

Time- and space-efficient polynomial computations

Bruno Grenet

Based on joint works with P. Giorgi, D. S. Roche and J.-G. Dumas



April 3., 2025

Computer Algebra 101

Complexity of polynomial multiplication: $M(n)$

- ▶ Classical: $O(n^2)$
- ▶ Karatsuba: $O(n^{\log_2 3}) = O(n^{1.585})$
- ▶ Toom-3: $O(n^{\log_3 5}) = O(n^{1.465})$
- ▶ FFT-based: $O(n \log n)$, given $\omega^{2^n} = 1$
 $O(n \log n \log \log n)$

[Karatsuba (1962)]

[Toom (1963), Cook (1966)]

[Cooley, Tukey (1965)]

[Cantor, Kaltofen (1991)]

Computer Algebra 101

Complexity of polynomial multiplication: $M(n)$

- ▶ Classical: $O(n^2)$
- ▶ Karatsuba: $O(n^{\log_2 3}) = O(n^{1.585})$ [Karatsuba (1962)]
- ▶ Toom-3: $O(n^{\log_3 5}) = O(n^{1.465})$ [Toom (1963), Cook (1966)]
- ▶ FFT-based: $O(n \log n)$, given $\omega^{2^n} = 1$
 $O(n \log n \log \log n)$ [Cooley, Tukey (1965)]
[Cantor, Kaltofen (1991)]

Complexity of other polynomial and power series operations

- ▶ Classical algorithms: $O(n^2)$
- ▶ Short and middle products: $M(n) + O(n)$ [Hanrot, Quercia, Zimmermann (2004)]
- ▶ Inversion, divisions: $O(M(n))$ [Strassen (1973), ...]
- ▶ Evaluation & interpolation: $O(M(n) \log n)$ [Borodin, Moenck (1974)]
- ▶ GCD: $O(M(n) \log n)$ [Brent, Gustavson, Yun (1980)]
- ▶ Power series composition: $O(M(n) \log n)$ [Kinoshita, Li (2024)]

Computer Algebra 101

Complexity of polynomial multiplication: $M(n)$

- ▶ Classical: $O(n^2)$
- ▶ Karatsuba: $O(n^{\log_2 3}) = O(n^{1.585})$ [Karatsuba (1962)]
- ▶ Toom-3: $O(n^{\log_3 5}) = O(n^{1.465})$ [Toom (1963), Cook (1966)]
- ▶ FFT-based: $O(n \log n)$, given $\omega^{2^n} = 1$
 $O(n \log n \log \log n)$ [Cooley, Tukey (1965)]
[Cantor, Kaltofen (1991)]

Complexity of other polynomial and power series operations

- ▶ Classical algorithms: $O(n^2)$
- ▶ Short and middle products: $M(n) + O(n)$ [Hanrot, Quercia, Zimmermann (2004)]
- ▶ Inversion, divisions: $O(M(n))$ [Strassen (1973), ...]
- ▶ Evaluation & interpolation: $O(M(n) \log n)$ [Borodin, Moenck (1974)]
- ▶ GCD: $O(M(n) \log n)$ [Brent, Gustavson, Yun (1980)]
- ▶ Power series composition: $O(M(n) \log n)$ [Kinoshita, Li (2024)]

What about space complexity?

Computer Algebra 101

Complexity of polynomial multiplication: $M(n)$

- ▶ Classical: $O(n^2)$ $O(1)$
- ▶ Karatsuba: $O(n^{\log_2 3}) = O(n^{1.585})$ $O(n)$ [Karatsuba (1962)]
- ▶ Toom-3: $O(n^{\log_3 5}) = O(n^{1.465})$ $O(n)$ [Toom (1963), Cook (1966)]
- ▶ FFT-based: $O(n \log n)$, given $\omega^{2^n} = 1$ $O(n)$ [Cooley, Tukey (1965)]
 $O(n \log n \log \log n)$ $O(n)$ [Cantor, Kaltofen (1991)]

Complexity of other polynomial and power series operations

- ▶ Classical algorithms: $O(n^2)$ $O(1)$ (usually)
- ▶ Short and middle products: $M(n) + O(n)$ $O(n)$ [Hanrot, Quercia, Zimmermann (2004)]
- ▶ Inversion, divisions: $O(M(n))$ $O(n)$ [Strassen (1973), ...]
- ▶ Evaluation & interpolation: $O(M(n) \log n)$ $O(n \log n)$ [von zur Gathen, Shoup (1992)]
- ▶ GCD: $O(M(n) \log n)$ $O(n \log n)$ [Brent, Gustavson, Yun (1980)]
- ▶ Power series composition: $O(M(n) \log n)$ 😊 [Kinoshita, Li (2024)]

What about space complexity?

