High-speed cryptography, part 4: fast multiplication and its applications

Daniel J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Survey paper:
cr.yp.to/papers.html#multapps

Integer-factorization bottleneck:
Given sequence of numbers, find nonempty subsequence with square product.
e.g. given $6, 7, 8, 10, 15$, discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers, find 1 as nontrivial product of powers.
e.g. given $6, 7, 8, 10, 15$, discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

eed cryptography,

tiplication
applications

. Bernstein
ty of Illinois at Chicago &
che Universiteit Eindhoven

paper:
to/papers.html#multapps

Integer-factorization bottleneck:
Given sequence of numbers,
find nonempty subsequence
with square product.
e.g. given $6, 7, 8, 10, 15$,
discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers,
find 1 as nontrivial
product of powers.
e.g. given $6, 7, 8, 10, 15$,
discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

Two ver
cryptogr
Multiply
multiply

graphy,

ns

n
is at Chicago &
siteit Eindhoven

s.html#multapps

Integer-factorization bottleneck:
Given sequence of numbers,
find nonempty subsequence
with square product.
e.g. given $6, 7, 8, 10, 15$,
discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers,
find 1 as nontrivial
product of powers.
e.g. given $6, 7, 8, 10, 15$,
discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

Two very commor
cryptographic bott
Multiply large poly
multiply large inte

ago &

hoven

ultapps

Integer-factorization bottleneck:
Given sequence of numbers,
find nonempty subsequence
with square product.
e.g. given $6, 7, 8, 10, 15$,
discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers,
find 1 as nontrivial
product of powers.
e.g. given $6, 7, 8, 10, 15$,
discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

Two very common
cryptographic bottlenecks:
Multiply large polynomials;
multiply large integers.

Integer-factorization bottleneck:
Given sequence of numbers,
find nonempty subsequence
with square product.
e.g. given $6, 7, 8, 10, 15$,
discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers,
find 1 as nontrivial
product of powers.
e.g. given $6, 7, 8, 10, 15$,
discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

Two very common
cryptographic bottlenecks:
Multiply large polynomials;
multiply large integers.

Integer-factorization bottleneck:
Given sequence of numbers,
find nonempty subsequence
with square product.
e.g. given $6, 7, 8, 10, 15$,
discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers,
find 1 as nontrivial
product of powers.
e.g. given $6, 7, 8, 10, 15$,
discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

Two very common
cryptographic bottlenecks:
Multiply large polynomials;
multiply large integers.

All of these computations
can be performed in
essentially *linear* time.

Integer-factorization bottleneck:
Given sequence of numbers,
find nonempty subsequence
with square product.
e.g. given $6, 7, 8, 10, 15$,
discover $6 \cdot 10 \cdot 15 = 30^2$.

Discrete-log bottleneck:
Given sequence of numbers,
find 1 as nontrivial
product of powers.
e.g. given $6, 7, 8, 10, 15$,
discover $6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

More generally: find kth power.

Two very common
cryptographic bottlenecks:
Multiply large polynomials;
multiply large integers.

All of these computations
can be performed in
essentially *linear* time.

Do real applications
reach large enough sizes
to benefit from these techniques?
In cryptanalysis, definitely.
In cryptography, sometimes:
Gaudry–Schost Kummer surface;
McBits; many more examples.

factorization bottleneck:
equence of numbers,
empty subsequence
uare product.
en $6, 7, 8, 10, 15,$
$6 \cdot 10 \cdot 15 = 30^2$.

-log bottleneck:
equence of numbers,
s nontrivial
of powers.
en $6, 7, 8, 10, 15,$
$6^3 7^0 8^{-2} 10^3 15^{-3} = 1$.

nerally: find kth power.

Two very common
cryptographic bottlenecks:
Multiply large polynomials;
multiply large integers.

All of these computations
can be performed in
essentially *linear* time.

Do real applications
reach large enough sizes
to benefit from these techniques?
In cryptanalysis, definitely.
In cryptography, sometimes:
Gaudry–Schost Kummer surface;
McBits; many more examples.

The fast

Use $(c_0,$
to repres

Summar
"f has
f does n

$f = f_0($
$(c_0, c_2, .$
$(c_1, c_3, .$
represen

$\mathbf{C}[x]$-mo
from $\mathbf{C}[$
maps f_0

on bottleneck:
numbers,
bsequence
ct.
$10, 15,$
$= 30^2.$

eneck:
numbers,
l
.
$10, 15,$
$0^3 15^{-3} = 1.$

nd kth power.

Two very common
cryptographic bottlenecks:
Multiply large polynomials;
multiply large integers.

All of these computations
can be performed in
essentially *linear* time.

Do real applications
reach large enough sizes
to benefit from these techniques?
In cryptanalysis, definitely.
In cryptography, sometimes:
Gaudry–Schost Kummer surface;
McBits; many more examples.

The fast Fourier tr

Use $(c_0, c_1, \ldots, c_n$
to represent $f = \sum$

Summary of repres
"f has n coeffs".
f does not determ

$f = f_0(x^2) + x f_1($
$(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/}$
$(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/}$
represent f_0, f_1 re

$\mathbf{C}[x]$-morphism y
from $\mathbf{C}[x][y]$ to $\mathbf{C}$
maps $f_0(y) + x f_1$

eck:

1.

wer.

Two very common cryptographic bottlenecks: Multiply large polynomials; multiply large integers.

All of these computations can be performed in essentially *linear* time.

Do real applications reach large enough sizes to benefit from these techniques? In cryptanalysis, definitely. In cryptography, sometimes: Gaudry–Schost Kummer surface; McBits; many more examples.

The fast Fourier transform

Use $(c_0, c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$ to represent $f = \sum_j c_j x^j \in$

Summary of representation s "f has n coeffs". Warning: f does not determine n.

$f = f_0(x^2) + x f_1(x^2)$ where $(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$, $(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$ represent f_0, f_1 respectively.

$\mathbf{C}[x]$-morphism $y \mapsto x^2$ from $\mathbf{C}[x][y]$ to $\mathbf{C}[x]$ maps $f_0(y) + x f_1(y)$ to f.

Two very common cryptographic bottlenecks: Multiply large polynomials; multiply large integers.

All of these computations can be performed in essentially *linear* time.

Do real applications reach large enough sizes to benefit from these techniques? In cryptanalysis, definitely. In cryptography, sometimes: Gaudry–Schost Kummer surface; McBits; many more examples.

The fast Fourier transform

Use $(c_0, c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$ to represent $f = \sum_j c_j x^j \in \mathbf{C}[x]$.

Summary of representation size: "f has n coeffs". Warning: f does not determine n.

$f = f_0(x^2) + x f_1(x^2)$ where $(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$, $(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$ represent f_0, f_1 respectively.

$\mathbf{C}[x]$-morphism $y \mapsto x^2$ from $\mathbf{C}[x][y]$ to $\mathbf{C}[x]$ maps $f_0(y) + x f_1(y)$ to f.

y common
aphic bottlenecks:
large polynomials;
large integers.

ese computations
performed in
ly *linear* time.

applications
rge enough sizes
fit from these techniques?
analysis, definitely.
ography, sometimes:
-Schost Kummer surface;
many more examples.

The fast Fourier transform

Use $(c_0, c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$
to represent $f = \sum_j c_j x^j \in \mathbf{C}[x]$.

Summary of representation size:
"f has n coeffs". Warning:
f does not determine n.

$f = f_0(x^2) + x f_1(x^2)$ where
$(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$,
$(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$
represent f_0, f_1 respectively.

$\mathbf{C}[x]$-morphism $y \mapsto x^2$
from $\mathbf{C}[x][y]$ to $\mathbf{C}[x]$
maps $f_0(y) + x f_1(y)$ to f.

Quickly
by evalu
$f(\alpha) =$
$f(-\alpha) =$

Evaluate
all $\alpha \in$ C
by evalu
for all β
plus 102

Apply th
$n \lg n$ ac
to evalu
for all α
if n is a

n
tlenecks:
ynomials;
gers.

utations
in
ime.

ns
n sizes
ese techniques?
efinitely.
ometimes:
ummer surface;
re examples.

The fast Fourier transform

Use $(c_0, c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$
to represent $f = \sum_j c_j x^j \in \mathbf{C}[x]$.

Summary of representation size:
"f has n coeffs". Warning:
f does not determine n.

$f = f_0(x^2) + x f_1(x^2)$ where
$(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$,
$(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$
represent f_0, f_1 respectively.

$\mathbf{C}[x]$-morphism $y \mapsto x^2$
from $\mathbf{C}[x][y]$ to $\mathbf{C}[x]$
maps $f_0(y) + x f_1(y)$ to f.

Quickly evaluate f
by evaluating $f_0(\alpha$
$f(\alpha) = f_0(\alpha^2) +$
$f(-\alpha) = f_0(\alpha^2) -$

Evaluate $f(\alpha)$ for,
all $\alpha \in \mathbf{C}$ with α^{10}
by evaluating $f_0(\beta$
for all $\beta \in \mathbf{C}$ with
plus 1024 adds, 51

Apply this recursiv
$n \lg n$ adds, $(n/2)$
to evaluate n-coef
for all $\alpha \in \mathbf{C}$ with
if n is a power of

ques?

face;

es.

The fast Fourier transform

Use $(c_0, c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$ to represent $f = \sum_j c_j x^j \in \mathbf{C}[x]$.

Summary of representation size: "f has n coeffs". Warning: f does not determine n.

$f = f_0(x^2) + x f_1(x^2)$ where $(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$, $(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$ represent f_0, f_1 respectively.

$\mathbf{C}[x]$-morphism $y \mapsto x^2$ from $\mathbf{C}[x][y]$ to $\mathbf{C}[x]$ maps $f_0(y) + x f_1(y)$ to f.

Quickly evaluate $f(\alpha), f(-\alpha$ by evaluating $f_0(\alpha^2)$; $f_1(\alpha^2$

$$f(\alpha) = f_0(\alpha^2) + \alpha f_1(\alpha^2)$$

$$f(-\alpha) = f_0(\alpha^2) - \alpha f_1(\alpha^2)$$

Evaluate $f(\alpha)$ for, e.g., all $\alpha \in \mathbf{C}$ with $\alpha^{1024} = 1$ by evaluating $f_0(\beta)$, $f_1(\beta)$ for all $\beta \in \mathbf{C}$ with $\beta^{512} = 1$; plus 1024 adds, 512 mults.

Apply this recursively $\Rightarrow$ $n \lg n$ adds, $(n/2) \lg n$ mult to evaluate n-coeff f for all $\alpha \in \mathbf{C}$ with $\alpha^n = 1$ if n is a power of 2.

The fast Fourier transform

Use $(c_0, c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$
to represent $f = \sum_j c_j x^j \in \mathbf{C}[x]$.

Summary of representation size:
"f has n coeffs". Warning:
f does not determine n.

$f = f_0(x^2) + x f_1(x^2)$ where
$(c_0, c_2, \ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$,
$(c_1, c_3, \ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$
represent f_0, f_1 respectively.

$\mathbf{C}[x]$-morphism $y \mapsto x^2$
from $\mathbf{C}[x][y]$ to $\mathbf{C}[x]$
maps $f_0(y) + x f_1(y)$ to f.

Quickly evaluate $f(\alpha), f(-\alpha)$
by evaluating $f_0(\alpha^2)$; $f_1(\alpha^2)$;

$$f(\alpha) = f_0(\alpha^2) + \alpha f_1(\alpha^2);$$
$$f(-\alpha) = f_0(\alpha^2) - \alpha f_1(\alpha^2).$$

Evaluate $f(\alpha)$ for, e.g.,
all $\alpha \in \mathbf{C}$ with $\alpha^{1024} = 1$
by evaluating $f_0(\beta)$, $f_1(\beta)$
for all $\beta \in \mathbf{C}$ with $\beta^{512} = 1$;
plus 1024 adds, 512 mults.

Apply this recursively $\Rightarrow$
$n \lg n$ adds, $(n/2) \lg n$ mults
to evaluate n-coeff f
for all $\alpha \in \mathbf{C}$ with $\alpha^n = 1$
if n is a power of 2.

Fourier transform

$c_1, \ldots, c_{n-1}) \in \mathbf{C}^n$
sent $f = \sum_j c_j x^j \in \mathbf{C}[x]$.

y of representation size:
n coeffs". Warning:
not determine n.

$x^2) + x f_1(x^2)$ where
$\ldots) \in \mathbf{C}^{\lceil n/2 \rceil}$,
$\ldots) \in \mathbf{C}^{\lfloor n/2 \rfloor}$
t f_0, f_1 respectively.

rphism $y \mapsto x^2$
$x][y]$ to $\mathbf{C}[x]$
$(y) + x f_1(y)$ to f.

Quickly evaluate $f(\alpha), f(-\alpha)$
by evaluating $f_0(\alpha^2)$; $f_1(\alpha^2)$;

$$f(\alpha) = f_0(\alpha^2) + \alpha f_1(\alpha^2);$$
$$f(-\alpha) = f_0(\alpha^2) - \alpha f_1(\alpha^2).$$

Evaluate $f(\alpha)$ for, e.g.,
all $\alpha \in \mathbf{C}$ with $\alpha^{1024} = 1$
by evaluating $f_0(\beta)$, $f_1(\beta)$
for all $\beta \in \mathbf{C}$ with $\beta^{512} = 1$;
plus 1024 adds, 512 mults.

Apply this recursively $\Rightarrow$
$n \lg n$ adds, $(n/2) \lg n$ mults
to evaluate n-coeff f
for all $\alpha \in \mathbf{C}$ with $\alpha^n = 1$
if n is a power of 2.

Another

If $f \in \mathbf{C}$
f mod x
$c_0 + c_1 x$
f mod x
$(c_0 + c_2$
f mod x
$(c_0 - c_2$

$\mathbf{C}[x]$-mo
$\mathbf{C}[x]/(x$
maps c_0
$((c_0 + c$
$(c_0 - c$

ransform

$_{-1}) \in \mathbf{C}^n$

$\sum_j c_j x^j \in \mathbf{C}[x]$.

sentation size:

Warning:

nine n.

(x^2) where

$/2]$,

$/2]$

spectively.

$\to x^2$

$[x]$

(y) to f.

Quickly evaluate $f(\alpha), f(-\alpha)$
by evaluating $f_0(\alpha^2)$; $f_1(\alpha^2)$;

$$f(\alpha) = f_0(\alpha^2) + \alpha f_1(\alpha^2);$$
$$f(-\alpha) = f_0(\alpha^2) - \alpha f_1(\alpha^2).$$

Evaluate $f(\alpha)$ for, e.g.,
all $\alpha \in \mathbf{C}$ with $\alpha^{1024} = 1$
by evaluating $f_0(\beta)$, $f_1(\beta)$
for all $\beta \in \mathbf{C}$ with $\beta^{512} = 1$;
plus 1024 adds, 512 mults.

Apply this recursively $\Rightarrow$
$n \lg n$ adds, $(n/2) \lg n$ mults
to evaluate n-coeff f
for all $\alpha \in \mathbf{C}$ with $\alpha^n = 1$
if n is a power of 2.

Another view of th

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 +$
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 +$
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 -$

$\mathbf{C}[x]$-morphism $\mathbf{C}[$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}$
maps $c_0 + c_1 x + c$
$((c_0 + c_2) + (c_1 +$
$(c_0 - c_2) + (c_1 -$

$\mathbf{C}[x]$.

size:

e

Quickly evaluate $f(\alpha), f(-\alpha)$
by evaluating $f_0(\alpha^2)$; $f_1(\alpha^2)$;

$$f(\alpha) = f_0(\alpha^2) + \alpha f_1(\alpha^2);$$
$$f(-\alpha) = f_0(\alpha^2) - \alpha f_1(\alpha^2).$$

Evaluate $f(\alpha)$ for, e.g.,
all $\alpha \in \mathbf{C}$ with $\alpha^{1024} = 1$
by evaluating $f_0(\beta)$, $f_1(\beta)$
for all $\beta \in \mathbf{C}$ with $\beta^{512} = 1$;
plus 1024 adds, 512 mults.

Apply this recursively $\Rightarrow$
$n \lg n$ adds, $(n/2) \lg n$ mults
to evaluate n-coeff f
for all $\alpha \in \mathbf{C}$ with $\alpha^n = 1$
if n is a power of 2.

Another view of the FFT

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 + c_3 x^3$ ther
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 + c_3)x$,
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 - c_3)x$.

$\mathbf{C}[x]$-morphism $\mathbf{C}[x]/(x^4 -$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 +$
maps $c_0 + c_1 x + c_2 x^2 + c_3 x$
$((c_0 + c_2) + (c_1 + c_3)x,$
$(c_0 - c_2) + (c_1 - c_3)x)$.

Quickly evaluate $f(\alpha), f(-\alpha)$
by evaluating $f_0(\alpha^2)$; $f_1(\alpha^2)$;

$$f(\alpha) = f_0(\alpha^2) + \alpha f_1(\alpha^2);$$
$$f(-\alpha) = f_0(\alpha^2) - \alpha f_1(\alpha^2).$$

Evaluate $f(\alpha)$ for, e.g.,
all $\alpha \in \mathbf{C}$ with $\alpha^{1024} = 1$
by evaluating $f_0(\beta)$, $f_1(\beta)$
for all $\beta \in \mathbf{C}$ with $\beta^{512} = 1$;
plus 1024 adds, 512 mults.

Apply this recursively $\Rightarrow$
$n \lg n$ adds, $(n/2) \lg n$ mults
to evaluate n-coeff f
for all $\alpha \in \mathbf{C}$ with $\alpha^n = 1$
if n is a power of 2.

Another view of the FFT

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 + c_3 x^3$ then
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 + c_3)x$,
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 - c_3)x$.

$\mathbf{C}[x]$-morphism $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow\!\!\!\!\rightarrow$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$
maps $c_0 + c_1 x + c_2 x^2 + c_3 x^3$ to
$((c_0 + c_2) + (c_1 + c_3)x,$
$(c_0 - c_2) + (c_1 - c_3)x)$.

evaluate $f(\alpha), f(-\alpha)$
ating $f_0(\alpha^2)$; $f_1(\alpha^2)$;
$= f_0(\alpha^2) + \alpha f_1(\alpha^2)$;
$= f_0(\alpha^2) - \alpha f_1(\alpha^2)$.

e $f(\alpha)$ for, e.g.,
C with $\alpha^{1024} = 1$
ating $f_0(\beta)$, $f_1(\beta)$
$\in$ **C** with $\beta^{512} = 1$;
4 adds, 512 mults.

his recursively $\Rightarrow$
dds, $(n/2)\lg n$ mults
ate n-coeff f
$\in$ **C** with $\alpha^n = 1$
power of 2.

Another view of the FFT

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 + c_3 x^3$ then
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 + c_3)x$,
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 - c_3)x$.

$\mathbf{C}[x]$-morphism $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow\!\!\!\!\rightarrow$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$
maps $c_0 + c_1 x + c_2 x^2 + c_3 x^3$ to
$((c_0 + c_2) + (c_1 + c_3)x,$
$\ (c_0 - c_2) + (c_1 - c_3)x)$.

If $f \in$ **C**
$f \bmod x$
$c_0 + c_1 x$
$f \bmod x$
$(c_0 + \alpha c$
$+ (c_2$
$f \bmod x$
$(c_0 - \alpha c$
$+ (c_2$

Given c_0
use n m
$c_0 + \alpha c_n$
$c_0 - \alpha c_n$

$f(\alpha), f(-\alpha)$

$\alpha^2); f_1(\alpha^2);$

$+ \alpha f_1(\alpha^2);$

$- \alpha f_1(\alpha^2).$

, e.g.,

$024 = 1$

$\beta), f_1(\beta)$

$\beta^{512} = 1;$

12 mults.

vely $\Rightarrow$

$\lg n$ mults

ff f

$\alpha^n = 1$

2.

Another view of the FFT

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 + c_3 x^3$ then
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 + c_3)x,$
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 - c_3)x.$

$\mathbf{C}[x]$-morphism $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow\!\!\!\!\rightarrow$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$
maps $c_0 + c_1 x + c_2 x^2 + c_3 x^3$ to
$((c_0 + c_2) + (c_1 + c_3)x,$
$(c_0 - c_2) + (c_1 - c_3)x).$

If $f \in \mathbf{C}[x]$ and
$f \bmod x^{2n} - \alpha^2 =$
$c_0 + c_1 x + \cdots + c$
$f \bmod x^n - \alpha =$
$(c_0 + \alpha c_n) + (c_1$
$+ (c_2 + \alpha c_{n+2}$
$f \bmod x^n + \alpha =$
$(c_0 - \alpha c_n) + (c_1$
$+ (c_2 - \alpha c_{n+2}$

Given $c_0, c_1, \ldots, c$
use n mults, $2n$ a
$c_0 + \alpha c_n, c_1 + \alpha c$
$c_0 - \alpha c_n, c_1 - \alpha c$

$\alpha)$

$);$

;

.

s

Another view of the FFT

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 + c_3 x^3$ then
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 + c_3)x$,
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 - c_3)x$.

$\mathbf{C}[x]$-morphism $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow\!\!\!\!\rightarrow$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$
maps $c_0 + c_1 x + c_2 x^2 + c_3 x^3$ to
$((c_0 + c_2) + (c_1 + c_3)x,$
$(c_0 - c_2) + (c_1 - c_3)x)$.

If $f \in \mathbf{C}[x]$ and
$f \bmod x^{2n} - \alpha^2 =$
$c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-}$
$f \bmod x^n - \alpha =$
$(c_0 + \alpha c_n) + (c_1 + \alpha c_{n+1})x$
$+ (c_2 + \alpha c_{n+2})x^2 + \cdots,$
$f \bmod x^n + \alpha =$
$(c_0 - \alpha c_n) + (c_1 - \alpha c_{n+1})x$
$+ (c_2 - \alpha c_{n+2})x^2 + \cdots.$

Given $c_0, c_1, \ldots, c_{2n-1} \in \mathbf{C}$,
use n mults, $2n$ adds to cor
$c_0 + \alpha c_n, c_1 + \alpha c_{n+1}, \ldots,$
$c_0 - \alpha c_n, c_1 - \alpha c_{n+1}, \ldots.$

Another view of the FFT

If $f \in \mathbf{C}[x]$ and
$f \bmod x^4 - 1 =$
$c_0 + c_1 x + c_2 x^2 + c_3 x^3$ then
$f \bmod x^2 - 1 =$
$(c_0 + c_2) + (c_1 + c_3)x$,
$f \bmod x^2 + 1 =$
$(c_0 - c_2) + (c_1 - c_3)x$.

$\mathbf{C}[x]$-morphism $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow\!\!\!\!\rightarrow$
$\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$
maps $c_0 + c_1 x + c_2 x^2 + c_3 x^3$ to
$((c_0 + c_2) + (c_1 + c_3)x,$
$(c_0 - c_2) + (c_1 - c_3)x)$.

If $f \in \mathbf{C}[x]$ and
$f \bmod x^{2n} - \alpha^2 =$
$c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1}$ then
$f \bmod x^n - \alpha =$
$(c_0 + \alpha c_n) + (c_1 + \alpha c_{n+1})x$
$\quad + (c_2 + \alpha c_{n+2})x^2 + \cdots$,
$f \bmod x^n + \alpha =$
$(c_0 - \alpha c_n) + (c_1 - \alpha c_{n+1})x$
$\quad + (c_2 - \alpha c_{n+2})x^2 + \cdots$.

Given $c_0, c_1, \ldots, c_{2n-1} \in \mathbf{C}$,
use n mults, $2n$ adds to compute
$c_0 + \alpha c_n, c_1 + \alpha c_{n+1}, \ldots,$
$c_0 - \alpha c_n, c_1 - \alpha c_{n+1}, \ldots.$

view of the FFT

$[x]$ and

$^4 - 1 =$

$+ c_2x^2 + c_3x^3$ then

$^2 - 1 =$

$) + (c_1 + c_3)x,$

$^2 + 1 =$

$) + (c_1 - c_3)x.$

rphism $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow\!\!\!\!\rightarrow$

$^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$

$+ c_1x + c_2x^2 + c_3x^3$ to

$_2) + (c_1 + c_3)x,$

$_2) + (c_1 - c_3)x).$

If $f \in \mathbf{C}[x]$ and

$f \bmod x^{2n} - \alpha^2 =$

$c_0 + c_1x + \cdots + c_{2n-1}x^{2n-1}$ then

$f \bmod x^n - \alpha =$

$(c_0 + \alpha c_n) + (c_1 + \alpha c_{n+1})x$

$\quad + (c_2 + \alpha c_{n+2})x^2 + \cdots,$

$f \bmod x^n + \alpha =$

$(c_0 - \alpha c_n) + (c_1 - \alpha c_{n+1})x$

$\quad + (c_2 - \alpha c_{n+2})x^2 + \cdots.$

Given $c_0, c_1, \ldots, c_{2n-1} \in \mathbf{C}$,

use n mults, $2n$ adds to compute

$c_0 + \alpha c_n, c_1 + \alpha c_{n+1}, \ldots,$

$c_0 - \alpha c_n, c_1 - \alpha c_{n+1}, \ldots.$

Apply th

f mod

f mod

$x - 1$

$=$

$f(1)$

(basic F

this view

he FFT

c_3x^3 then

$c_3)x$,

$c_3)x$.

