)}}"'] \arrow[d] & \text{Is } x \text{ smooth?} \arrow[dl, bend left, "\substack{\text{batch time}\\ \text{at worst } b(\lg b)^{3+o(1)};\\ \text{usually negligible}}"] \\
\substack{\text{Small factors of } x\\ \text{if } x \text{ is smooth}} &
\end{tikzcd}
$$

The usually-better algorithm:

1. Compute $z = \prod_{p \in P} p$.
2. Compute $z \bmod x$ for each $x \in S$.
3. Repeatedly divide x by $\gcd\{z \bmod x, x\}$.

Step 3 might take many iterations.

Better, guaranteeing $b(\lg b)^{2+o(1)}$:
Compute $(z \bmod x)^{\text{big}} \bmod x$.
(2004 Bernstein; many precedents)

Many constant-factor speedups:
FFT doubling (2004 Kramer) et al.

In newer algorithms for factorization, discrete logs, etc.: Often numbers are sieveable. (introduced by 1977 Schroeppel)

Sieve up to (largest prime)$^\theta$; discard if not too promising; then use batch smoothness method.

Total time is roughly $RS^\theta T^{1-\theta}$ where R is smoothness ratio, S is sieve time per number, T is batch time per number. (see 1982 Pomerance)

S is annoyingly high if
sieve doesn't fit into DRAM,
so take $\theta < 1$.
(standard; e.g. factorization of
RSA-155 used non-optimal $\theta = 0.8$)

Consequence: Reducing T helps.

When T is small enough,
should choose θ to sieve in L2 cache,
maybe even L1 cache,
so as to reduce S further;
makes T even more important.
(2000 Bernstein)

`http://cr.yp.to/papers.html`

`#dcba` "Factoring into coprimes in essentially linear time"

`#sf` "How to find small factors of integers"

`#multapps` "Fast multiplication and its applications"

`#smoothparts` "How to find smooth parts of integers"

Forthcoming: "Sieving in cache"