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} f_3 \ f_2 \ f_1 \ f_0 \\ \times \quad g_3 \ g_2 \ g_1 \ g_0 \\ \hline f_3 g_0 \ f_2 g_0 \ f_1 g_0 \ f_0 g_0 \\ + \quad f_3 g_1 \ f_2 g_1 \ f_1 g_1 \ f_0 g_1 \\ + \quad f_3 g_2 \ f_2 g_2 \ f_1 g_2 \ f_0 g_2 \\ + \quad f_3 g_3 \ f_2 g_3 \ f_1 g_3 \ f_0 g_3 \\ \hline = \quad h_6 \ h_5 \ h_4 \ h_3 \ h_2 \ h_1 \ h_0 \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} f_3 f_2 f_1 f_0 \\ \times g_3 g_2 g_1 g_0 \\ \hline f_3 g_0 f_2 g_0 f_1 g_0 f_0 g_0 \\ + f_3 g_1 f_2 g_1 f_1 g_1 f_0 g_1 \\ + f_3 g_2 f_2 g_2 f_1 g_2 f_0 g_2 \\ + f_3 g_3 f_2 g_3 f_1 g_3 f_0 g_3 \\ \hline h_6 h_5 h_4 h_3 h_2 h_1 h_0 \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

$$\begin{array}{r} f_1 f_0 \\ \times g_1 g_0 \\ \hline f_1 \times g_1 \quad f_0 \times g_0 \\ + f_1 \times g_0 \\ + f_0 \times g_1 \end{array}$$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} f_3 f_2 f_1 f_0 \\ \times g_3 g_2 g_1 g_0 \\ \hline f_3 g_0 f_2 g_0 f_1 g_0 f_0 g_0 \\ + f_3 g_1 f_2 g_1 f_1 g_1 f_0 g_1 \\ + f_3 g_2 f_2 g_2 f_1 g_2 f_0 g_2 \\ + f_3 g_3 f_2 g_3 f_1 g_3 f_0 g_3 \\ \hline h_6 h_5 h_4 h_3 h_2 h_1 h_0 \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

$$\begin{array}{r} f_1 f_0 \\ \times g_1 g_0 \\ \hline f_1 \times g_1 \quad f_0 \times g_0 \\ + f_1 \times g_0 \\ + f_0 \times g_1 \end{array}$$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} f_3 f_2 f_1 f_0 \\ \times g_3 g_2 g_1 g_0 \\ \hline f_3 g_0 f_2 g_0 f_1 g_0 f_0 g_0 \\ + f_3 g_1 f_2 g_1 f_1 g_1 f_0 g_1 \\ + f_3 g_2 f_2 g_2 f_1 g_2 f_0 g_2 \\ + f_3 g_3 f_2 g_3 f_1 g_3 f_0 g_3 \\ \hline h_6 h_5 h_4 h_3 h_2 h_1 h_0 \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

$$\begin{array}{r} f_1 f_0 \\ \times g_1 g_0 \\ \hline f_1 \times g_1 \quad f_0 \times g_0 \\ - (f_0 - f_1)(g_0 - g_1) \\ + f_0 \times g_0 \\ + f_1 \times g_1 \end{array} \quad \begin{array}{l} f_0 - f_1 \\ g_0 - g_1 \end{array}$$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} f_3 f_2 f_1 f_0 \\ \times g_3 g_2 g_1 g_0 \\ \hline f_3 g_0 f_2 g_0 f_1 g_0 f_0 g_0 \\ + f_3 g_1 f_2 g_1 f_1 g_1 f_0 g_1 \\ + f_3 g_2 f_2 g_2 f_1 g_2 f_0 g_2 \\ + f_3 g_3 f_2 g_3 f_1 g_3 f_0 g_3 \\ \hline = h_6 h_5 h_4 h_3 h_2 h_1 h_0 \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

$$\begin{array}{r} f_1 f_0 \\ \times g_1 g_0 \\ \hline f_1 \times g_1 \quad f_0 \times g_0 \\ - (f_0 - f_1)(g_0 - g_1) \\ + f_0 \times g_0 \\ + f_1 \times g_1 \\ \hline f_0 - f_1 \\ g_0 - g_1 \end{array}$$

- ▶ Time: $O(n^{\log 3}) = O(n^{1.585})$
- ▶ Space: $O(n)$

What is space complexity?