$x]/(x^4-1) \hookrightarrow\!\!\!\!\twoheadrightarrow$

$\mathbf{C}[x]/(x^2+1)$

$c_2x^2 + c_3x^3$ to

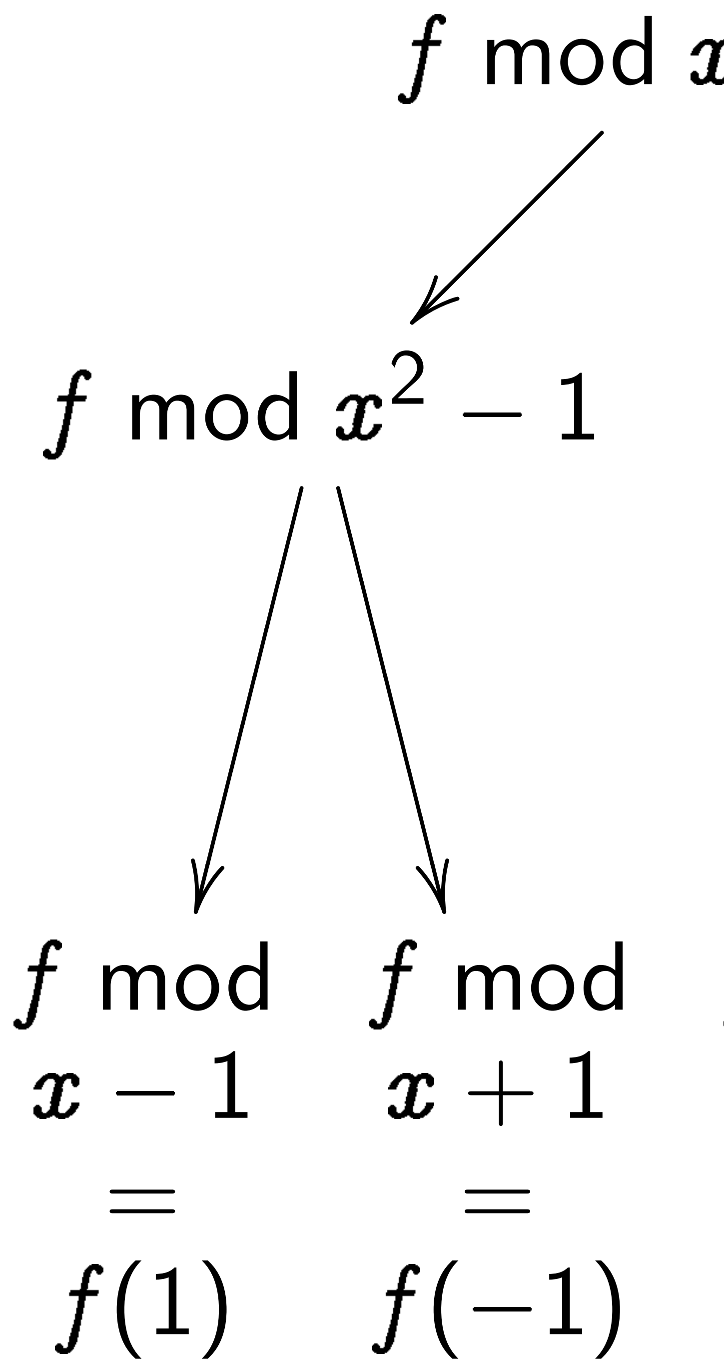
$c_3)x$,

$c_3)x)$.

If $f \in \mathbf{C}[x]$ and
$f \bmod x^{2n} - \alpha^2 =$
$c_0 + c_1x + \cdots + c_{2n-1}x^{2n-1}$ then
$f \bmod x^n - \alpha =$
$(c_0 + \alpha c_n) + (c_1 + \alpha c_{n+1})x$
$\quad + (c_2 + \alpha c_{n+2})x^2 + \cdots,$
$f \bmod x^n + \alpha =$
$(c_0 - \alpha c_n) + (c_1 - \alpha c_{n+1})x$
$\quad + (c_2 - \alpha c_{n+2})x^2 + \cdots.$

Given $c_0, c_1, \ldots, c_{2n-1} \in \mathbf{C}$,
use n mults, $2n$ adds to compute
$c_0 + \alpha c_n, c_1 + \alpha c_{n+1}, \ldots,$
$c_0 - \alpha c_n, c_1 - \alpha c_{n+1}, \ldots.$

Apply this recursiv



(basic FFT idea: 1
this view: 1972 Fi

n

1) $\hookrightarrow\!\!\twoheadrightarrow$

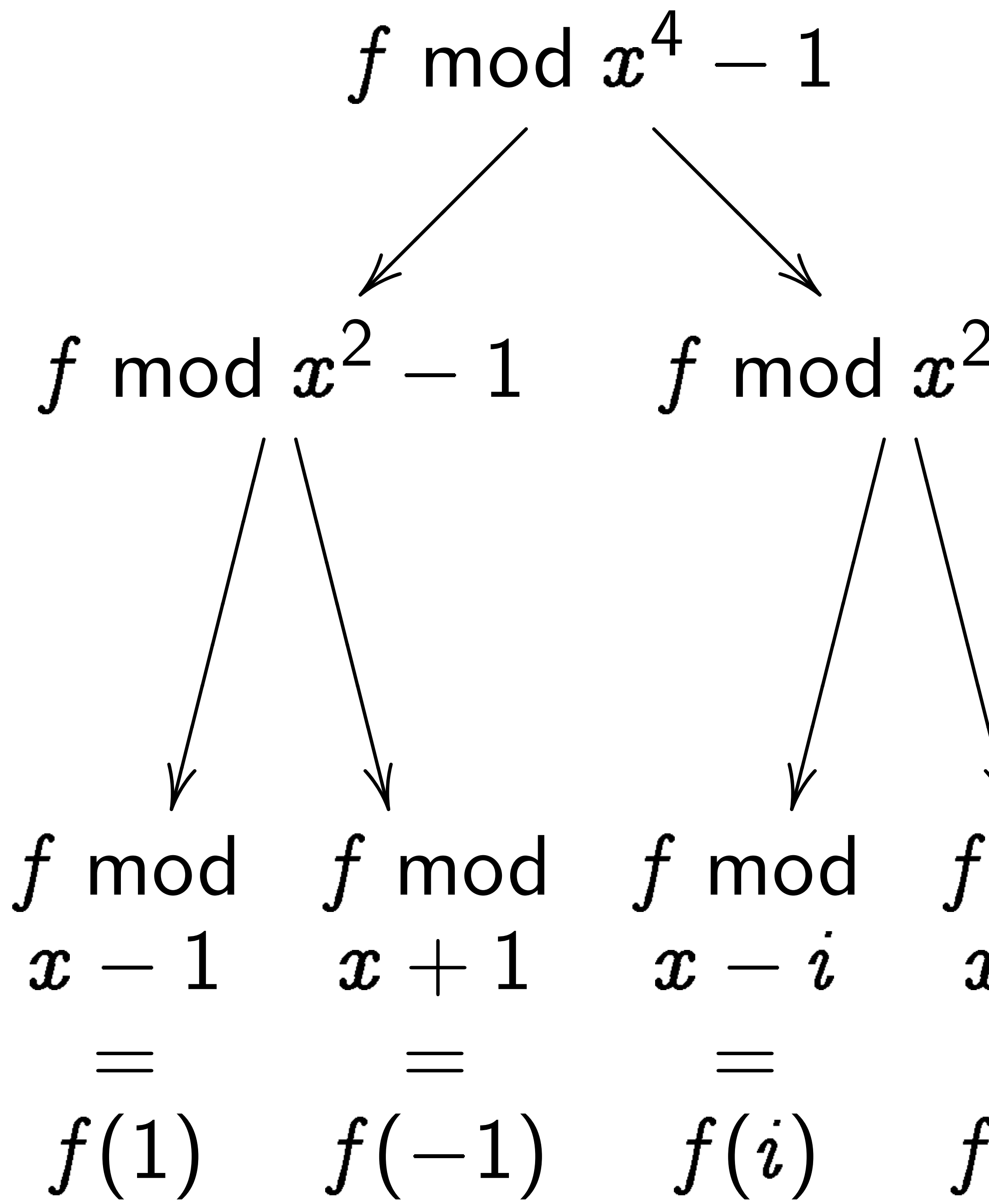
- 1)

^{3} to

If $f \in \mathbf{C}[x]$ and
$f \bmod x^{2n} - \alpha^2 =$
$c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1}$ then
$f \bmod x^n - \alpha =$
$(c_0 + \alpha c_n) + (c_1 + \alpha c_{n+1})x$
$\quad + (c_2 + \alpha c_{n+2})x^2 + \cdots,$
$f \bmod x^n + \alpha =$
$(c_0 - \alpha c_n) + (c_1 - \alpha c_{n+1})x$
$\quad + (c_2 - \alpha c_{n+2})x^2 + \cdots.$

Given $c_0, c_1, \ldots, c_{2n-1} \in \mathbf{C}$,
use n mults, $2n$ adds to compute
$c_0 + \alpha c_n, c_1 + \alpha c_{n+1}, \ldots,$
$c_0 - \alpha c_n, c_1 - \alpha c_{n+1}, \ldots.$

Apply this recursively:



(basic FFT idea: 1866 Gaus
this view: 1972 Fiduccia)

If $f \in \mathbf{C}[x]$ and

$f \bmod x^{2n} - \alpha^2 =$

$c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1}$ then

$f \bmod x^n - \alpha =$

$$(c_0 + \alpha c_n) + (c_1 + \alpha c_{n+1})x + (c_2 + \alpha c_{n+2})x^2 + \cdots,$$

$f \bmod x^n + \alpha =$

$$(c_0 - \alpha c_n) + (c_1 - \alpha c_{n+1})x + (c_2 - \alpha c_{n+2})x^2 + \cdots.$$

Given $c_0, c_1, \ldots, c_{2n-1} \in \mathbf{C}$,

use n mults, $2n$ adds to compute

$c_0 + \alpha c_n, c_1 + \alpha c_{n+1}, \ldots,$

$c_0 - \alpha c_n, c_1 - \alpha c_{n+1}, \ldots.$

Apply this recursively:

$f \bmod x^4 - 1$

$f \bmod x^2 - 1$ $\qquad$ $f \bmod x^2 + 1$

$f \bmod x - 1 = f(1)$	$f \bmod x + 1 = f(-1)$	$f \bmod x - i = f(i)$	$f \bmod x + i = f(-i)$

(basic FFT idea: 1866 Gauss;

this view: 1972 Fiduccia)

$[x]$ and

$^{2n} - \alpha^2 =$

$+ \cdots + c_{2n-1}x^{2n-1}$ then

$^{n} - \alpha =$

$_n) + (c_1 + \alpha c_{n+1})x$

$+ \alpha c_{n+2})x^2 + \cdots,$

$^{n} + \alpha =$

$_n) + (c_1 - \alpha c_{n+1})x$

$- \alpha c_{n+2})x^2 + \cdots.$

$, c_1, \ldots, c_{2n-1} \in \mathbf{C},$

ults, $2n$ adds to compute

$_n, c_1 + \alpha c_{n+1}, \ldots,$

$_n, c_1 - \alpha c_{n+1}, \ldots.$

Apply this recursively:

$f \bmod x^4 - 1$

$f \bmod x^2 - 1$ $\quad$ $f \bmod x^2 + 1$

f mod $x - 1$ $=$ $f(1)$	f mod $x + 1$ $=$ $f(-1)$	f mod $x - i$ $=$ $f(i)$	f mod $x + i$ $=$ $f(-i)$

(basic FFT idea: 1866 Gauss;
this view: 1972 Fiducciа)

1966 Sa

Can very

in $\mathbf{C}[x]/$

by mapp

Given f,

compute

using T

Compute

Given f,

compute

its image

=

$_{2n-1}x^{2n-1}$ then

$+ \alpha c_{n+1})x$

$)x^2 + \cdots,$

$- \alpha c_{n+1})x$

$)x^2 + \cdots.$

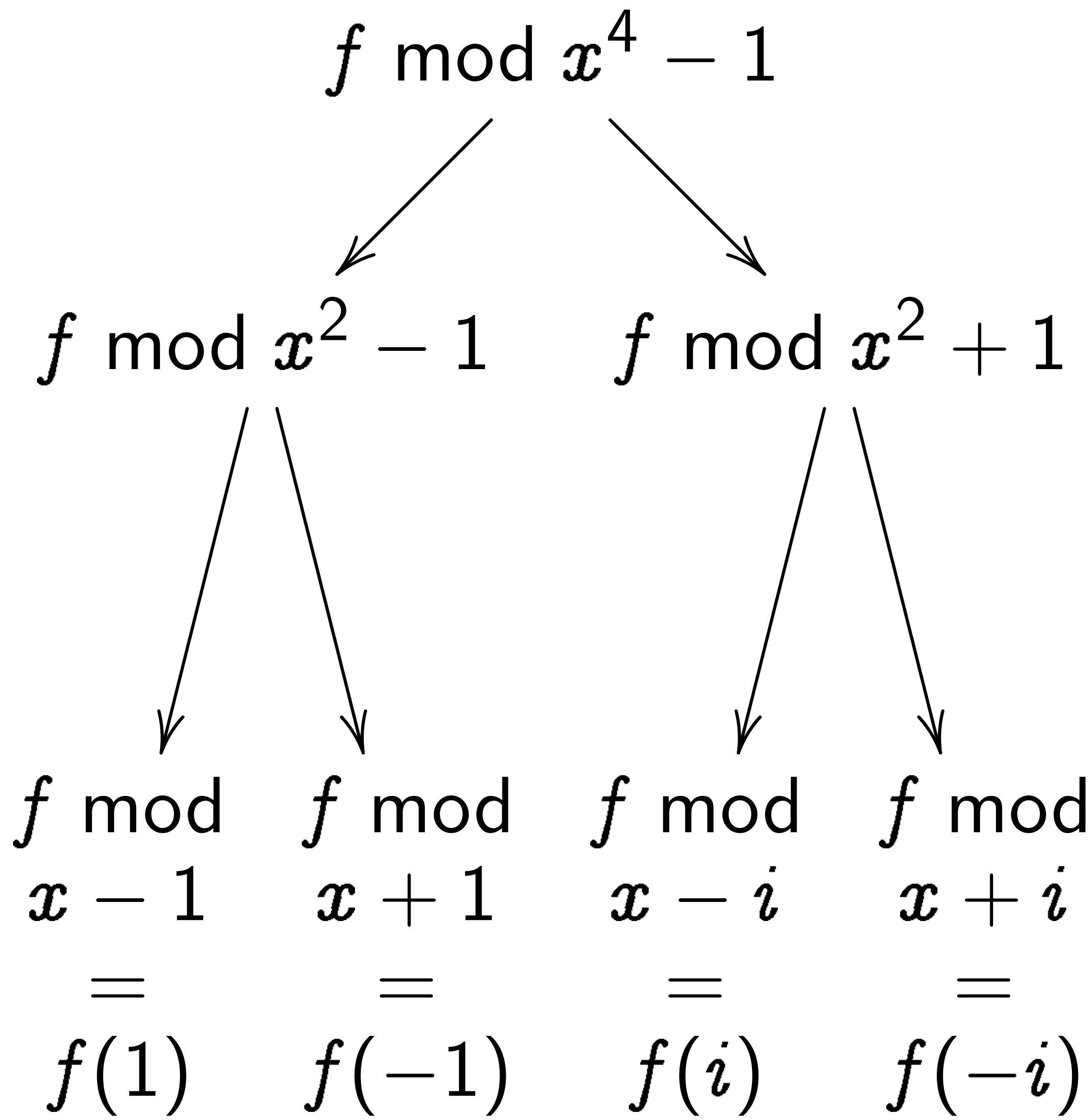
$_{2n-1} \in \mathbf{C}$,

dds to compute

$_{n+1}, \ldots,$

$_{n+1}, \ldots.$

Apply this recursively:



(basic FFT idea: 1866 Gauss; this view: 1972 Fiduccia)

1966 Sande, 1966

Can very quickly m

in $\mathbf{C}[x]/(x^n - 1)$ c

by mapping $\mathbf{C}[x]/$

Given $f, g \in \mathbf{C}[x]/$

compute fg as T^{-}

using $T : \mathbf{C}[x]/(x^n$

Compute T quickl

Given $f, g \in \mathbf{C}[x]$,

compute fg from

its image in $\mathbf{C}[x]/$

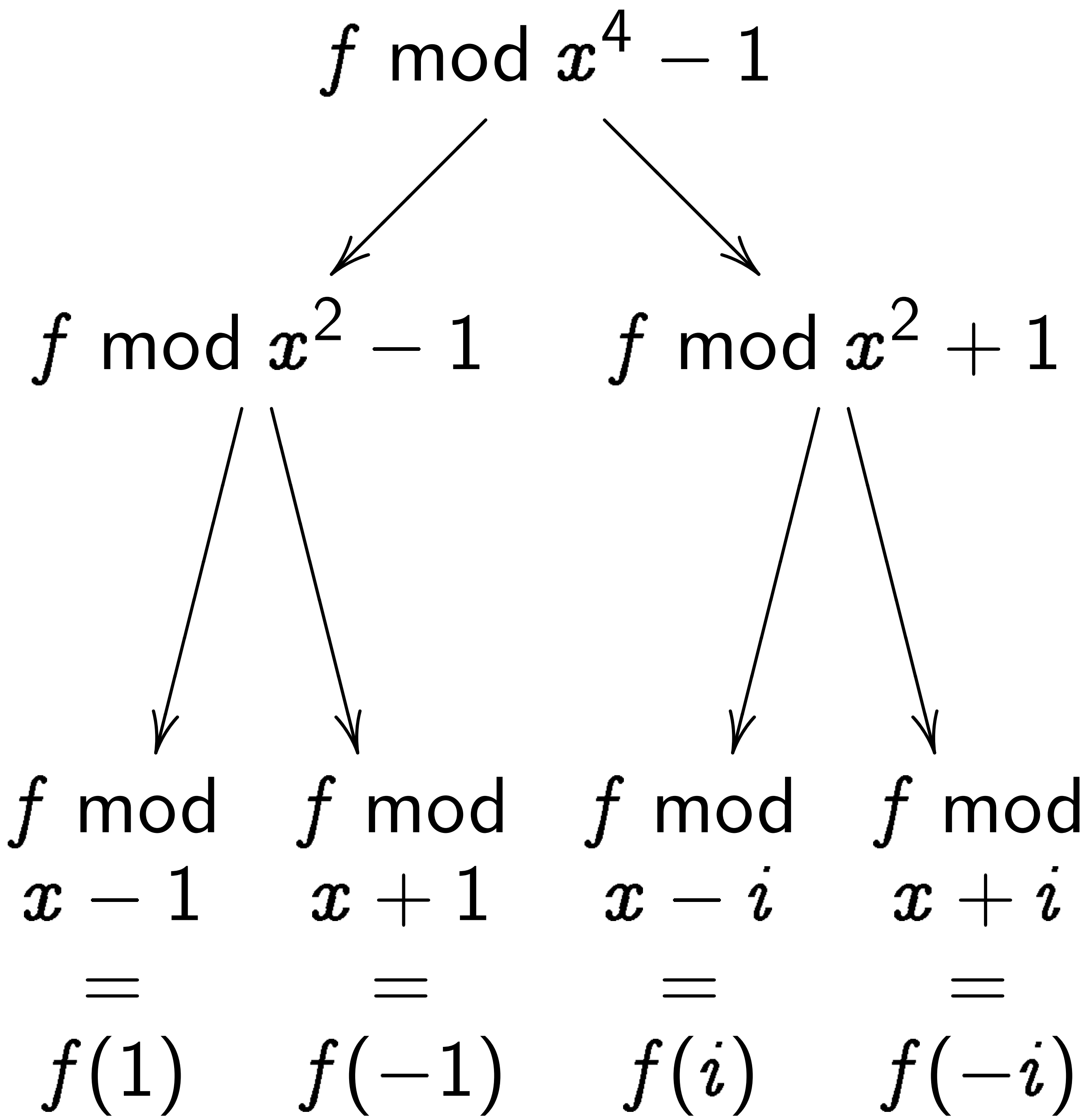
^{1} then

c

c

mpute

Apply this recursively:



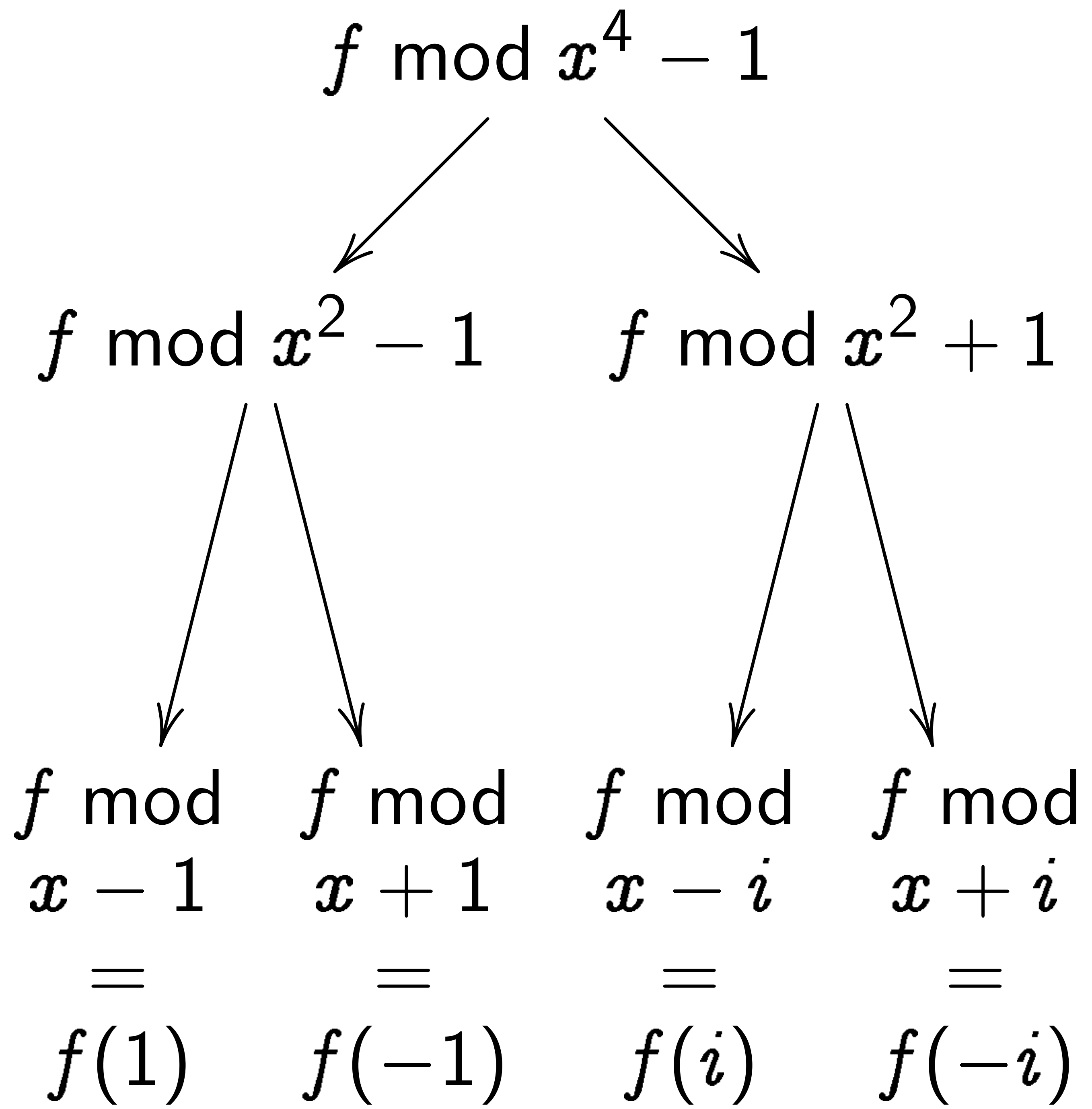
(basic FFT idea: 1866 Gauss;
this view: 1972 Fiduccia)

1966 Sande, 1966 Stockham
Can very quickly multiply
in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or
by mapping $\mathbf{C}[x]/(x^n - 1)$ t

Given $f, g \in \mathbf{C}[x]/(x^n - 1)$:
compute fg as $T^{-1}(T(f)T($
using $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\!\rightarrow$
Compute T quickly by the F

Given $f, g \in \mathbf{C}[x]$, $\deg fg <$
compute fg from
its image in $\mathbf{C}[x]/(x^n - 1)$.

Apply this recursively:



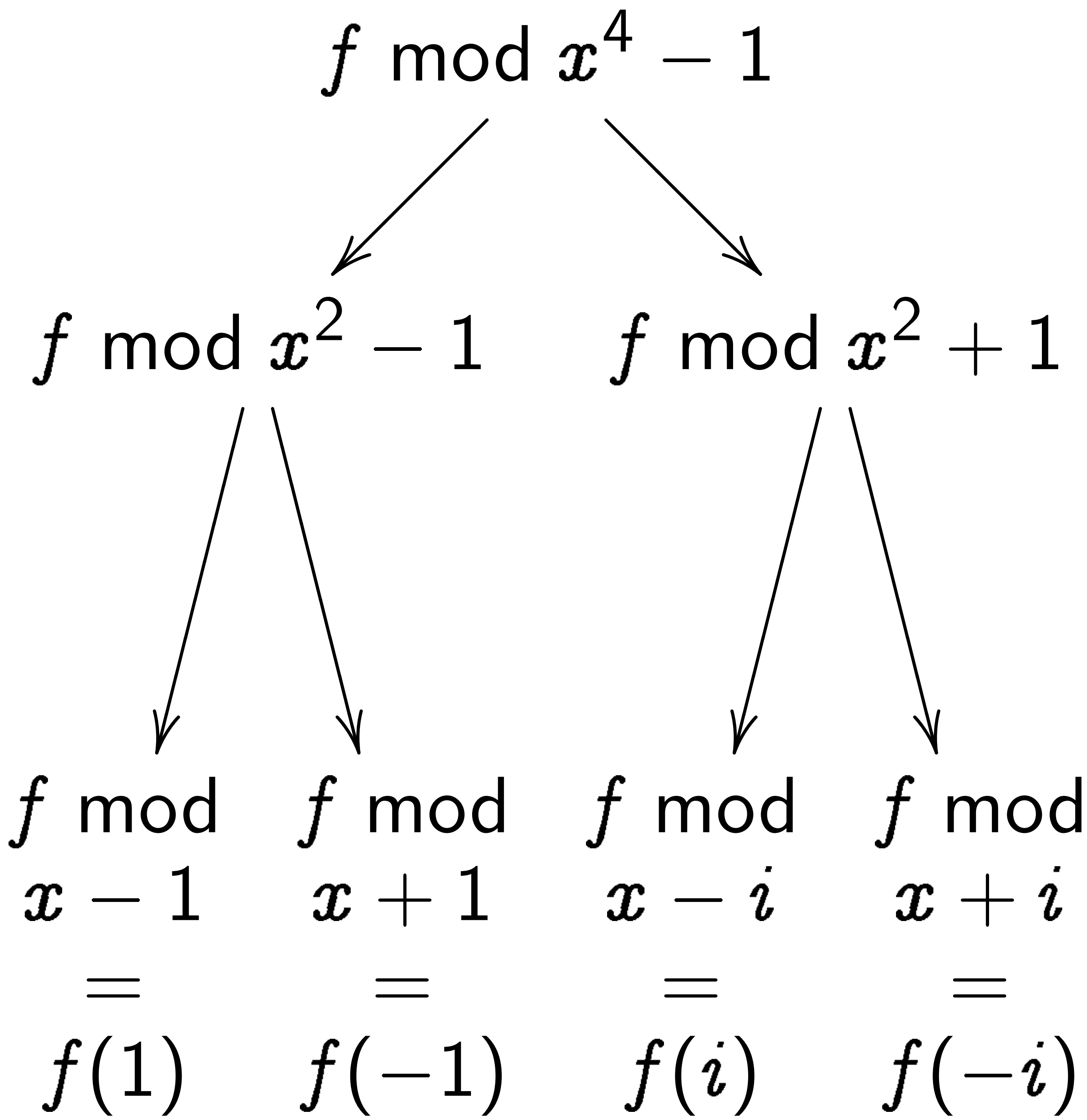
(basic FFT idea: 1866 Gauss; this view: 1972 Fiduccia)

1966 Sande, 1966 Stockham: Can very quickly multiply in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$ by mapping $\mathbf{C}[x]/(x^n - 1)$ to $\mathbf{C}^n$.

