# Applications of fast multiplication

D. J. Bernstein
University of Illinois at Chicago

# Power-series product

Recall: a power series $f \in A[[x]]$ is a formal sum $f_0 + f_1 x + f_2 x^2 + \cdots$ with each $f_j \in A$.

Approximate $f$ by the polynomial $f \bmod x^n = f_0 + \cdots + f_{n-1} x^{n-1}$.

Given $f \bmod x^n$ and $g \bmod x^n$, can compute $fg \bmod x^n$ with $A$-complexity $O(n \lg n \lg\lg n)$.

## Power-series reciprocal

$f \in A[[x]]$ with $f_0 = 1$.
Given approximation to $f$.
Want approximation to $1/f$.

Fact: If $(1/f) \bmod x^n = z$
then $(1/f) \bmod x^{2n} =$
$z - (fz - 1)z \bmod x^{2n}$.

$A$-complexity $O(n \lg n \lg \lg n)$
for $(1/f) \bmod x^n$ given $f \bmod x^n$.

## Newton's method

Differentiable partial function $p$.
Want to find a root of $p$.

General idea:
If $z$ is "close" to a root of $p$
then $z - p(z)/p'(z)$ is "closer."
Fast convergence to simple roots.

For $p = (z \mapsto 1 - 1/fz)$:
$p/p' = (z \mapsto (fz - 1)z)$.

## Power-series quotient

$f, g \in A[[x]]$ with $f_0 = 1$.

$A$-complexity $O(n \lg n \lg \lg n)$
for $(g/f) \bmod x^n$
given $f \bmod x^n$, $g \bmod x^n$.

More precisely:
$4 + o(1)$ times multiplication.
(Cook; Sieveking; Kung; Brent)

Eliminate redundant FFTs.
Use higher-order iteration.
Merge quotient with reciprocal.

$13/6 + o(1)$ times multiplication.

(Schönhage; A. Karp, Markstein, U.S. Patent 5,341,321; Brent; Harley; Zimmermann; Bernstein)

<u>What about **Z**?</u>

Circuit of size $O(n \lg n \lg \lg n)$
can compute $n$-bit approximation
to a quotient in **R**.

Same idea as in $A[[x]]$;
more numerical analysis.

Or a quotient in $\mathbf{Z}_2$:
given $g \in \mathbf{Z}$ and odd $f \in \mathbf{Z}$,
find $h \in \mathbf{Z}$ with $hf \equiv g \pmod{2^n}$.

## Power-series logarithm

**R**-complexity $(12 + o(1))n \lg n$
to multiply in $\mathbf{R}[[x]]$.

Given $f \in \mathbf{R}[[x]]$, $f_0 = 1$.
Want $\log f$.

Use $(\log f)' = f'/f$.
**R**-complexity $(26 + o(1))n \lg n$.

## Power-series exponential

Given $f \in \mathbf{R}[[x]]$, $f_0 = 0$.
Want $\exp f$.

Use Newton's method to find root of $p = (z \mapsto \log z - f)$.
Note $p/p' = (z \mapsto (\log z - f)z)$.

$\mathbf{R}$-complexity $(34 + o(1))n \lg n$.

## Counting smooth polynomials

A polynomial in $\mathbf{F}_2[t]$ is **smooth** if it is a product of polynomials of degree $\le 30$.

$\sum_{n \in \mathbf{F}_2[t],\, n \text{ smooth}} x^{\deg n}$
$= \prod_{k \le 30} 1/(1 - x^k)^{c_k}$
$= \exp \sum_{k \le 30} c_k (x^k + \frac{1}{2} x^{2k} + \cdots)$
where $c_k = (1/k) \sum_{d|k} 2^d \mu(k/d)$.

Not so easy to approximate $\log f$ or $\exp f$ for $f \in \mathbf{R}$.

Circuit size $n(\lg n)^{O(1)}$ using arithmetic-geometric mean or fast Taylor-series summation.

(Gauss; Legendre; Landen; Beeler; Gosper; Schroeppel; Salamin; Brent)

# Multiplying many numbers

Given $x_1, x_2, \ldots, x_m \in \mathbf{Z}$,
$n$ bits together, $m \geq 1$.
Want $x_1 x_2 \cdots x_m$.

Method for $m$ even: $x_1 x_2 \cdots x_m$
$= (x_1 \cdots x_{m/2})(x_{m/2+1} \cdots x_m)$.

Circuit size $O(n \lg n \lg \lg n \lg m)$.

Need a balanced splitting.
Otherwise too much recursion.

Can measure balance
by total bits instead of $m$.
Replaces $\lg m$ by
entropy of $x_j$ size distribution.
(Strassen)

## Continued fractions

$5 + 1/(2 + 1/(1 + 1/(1 + 1/3)))$
$= 97/18.$

$C(5)C(2)C(1)C(1)C(3) = \begin{pmatrix} 97 & 27 \\ 18 & 5 \end{pmatrix}$
where $C(a) = \begin{pmatrix} a & 1 \\ 1 & 0 \end{pmatrix}$.

Given $a_1, a_2, \ldots, a_m$,
can quickly compute
$C(a_1)C(a_2) \cdots C(a_m)$.

Given $f, g \in \mathbf{Z}$,
can quickly compute $\gcd\{f, g\}$
and the continued fraction for $f/g$.

Circuit size $O(n(\lg n)^2 \lg\lg n)$.

(Lehmer; Knuth; Schönhage;
Brent, Gustavson, Yun)

<u>Multipoint evaluation</u>

Given positive $f, q_1, \ldots, q_m \in \mathbf{Z}$.
Want each $f \bmod q_j$.

Method for $m$ even:
Recursively do the same for
$f, q_1 q_2, \ldots, q_{m-1} q_m$.

Circuit size $O(n \lg n \lg \lg n \lg m)$.

(Borodin, Moenck)

## Finding small factors

Given a set $P$ of primes,
a set $S$ of nonzero integers.
Want to partly factor $S$ using $P$.

Method: Find $g = \prod_{f \in S} f$.
Find $Q = \{q \in P : g \bmod q = 0\}$.
If $\#S \leq 1$, print $(Q, S)$ and stop.
Choose $T \subseteq S$, half size.
Handle $Q, T$. Handle $Q, S - T$.

Circuit size $n(\lg n)^{O(1)}$.

In particular: Given $y$ integers, each with $(\lg y)^{O(1)}$ bits, can recognize and factor the $y$-smooth integers. Circuit size $(\lg y)^{O(1)}$ per integer.

# Factoring into coprimes

Given a set $S$ of positive integers:
Can find a coprime set $P$
and completely factor $S$ using $P$.

Coprime means $\gcd\{q, q'\} = 1$
for all $q, q' \in P$ with $q \neq q'$.

Circuit size $n(\lg n)^{O(1)}$.