Algebraic RAM

- ▶ Algebraic registers containing one ring element each for some ring R
- ▶ Registers for pointers of size $O(\log n)$ $n = \#$ input registers

Space complexity

- ▶ Number of registers used, not counting the input and output registers
- ▶ Distinction between algebraic registers and pointers

Relation with standard complexity classes

- ▶ Input size if $\#R = p$: $N = O(n \log p)$ $\log N = O(\log n + \log \log p)$
- ▶ One pointer $\rightarrow O(\log n) = O(\log N)$ bit-size
- ▶ One algebraic register \rightarrow depends on the *relative* size of p and n
 - ▶ $\log p = \omega(\log n)$: one register is larger than $\log N$
 - ▶ $\log p = O(\log n)$: one register is $O(\log N)$

Rule of thumb: constant number of registers (of both kinds) \simeq complexity class L

Permission models

ro/wo: read-only inputs & write-only output

- 👍 classical model in complexity theory, *easy* composition of alg.
- 👎 further from practice, composition does not preserve time
- 👎 $\Omega(n^2)$ lower bounds on time \times space for multiplication

[Abrahamson (1985)]

ro/rw: read-only inputs & read-write output

- 👍 closer to practice, allows parallel access to the inputs
- 👎 still restrictive in a sequential model

rw/rw: read-write inputs & read-write output, *inputs restored at the end*

- 👍 still consistent with practice, allows recursive calls / use as subroutines
- 👎 not suitable for parallel programming

Goal: time- and space-efficient algorithms in the ro/rw and rw/rw models

Contents

1. Algorithms in the `ro/rw` model

2. Algorithms in the `rw/rw` model

Known results for multiplication algorithms

algorithm	model	time	alg. sp.	# pointers	
Classical	ro/wo	$O(n^2)$	$O(1)$	$O(1)$	folklore
Karatsuba	ro/wo	$O(n^{\log_3 3})$	$O(n)$	$O(\log n)$	Karatsuba (1962)
	ro/rw		n	$O(\log n)$	Thomé (2002)
	ro/rw		$O(1)$	$O(\log n)$	Roche (2009)
Toom-Cook	ro/wo	$O(n^{\log_3 5})$	$O(n)$	$O(\log n)$	Toom (1963)
FFT-based (given $\omega^{2^n} = 1$)	ro/wo	$O(n \log n)$	$O(n)$	$O(1)$	Cooley, Tukey (1965)
	ro/rw		$O(1)$	$O(1)$	Roche (2009) (if $n = 2^k$)
	ro/rw		$O(1)$	$O(1)$	Harvey, Roche (2010)

Known results for multiplication algorithms

algorithm	model	time	alg. sp.	# pointers	
Classical	ro/wo	$O(n^2)$	$O(1)$	$O(1)$	folklore
Karatsuba	ro/wo	$O(n^{\log_3 3})$	$O(n)$	$O(\log n)$	Karatsuba (1962)
	ro/rw		n	$O(\log n)$	Thomé (2002)
	ro/rw		$O(1)$	$O(\log n)$	Roche (2009)
Toom-Cook	ro/wo	$O(n^{\log_3 5})$	$O(n)$	$O(\log n)$	Toom (1963)
FFT-based (given $\omega^{2^n} = 1$)	ro/wo	$O(n \log n)$	$O(n)$	$O(1)$	Cooley, Tukey (1965)
	ro/rw		$O(1)$	$O(1)$	Roche (2009) (if $n = 2^k$)
	ro/rw		$O(1)$	$O(1)$	Harvey, Roche (2010)

Our result

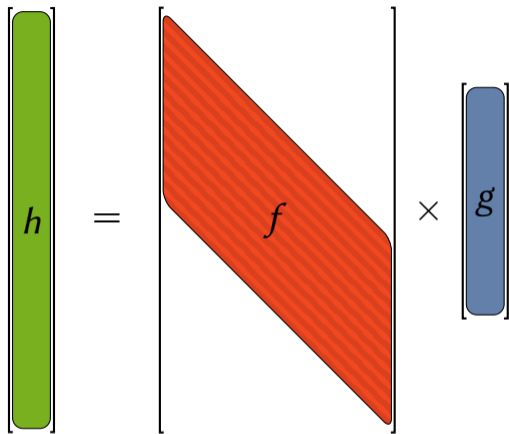
[Giorgi, G., Roche (2019)]