Given $f, g \in \mathbf{C}[x]/(x^n - 1)$: compute fg as $T^{-1}(T(f)T(g))$ using $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\rightarrow \mathbf{C}^n$. Compute T quickly by the FFT.

Given $f, g \in \mathbf{C}[x]$, $\deg fg < n$: compute fg from its image in $\mathbf{C}[x]/(x^n - 1)$.

Apply this recursively:



(basic FFT idea: 1866 Gauss;
this view: 1972 Fiduccia)

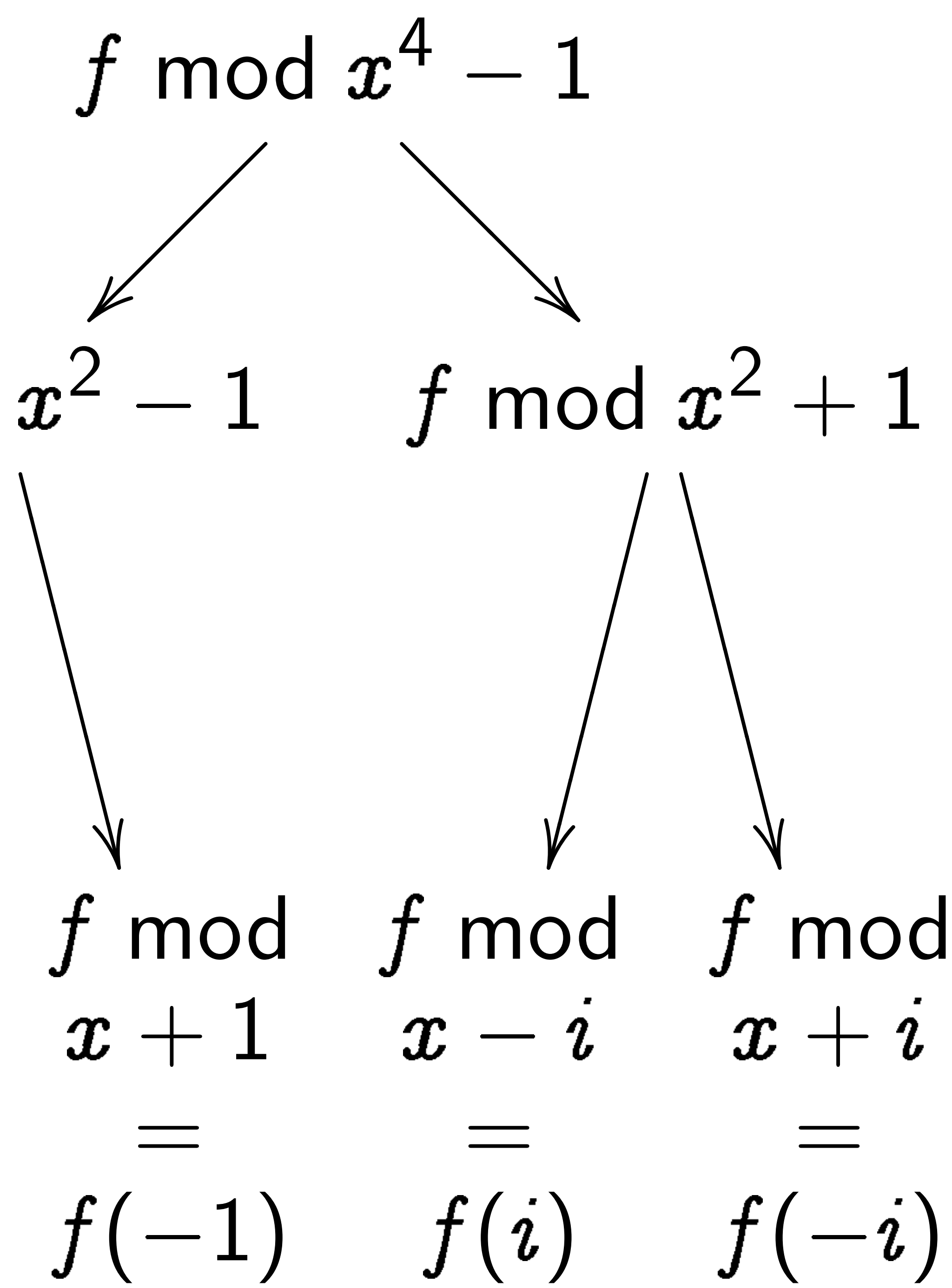
1966 Sande, 1966 Stockham:
Can very quickly multiply
in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$
by mapping $\mathbf{C}[x]/(x^n - 1)$ to $\mathbf{C}^n$.

Given $f, g \in \mathbf{C}[x]/(x^n - 1)$:
compute fg as $T^{-1}(T(f)T(g))$
using $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\!\rightarrow \mathbf{C}^n$.
Compute T quickly by the FFT.

Given $f, g \in \mathbf{C}[x]$, $\deg fg < n$:
compute fg from
its image in $\mathbf{C}[x]/(x^n - 1)$.

Later authors: Replace $\mathbf{C}$ with,
e.g., $R = \mathbf{Z}/(3 \cdot 2^{41} + 1)$;
23 has order 2^{41} in R^*.

his recursively:



FT idea: 1866 Gauss;
v: 1972 Fiduccia)

1966 Sande, 1966 Stockham:
Can very quickly multiply
in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$
by mapping $\mathbf{C}[x]/(x^n - 1)$ to $\mathbf{C}^n$.

Given $f, g \in \mathbf{C}[x]/(x^n - 1)$:
compute fg as $T^{-1}(T(f)T(g))$
using $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\!\rightarrow \mathbf{C}^n$.
Compute T quickly by the FFT.

Given $f, g \in \mathbf{C}[x]$, $\deg fg < n$:
compute fg from
its image in $\mathbf{C}[x]/(x^n - 1)$.

Later authors: Replace $\mathbf{C}$ with,
e.g., $R = \mathbf{Z}/(3 \cdot 2^{41} + 1)$;
23 has order 2^{41} in R^*.

<u>Multiplic</u>

Given r,
in time
where b

(1971 Po
1971 Nic
1971 Sch

Also tim
where b
Given r,
compute

(reductio
1966 Co

vely:

$^4 - 1$

$f \bmod x^2 + 1$

$f \bmod x - i = f(i)$ $\quad$ $f \bmod x + i = f(-i)$

1866 Gauss;
duccia)

1966 Sande, 1966 Stockham:
Can very quickly multiply
in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$
by mapping $\mathbf{C}[x]/(x^n - 1)$ to $\mathbf{C}^n$.

Given $f, g \in \mathbf{C}[x]/(x^n - 1)$:
compute fg as $T^{-1}(T(f)T(g))$
using $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\!\rightarrow \mathbf{C}^n$.
Compute T quickly by the FFT.

Given $f, g \in \mathbf{C}[x]$, $\deg fg < n$:
compute fg from
its image in $\mathbf{C}[x]/(x^n - 1)$.

Later authors: Replace $\mathbf{C}$ with,
e.g., $R = \mathbf{Z}/(3 \cdot 2^{41} + 1)$;
23 has order 2^{41} in R^*.

Multiplication and

Given $r, s \in \mathbf{Z}$, ca
in time $\leq b(\lg b)^{1+}$
where b is number

(1971 Pollard; ind
1971 Nicholson; in
1971 Schönhage S

Also time $\leq b(\lg b$
where b is number
Given $r, s \in \mathbf{Z}$ wit
compute $\lfloor r/s \rfloor$ an

(reduction to prod
1966 Cook)

$+ 1$

mod

$c + i$

$=$

$(-i)$

s;

1966 Sande, 1966 Stockham:
Can very quickly multiply
in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$
by mapping $\mathbf{C}[x]/(x^n - 1)$ to $\mathbf{C}^n$.

Given $f, g \in \mathbf{C}[x]/(x^n - 1)$:
compute fg as $T^{-1}(T(f)T(g))$
using $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\!\!\twoheadrightarrow \mathbf{C}^n$.
Compute T quickly by the FFT.

Given $f, g \in \mathbf{C}[x]$, $\deg fg < n$:
compute fg from
its image in $\mathbf{C}[x]/(x^n - 1)$.

Later authors: Replace $\mathbf{C}$ with,
e.g., $R = \mathbf{Z}/(3 \cdot 2^{41} + 1)$;
23 has order 2^{41} in R^*.

Multiplication and division

Given $r, s \in \mathbf{Z}$, can compute
in time $\le b(\lg b)^{1+o(1)}$
where b is number of input b

(1971 Pollard; independently
1971 Nicholson; independent
1971 Schönhage Strassen)

Also time $\le b(\lg b)^{1+o(1)}$
where b is number of input b
Given $r, s \in \mathbf{Z}$ with $s \ne 0$,
compute $\lfloor r/s \rfloor$ and $r \bmod s$

(reduction to product:
1966 Cook)

1966 Sande, 1966 Stockham:
Can very quickly multiply
in $\mathbf{C}[x]/(x^n-1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$
by mapping $\mathbf{C}[x]/(x^n-1)$ to $\mathbf{C}^n$.

Given $f, g \in \mathbf{C}[x]/(x^n-1)$:
compute fg as $T^{-1}(T(f)T(g))$
using $T : \mathbf{C}[x]/(x^n-1) \hookrightarrow\!\!\!\!\to \mathbf{C}^n$.
Compute T quickly by the FFT.

Given $f, g \in \mathbf{C}[x]$, $\deg fg < n$:
compute fg from
its image in $\mathbf{C}[x]/(x^n-1)$.

Later authors: Replace $\mathbf{C}$ with,
e.g., $R = \mathbf{Z}/(3 \cdot 2^{41}+1)$;
23 has order 2^{41} in R^*.

<u>Multiplication and division</u>

Given $r, s \in \mathbf{Z}$, can compute rs
in time $\le b(\lg b)^{1+o(1)}$
where b is number of input bits.

(1971 Pollard; independently
1971 Nicholson; independently
1971 Schönhage Strassen)

Also time $\le b(\lg b)^{1+o(1)}$
where b is number of input bits:
Given $r, s \in \mathbf{Z}$ with $s \ne 0$,
compute $\lfloor r/s \rfloor$ and $r \bmod s$.

(reduction to product:
1966 Cook)

nde, 1966 Stockham:

y quickly multiply

$(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$

ping $\mathbf{C}[x]/(x^n - 1)$ to $\mathbf{C}^n$.

$g \in \mathbf{C}[x]/(x^n - 1)$:

e fg as $T^{-1}(T(f)T(g))$

: $\mathbf{C}[x]/(x^n - 1) \hookrightarrow\!\!\!\!\rightarrow \mathbf{C}^n$.

e T quickly by the FFT.

$g \in \mathbf{C}[x]$, deg $fg < n$:

e fg from

e in $\mathbf{C}[x]/(x^n - 1)$.

thors: Replace **C** with,

$= \mathbf{Z}/(3 \cdot 2^{41} + 1)$;

rder 2^{41} in R^*.

Multiplication and division

Given $r, s \in \mathbf{Z}$, can compute rs in time $\le b(\lg b)^{1+o(1)}$ where b is number of input bits.

(1971 Pollard; independently 1971 Nicholson; independently 1971 Schönhage Strassen)

Also time $\le b(\lg b)^{1+o(1)}$ where b is number of input bits: Given $r, s \in \mathbf{Z}$ with $s \ne 0$, compute $\lfloor r/s \rfloor$ and $r \bmod s$.

(reduction to product: 1966 Cook)

Product

Time $\le$

where b

Given x_1

compute

Actually

product

Root is

Has left

product

Also righ

product

Stockham:

multiply

or $\mathbf{C}[x]$ or $\mathbf{R}[x]$

$(x^n - 1)$ to $\mathbf{C}^n$.

$/(x^n - 1)$:

$^{-1}(T(f)T(g))$

$^n - 1) \hookrightarrow \mathbf{C}^n$.

y by the FFT.

$\deg fg < n$:

$(x^n - 1)$.

place $\mathbf{C}$ with,

$^{41} + 1)$;

n R^*.

Multiplication and division

Given $r, s \in \mathbf{Z}$, can compute rs in time $\le b(\lg b)^{1+o(1)}$ where b is number of input bits.

(1971 Pollard; independently 1971 Nicholson; independently 1971 Schönhage Strassen)

Also time $\le b(\lg b)^{1+o(1)}$ where b is number of input bits: Given $r, s \in \mathbf{Z}$ with $s \neq 0$, compute $\lfloor r/s \rfloor$ and $r \bmod s$.

(reduction to product: 1966 Cook)

Product trees

Time $\le b(\lg b)^{2+o}$

where b is number

Given $x_1, x_2, \ldots,$

compute $x_1 x_2 \cdots$

Actually compute

product tree of x

Root is $x_1 x_2 \cdots x$

Has left subtree if

product tree of x_1

Also right subtree

product tree of $x_{\lceil}$

n:

$\mathbf{R}[x]$
to $\mathbf{C}^n$.

$(g))$
$\mathbf{C}^n$.
FFT.

n:

ith,

Multiplication and division

Given $r, s \in \mathbf{Z}$, can compute rs
in time $\le b(\lg b)^{1+o(1)}$
where b is number of input bits.

(1971 Pollard; independently
1971 Nicholson; independently
1971 Schönhage Strassen)

Also time $\le b(\lg b)^{1+o(1)}$
where b is number of input bits:
Given $r, s \in \mathbf{Z}$ with $s \ne 0$,
compute $\lfloor r/s \rfloor$ and $r \bmod s$.

(reduction to product:
1966 Cook)

Product trees

Time $\le b(\lg b)^{2+o(1)}$
where b is number of input
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$,
compute $x_1 x_2 \cdots x_n$.

Actually compute
product tree of $x_1, x_2, \ldots,$
Root is $x_1 x_2 \cdots x_n$.
Has left subtree if $n \ge 2$:
product tree of $x_1, \ldots, x_{\lceil n/}$
Also right subtree if $n \ge 2$:
product tree of $x_{\lceil n/2 \rceil + 1}, \ldots$

Multiplication and division

Given $r, s \in \mathbf{Z}$, can compute rs in time $\leq b(\lg b)^{1+o(1)}$ where b is number of input bits.

(1971 Pollard; independently 1971 Nicholson; independently 1971 Schönhage Strassen)

Also time $\leq b(\lg b)^{1+o(1)}$ where b is number of input bits: Given $r, s \in \mathbf{Z}$ with $s \neq 0$, compute $\lfloor r/s \rfloor$ and $r \bmod s$.

(reduction to product: 1966 Cook)

Product trees

Time $\leq b(\lg b)^{2+o(1)}$ where b is number of input bits: Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$, compute $x_1 x_2 \cdots x_n$.

Actually compute **product tree** of $x_1, x_2, \ldots, x_n$. Root is $x_1 x_2 \cdots x_n$. Has left subtree if $n \geq 2$: product tree of $x_1, \ldots, x_{\lceil n/2 \rceil}$. Also right subtree if $n \geq 2$: product tree of $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$.

cation and division

$s \in \mathbf{Z}$, can compute rs
$\leq b(\lg b)^{1+o(1)}$
is number of input bits.

ollard; independently
cholson; independently
hönhage Strassen)

ne $\leq b(\lg b)^{1+o(1)}$
is number of input bits:
$s \in \mathbf{Z}$ with $s \neq 0$,
e $\lfloor r/s \rfloor$ and $r \bmod s$.

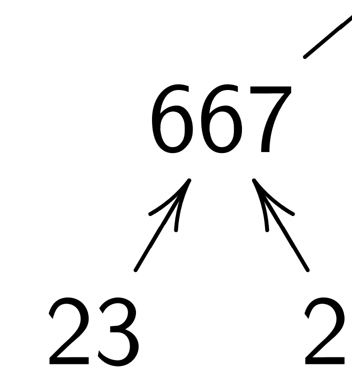
on to product:
ok)

Product trees

Time $\leq b(\lg b)^{2+o(1)}$
where b is number of input bits:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$,
compute $x_1 x_2 \cdots x_n$.

Actually compute
product tree of $x_1, x_2, \ldots, x_n$.
Root is $x_1 x_2 \cdots x_n$.
Has left subtree if $n \geq 2$:
product tree of $x_1, \ldots, x_{\lceil n/2 \rceil}$.
Also right subtree if $n \geq 2$:
product tree of $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$.

e.g. tree



Tree has
Each lev

Obtain e
in time
by multi

division

n compute rs
$+o(1)$
r of input bits.

ependently
ndependently
Strassen)

$)^{1+o(1)}$
r of input bits:
h $s \neq 0$,
d $r \bmod s$.

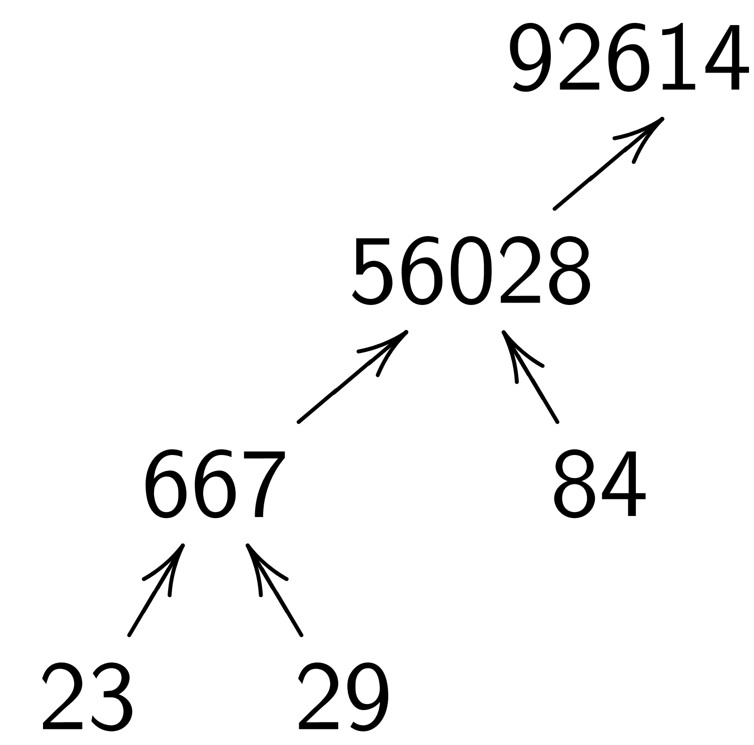
luct:

Product trees

Time $\le b(\lg b)^{2+o(1)}$
where b is number of input bits:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$,
compute $x_1 x_2 \cdots x_n$.

Actually compute
product tree of $x_1, x_2, \ldots, x_n$.
Root is $x_1 x_2 \cdots x_n$.
Has left subtree if $n \ge 2$:
product tree of $x_1, \ldots, x_{\lceil n/2 \rceil}$.
Also right subtree if $n \ge 2$:
product tree of $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$.

e.g. tree for 23, 29



Tree has $\le (\lg b)^{1+}$
Each level has $\le b$

Obtain each level
in time $\le b(\lg b)^{1+}$
by multiplying low

e rs

bits.

y

tly

bits:

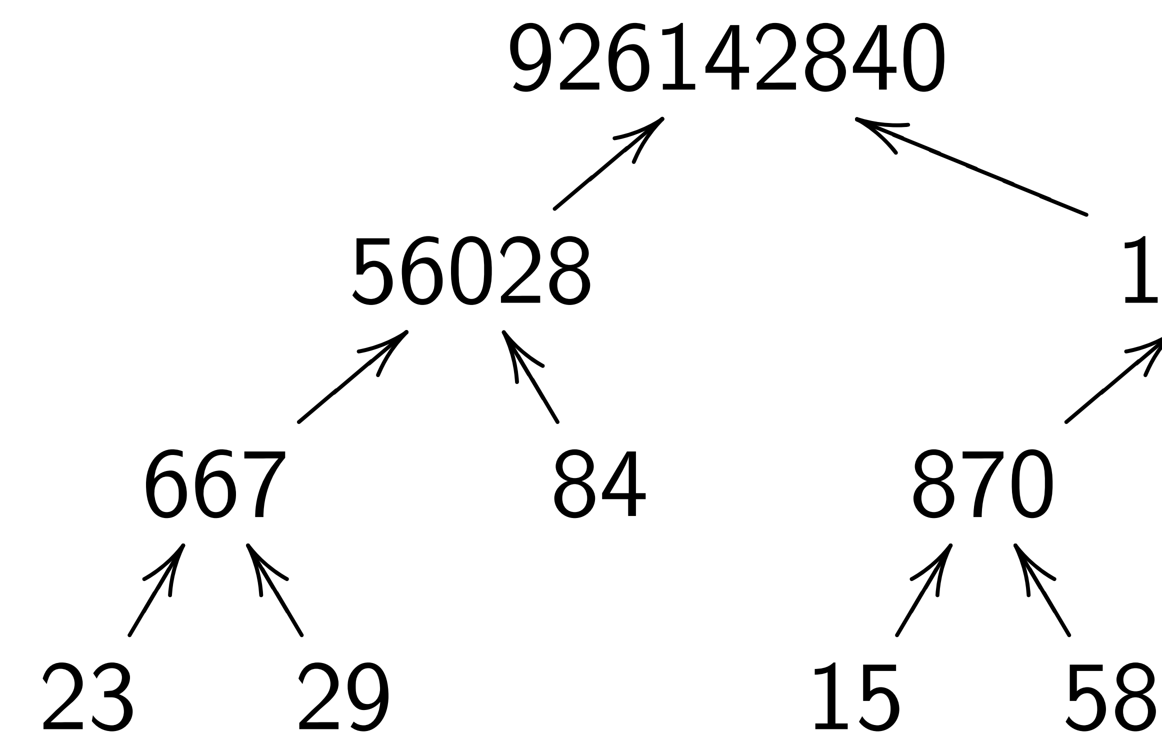
3.

Product trees

Time $\leq b(\lg b)^{2+o(1)}$
where b is number of input bits:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$,
compute $x_1 x_2 \cdots x_n$.

Actually compute
product tree of $x_1, x_2, \ldots, x_n$.
Root is $x_1 x_2 \cdots x_n$.
Has left subtree if $n \geq 2$:
product tree of $x_1, \ldots, x_{\lceil n/2 \rceil}$.
Also right subtree if $n \geq 2$:
product tree of $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$.

e.g. tree for 23, 29, 84, 15, 58



Tree has $\leq (\lg b)^{1+o(1)}$ level
Each level has $\leq b(\lg b)^{0+o(1}$

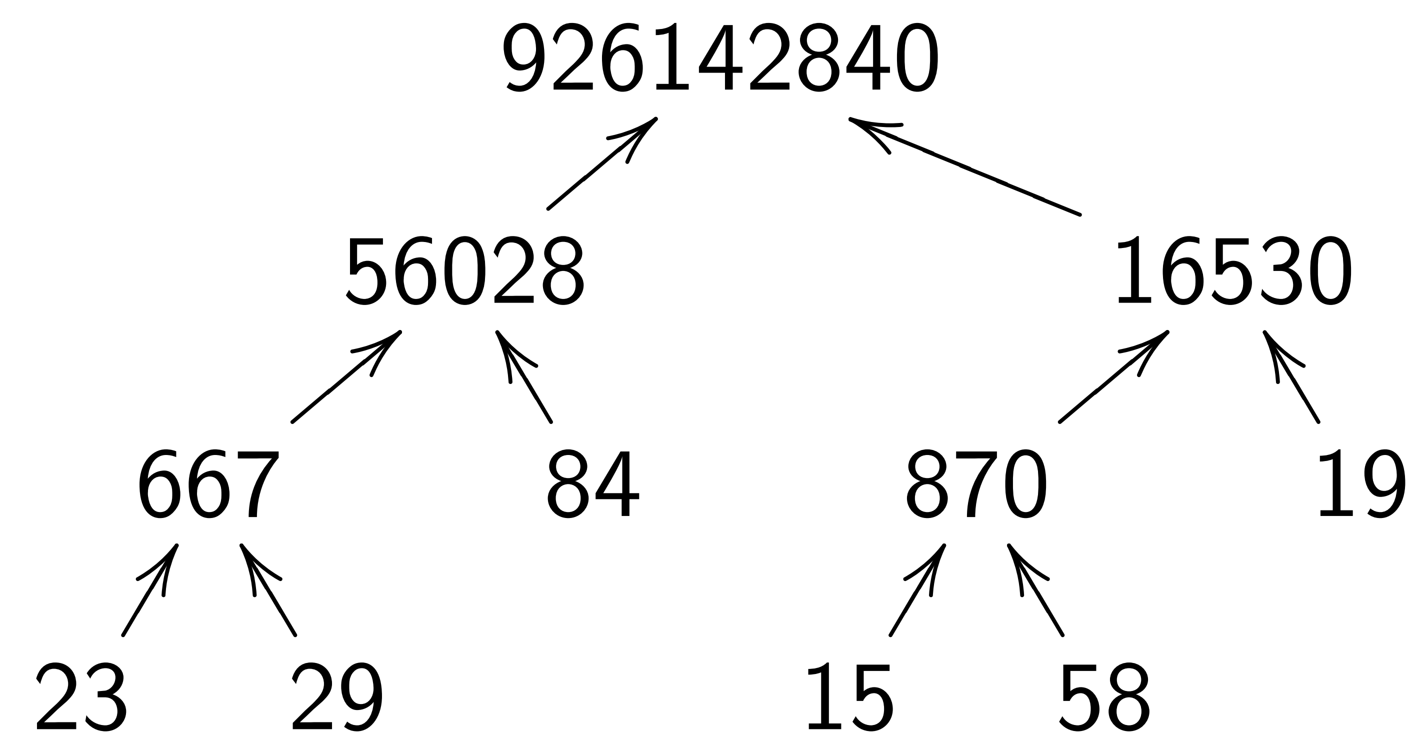
Obtain each level
in time $\leq b(\lg b)^{1+o(1)}$
by multiplying lower-level pa

Product trees

Time $\leq b(\lg b)^{2+o(1)}$
where b is number of input bits:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$,
compute $x_1 x_2 \cdots x_n$.

Actually compute
product tree of $x_1, x_2, \ldots, x_n$.
Root is $x_1 x_2 \cdots x_n$.
Has left subtree if $n \geq 2$:
product tree of $x_1, \ldots, x_{\lceil n/2 \rceil}$.
Also right subtree if $n \geq 2$:
product tree of $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$.

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



Tree has $\leq (\lg b)^{1+o(1)}$ levels.
Each level has $\leq b(\lg b)^{0+o(1)}$ bits.

Obtain each level
in time $\leq b(\lg b)^{1+o(1)}$
by multiplying lower-level pairs.

trees

$b(\lg b)^{2+o(1)}$

is number of input bits:

$_1, x_2, \ldots, x_n \in \mathbf{Z}$,

e $x_1 x_2 \cdots x_n$.

compute

tree of $x_1, x_2, \ldots, x_n$.

$x_1 x_2 \cdots x_n$.

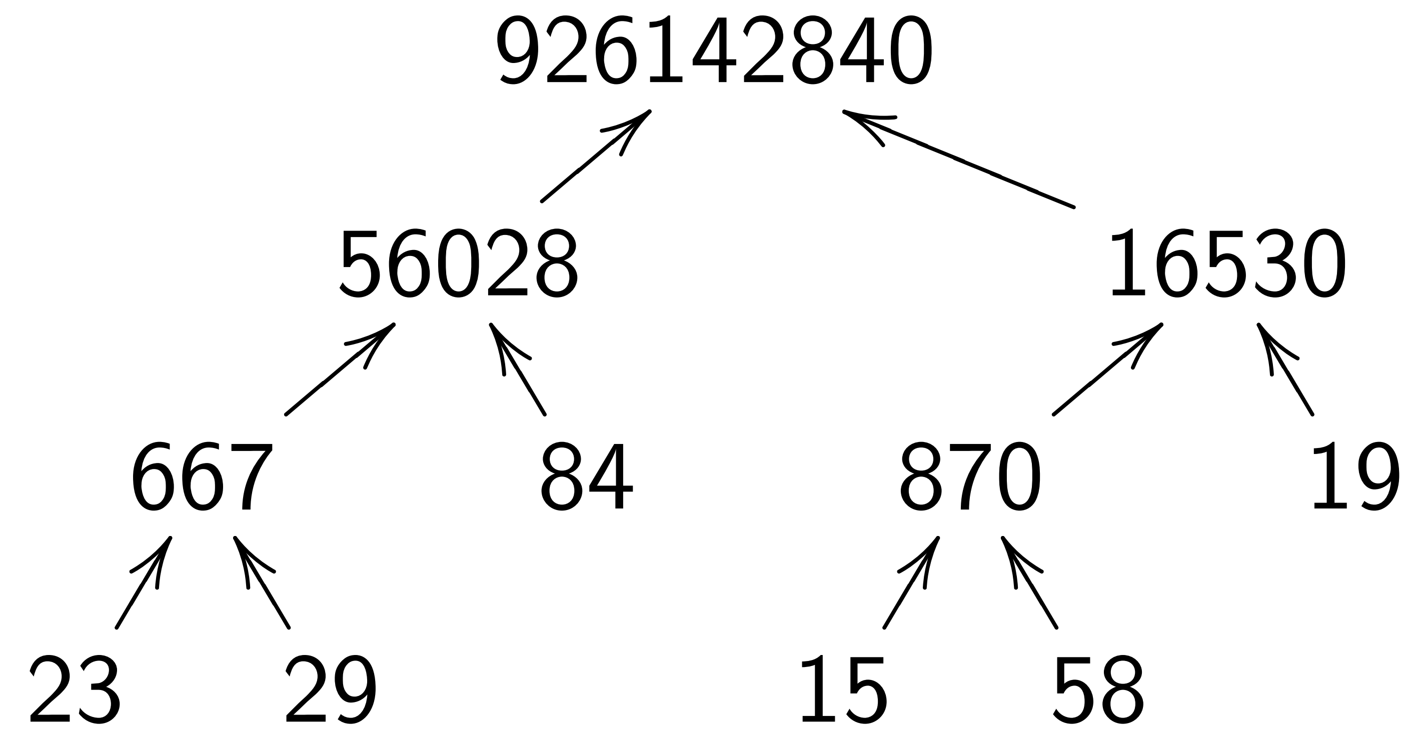
subtree if $n \geq 2$:

tree of $x_1, \ldots, x_{\lceil n/2 \rceil}$.

nt subtree if $n \geq 2$:

tree of $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$.

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



Tree has $\leq (\lg b)^{1+o(1)}$ levels.

Each level has $\leq b(\lg b)^{0+o(1)}$ bits.

Obtain each level

in time $\leq b(\lg b)^{1+o(1)}$

by multiplying lower-level pairs.

Remaind

Remain

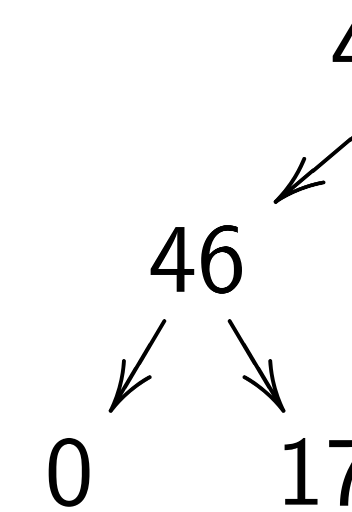
of $r, x_1,$

one nod

in produ

e.g. rem

2230928



(1)

r of input bits:

$x_n \in \mathbf{Z}$,

x_n.

$_1, x_2, \ldots, x_n$.

n.

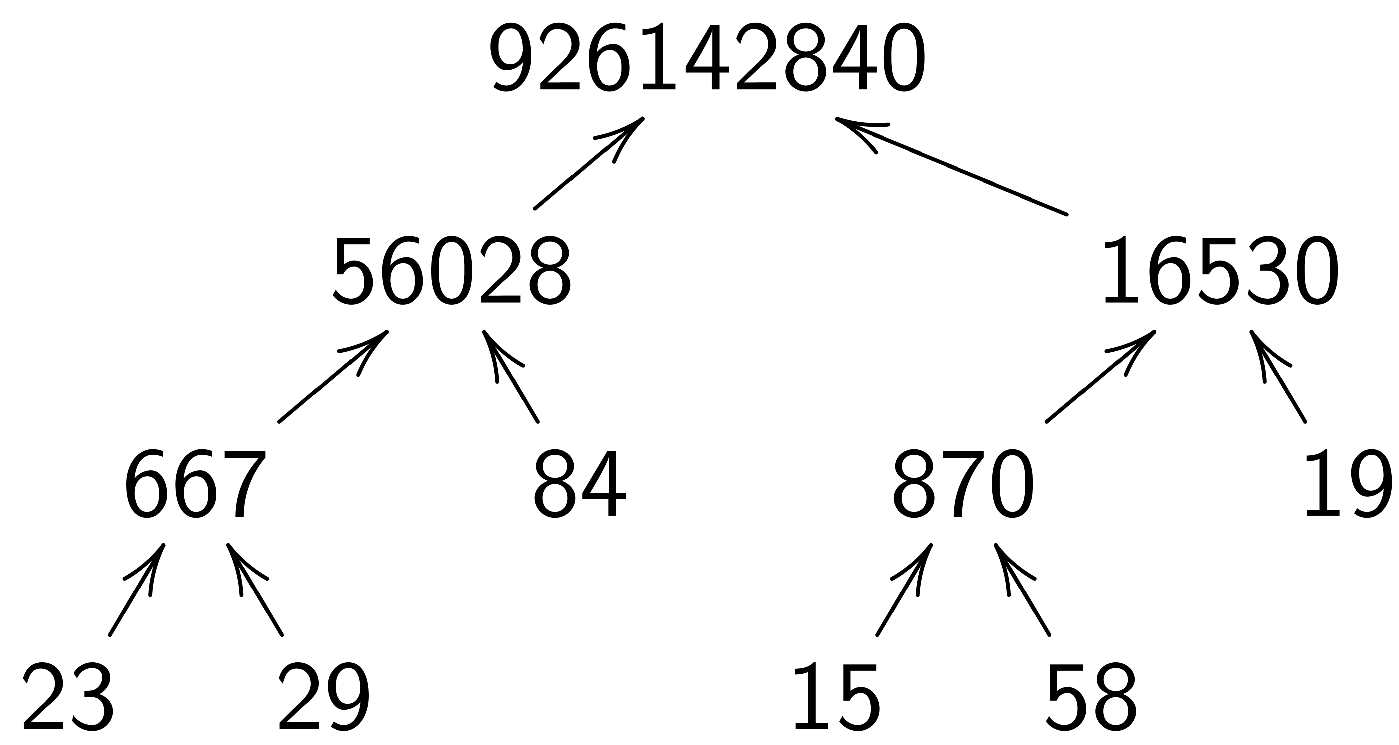
$n \geq 2$:

$, \ldots, x_{\lceil n/2 \rceil}$.

if $n \geq 2$:

$n/2\rceil+1, \ldots, x_n$.

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



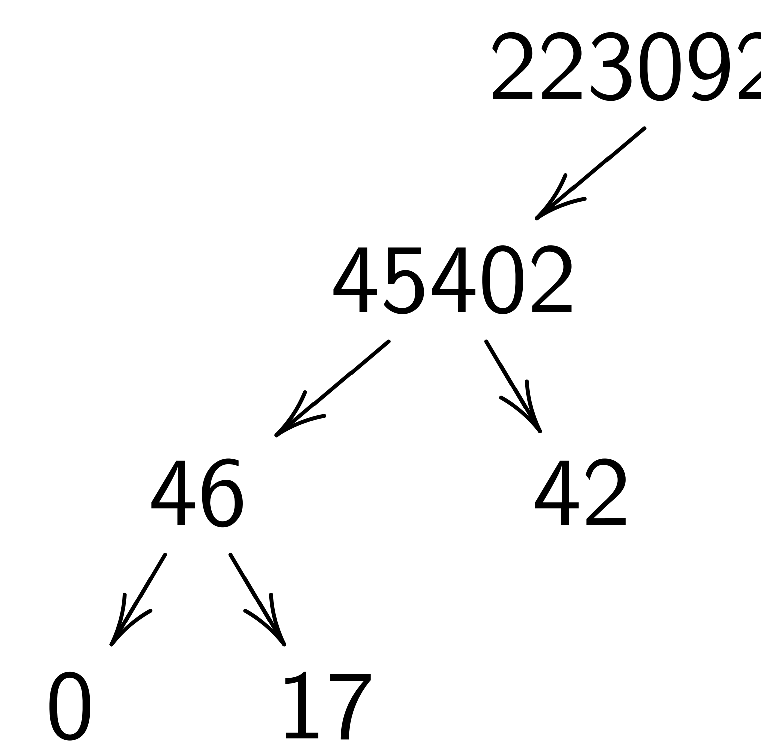
Tree has $\leq (\lg b)^{1+o(1)}$ levels.
Each level has $\leq b(\lg b)^{0+o(1)}$ bits.

Obtain each level
in time $\leq b(\lg b)^{1+o(1)}$
by multiplying lower-level pairs.

Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_r$
one node $r \bmod t$
in product tree of

e.g. remainder tree
223092870, 23, 29,



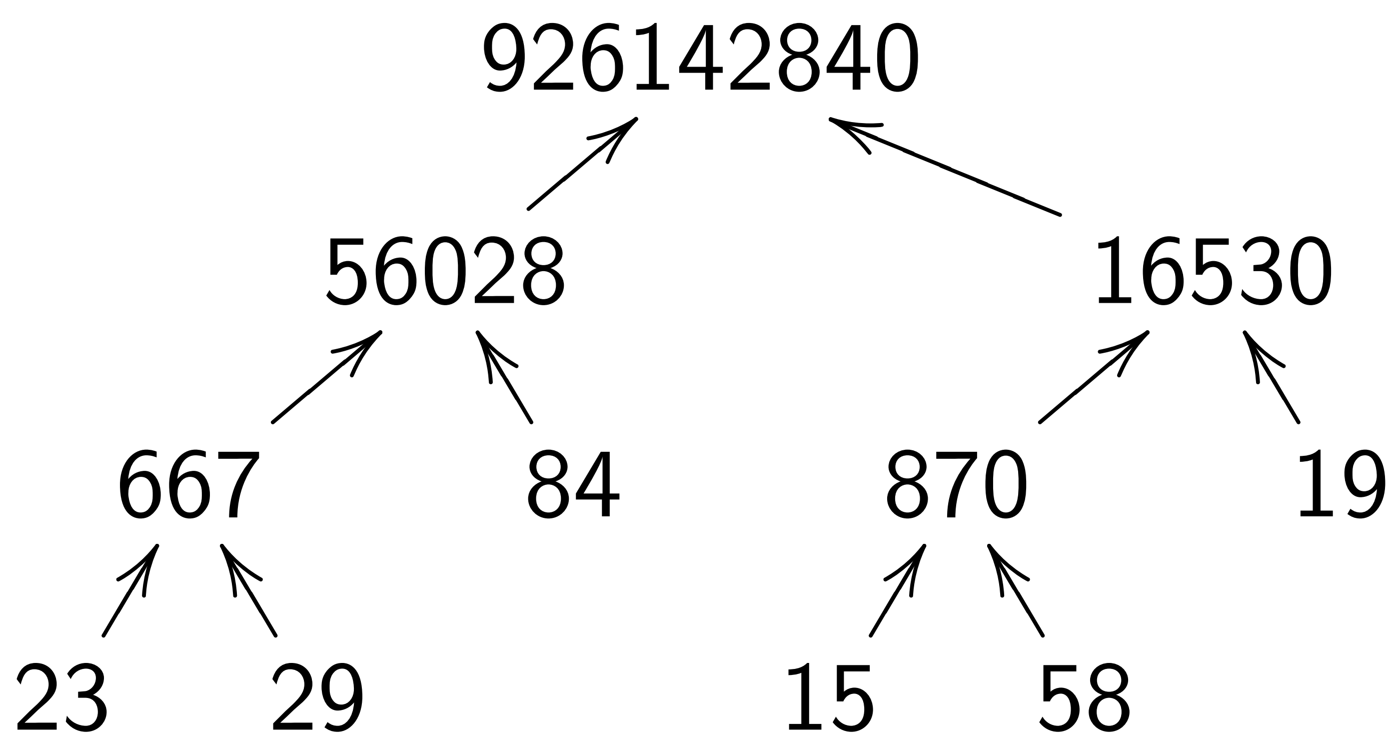
bits:

x_n.

$2\rceil$.

$\ldots, x_n$.

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



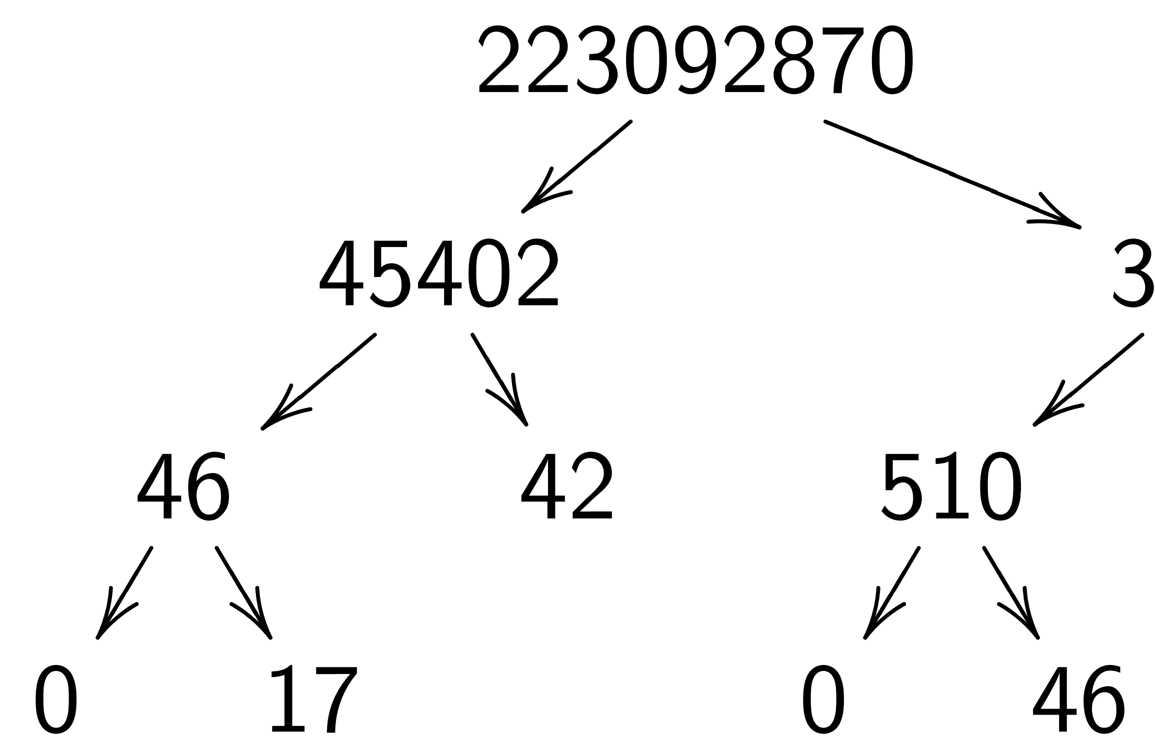
Tree has $\le (\lg b)^{1+o(1)}$ levels.
Each level has $\le b(\lg b)^{0+o(1)}$ bits.

Obtain each level
in time $\le b(\lg b)^{1+o(1)}$
by multiplying lower-level pairs.

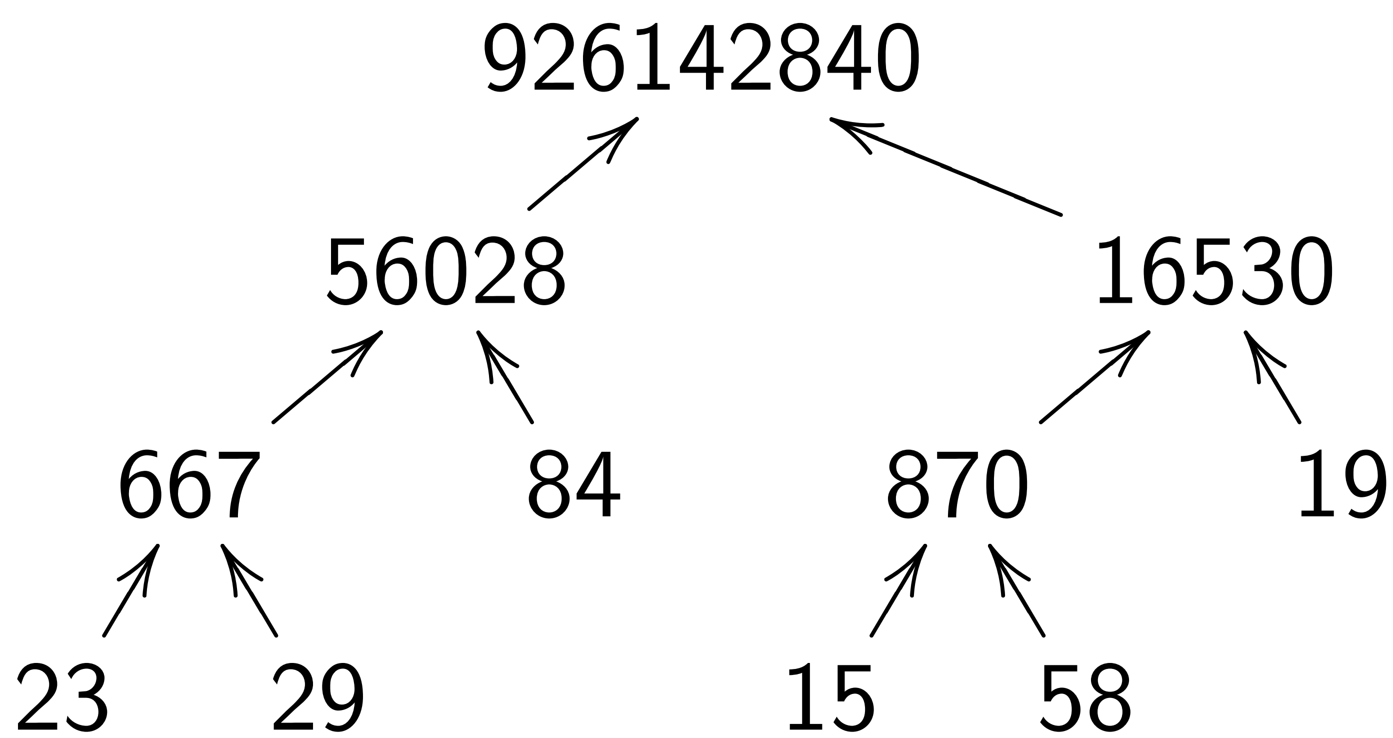
Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_n$ has
one node $r \bmod t$ for each n
in product tree of $x_1, x_2, \ldots$

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



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



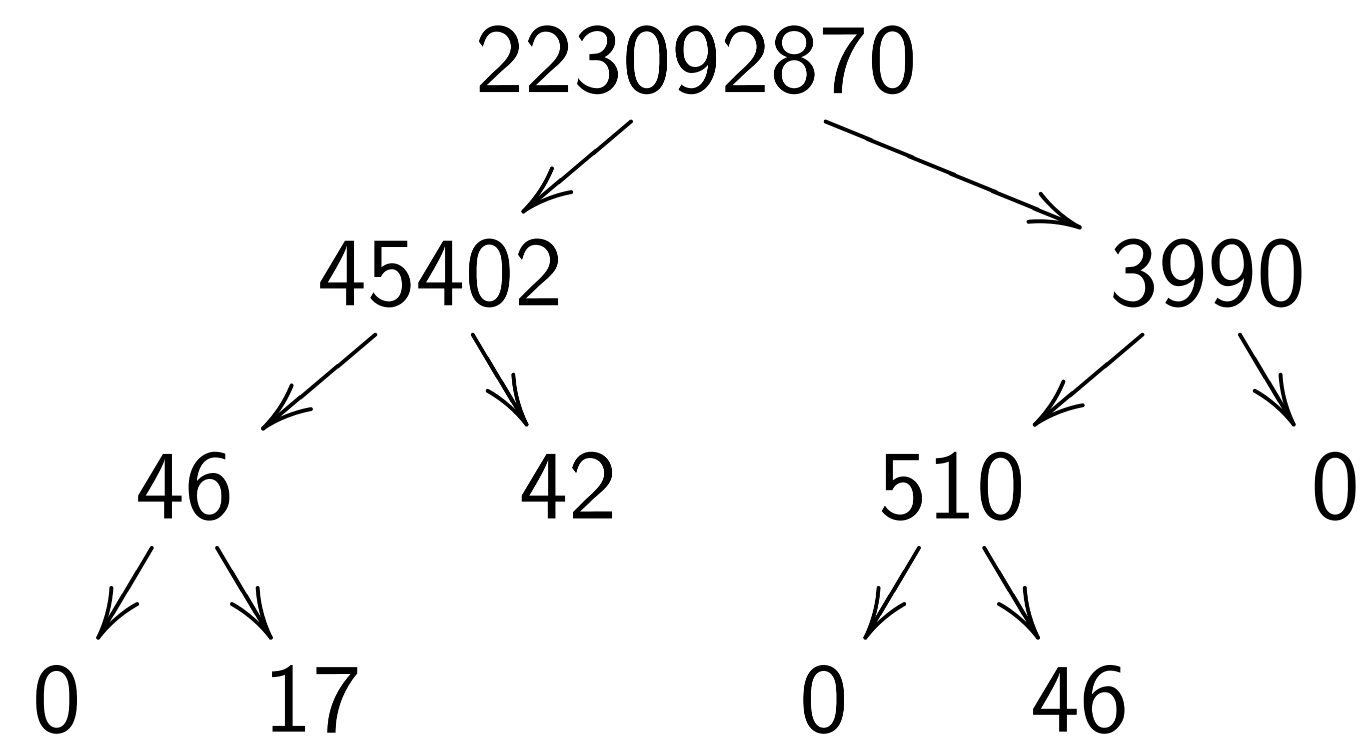
Tree has $\leq (\lg b)^{1+o(1)}$ levels.
Each level has $\leq b(\lg b)^{0+o(1)}$ bits.

Obtain each level
in time $\leq b(\lg b)^{1+o(1)}$
by multiplying lower-level pairs.

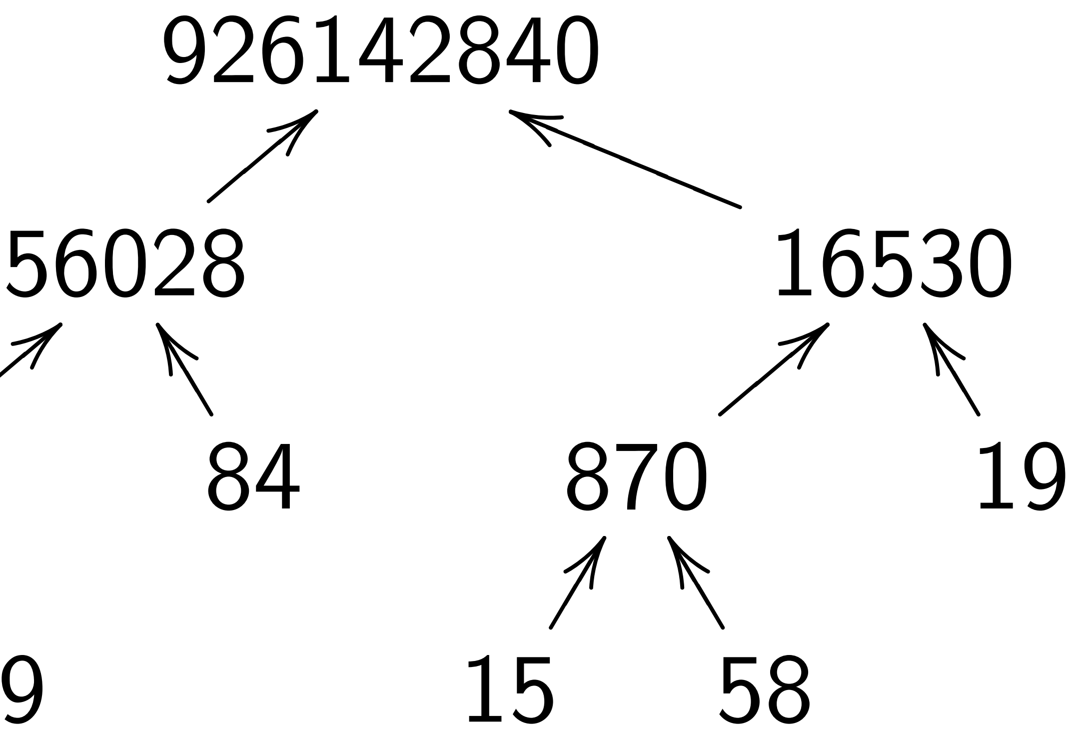
Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_n$ has
one node $r \bmod t$ for each node t
in product tree of $x_1, x_2, \ldots, x_n$.

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



for 23, 29, 84, 15, 58, 19:



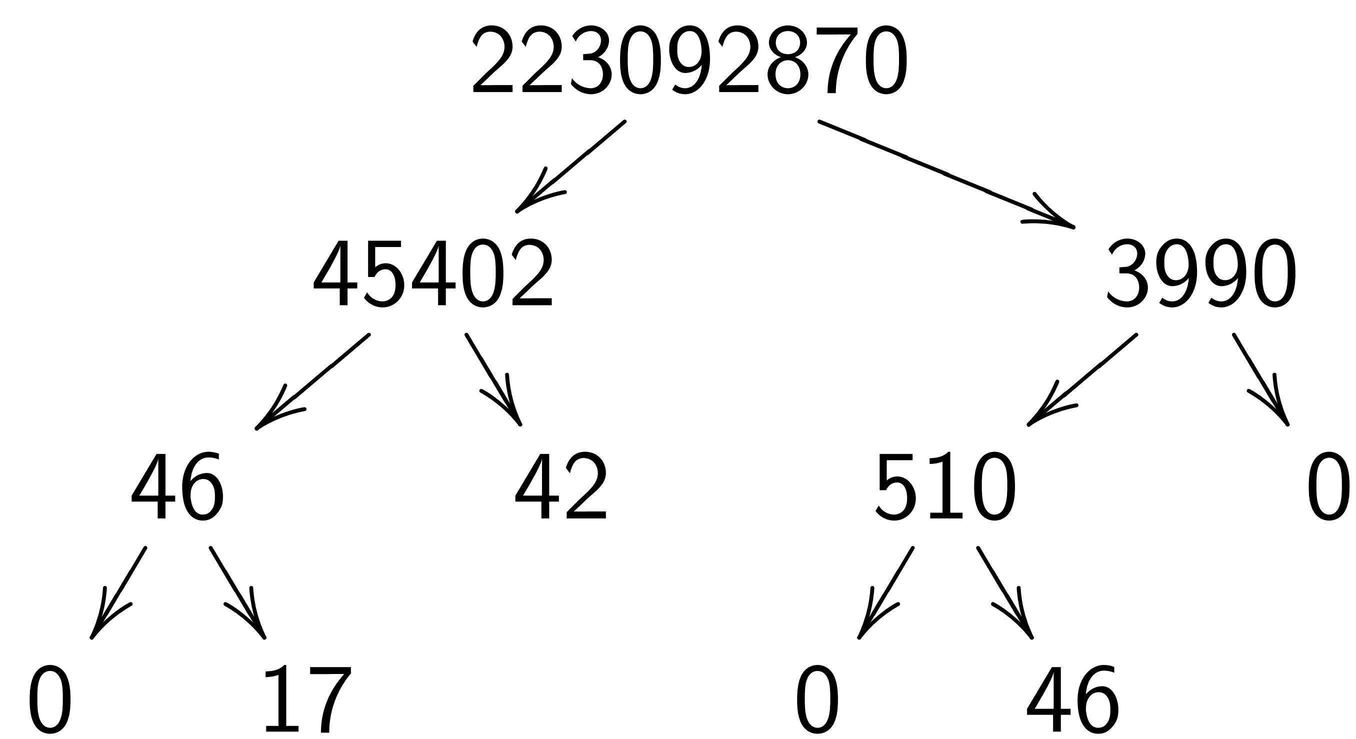
s $\leq (\lg b)^{1+o(1)}$ levels.
vel has $\leq b(\lg b)^{0+o(1)}$ bits.

each level
$\leq b(\lg b)^{1+o(1)}$
plying lower-level pairs.

Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_n$ has
one node $r \bmod t$ for each node t
in product tree of $x_1, x_2, \ldots, x_n$.

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



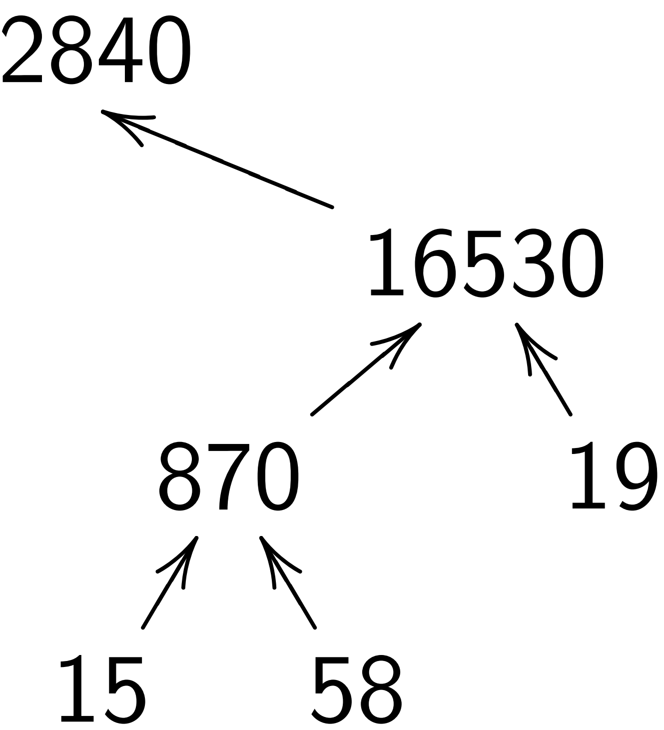
Time $\leq$
Given r
nonzero
compute
of $r, x_1,$

In partic
r mod x

In partic
$x_1, \ldots,$

(1972 M
for "sing
whateve

$, 84, 15, 58, 19$:



$+o(1)$ levels.

$b(\lg b)^{0+o(1)}$ bits.

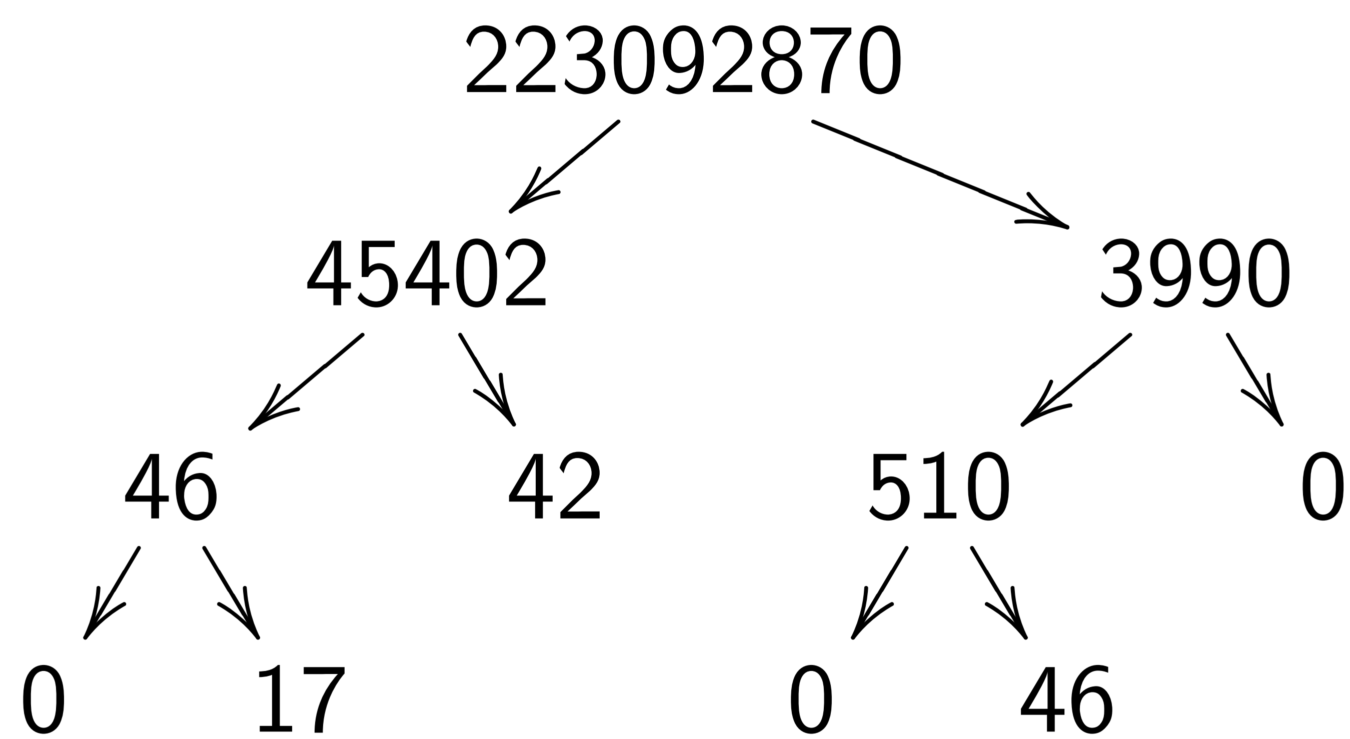
$+o(1)$

er-level pairs.

Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_n$ has
one node $r \bmod t$ for each node t
in product tree of $x_1, x_2, \ldots, x_n$.

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



Time $\le b(\lg b)^{2+o}$

Given $r \in \mathbf{Z}$ and
nonzero $x_1, \ldots, x_$
compute remainde
of $r, x_1, \ldots, x_n$.

In particular, comp
$r \bmod x_1, \ldots, r$ m

In particular, see w
$x_1, \ldots, x_n$ divide

(1972 Moenck Bo
for "single precisio
whatever exactly t

3, 19:

6530

19

s.

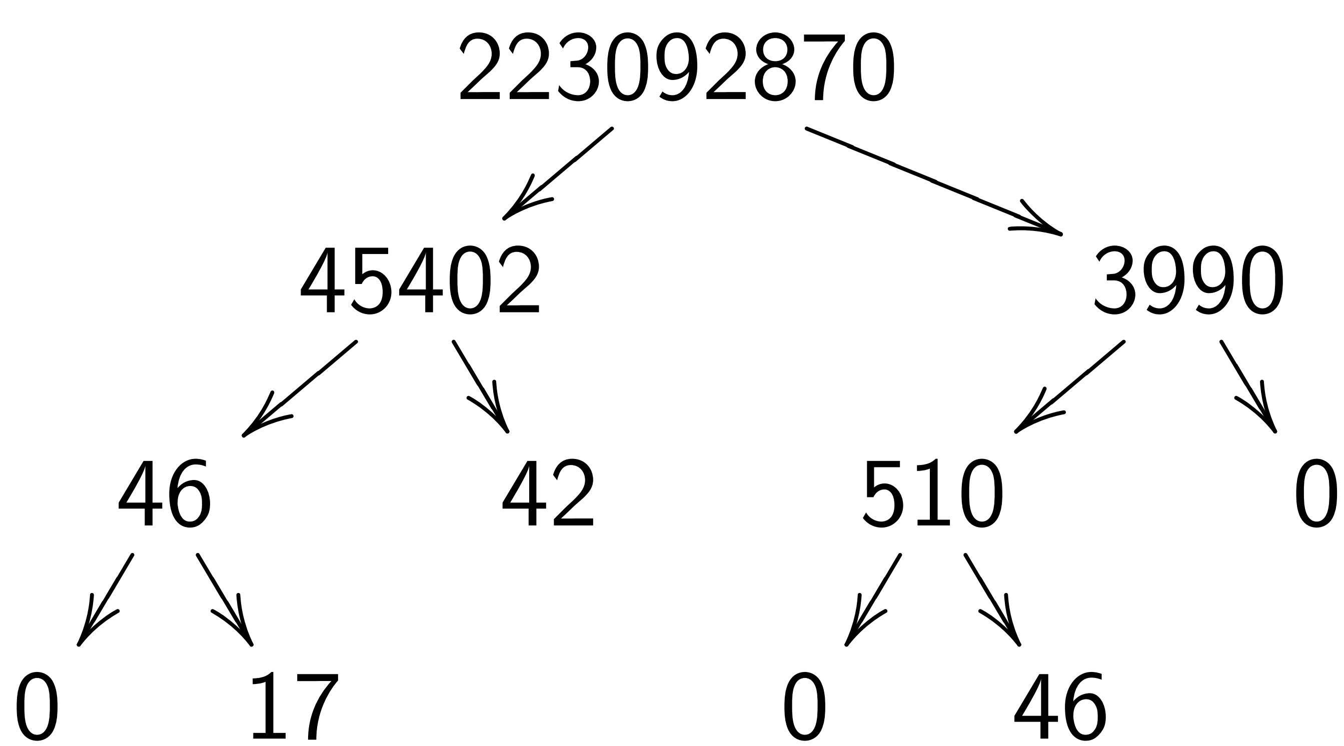
1) bits.

irs.

Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_n$ has
one node $r \bmod t$ for each node t
in product tree of $x_1, x_2, \ldots, x_n$.

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



Time $\le b(\lg b)^{2+o(1)}$:
Given $r \in \mathbf{Z}$ and
nonzero $x_1, \ldots, x_n \in \mathbf{Z}$,
compute remainder tree
of $r, x_1, \ldots, x_n$.

In particular, compute
$r \bmod x_1, \ldots, r \bmod x_n$.

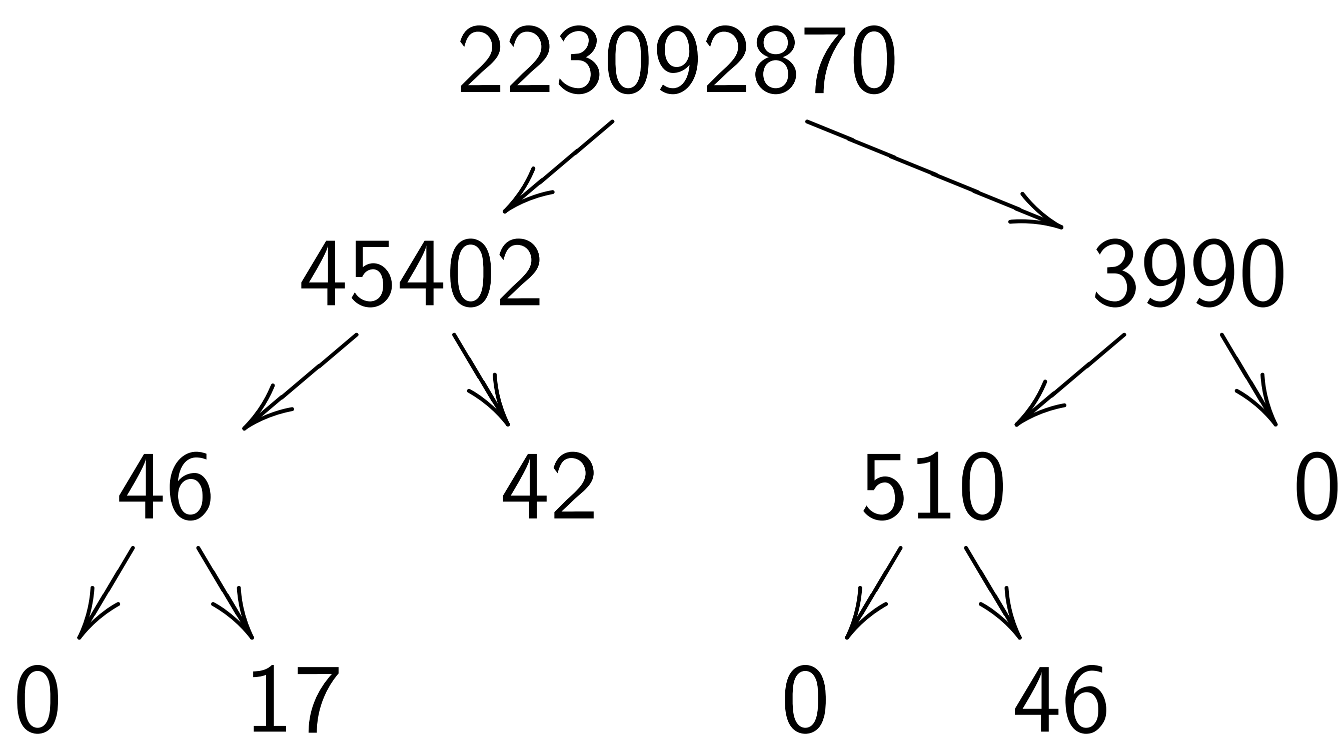
In particular, see which of
$x_1, \ldots, x_n$ divide r.

(1972 Moenck Borodin,
for "single precision" x_i's,
whatever exactly that means

Remainder trees

Remainder tree
of $r, x_1, x_2, \ldots, x_n$ has
one node $r \bmod t$ for each node t
in product tree of $x_1, x_2, \ldots, x_n$.

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



Time $\le b(\lg b)^{2+o(1)}$:
Given $r \in \mathbf{Z}$ and
nonzero $x_1, \ldots, x_n \in \mathbf{Z}$,
compute remainder tree
of $r, x_1, \ldots, x_n$.

In particular, compute
$r \bmod x_1, \ldots, r \bmod x_n$.

In particular, see which of
$x_1, \ldots, x_n$ divide r.

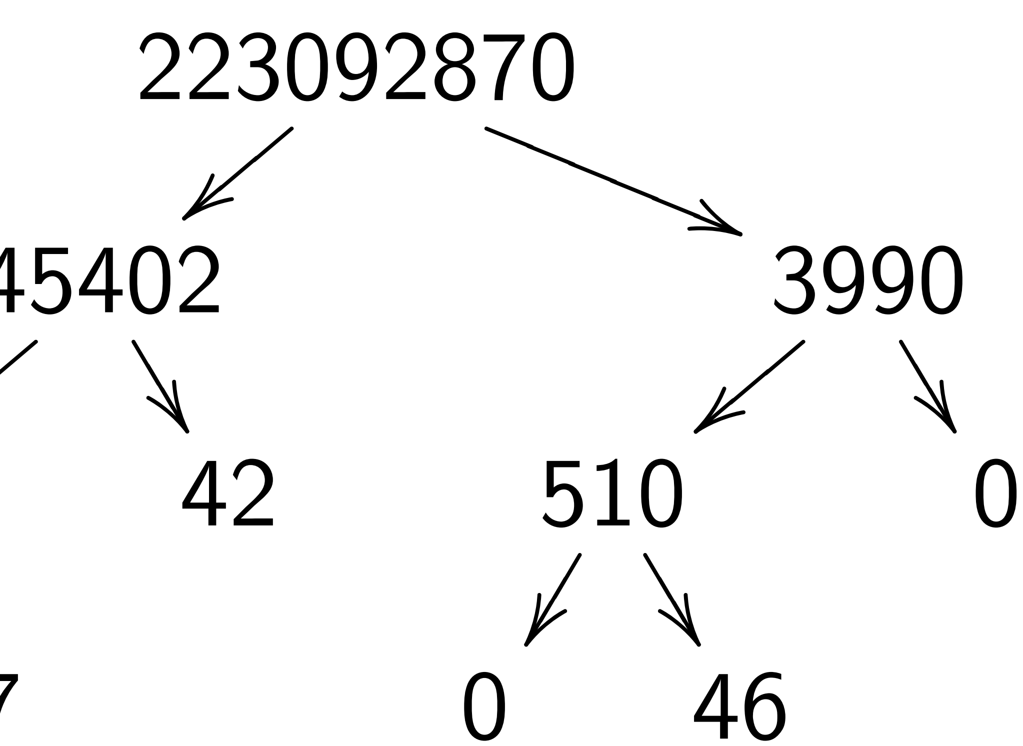
(1972 Moenck Borodin,
for "single precision" x_i's,
whatever exactly that means)

der trees

der tree

$x_2, \ldots, x_n$ has
e r mod t for each node t
ct tree of $x_1, x_2, \ldots, x_n$.

ainder tree of
70, 23, 29, 84, 15, 58, 19:



Time $\leq b(\lg b)^{2+o(1)}$:
Given $r \in \mathbf{Z}$ and
nonzero $x_1, \ldots, x_n \in \mathbf{Z}$,
compute remainder tree
of $r, x_1, \ldots, x_n$.

In particular, compute
$r \bmod x_1, \ldots, r \bmod x_n$.

In particular, see which of
$x_1, \ldots, x_n$ divide r.

(1972 Moenck Borodin,
for "single precision" x_i's,
whatever exactly that means)

Small pr

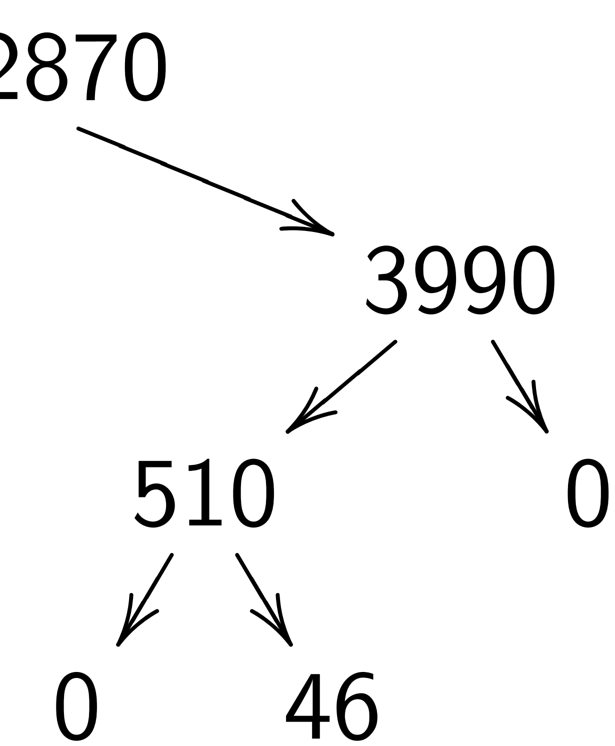
Time $\leq$
Given x_1
finite set
$\{p \in Q$:

In partic
see whet
any of x

Algorith
1. Use a
comp
2. Use a
which

$_n$ has
for each node t
$x_1, x_2, \ldots, x_n$.

e of
$84, 15, 58, 19$:



Time $\le b(\lg b)^{2+o(1)}$:
Given $r \in \mathbf{Z}$ and
nonzero $x_1, \ldots, x_n \in \mathbf{Z}$,
compute remainder tree
of $r, x_1, \ldots, x_n$.

In particular, compute
$r \bmod x_1, \ldots, r \bmod x_n$.

In particular, see which of
$x_1, \ldots, x_n$ divide r.

(1972 Moenck Borodin,
for "single precision" x_i's,
whatever exactly that means)

Small primes, unic

Time $\le b(\lg b)^{2+o}$
Given $x_1, x_2, \ldots,$
finite set $Q \subseteq \mathbf{Z} -$
$\{p \in Q : x_1 x_2 \cdots$

In particular, wher
see whether p divi
any of $x_1, x_2, \ldots,$

Algorithm:
1. Use a product t
compute $r = x$
2. Use a remainde
which $p \in Q$ di

ode t

, x_n.

, 19:

990

0

Time $\le b(\lg b)^{2+o(1)}$:
Given $r \in \mathbf{Z}$ and
nonzero $x_1, \ldots, x_n \in \mathbf{Z}$,
compute remainder tree
of $r, x_1, \ldots, x_n$.

In particular, compute
$r \bmod x_1, \ldots, r \bmod x_n$.

In particular, see which of
$x_1, \ldots, x_n$ divide r.

(1972 Moenck Borodin,
for "single precision" x_i's,
whatever exactly that means)

Small primes, union

Time $\le b(\lg b)^{2+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ an
finite set $Q \subseteq \mathbf{Z} - \{0\}$, com
$\{p \in Q : x_1 x_2 \cdots x_n \bmod p$

In particular, when p is prim
see whether p divides
any of $x_1, x_2, \ldots, x_n$.

Algorithm:

1. Use a product tree to
   compute $r = x_1 x_2 \cdots x_n$
2. Use a remainder tree to s
   which $p \in Q$ divide r.

Time $\le b(\lg b)^{2+o(1)}$:
Given $r \in \mathbf{Z}$ and nonzero $x_1, \ldots, x_n \in \mathbf{Z}$, compute remainder tree of $r, x_1, \ldots, x_n$.

In particular, compute $r \bmod x_1, \ldots, r \bmod x_n$.

In particular, see which of $x_1, \ldots, x_n$ divide r.

(1972 Moenck Borodin, for "single precision" x_i's, whatever exactly that means)

Small primes, union

Time $\le b(\lg b)^{2+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set $Q \subseteq \mathbf{Z} - \{0\}$, compute $\{p \in Q : x_1 x_2 \cdots x_n \bmod p = 0\}$.

In particular, when p is prime, see whether p divides any of $x_1, x_2, \ldots, x_n$.

Algorithm:

1. Use a product tree to compute $r = x_1 x_2 \cdots x_n$.
2. Use a remainder tree to see which $p \in Q$ divide r.

$b(\lg b)^{2+o(1)}$:
$\in$ **Z** and
$x_1, \ldots, x_n \in$ **Z**,
e remainder tree
$\ldots, x_n$.

cular, compute
$_1, \ldots, r \bmod x_n$.

cular, see which of
x_n divide r.

loenck Borodin,
gle precision" x_i's,
r exactly that means)

Small primes, union

Time $\le b(\lg b)^{2+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in$ **Z** and
finite set $Q \subseteq$ **Z** $- \{0\}$, compute
$\{p \in Q : x_1 x_2 \cdots x_n \bmod p = 0\}$.

In particular, when p is prime,
see whether p divides
any of $x_1, x_2, \ldots, x_n$.

Algorithm:

1. Use a product tree to compute $r = x_1 x_2 \cdots x_n$.
2. Use a remainder tree to see which $p \in Q$ divide r.

Small pr

Time $\le$
Given x_1
finite set
compute
$\ldots, \{p$
(2000 B

Algorith

1. Repla
   $\{p \in$
2. If n =
3. Recur
4. Recur

(1):

$_n \in$ **Z**,
er tree

oute
nod x_n.

which of
r.

rodin,
on'' x_i's,
that means)

Small primes, union

Time $\le b(\lg b)^{2+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in$ **Z** and finite set $Q \subseteq$ **Z** $- \{0\}$, compute $\{p \in Q : x_1 x_2 \cdots x_n \bmod p = 0\}$.

In particular, when p is prime, see whether p divides any of $x_1, x_2, \ldots, x_n$.

Algorithm:

1. Use a product tree to compute $r = x_1 x_2 \cdots x_n$.
2. Use a remainder tree to see which $p \in Q$ divide r.

Small primes, sepa

Time $\le b(\lg b)^{3+o}$
Given $x_1, x_2, \ldots, x$
finite set Q of prin
compute $\{p \in Q :$
$\ldots, \{p \in Q : x_n$ m
(2000 Bernstein)

Algorithm for $n \ge$

1. Replace Q with
   $\{p \in Q : x_1 \cdots$
2. If $n = 1$, print
3. Recurse on $x_1,$
4. Recurse on $x_{\lceil n}$

s)

Small primes, union

Time $\leq b(\lg b)^{2+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set $Q \subseteq \mathbf{Z} - \{0\}$, compute $\{p \in Q : x_1 x_2 \cdots x_n \bmod p = 0\}$.

In particular, when p is prime, see whether p divides any of $x_1, x_2, \ldots, x_n$.

Algorithm:

1. Use a product tree to compute $r = x_1 x_2 \cdots x_n$.
2. Use a remainder tree to see which $p \in Q$ divide r.

Small primes, separately

Time $\leq b(\lg b)^{3+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ an
finite set Q of primes,
compute $\{p \in Q : x_1 \bmod p$
$\ldots, \{p \in Q : x_n \bmod p = 0$
(2000 Bernstein)

Algorithm for $n \geq 1$:

1. Replace Q with
   $\{p \in Q : x_1 \cdots x_n \bmod p$
2. If $n = 1$, print Q and sto
3. Recurse on $x_1, \ldots, x_{\lceil n/2}$
4. Recurse on $x_{\lceil n/2 \rceil + 1}, \ldots$

Small primes, union

Time $\le b(\lg b)^{2+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set $Q \subseteq \mathbf{Z} - \{0\}$, compute $\{p \in Q : x_1 x_2 \cdots x_n \bmod p = 0\}$.

In particular, when p is prime, see whether p divides any of $x_1, x_2, \ldots, x_n$.

Algorithm:

1. Use a product tree to compute $r = x_1 x_2 \cdots x_n$.
2. Use a remainder tree to see which $p \in Q$ divide r.

Small primes, separately

Time $\le b(\lg b)^{3+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes, compute $\{p \in Q : x_1 \bmod p = 0\}$, $\ldots$, $\{p \in Q : x_n \bmod p = 0\}$.
(2000 Bernstein)

Algorithm for $n \ge 1$:

1. Replace Q with $\{p \in Q : x_1 \cdots x_n \bmod p = 0\}$.
2. If $n = 1$, print Q and stop.
3. Recurse on $x_1, \ldots, x_{\lceil n/2 \rceil}, Q$.
4. Recurse on $x_{\lceil n/2 \rceil + 1}, \ldots, x_n, Q$.

rimes, union

$b(\lg b)^{2+o(1)}$:

$_1, x_2, \ldots, x_n \in \mathbf{Z}$ and

t $Q \subseteq \mathbf{Z} - \{0\}$, compute

: $x_1 x_2 \cdots x_n \bmod p = 0\}$.

cular, when p is prime,

ther p divides

$_1, x_2, \ldots, x_n$.

m:

a product tree to

pute $r = x_1 x_2 \cdots x_n$.

a remainder tree to see

h $p \in Q$ divide r.

Small primes, separately

Time $\le b(\lg b)^{3+o(1)}$:

Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and

finite set Q of primes,

compute $\{p \in Q : x_1 \bmod p = 0\}$,

$\ldots$, $\{p \in Q : x_n \bmod p = 0\}$.

(2000 Bernstein)

Algorithm for $n \ge 1$:

1. Replace Q with $\{p \in Q : x_1 \cdots x_n \bmod p = 0\}$.
2. If $n = 1$, print Q and stop.
3. Recurse on $x_1, \ldots, x_{\lceil n/2 \rceil}, Q$.
4. Recurse on $x_{\lceil n/2 \rceil + 1}, \ldots, x_n, Q$.

Factor

over

2543,

ove

2, 3, 7

2543

over

2, 17

Each lev

on

$^{(1)}$:

$x_n \in \mathbf{Z}$ and

$\{0\}$, compute

$x_n \bmod p = 0\}$.

p is prime,

des

x_n.

tree to

$_1 x_2 \cdots x_n$.

er tree to see

vide r.

Small primes, separately

Time $\le b(\lg b)^{3+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and
finite set Q of primes,
compute $\{p \in Q : x_1 \bmod p = 0\}$,
$\ldots$, $\{p \in Q : x_n \bmod p = 0\}$.
(2000 Bernstein)

Algorithm for $n \ge 1$:

1. Replace Q with
   $\{p \in Q : x_1 \cdots x_n \bmod p = 0\}$.
2. If $n = 1$, print Q and stop.
3. Recurse on $x_1, \ldots, x_{\lceil n/2 \rceil}, Q$.
4. Recurse on $x_{\lceil n/2 \rceil + 1}, \ldots, x_n, Q$.

Factor 2543, 676
over {2, 3, 5, 7,

2543, 6766
over
2, 3, 7, 17

2543	6766
over	over
2, 17	2, 17

Each level has $\le b$

d
pute
$= 0\}$.
e,
.
see

Small primes, separately

Time $\le b(\lg b)^{3+o(1)}$:
Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes, compute $\{p \in Q : x_1 \bmod p = 0\}$, $\ldots$, $\{p \in Q : x_n \bmod p = 0\}$.
(2000 Bernstein)

Algorithm for $n \ge 1$:

1. Replace Q with $\{p \in Q : x_1 \cdots x_n \bmod p = 0\}$.
2. If $n = 1$, print Q and stop.
3. Recurse on $x_{1}, \ldots, x_{\lceil n/2 \rceil}, Q$.
4. Recurse on $x_{\lceil n/2 \rceil + 1}, \ldots, x_n, Q$.

Factor $2543, 6766, 8967, 75$
over $\{\underline{2}, \underline{3}, 5, \underline{7}, 11, 13, \underline{17}$

$2543, 6766$ over $\underline{2}, 3, 7, \underline{17}$

$8967, 759$ over $\underline{2}, \underline{3}, \underline{7}, 1$

2543 over $2, 17$

6766 over $\underline{2}, \underline{17}$

8967 over $2, \underline{3}, \underline{7}$

75 o $\underline{2},$

Each level has $\le b(\lg b)^{0+o(1}$

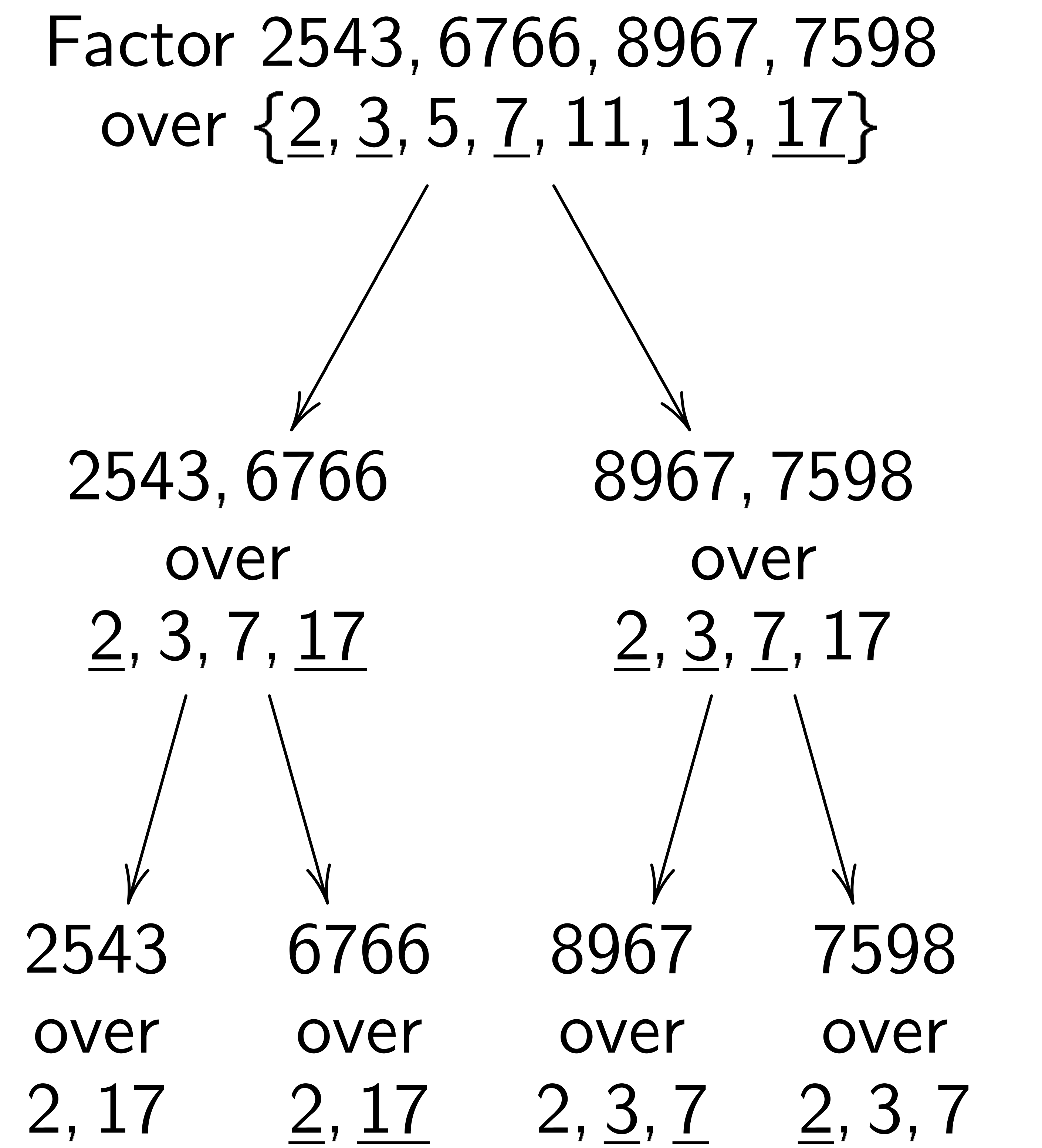
Small primes, separately

Time $\le b(\lg b)^{3+o(1)}$:

Given $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes, compute $\{p \in Q : x_1 \bmod p = 0\}$, $\ldots$, $\{p \in Q : x_n \bmod p = 0\}$.
(2000 Bernstein)

Algorithm for $n \ge 1$:

1. Replace Q with $\{p \in Q : x_1 \cdots x_n \bmod p = 0\}$.
2. If $n = 1$, print Q and stop.
3. Recurse on $x_1, \ldots, x_{\lceil n/2 \rceil}, Q$.
4. Recurse on $x_{\lceil n/2 \rceil + 1}, \ldots, x_n, Q$.



Each level has $\le b(\lg b)^{0+o(1)}$ bits.

rimes, separately

$b(\lg b)^{3+o(1)}$:

$_1, x_2, \ldots, x_n \in \mathbf{Z}$ and

t Q of primes,

e $\{p \in Q : x_1 \bmod p = 0\}$,

$\in Q : x_n \bmod p = 0\}$.

ernstein)

m for $n \geq 1$:

ace Q with

$Q : x_1 \cdots x_n \bmod p = 0\}$.

$= 1$, print Q and stop.

rse on $x_1, \ldots, x_{\lceil n/2 \rceil}, Q$.

rse on $x_{\lceil n/2 \rceil + 1}, \ldots, x_n, Q$.

Factor $2543, 6766, 8967, 7598$
over $\{\underline{2}, \underline{3}, 5, \underline{7}, 11, 13, \underline{17}\}$

$2543, 6766$ over $\underline{2}, 3, 7, \underline{17}$

$8967, 7598$ over $\underline{2}, \underline{3}, \underline{7}, 17$

2543 over $2, 17$

6766 over $\underline{2}, \underline{17}$

8967 over $2, \underline{3}, \underline{7}$

7598 over $\underline{2}, 3, 7$

Each level has $\leq b(\lg b)^{0+o(1)}$ bits.

Exponen

Time $\leq$

Given no

find e, p

Algorith

1. If x n
   Print
2. Find
   with
3. If r n
   $2f +$
4. Print

arately

$^{(1)}$:

$x_n \in \mathbf{Z}$ and

mes,

$x_1 \bmod p = 0\}$,

mod $p = 0\}$.

1:

n

$x_n \bmod p = 0\}$.

Q and stop.

$\ldots, x_{\lceil n/2 \rceil}, Q$.

$/2\rceil+1}, \ldots, x_n, Q$.

Factor $2543, 6766, 8967, 7598$
over $\{\underline{2}, \underline{3}, 5, \underline{7}, 11, 13, \underline{17}\}$

$2543, 6766$ over $\underline{2}, 3, 7, \underline{17}$

$8967, 7598$ over $\underline{2}, \underline{3}, \underline{7}, 17$

2543 over $2, 17$

6766 over $\underline{2}, \underline{17}$

8967 over $2, \underline{3}, \underline{7}$

7598 over $\underline{2}, 3, 7$

Each level has $\le b(\lg b)^{0+o(1)}$ bits.

Exponents of a sm

Time $\le b(\lg b)^{2+o}$

Given nonzero p, x

find $e, p^e, x/p^e$ wi

Algorithm:

1. If $x \bmod p \ne 0$
   Print $0, 1, x$ an
2. Find $f, (p^2)^f, r$
   with maximal f
3. If $r \bmod p = 0$:
   $2f + 2, (p^2)^f p^2$
4. Print $2f + 1, ($

d

$= 0\}$,

$\}$.

$= 0\}$.

p.

$\rceil, Q$.

$, x_n, Q$.

Factor $2543, 6766, 8967, 7598$
over $\{\underline{2}, \underline{3}, 5, \underline{7}, 11, 13, \underline{17}\}$

$2543, 6766$ over $\underline{2}, 3, 7, \underline{17}$

$8967, 7598$ over $\underline{2}, \underline{3}, \underline{7}, 17$

2543 over $2, 17$

6766 over $\underline{2}, \underline{17}$

8967 over $2, \underline{3}, \underline{7}$

7598 over $\underline{2}, 3, 7$

Each level has $\le b(\lg b)^{0+o(1)}$ bits.

Exponents of a small prime

Time $\le b(\lg b)^{2+o(1)}$:
Given nonzero $p, x \in \mathbf{Z}$,
find $e, p^e, x/p^e$ with maxima

Algorithm:

1. If $x \bmod p \ne 0$:
   Print $0, 1, x$ and stop.
2. Find $f, (p^2)^f, r = (x/p)/$
   with maximal f.
3. If $r \bmod p = 0$: Print
   $2f + 2, (p^2)^f p^2, r/p$ and
4. Print $2f + 1, (p^2)^f p, r$.

Factor 2543, 6766, 8967, 7598
over $\{\underline{2}, \underline{3}, 5, \underline{7}, 11, 13, \underline{17}\}$

- 2543, 6766 over $\underline{2}, 3, 7, \underline{17}$
  - 2543 over 2, 17
  - 6766 over $\underline{2}, \underline{17}$
- 8967, 7598 over $\underline{2}, \underline{3}, \underline{7}, 17$
  - 8967 over $2, \underline{3}, \underline{7}$
  - 7598 over $\underline{2}, 3, 7$

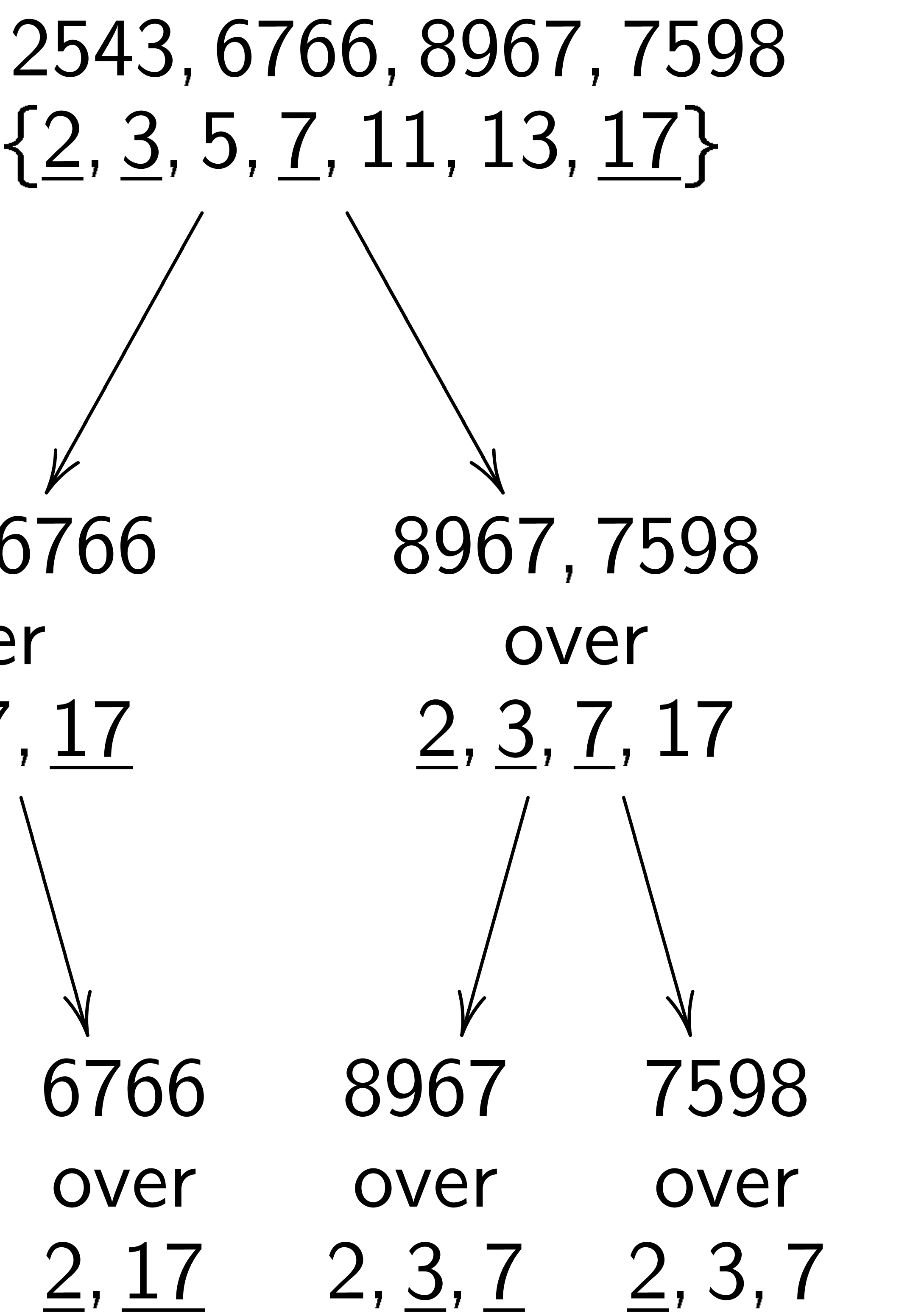
Each level has $\leq b(\lg b)^{0+o(1)}$ bits.

Exponents of a small prime

Time $\leq b(\lg b)^{2+o(1)}$:
Given nonzero $p, x \in \mathbf{Z}$,
find $e, p^e, x/p^e$ with maximal e.

Algorithm:

1. If $x \bmod p \neq 0$:
   Print $0, 1, x$ and stop.
2. Find $f, (p^2)^f, r = (x/p)/(p^2)^f$
   with maximal f.
3. If $r \bmod p = 0$: Print
   $2f + 2, (p^2)^f p^2, r/p$ and stop.
4. Print $2f + 1, (p^2)^f p, r$.



vel has $\leq b(\lg b)^{0+o(1)}$ bits.

Exponents of a small prime

Time $\leq b(\lg b)^{2+o(1)}$:
Given nonzero $p, x \in \mathbf{Z}$,
find $e, p^e, x/p^e$ with maximal e.

Algorithm:

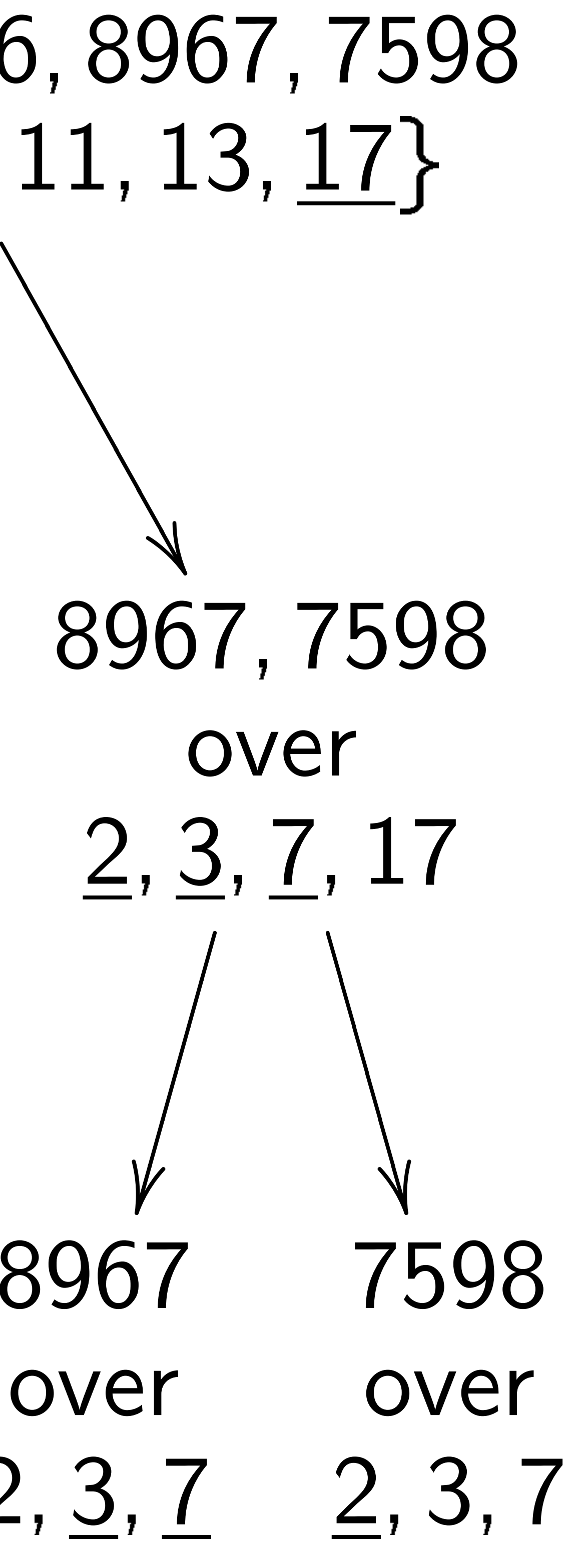
1. If $x \bmod p \neq 0$:
   Print $0, 1, x$ and stop.
2. Find $f, (p^2)^f, r = (x/p)/(p^2)^f$
   with maximal f.
3. If $r \bmod p = 0$: Print
   $2f + 2, (p^2)^f p^2, r/p$ and stop.
4. Print $2f + 1, (p^2)^f p, r$.

Exponen

Time $\leq$
Given fi
and non
$e, \prod_{p \in Q}$

Algorith

1. Repla
   $\{p \in$
2. Find
   $s = \prod$
3. Find
4. Outp
   where



$b(\lg b)^{0+o(1)}$ bits.

Exponents of a small prime

Time $\le b(\lg b)^{2+o(1)}$:

Given nonzero $p, x \in \mathbf{Z}$,

find $e, p^e, x/p^e$ with maximal e.

Algorithm:

1. If $x \bmod p \ne 0$:
   Print $0, 1, x$ and stop.
2. Find $f, (p^2)^f, r = (x/p)/(p^2)^f$ with maximal f.
3. If $r \bmod p = 0$: Print $2f + 2, (p^2)^f p^2, r/p$ and stop.
4. Print $2f + 1, (p^2)^f p, r$.

Exponents of smal

Time $\le b(\lg b)^{3+o}$

Given finite set Q

and nonzero $x \in \mathbf{Z}$

$e, \prod_{p \in Q} p^{e(p)}, x/\prod$

Algorithm:

1. Replace Q with
   $\{p \in Q : x$ moc
2. Find maximal f
   $s = \prod(p^2)^{f(p^2)}$
3. Find $T = \{p \in$
4. Output $e, s\prod_p$
   where $e(p) = 2$

598
}

98

7

,
598
ver
3, 7

1) bits.

Exponents of a small prime

Time $\le b(\lg b)^{2+o(1)}$:
Given nonzero $p, x \in \mathbf{Z}$,
find $e, p^e, x/p^e$ with maximal e.

Algorithm:

1. If $x \bmod p \ne 0$:
   Print $0, 1, x$ and stop.
2. Find $f, (p^2)^f, r = (x/p)/(p^2)^f$
   with maximal f.
3. If $r \bmod p = 0$: Print
   $2f+2, (p^2)^f p^2, r/p$ and stop.
4. Print $2f+1, (p^2)^f p, r$.

Exponents of small primes

Time $\le b(\lg b)^{3+o(1)}$:
Given finite set Q of primes
and nonzero $x \in \mathbf{Z}$, find ma
$e, \prod_{p\in Q} p^{e(p)}, x/\prod_{p\in Q} p^{e(p}$

Algorithm:

1. Replace Q with
   $\{p \in Q : x \bmod p = 0\}$.
2. Find maximal f, s, r with
   $s = \prod (p^2)^{f(p^2)}$, $r = (x/$
3. Find $T = \{p \in Q : r \bmod$
4. Output $e, s \prod_{p \in T} p, r/$
   where $e(p) = 2f(p^2) + [$

Exponents of a small prime

Time $\le b(\lg b)^{2+o(1)}$:
Given nonzero $p, x \in \mathbf{Z}$,
find $e, p^e, x/p^e$ with maximal e.

Algorithm:

1. If $x \bmod p \ne 0$:
   Print $0, 1, x$ and stop.
2. Find $f, (p^2)^f, r = (x/p)/(p^2)^f$
   with maximal f.
3. If $r \bmod p = 0$: Print
   $2f + 2, (p^2)^f p^2, r/p$ and stop.
4. Print $2f + 1, (p^2)^f p, r$.

Exponents of small primes

Time $\le b(\lg b)^{3+o(1)}$:
Given finite set Q of primes
and nonzero $x \in \mathbf{Z}$, find maximal
$e, \prod_{p \in Q} p^{e(p)}, x/\prod_{p \in Q} p^{e(p)}$.

Algorithm:

1. Replace Q with
   $\{p \in Q : x \bmod p = 0\}$.
2. Find maximal f, s, r with
   $s = \prod (p^2)^{f(p^2)}$, $r = (x/\prod p)/s$.
3. Find $T = \{p \in Q : r \bmod p = 0\}$.
4. Output $e, s\prod_{p \in T} p, r/\prod_{p \in T} p$
   where $e(p) = 2f(p^2) + [p \in T]$.

nts of a small prime

$b(\lg b)^{2+o(1)}$:

onzero $p, x \in \mathbf{Z}$,

$^e, x/p^e$ with maximal e.

m:

mod $p \neq 0$:

$0, 1, x$ and stop.

$f, (p^2)^f, r = (x/p)/(p^2)^f$

maximal f.

nod $p = 0$: Print

$2, (p^2)^f p^2, r/p$ and stop.

$2f + 1, (p^2)^f p, r$.

Exponents of small primes

Time $\leq b(\lg b)^{3+o(1)}$:
Given finite set Q of primes
and nonzero $x \in \mathbf{Z}$, find maximal
$e, \prod_{p \in Q} p^{e(p)}, x / \prod_{p \in Q} p^{e(p)}$.

Algorithm:

1. Replace Q with $\{p \in Q : x \bmod p = 0\}$.
2. Find maximal f, s, r with $s = \prod (p^2)^{f(p^2)}$, $r = (x / \prod p)/s$.
3. Find $T = \{p \in Q : r \bmod p = 0\}$.
4. Output $e, s \prod_{p \in T} p, r / \prod_{p \in T} p$ where $e(p) = 2f(p^2) + [p \in T]$.

Smooth

Time $\leq$
Given no
and finit
compute
Q-smoot
Q-smoot

Q-smoot
of power

Q-smoot
largest Q
In partic
$x_1, x_2, .$

nall prime

(1):

c ∈ **Z**,

th maximal e.

:

d stop.

$= (x/p)/(p^2)^f$

f.

: Print

$^2, r/p$ and stop.

$p^2)^f p, r$.

Exponents of small primes

Time $\le b(\lg b)^{3+o(1)}$:
Given finite set Q of primes
and nonzero $x \in \mathbf{Z}$, find maximal
$e, \prod_{p \in Q} p^{e(p)}, x / \prod_{p \in Q} p^{e(p)}$.

Algorithm:

1. Replace Q with
   $\{p \in Q : x \bmod p = 0\}$.
2. Find maximal f, s, r with
   $s = \prod (p^2)^{f(p^2)}$, $r = (x / \prod p)/s$.
3. Find $T = \{p \in Q : r \bmod p = 0\}$.
4. Output $e, s \prod_{p \in T} p, r / \prod_{p \in T} p$
   where $e(p) = 2f(p^2) + [p \in T]$.

Smooth parts, old

Time $\le b(\lg b)^{3+o}$
Given nonzero x_1,
and finite set Q of
compute Q-smoot
Q-smooth part of
Q-smooth part of

Q-smooth means p
of powers of eleme

Q-smooth part me
largest Q-smooth
In particular, see w
$x_1, x_2, \ldots, x_n$ are

al e.

$(p^2)^f$

stop.

Exponents of small primes

Time $\le b(\lg b)^{3+o(1)}$:
Given finite set Q of primes and nonzero $x \in \mathbf{Z}$, find maximal $e, \prod_{p \in Q} p^{e(p)}, x / \prod_{p \in Q} p^{e(p)}$.

Algorithm:

1. Replace Q with $\{p \in Q : x \bmod p = 0\}$.
2. Find maximal f, s, r with $s = \prod (p^2)^{f(p^2)}$, $r = (x / \prod p)/s$.
3. Find $T = \{p \in Q : r \bmod p = 0\}$.
4. Output $e, s \prod_{p \in T} p, r / \prod_{p \in T} p$ where $e(p) = 2f(p^2) + [p \in T]$.

Smooth parts, old approach

Time $\le b(\lg b)^{3+o(1)}$:
Given nonzero $x_1, x_2, \ldots, x_{n}$
and finite set Q of primes,
compute Q-smooth part of x
Q-smooth part of x_2, ...,
Q-smooth part of x_n.

Q-smooth means product of powers of elements of Q.

Q-smooth part means largest Q-smooth divisor.
In particular, see which of $x_1, x_2, \ldots, x_n$ are smooth.

Exponents of small primes

Time $\le b(\lg b)^{3+o(1)}$:
Given finite set Q of primes and nonzero $x \in \mathbf{Z}$, find maximal $e, \prod_{p\in Q} p^{e(p)}, x/\prod_{p\in Q} p^{e(p)}$.

Algorithm:

1. Replace Q with $\{p \in Q : x \bmod p = 0\}$.
2. Find maximal f, s, r with $s = \prod (p^2)^{f(p^2)}$, $r = (x/\prod p)/s$.
3. Find $T = \{p \in Q : r \bmod p = 0\}$.
4. Output $e, s\prod_{p\in T} p, r/\prod_{p\in T} p$ where $e(p) = 2f(p^2) + [p \in T]$.

Smooth parts, old approach

Time $\le b(\lg b)^{3+o(1)}$:
Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes, compute Q-smooth part of x_1, Q-smooth part of x_2, ..., Q-smooth part of x_n.

Q-smooth means product of powers of elements of Q.

Q-smooth part means largest Q-smooth divisor.
In particular, see which of $x_1, x_2, \ldots, x_n$ are smooth.

nts of small primes

$b(\lg b)^{3+o(1)}$:

nite set Q of primes

zero $x \in \mathbf{Z}$, find maximal

$p^{e(p)}, x/\prod_{p \in Q} p^{e(p)}$.

m:

ace Q with

$Q : x \bmod p = 0\}$.

maximal f, s, r with

$\prod(p^2)^{f(p^2)}$, $r = (x/\prod p)/s$.

$T = \{p \in Q : r \bmod p = 0\}$.

ut $e, s\prod_{p \in T} p, r/\prod_{p \in T} p$

e $e(p) = 2f(p^2) + [p \in T]$.

Smooth parts, old approach

Time $\le b(\lg b)^{3+o(1)}$:
Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
and finite set Q of primes,
compute Q-smooth part of x_1,
Q-smooth part of x_2, ...,
Q-smooth part of x_n.

Q-smooth means product
of powers of elements of Q.

Q-smooth part means
largest Q-smooth divisor.
In particular, see which of
$x_1, x_2, \ldots, x_n$ are smooth.

Algorith

1. Find
   ..., $($
2. For e
   Find
   $s = \prod$
   Print

e.g. fact
over {2,
2543 ove
6766 ove
8967 ove
7598 ove

ll primes

$^{(1)}$:

of primes

Z, find maximal

$\prod_{p \in Q} p^{e(p)}$.

n

d $p = 0\}$.

f, s, r with

, $r = (x / \prod p)/s$.

Q : r mod $p = 0\}$.

$_{\in T}\, p, r / \prod_{p \in T} p$

$2f(p^2) + [p \in T]$.

Smooth parts, old approach

Time $\leq b(\lg b)^{3+o(1)}$:
Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
and finite set Q of primes,
compute Q-smooth part of x_1,
Q-smooth part of x_2, ...,
Q-smooth part of x_n.

Q-smooth means product
of powers of elements of Q.

Q-smooth part means
largest Q-smooth divisor.
In particular, see which of
$x_1, x_2, \ldots, x_n$ are smooth.

Algorithm:

1. Find $Q_1 = \{p :$
   ..., $Q_n = \{p :$
2. For each i sepa
   Find maximal e
   $s = \prod_{p \in Q_i} p^{e(p}$
   Print s.

e.g. factor 2543, 6
over $\{2, 3, 5, 7, 11,$
2543 over $\{\}$, smo
6766 over $\{2, 17\}$,
8967 over $\{3, 7\}$, s
7598 over $\{2\}$, sm

ximal

).

n

$\prod p)/s$.

$p = 0\}$.

$]_{p \in T}\, p$

$p \in T]$.

Smooth parts, old approach

Time $\le b(\lg b)^{3+o(1)}$:
Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
and finite set Q of primes,
compute Q-smooth part of x_1,
Q-smooth part of x_2, ...,
Q-smooth part of x_n.

Q-smooth means product
of powers of elements of Q.

Q-smooth part means
largest Q-smooth divisor.
In particular, see which of
$x_1, x_2, \ldots, x_n$ are smooth.

Algorithm:

1. Find $Q_1 = \{p : x_1 \bmod p$
   $\ldots,\ Q_n = \{p : x_n \bmod p$
2. For each i separately:
   Find maximal e, s, r with
   $s = \prod_{p \in Q_i} p^{e(p)}$, $r = x_{i/}$
   Print s.

e.g. factor 2543, 6766, 8967
over $\{2, 3, 5, 7, 11, 13, 17\}$:
2543 over $\{\}$, smooth part 1
6766 over $\{2, 17\}$, smooth p
8967 over $\{3, 7\}$, smooth pa
7598 over $\{2\}$, smooth part

Smooth parts, old approach

Time $\le b(\lg b)^{3+o(1)}$:

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes, compute Q-smooth part of x_1, Q-smooth part of x_2, ..., Q-smooth part of x_n.

Q-smooth means product of powers of elements of Q.

Q-smooth part means largest Q-smooth divisor. In particular, see which of $x_1, x_2, \ldots, x_n$ are smooth.

Algorithm:

1. Find $Q_1 = \{p : x_1 \bmod p = 0\}$, $\ldots$, $Q_n = \{p : x_n \bmod p = 0\}$.
2. For each i separately: Find maximal e, s, r with $s = \prod_{p \in Q_i} p^{e(p)}$, $r = x_i/s$. Print s.

e.g. factor $2543, 6766, 8967, 7598$ over $\{2, 3, 5, 7, 11, 13, 17\}$:
2543 over $\{\}$, smooth part 1;
6766 over $\{2, 17\}$, smooth part 34;
8967 over $\{3, 7\}$, smooth part 147;
7598 over $\{2\}$, smooth part 2.

parts, old approach

$b(\lg b)^{3+o(1)}$:

onzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$

e set Q of primes,

e Q-smooth part of x_1,

th part of x_2, ...,

th part of x_n.

th means product

rs of elements of Q.

th part means

Q-smooth divisor.

ular, see which of

$\ldots, x_n$ are smooth.

Algorithm:

1. Find $Q_1 = \{p : x_1 \bmod p = 0\}$, ..., $Q_n = \{p : x_n \bmod p = 0\}$.
2. For each i separately:
   Find maximal e, s, r with $s = \prod_{p \in Q_i} p^{e(p)}$, $r = x_i/s$.
   Print s.

e.g. factor 2543, 6766, 8967, 7598 over $\{2, 3, 5, 7, 11, 13, 17\}$:

2543 over $\{\}$, smooth part 1;

6766 over $\{2, 17\}$, smooth part 34;

8967 over $\{3, 7\}$, smooth part 147;

7598 over $\{2\}$, smooth part 2.

Smooth

Recall cr

find kth

product

$x_1, x_2, .$

Choose

Define Q

See whic

are y-sm

Know th

Do linea

on the e

approach

(1):

$x_2, \ldots, x_n \in \mathbf{Z}$

f primes,

h part of x_1,

x_2, ...,

x_n.

product

ents of Q.

eans

divisor.

which of

smooth.

Algorithm:

1. Find $Q_1 = \{p : x_1 \bmod p = 0\}$, ..., $Q_n = \{p : x_n \bmod p = 0\}$.
2. For each i separately:
   Find maximal e, s, r with
   $s = \prod_{p \in Q_i} p^{e(p)}$, $r = x_i/s$.
   Print s.

e.g. factor 2543, 6766, 8967, 7598 over $\{2, 3, 5, 7, 11, 13, 17\}$:
2543 over $\{\}$, smooth part 1;
6766 over $\{2, 17\}$, smooth part 34;
8967 over $\{3, 7\}$, smooth part 147;
7598 over $\{2\}$, smooth part 2.

Smooth multiplica

Recall cryptanalyti
find kth power no
product of powers
$x_1, x_2, \ldots, x_n$.

Choose y; imagine
Define Q as set of
See which of x_1, x
are y-smooth, i.e.,
Know their factori
Do linear algebra
on the exponent v

$_n \in \mathbf{Z}$

$x_1,$

Algorithm:

1. Find $Q_1 = \{p : x_1 \bmod p = 0\}$, $\ldots, Q_n = \{p : x_n \bmod p = 0\}$.
2. For each i separately: Find maximal e, s, r with $s = \prod_{p \in Q_i} p^{e(p)}$, $r = x_i/s$. Print s.

e.g. factor $2543, 6766, 8967, 7598$ over $\{2, 3, 5, 7, 11, 13, 17\}$:
2543 over $\{\}$, smooth part 1;
6766 over $\{2, 17\}$, smooth part 34;
8967 over $\{3, 7\}$, smooth part 147;
7598 over $\{2\}$, smooth part 2.

Smooth multiplicative deper

Recall cryptanalytic bottlene find kth power nontrivially a product of powers of $x_1, x_2, \ldots, x_n$.

Choose y; imagine $y = 2^{40}$.
Define Q as set of primes $\leq$
See which of $x_1, x_2, \ldots, x_n$ are y-smooth, i.e., Q-smoot
Know their factorizations.
Do linear algebra over $\mathbf{Z}/k$ on the exponent vectors.

Algorithm:

1. Find $Q_1 = \{p : x_1 \bmod p = 0\}$,
   $\ldots, Q_n = \{p : x_n \bmod p = 0\}$.
2. For each i separately:
   Find maximal e, s, r with
   $s = \prod_{p \in Q_i} p^{e(p)}$, $r = x_i/s$.
   Print s.

e.g. factor $2543, 6766, 8967, 7598$
over $\{2, 3, 5, 7, 11, 13, 17\}$:
2543 over $\{\}$, smooth part 1;
6766 over $\{2, 17\}$, smooth part 34;
8967 over $\{3, 7\}$, smooth part 147;
7598 over $\{2\}$, smooth part 2.

Smooth multiplicative dependencies

Recall cryptanalytic bottleneck:
find kth power nontrivially as
product of powers of
$x_1, x_2, \ldots, x_n$.

Choose y; imagine $y = 2^{40}$.
Define Q as set of primes $\le y$.
See which of $x_1, x_2, \ldots, x_n$
are y-smooth, i.e., Q-smooth.
Know their factorizations.
Do linear algebra over $\mathbf{Z}/k$
on the exponent vectors.

m:

$Q_1 = \{p : x_1 \bmod p = 0\}$,

$Q_n = \{p : x_n \bmod p = 0\}$.

ach i separately:

maximal e, s, r with

$\prod_{p \in Q_i} p^{e(p)}$, $r = x_i/s$.

s.

tor $2543, 6766, 8967, 7598$

$3, 5, 7, 11, 13, 17\}$:

er $\{\}$, smooth part 1;

er $\{2, 17\}$, smooth part 34;

er $\{3, 7\}$, smooth part 147;

er $\{2\}$, smooth part 2.

Smooth multiplicative dependencies

Recall cryptanalytic bottleneck: find kth power nontrivially as product of powers of $x_1, x_2, \ldots, x_n$.

Choose y; imagine $y = 2^{40}$. Define Q as set of primes $\le y$. See which of $x_1, x_2, \ldots, x_n$ are y-smooth, i.e., Q-smooth. Know their factorizations. Do linear algebra over $\mathbf{Z}/k$ on the exponent vectors.

Smooth

Given no

and finit

Time typ

to obtai

(2004 Fr

Morain V

Algorith

Compute

Compute

For each

Replace

$x_i/\gcd\{$

repeated

$x_1 \bmod p = 0\}$,
$x_n \bmod p = 0\}$.
arately:
e, s, r with
$p)$, $r = x_i/s$.

5766, 8967, 7598
13, 17\}:
ooth part 1;
smooth part 34;
smooth part 147;
nooth part 2.

Smooth multiplicative dependencies

Recall cryptanalytic bottleneck: find kth power nontrivially as product of powers of $x_1, x_2, \ldots, x_n$.

Choose y; imagine $y = 2^{40}$. Define Q as set of primes $\le y$. See which of $x_1, x_2, \ldots, x_n$ are y-smooth, i.e., Q-smooth. Know their factorizations. Do linear algebra over $\mathbf{Z}/k$ on the exponent vectors.

Smooth parts, new

Given nonzero x_1,
and finite set Q of
Time typically $\le b$
to obtain smooth
(2004 Franke Klei
Morain Wirth, in E

Algorithm:
Compute $r = \prod_{p\in}$
Compute $r \bmod x$
For each i separat
Replace x_i by
$x_i/\gcd\{x_i, r \bmod$
repeatedly until gc

$= 0\}$,
$= 0\}$.

$/s$.

, 7598

1;
part 34;
rt 147;
2.

Smooth multiplicative dependencies

Recall cryptanalytic bottleneck:
find kth power nontrivially as
product of powers of
$x_1, x_2, \ldots, x_n$.

Choose y; imagine $y = 2^{40}$.
Define Q as set of primes $\le y$.
See which of $x_1, x_2, \ldots, x_n$
are y-smooth, i.e., Q-smooth.
Know their factorizations.
Do linear algebra over $\mathbf{Z}/k$
on the exponent vectors.

Smooth parts, new approach

Given nonzero $x_1, x_2, \ldots, x$
and finite set Q of primes:
Time typically $\le b(\lg b)^{2+o(}$
to obtain smooth parts of x
(2004 Franke Kleinjung
Morain Wirth, in ECPP con

Algorithm:
Compute $r = \prod_{p \in Q} p$.
Compute $r \bmod x_1, \ldots, r$ m
For each i separately:
Replace x_i by
$x_i / \gcd\{x_i, r \bmod x_i\}$
repeatedly until gcd is 1.

Smooth multiplicative dependencies

Recall cryptanalytic bottleneck:
find kth power nontrivially as product of powers of $x_1, x_2, \ldots, x_n$.

Choose y; imagine $y = 2^{40}$.
Define Q as set of primes $\leq y$.
See which of $x_1, x_2, \ldots, x_n$ are y-smooth, i.e., Q-smooth.
Know their factorizations.
Do linear algebra over $\mathbf{Z}/k$ on the exponent vectors.

Smooth parts, new approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes:
Time typically $\leq b(\lg b)^{2+o(1)}$ to obtain smooth parts of x's.
(2004 Franke Kleinjung Morain Wirth, in ECPP context)

Algorithm:
Compute $r = \prod_{p \in Q} p$.
Compute $r \bmod x_1, \ldots, r \bmod x_n$.
For each i separately:
Replace x_i by $x_i / \gcd\{x_i, r \bmod x_i\}$ repeatedly until gcd is 1.

multiplicative dependencies

ryptanalytic bottleneck:
power nontrivially as
of powers of
$\ldots, x_n$.

y; imagine $y = 2^{40}$.
Q as set of primes $\le y$.
ch of $x_1, x_2, \ldots, x_n$
nooth, i.e., Q-smooth.
heir factorizations.
r algebra over $\mathbf{Z}/k$
xponent vectors.

Smooth parts, new approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
and finite set Q of primes:
Time typically $\le b(\lg b)^{2+o(1)}$
to obtain smooth parts of x's.
(2004 Franke Kleinjung
Morain Wirth, in ECPP context)

Algorithm:
Compute $r = \prod_{p \in Q} p$.
Compute $r \bmod x_1, \ldots, r \bmod x_n$.
For each i separately:
Replace x_i by
$x_i/\gcd\{x_i, r \bmod x_i\}$
repeatedly until gcd is 1.

Slight va
Time alv

Compute
$\gcd\{x_i,$
where k

Subrouti
takes tin
(1971 Sc
core idea
$b(\lg b)^{5+}$

Or, to se
see if $(r$

tive dependencies

ic bottleneck:
ntrivially as
of

e $y = 2^{40}$.
f primes $\le y$.
$_2, \ldots, x_n$
Q-smooth.
zations.
over $\mathbf{Z}/k$
ectors.

Smooth parts, new approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
and finite set Q of primes:
Time typically $\le b(\lg b)^{2+o(1)}$
to obtain smooth parts of x's.
(2004 Franke Kleinjung
Morain Wirth, in ECPP context)

Algorithm:
Compute $r = \prod_{p \in Q} p$.
Compute $r \bmod x_1, \ldots, r \bmod x_n$.
For each i separately:
Replace x_i by
$x_i / \gcd\{x_i, r \bmod x_i\}$
repeatedly until gcd is 1.

Slight variant (200
Time always $\le b$(l

Compute smooth
gcd$\{x_i, (r \bmod x_i$
where $k = \lceil \lg \lg x$

Subroutine: Comp
takes time $\le b(\lg$
(1971 Schönhage;
core idea: 1938 Le
$b(\lg b)^{5+o(1)}$: 1971

Or, to see if x_i is
see if $(r \bmod x_i)^{2^{\cdot}}$

ndencies

eck:

ns

y.

h.

Smooth parts, new approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
and finite set Q of primes:
Time typically $\le b(\lg b)^{2+o(1)}$
to obtain smooth parts of x's.
(2004 Franke Kleinjung
Morain Wirth, in ECPP context)

Algorithm:
Compute $r = \prod_{p \in Q} p$.
Compute $r \bmod x_1, \ldots, r \bmod x_n$.
For each i separately:
Replace x_i by
$x_i / \gcd\{x_i, r \bmod x_i\}$
repeatedly until gcd is 1.

Slight variant (2004 Bernste
Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i
$\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x$
where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd
takes time $\le b(\lg b)^{2+o(1)}$.
(1971 Schönhage;
core idea: 1938 Lehmer;
$b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth,
see if $(r \bmod x_i)^{2^k} \bmod x_i =$

Smooth parts, new approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$ and finite set Q of primes: Time typically $\le b(\lg b)^{2+o(1)}$ to obtain smooth parts of x's. (2004 Franke Kleinjung Morain Wirth, in ECPP context)

Algorithm:
Compute $r = \prod_{p \in Q} p$.
Compute $r \bmod x_1, \ldots, r \bmod x_n$.
For each i separately:
Replace x_i by $x_i/\gcd\{x_i, r \bmod x_i\}$ repeatedly until gcd is 1.

Slight variant (2004 Bernstein): Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i as $\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x_i\}$ where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd takes time $\le b(\lg b)^{2+o(1)}$. (1971 Schönhage; core idea: 1938 Lehmer; $b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth, see if $(r \bmod x_i)^{2^k} \bmod x_i = 0$.

parts, new approach

onzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$
te set Q of primes:
pically $\le b(\lg b)^{2+o(1)}$
n smooth parts of x's.
ranke Kleinjung
Wirth, in ECPP context)

m:
e $r = \prod_{p \in Q} p$.
e $r \bmod x_1, \ldots, r \bmod x_n$.
n i separately:
x_i by
$x_i, r \bmod x_i\}$
lly until gcd is 1.

Slight variant (2004 Bernstein):
Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i as
$\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x_i\}$
where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd
takes time $\le b(\lg b)^{2+o(1)}$.
(1971 Schönhage;
core idea: 1938 Lehmer;
$b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth,
see if $(r \bmod x_i)^{2^k} \bmod x_i = 0$.

Minor pr
finds the
but does

v approach

$x_2, \ldots, x_n \in \mathbf{Z}$

f primes:

$b(\lg b)^{2+o(1)}$

parts of x's.

njung

ECPP context)

$_{\in Q}\, p$.

$_1, \ldots, r \bmod x_n$.

ely:

$x_i\}$

cd is 1.

Slight variant (2004 Bernstein):
Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i as
$\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x_i\}$
where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd
takes time $\le b(\lg b)^{2+o(1)}$.
(1971 Schönhage;
core idea: 1938 Lehmer;
$b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth,
see if $(r \bmod x_i)^{2^k} \bmod x_i = 0$.

Minor problem: N

finds the smooth r

but doesn't factor

n

$_n \in \mathbf{Z}$

1)

's.

text)

od x_n.

Slight variant (2004 Bernstein):
Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i as
$\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x_i\}$
where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd
takes time $\le b(\lg b)^{2+o(1)}$.
(1971 Schönhage;
core idea: 1938 Lehmer;
$b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth,
see if $(r \bmod x_i)^{2^k} \bmod x_i = 0$.

Minor problem: New algorit
finds the smooth numbers
but doesn't factor them.

Slight variant (2004 Bernstein):
Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i as
$\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x_i\}$
where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd
takes time $\le b(\lg b)^{2+o(1)}$.
(1971 Schönhage;
core idea: 1938 Lehmer;
$b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth,
see if $(r \bmod x_i)^{2^k} \bmod x_i = 0$.

Minor problem: New algorithm
finds the smooth numbers
but doesn't factor them.

Slight variant (2004 Bernstein):
Time always $\le b(\lg b)^{2+o(1)}$.

Compute smooth part of x_i as
$\gcd\{x_i, (r \bmod x_i)^{2^k} \bmod x_i\}$
where $k = \lceil \lg \lg x_i \rceil$.

Subroutine: Computing gcd
takes time $\le b(\lg b)^{2+o(1)}$.
(1971 Schönhage;
core idea: 1938 Lehmer;
$b(\lg b)^{5+o(1)}$: 1971 Knuth)

Or, to see if x_i is smooth,
see if $(r \bmod x_i)^{2^k} \bmod x_i = 0$.

Minor problem: New algorithm
finds the smooth numbers
but doesn't factor them.

Solution:
Feed the smooth numbers
to the old algorithm.
Very few smooth numbers,
so this is very fast.

Bottom line for cryptanalysis:
time per input number to
find and factor smooth numbers
has dropped by $(\lg b)^{1+o(1)}$.

ariant (2004 Bernstein):
ways $\leq b(\lg b)^{2+o(1)}$.

e smooth part of x_i as
$(r \bmod x_i)^{2^k} \bmod x_i\}$
$= \lceil \lg \lg x_i \rceil$.

ine: Computing gcd
ne $\leq b(\lg b)^{2+o(1)}$.
chönhage;
a: 1938 Lehmer;
$\cdot o(1)$: 1971 Knuth)

ee if x_i is smooth,
$\bmod x_i)^{2^k} \bmod x_i = 0$.

Minor problem: New algorithm
finds the smooth numbers
but doesn't factor them.

Solution:
Feed the smooth numbers
to the old algorithm.
Very few smooth numbers,
so this is very fast.

Bottom line for cryptanalysis:
time per input number to
find and factor smooth numbers
has dropped by $(\lg b)^{1+o(1)}$.

Is smoot

After fin
do first s
Throw a
only onc
numbers
repeat u

Don't w
Want sn
they are
divide th

04 Bernstein):
g $b)^{2+o(1)}$.

part of x_i as
$)^{2^k}$ mod $x_i\}$
$i\rceil$.

puting gcd
$b)^{2+o(1)}$.

ehmer;
1 Knuth)

smooth,
^{k} mod $x_i = 0$.

Minor problem: New algorithm
finds the smooth numbers
but doesn't factor them.

Solution:
Feed the smooth numbers
to the old algorithm.
Very few smooth numbers,
so this is very fast.

Bottom line for cryptanalysis:
time per input number to
find and factor smooth numbers
has dropped by $(\lg b)^{1+o(1)}$.

Is smooth the righ

After finding smoo
do first step of lin
Throw away prime
only once; throw a
numbers with thos
repeat until stable

Don't want *all* sm
Want smooth num
they are built from
divide the *other* n

ein):

.

as

$_i\}$

$= 0$.

Minor problem: New algorithm
finds the smooth numbers
but doesn't factor them.

Solution:
Feed the smooth numbers
to the old algorithm.
Very few smooth numbers,
so this is very fast.

Bottom line for cryptanalysis:
time per input number to
find and factor smooth numbers
has dropped by $(\lg b)^{1+o(1)}$.

Is smooth the right question

After finding smooth numbe
do first step of linear algebra
Throw away primes that app
only once; throw away
numbers with those primes;
repeat until stable.

Don't want *all* smooth num
Want smooth numbers only
they are built from primes th
divide the *other* numbers.

Minor problem: New algorithm finds the smooth numbers but doesn't factor them.

Solution:
Feed the smooth numbers to the old algorithm.
Very few smooth numbers, so this is very fast.

Bottom line for cryptanalysis: time per input number to find and factor smooth numbers has dropped by $(\lg b)^{1+o(1)}$.

Is smooth the right question?

After finding smooth numbers, do first step of linear algebra: Throw away primes that appear only once; throw away numbers with those primes; repeat until stable.

Don't want *all* smooth numbers. Want smooth numbers only if they are built from primes that divide the *other* numbers.

roblem: New algorithm
e smooth numbers
sn't factor them.

:
e smooth numbers
ld algorithm.
v smooth numbers,
s very fast.

line for cryptanalysis:
r input number to
factor smooth numbers
oped by $(\lg b)^{1+o(1)}$.

Is smooth the right question?

After finding smooth numbers, do first step of linear algebra: Throw away primes that appear only once; throw away numbers with those primes; repeat until stable.

Don't want *all* smooth numbers. Want smooth numbers only if they are built from primes that divide the *other* numbers.

An alter

Given no
Comput
Comput
(r/x_n) r
For each
$((r/x_i)$
where k

Finds x_i
are divis
Time $\leq$

(2004 B

ew algorithm
numbers
them.

numbers
m.
numbers,
.

yptanalysis:
mber to
ooth numbers
g $b)^{1+o(1)}$.

Is smooth the right question?

After finding smooth numbers, do first step of linear algebra: Throw away primes that appear only once; throw away numbers with those primes; repeat until stable.

Don't want *all* smooth numbers. Want smooth numbers only if they are built from primes that divide the *other* numbers.

An alternate appro

Given nonzero x_1,
Compute $r = x_1x$
Compute (r/x_1) m
(r/x_n) mod x_n.
For each i separat
$((r/x_i) \bmod x_i)^{2^k}$
where $k = \lceil \lg \lg x$

Finds x_i iff all prin
are divisors of othe
Time $\le b(\lg b)^{2+o}$

(2004 Bernstein)

hm

s:

bers

Is smooth the right question?

After finding smooth numbers, do first step of linear algebra: Throw away primes that appear only once; throw away numbers with those primes; repeat until stable.

Don't want *all* smooth numbers. Want smooth numbers only if they are built from primes that divide the *other* numbers.

An alternate approach

Given nonzero $x_1, x_2, \ldots, x$
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1$, . .
$(r/x_n) \bmod x_n$.
For each i separately: see if
$((r/x_i) \bmod x_i)^{2^k} \bmod x_i =$
where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i are divisors of other x's.
Time $\le b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Is smooth the right question?

After finding smooth numbers, do first step of linear algebra: Throw away primes that appear only once; throw away numbers with those primes; repeat until stable.

Don't want *all* smooth numbers. Want smooth numbers only if they are built from primes that divide the *other* numbers.

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1, \ldots,$ $(r/x_n) \bmod x_n$.
For each i separately: see if $((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$ where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i are divisors of other x's.
Time $\le b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

th the right question?

ding smooth numbers,
step of linear algebra:
way primes that appear
ce; throw away
s with those primes;
ntil stable.

ant *all* smooth numbers.
nooth numbers only if
built from primes that
ne *other* numbers.

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1$, $\ldots$, $(r/x_n) \bmod x_n$.
For each i separately: see if $((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$ where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i are divisors of other x's.
Time $\leq b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Compute
(r/x_n) r
r mod x
(1972 M

t question?

oth numbers,
ear algebra:
es that appear
away
se primes;
.

ooth numbers.
nbers only if
n primes that
umbers.

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute (r/x_1) mod x_1, . . . ,
(r/x_n) mod x_n.
For each i separately: see if
$((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$
where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i
are divisors of other x's.
Time $\le b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Compute (r/x_1) m
(r/x_n) mod x_n by
r mod $x_1^2, \ldots, r$ m
(1972 Moenck Bo

n?

ers,

a:

pear

bers.

if

hat

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$.
For each i separately: see if
$((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$
where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i
are divisors of other x's.
Time $\leq b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Compute $(r/x_1) \bmod x_1$, . .
$(r/x_n) \bmod x_n$ by computin
$r \bmod x_1^2, \ldots, r \bmod x_n^2$.
(1972 Moenck Borodin)

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$.
For each i separately: see if
$((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$
where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i
are divisors of other x's.
Time $\le b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$ by computing
$r \bmod x_1^2, \ldots, r \bmod x_n^2$.
(1972 Moenck Borodin)

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$.
For each i separately: see if
$((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$
where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i
are divisors of other x's.
Time $\le b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$ by computing
$r \bmod x_1^2, \ldots, r \bmod x_n^2$.
(1972 Moenck Borodin)

Problem: Recognizing the
interesting x's is not enough;
also need their factorizations.

An alternate approach

Given nonzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
Compute $r = x_1 x_2 \cdots x_n$.
Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$.
For each i separately: see if
$((r/x_i) \bmod x_i)^{2^k} \bmod x_i = 0$
where $k = \lceil \lg \lg x_i \rceil$.

Finds x_i iff all primes in x_i
are divisors of other x's.
Time $\le b(\lg b)^{2+o(1)}$.

(2004 Bernstein)

Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$ by computing
$r \bmod x_1^2, \ldots, r \bmod x_n^2$.
(1972 Moenck Borodin)

Problem: Recognizing the
interesting x's is not enough;
also need their factorizations.

Solution:
Again, very few of them.
Have ample time to
use rho method (1974 Pollard)
or use ECM (1987 Lenstra)
or factor into coprimes.

nate approach

nzero $x_1, x_2, \ldots, x_n \in \mathbf{Z}$:
e $r = x_1 x_2 \cdots x_n$.
e $(r/x_1) \bmod x_1$, ...,
mod x_n.
i separately: see if
mod $x_i)^{2^k} \bmod x_i = 0$
$= \lceil \lg \lg x_i \rceil$.

iff all primes in x_i
ors of other x's.
$b(\lg b)^{2+o(1)}$.

ernstein)

Compute $(r/x_1) \bmod x_1$, ...,
$(r/x_n) \bmod x_n$ by computing
$r \bmod x_1^2, \ldots, r \bmod x_n^2$.
(1972 Moenck Borodin)

Problem: Recognizing the
interesting x's is not enough;
also need their factorizations.

Solution:
Again, very few of them.
Have ample time to
use rho method (1974 Pollard)
or use ECM (1987 Lenstra)
or factor into coprimes.

Factorin

Time $\leq$
Given po
find copr
and com
of each

(announ
journal v

Immedia
for the c
Subsequ
constant

oach

$x_2, \ldots, x_n \in$ **Z**:

$_2 \cdots x_n$.

mod x_1, . . . ,

ely: see if

mod $x_i = 0$

$_i]$.

mes in x_i

er x's.

$^{(1)}$.

Compute (r/x_1) mod x_1, . . . ,
(r/x_n) mod x_n by computing
r mod $x_1^2, \ldots, r$ mod x_n^2.
(1972 Moenck Borodin)

Problem: Recognizing the interesting x's is not enough; also need their factorizations.

Solution:
Again, very few of them.
Have ample time to
use rho method (1974 Pollard)
or use ECM (1987 Lenstra)
or factor into coprimes.

Factoring into cop

Time $\le b(\lg b)^{O(1)}$

Given positive x_1,

find coprime set Q

and complete fact

of each x_i over Q.

(announced 1995

journal version: 20

Immediately gives

for the other facto

Subsequent resear

constant-factor sp

$_n \in \mathbf{Z}$:

. . ,

= 0

Compute $(r/x_1) \bmod x_1, \ldots,$ $(r/x_n) \bmod x_n$ by computing $r \bmod x_1^2, \ldots, r \bmod x_n^2$. (1972 Moenck Borodin)

Problem: Recognizing the interesting x's is not enough; also need their factorizations.

Solution:
Again, very few of them.
Have ample time to
use rho method (1974 Pollard)
or use ECM (1987 Lenstra)
or factor into coprimes.

Factoring into coprimes

Time $\le b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring probl
Subsequent research: lg spe
constant-factor speedups, et

Compute $(r/x_1) \bmod x_1, \ldots,$
$(r/x_n) \bmod x_n$ by computing
$r \bmod x_1^2, \ldots, r \bmod x_n^2$.
(1972 Moenck Borodin)

Problem: Recognizing the interesting x's is not enough; also need their factorizations.

Solution:
Again, very few of them.
Have ample time to
use rho method (1974 Pollard)
or use ECM (1987 Lenstra)
or factor into coprimes.

Factoring into coprimes

Time $\le b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$,
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring problems.
Subsequent research: lg speedups,
constant-factor speedups, etc.

e (r/x_1) mod x_1, . . . ,
mod x_n by computing
$_1^2, \ldots, r$ mod x_n^2.
loenck Borodin)

: Recognizing the
ng x's is not enough;
d their factorizations.

:
very few of them.
nple time to
method (1974 Pollard)
CM (1987 Lenstra)
r into coprimes.

Factoring into coprimes

Time $\le b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$,
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring problems.
Subsequent research: lg speedups,
constant-factor speedups, etc.

Typical a
detectin

Does 91
equal 15

Each sid
$\approx$ 1946

mod x_1, . . . ,
y computing
nod x_n^2.
rodin)

zing the
not enough;
ctorizations.

f them.
to
1974 Pollard)
7 Lenstra)
times.

Factoring into coprimes

Time $\leq b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$,
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring problems.
Subsequent research: lg speedups,
constant-factor speedups, etc.

Typical application
detecting multiplic

Does $91^{1952681}11$
equal $1547^{1708632}$

Each side has loga
$\approx$ 19466590.6748

. . ,

ng

n;

s.

rd)

Factoring into coprimes

Time $\le b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$,
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring problems.
Subsequent research: lg speedups,
constant-factor speedups, etc.

Typical application:
detecting multiplicative relat

Does $91^{1952681} 119^{1513335} 22$
equal $1547^{1708632} 6898073^{43}$

Each side has logarithm
≈ 19466590.674872.

Factoring into coprimes

Time $\le b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$,
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring problems.
Subsequent research: lg speedups,
constant-factor speedups, etc.

Typical application:
detecting multiplicative relations.

Does $91^{1952681} 119^{1513335} 221^{634643}$
equal $1547^{1708632} 6898073^{439346}$?

Each side has logarithm
≈ 19466590.674872.

Factoring into coprimes

Time $\le b(\lg b)^{O(1)}$:
Given positive $x_1, x_2, \ldots, x_n$,
find coprime set Q
and complete factorization
of each x_i over Q.

(announced 1995 Bernstein;
journal version: 2005)

Immediately gives $b(\lg b)^{O(1)}$
for the other factoring problems.
Subsequent research: lg speedups,
constant-factor speedups, etc.

Typical application:
detecting multiplicative relations.

Does $91^{1952681} 119^{1513335} 221^{634643}$
equal $1547^{1708632} 6898073^{439346}$?

Each side has logarithm
≈ 19466590.674872.

More generally:
What is kernel of $(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e}$?

Kernel lets us find relations,
not just verify relations.

g into coprimes

$b(\lg b)^{O(1)}$:

ositive $x_1, x_2, \ldots, x_n$,

rime set Q

nplete factorization

x_i over Q.

ced 1995 Bernstein;

version: 2005)

ntely gives $b(\lg b)^{O(1)}$

other factoring problems.

ent research: lg speedups,

t-factor speedups, etc.

Typical application:
detecting multiplicative relations.

Does $91^{1952681} 119^{1513335} 221^{634643}$
equal $1547^{1708632} 6898073^{439346}$?

Each side has logarithm
≈ 19466590.674872.

More generally:
What is kernel of $(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e}$?

Kernel lets us find relations,
not just verify relations.

Factor ir

$91 = 7 \cdot$

$221 = 1$

6898073

$(a, b, c, c$

$91^a 119^b$

7^{a+b-d-}

Kernel is

$(1, 1, 1, 2$

rimes

):

$x_2, \ldots, x_n$,

)

orization

.

Bernstein;

005)

$b(\lg b)^{O(1)}$

ring problems.

ch: lg speedups,

eedups, etc.

Typical application:
detecting multiplicative relations.

Does $91^{1952681}119^{1513335}221^{634643}$
equal $1547^{1708632}6898073^{439346}$?

Each side has logarithm
≈ 19466590.674872.

More generally:
What is kernel of $(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e}$?

Kernel lets us find relations,
not just verify relations.

Factor into coprim

$91 = 7 \cdot 13$; $119 =$

$221 = 13 \cdot 17$; 154

$6898073 = 7^4 \cdot 13$

$(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-}$
$7^{a+b-d-4e} 13^{a+c-}$

Kernel is generate
$(1, 1, 1, 2, 0)$ and $($

n,

)

ems.

edups,

tc.

Typical application:
detecting multiplicative relations.

Does $91^{1952681} 119^{1513335} 221^{634643}$ equal $1547^{1708632} 6898073^{439346}$?

Each side has logarithm ≈ 19466590.674872.

More generally:
What is kernel of $(a, b, c, d, e) \mapsto 91^a 119^b 221^c 1547^{-d} 6898073^{-e}$?

Kernel lets us find relations, not just verify relations.

Factor into coprimes:
$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;
$221 = 13 \cdot 17$; $1547 = 7 \cdot 13$
$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 689807$
$7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c}$

Kernel is generated by
$(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1$

Typical application:
detecting multiplicative relations.

Does $91^{1952681}119^{1513335}221^{634643}$ equal $1547^{1708632}6898073^{439346}$?

Each side has logarithm ≈ 19466590.674872.

More generally:
What is kernel of $(a, b, c, d, e) \mapsto 91^a 119^b 221^c 1547^{-d} 6898073^{-e}$?

Kernel lets us find relations,
not just verify relations.

Factor into coprimes:
$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;
$221 = 13 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;
$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e} =$
$7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

Kernel is generated by
$(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1)$.

Typical application:
detecting multiplicative relations.

Does $91^{1952681}119^{1513335}221^{634643}$ equal $1547^{1708632}6898073^{439346}$?

Each side has logarithm ≈ 19466590.674872.

More generally:
What is kernel of $(a, b, c, d, e) \mapsto 91^a 119^b 221^c 1547^{-d} 6898073^{-e}$?

Kernel lets us find relations, not just verify relations.

Factor into coprimes:
$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;
$221 = 13 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;
$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e} = 7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

Kernel is generated by $(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1)$.

Factoring into coprimes remains fast for larger numbers.
Factoring into primes does not.

application:
g multiplicative relations.

$^{1952681}119^{1513335}221^{634643}$

$47^{1708632}6898073^{439346}$?

e has logarithm
590.674872.

nerally:
kernel of $(a, b, c, d, e) \mapsto$
$221^c 1547^{-d} 6898073^{-e}$?

ets us find relations,
verify relations.

Factor into coprimes:

$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;
$221 = 13 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;
$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e} =$
$7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

Kernel is generated by
$(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1)$.

Factoring into coprimes
remains fast for larger numbers.
Factoring into primes does not.

Can app
in more
replace i

Typical a
Take a s
What ar

One ans
for $\{h \in$
as a vect
Factor g
This list
all irredu

(1993 N

n:

cative relations.

$9^{1513335} 221^{634643}$

6898073^{439346}?

rithm

72.

$(a, b, c, d, e) \mapsto$

$^{-d} 6898073^{-e}$?

relations,

tions.

Factor into coprimes:

$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;

$221 = 13 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;

$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$

$91^a 119^b 221^c 1547^{-d} 6898073^{-e} =$

$7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

Kernel is generated by

$(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1)$.

Factoring into coprimes

remains fast for larger numbers.

Factoring into primes does not.

Can apply same al

in more generality

replace integers wi

Typical applicatior

Take a squarefree

What are g's irred

One answer: Find

for $\{h \in (\mathbf{Z}/2)[x]$

as a vector space

Factor $g, h_1, h_2, \ldots$

This list of coprim

all irreducible divis

(1993 Niederreiter

tions.

1^{634643}

^{9346}?

$e) \mapsto$

3^{-e}?

Factor into coprimes:

$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;

$221 = 13 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;

$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$

$91^a 119^b 221^c 1547^{-d} 6898073^{-e} =$

$7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

Kernel is generated by

$(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1)$.

Factoring into coprimes

remains fast for larger numbers.

Factoring into primes does not.

Can apply same algorithms

in more generality: e.g.,

replace integers with polyno

Typical application:

Take a squarefree $g \in (\mathbf{Z}/2)$

What are g's irreducible divi

One answer: Find basis $h_1,$

for $\{h \in (\mathbf{Z}/2)[x] : (gh)' =$

as a vector space over $\mathbf{Z}/2$.

Factor $g, h_1, h_2, \ldots$ into cop

This list of coprimes contain

all irreducible divisors of g.

(1993 Niederreiter, 1994 Gö

Factor into coprimes:

$91 = 7 \cdot 13$; $119 = 7 \cdot 17$;
$221 = 13 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;
$6898073 = 7^4 \cdot 13^2 \cdot 17$.

$(a, b, c, d, e) \mapsto$
$91^a 119^b 221^c 1547^{-d} 6898073^{-e} =$
$7^{a+b-d-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

Kernel is generated by
$(1, 1, 1, 2, 0)$ and $(3, 2, 0, 1, 1)$.

Factoring into coprimes
remains fast for larger numbers.
Factoring into primes does not.

Can apply same algorithms
in more generality: e.g.,
replace integers with polynomials.

Typical application:
Take a squarefree $g \in (\mathbf{Z}/2)[x]$.
What are g's irreducible divisors?

One answer: Find basis $h_1, h_2, \ldots$
for $\{h \in (\mathbf{Z}/2)[x] : (gh)' = h^2\}$
as a vector space over $\mathbf{Z}/2$.
Factor $g, h_1, h_2, \ldots$ into coprimes.
This list of coprimes contains
all irreducible divisors of g.

(1993 Niederreiter, 1994 Göttfert)

nto coprimes:

13; $119 = 7 \cdot 17$;

$3 \cdot 17$; $1547 = 7 \cdot 13 \cdot 17$;

$= 7^4 \cdot 13^2 \cdot 17$.

$d, e) \mapsto$

$221^c 1547^{-d} 6898073^{-e} =$

$^{-4e} 13^{a+c-d-2e} 17^{b+c-d-e}$.

s generated by

$2, 0)$ and $(3, 2, 0, 1, 1)$.

g into coprimes

fast for larger numbers.

g into primes does not.

Can apply same algorithms
in more generality: e.g.,
replace integers with polynomials.

Typical application:
Take a squarefree $g \in (\mathbf{Z}/2)[x]$.
What are g's irreducible divisors?

One answer: Find basis $h_1, h_2, \ldots$
for $\{h \in (\mathbf{Z}/2)[x] : (gh)' = h^2\}$
as a vector space over $\mathbf{Z}/2$.
Factor $g, h_1, h_2, \ldots$ into coprimes.
This list of coprimes contains
all irreducible divisors of g.

(1993 Niederreiter, 1994 Göttfert)

More ex

of facto

1890 Sti

1985 Ka

Dora Di

Bach Mi

zur Gath

1989 Po

Teitelba

Bach Dr

1994 Bu

Bernstei

Cohen D

Storjoha

cr.yp.t

mes:

$= 7 \cdot 17$;

$7 = 7 \cdot 13 \cdot 17$;

$^2 \cdot 17$.

$^{-d}6898073^{-e} =$

$^{d-2e}17^{b+c-d-e}$.

d by

$3, 2, 0, 1, 1)$.

rimes

rger numbers.

mes does not.

Can apply same algorithms in more generality: e.g., replace integers with polynomials.

Typical application:
Take a squarefree $g \in (\mathbf{Z}/2)[x]$.
What are g's irreducible divisors?

One answer: Find basis $h_1, h_2, \ldots$ for $\{h \in (\mathbf{Z}/2)[x] : (gh)' = h^2\}$ as a vector space over $\mathbf{Z}/2$.
Factor $g, h_1, h_2, \ldots$ into coprimes.
This list of coprimes contains all irreducible divisors of g.

(1993 Niederreiter, 1994 Göttfert)

More examples, ap
of factoring into c
1890 Stieltjes; 197
1985 Kaltofen; 19
Dora DiCrescenzo
Bach Miller Shallit
zur Gathen; 1986
1989 Pohst Zasse
Teitelbaum; 1990
Bach Driscoll Sha
1994 Buchmann L
Bernstein; 1997 Si
Cohen Diaz y Dia
Storjohann; ...
cr.yp.to/copri

$\cdot 17$;

$^{-e} =$

$^{-d-e}$.

).

ers.

ot.

Can apply same algorithms
in more generality: e.g.,
replace integers with polynomials.

Typical application:
Take a squarefree $g \in (\mathbf{Z}/2)[x]$.
What are g's irreducible divisors?

One answer: Find basis $h_1, h_2, \ldots$
for $\{h \in (\mathbf{Z}/2)[x] : (gh)' = h^2\}$
as a vector space over $\mathbf{Z}/2$.
Factor $g, h_1, h_2, \ldots$ into coprimes.
This list of coprimes contains
all irreducible divisors of g.

(1993 Niederreiter, 1994 Göttfert)

More examples, applications
of factoring into coprimes: s
1890 Stieltjes; 1974 Collins;
1985 Kaltofen; 1985 Della
Dora DiCrescenzo Duval; 19
Bach Miller Shallit; 1986 vo
zur Gathen; 1986 Lüneburg;
1989 Pohst Zassenhaus; 199
Teitelbaum; 1990 Smedley;
Bach Driscoll Shallit; 1994 C
1994 Buchmann Lenstra; 19
Bernstein; 1997 Silverman; 1
Cohen Diaz y Diaz Olivier; 1
Storjohann; . . .
cr.yp.to/coprimes.html

Can apply same algorithms in more generality: e.g., replace integers with polynomials.

Typical application:
Take a squarefree $g \in (\mathbf{Z}/2)[x]$.
What are g's irreducible divisors?

One answer: Find basis $h_1, h_2, \ldots$ for $\{h \in (\mathbf{Z}/2)[x] : (gh)' = h^2\}$ as a vector space over $\mathbf{Z}/2$.
Factor $g, h_1, h_2, \ldots$ into coprimes.
This list of coprimes contains all irreducible divisors of g.

(1993 Niederreiter, 1994 Göttfert)

More examples, applications of factoring into coprimes: see 1890 Stieltjes; 1974 Collins; 1985 Kaltofen; 1985 Della Dora DiCrescenzo Duval; 1986 Bach Miller Shallit; 1986 von zur Gathen; 1986 Lüneburg; 1989 Pohst Zassenhaus; 1990 Teitelbaum; 1990 Smedley; 1993 Bach Driscoll Shallit; 1994 Ge; 1994 Buchmann Lenstra; 1996 Bernstein; 1997 Silverman; 1998 Cohen Diaz y Diaz Olivier; 1998 Storjohann; . . .

`cr.yp.to/coprimes.html`

ly same algorithms
generality: e.g.,
ntegers with polynomials.

application:
squarefree $g \in (\mathbf{Z}/2)[x]$.
e g's irreducible divisors?

wer: Find basis $h_1, h_2, \ldots$
$\in (\mathbf{Z}/2)[x] : (gh)' = h^2\}$
tor space over $\mathbf{Z}/2$.
$, h_1, h_2, \ldots$ into coprimes.
of coprimes contains
ucible divisors of g.

iederreiter, 1994 Göttfert)

More examples, applications of factoring into coprimes: see 1890 Stieltjes; 1974 Collins; 1985 Kaltofen; 1985 Della Dora DiCrescenzo Duval; 1986 Bach Miller Shallit; 1986 von zur Gathen; 1986 Lüneburg; 1989 Pohst Zassenhaus; 1990 Teitelbaum; 1990 Smedley; 1993 Bach Driscoll Shallit; 1994 Ge; 1994 Buchmann Lenstra; 1996 Bernstein; 1997 Silverman; 1998 Cohen Diaz y Diaz Olivier; 1998 Storjohann; . . .

cr.yp.to/coprimes.html

Exercise
how wou
shared a

2012 He
Wustrow
best-pap
USENIX
2012 Le
Bos–Kle
indepen
Whit is

RSA key
use such
this does

lgorithms
: e.g.,
ith polynomials.

n:
$g \in (\mathbf{Z}/2)[x]$.
ucible divisors?

basis $h_1, h_2, \ldots$
$: (gh)' = h^2\}$
over $\mathbf{Z}/2$.
. into coprimes.
es contains
sors of g.

, 1994 Göttfert)

More examples, applications of factoring into coprimes: see 1890 Stieltjes; 1974 Collins; 1985 Kaltofen; 1985 Della Dora DiCrescenzo Duval; 1986 Bach Miller Shallit; 1986 von zur Gathen; 1986 Lüneburg; 1989 Pohst Zassenhaus; 1990 Teitelbaum; 1990 Smedley; 1993 Bach Driscoll Shallit; 1994 Ge; 1994 Buchmann Lenstra; 1996 Bernstein; 1997 Silverman; 1998 Cohen Diaz y Diaz Olivier; 1998 Storjohann; . . .

cr.yp.to/coprimes.html

Exercise: Given 2^2
how would you ch
shared among tho

2012 Heninger–Du
Wustrow–Halderm
best-paper award
USENIX Security
2012 Lenstra–Hug
Bos–Kleinjung–Wa
independent "Ron
Whit is right" pap

RSA keys on the I
use such bad rand
this does find fact

mials.

$)[x]$.

sors?

$h_2, \ldots$

$h^2\}$

rimes.

ns

ttfert)

More examples, applications of factoring into coprimes: see 1890 Stieltjes; 1974 Collins; 1985 Kaltofen; 1985 Della Dora DiCrescenzo Duval; 1986 Bach Miller Shallit; 1986 von zur Gathen; 1986 Lüneburg; 1989 Pohst Zassenhaus; 1990 Teitelbaum; 1990 Smedley; 1993 Bach Driscoll Shallit; 1994 Ge; 1994 Buchmann Lenstra; 1996 Bernstein; 1997 Silverman; 1998 Cohen Diaz y Diaz Olivier; 1998 Storjohann; . . .

cr.yp.to/coprimes.html

Exercise: Given 2^{23} RSA key how would you check for pri shared among those keys?

2012 Heninger–Durumeric– Wustrow–Halderman, best-paper award at USENIX Security Symposiur 2012 Lenstra–Hughes–Augie Bos–Kleinjung–Wachter, independent “Ron was wron Whit is right” paper, Crypto

RSA keys on the Internet use such bad randomness th this does find factors!

More examples, applications of factoring into coprimes: see 1890 Stieltjes; 1974 Collins; 1985 Kaltofen; 1985 Della Dora DiCrescenzo Duval; 1986 Bach Miller Shallit; 1986 von zur Gathen; 1986 Lüneburg; 1989 Pohst Zassenhaus; 1990 Teitelbaum; 1990 Smedley; 1993 Bach Driscoll Shallit; 1994 Ge; 1994 Buchmann Lenstra; 1996 Bernstein; 1997 Silverman; 1998 Cohen Diaz y Diaz Olivier; 1998 Storjohann; . . .

`cr.yp.to/coprimes.html`

Exercise: Given 2^{23} RSA keys, how would you check for primes shared among those keys?

2012 Heninger–Durumeric–Wustrow–Halderman, best-paper award at USENIX Security Symposium; 2012 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter, independent "Ron was wrong, Whit is right" paper, Crypto:

RSA keys on the Internet use such bad randomness that this does find factors!