Any linear-space multiplication algorithm can be made constant-space with the same asymptotic time complexity in the ro/rw model

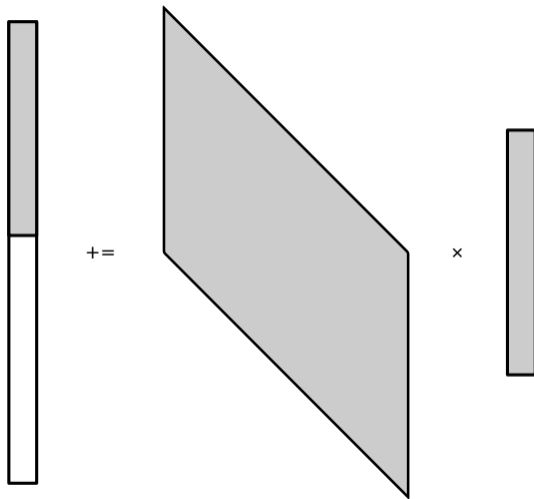
$h = f \times g$ as a matrix-vector product

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \\ h_{10} \end{bmatrix} = \begin{bmatrix} f_0 & & & & & & & & & & & \\ f_1 & f_0 & & & & & & & & & & \\ f_2 & f_1 & f_0 & & & & & & & & & \\ f_3 & f_2 & f_1 & f_0 & & & & & & & & \\ f_4 & f_3 & f_2 & f_1 & f_0 & & & & & & & \\ f_5 & f_4 & f_3 & f_2 & f_1 & f_0 & & & & & & \\ & f_5 & f_4 & f_3 & f_2 & f_1 & & & & & & \\ & & f_5 & f_4 & f_3 & f_2 & & & & & & \\ & & & f_5 & f_4 & f_3 & & & & & & \\ & & & & f_5 & f_4 & & & & & & \\ & & & & & f_5 & f_4 & & & & & \\ & & & & & & f_5 & f_4 & & & & \\ & & & & & & & f_5 & f_4 & & & \\ & & & & & & & & f_5 & & & \\ & & & & & & & & & f_5 & & \\ & & & & & & & & & & f_5 & \\ & & & & & & & & & & & f_5 \end{bmatrix} \times \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}$$

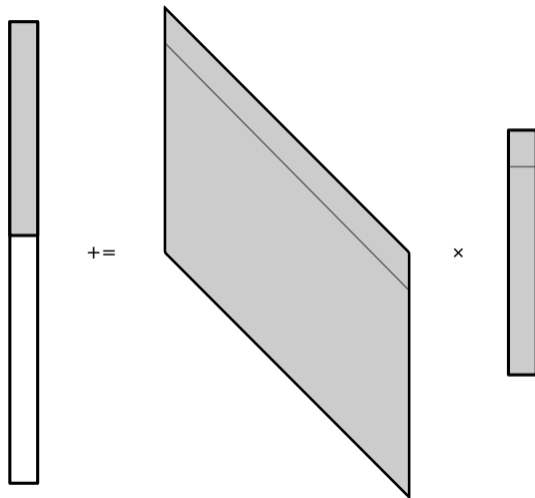
$h = f \times g$ as a matrix-vector product



Proof sketch: *semi-cumulative* algorithm

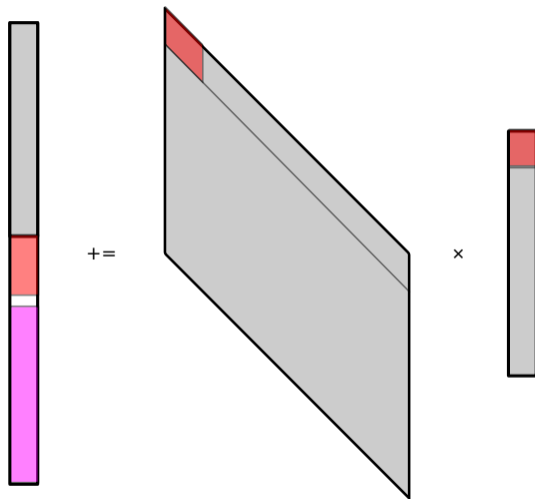


Proof sketch: *semi-cumulative* algorithm



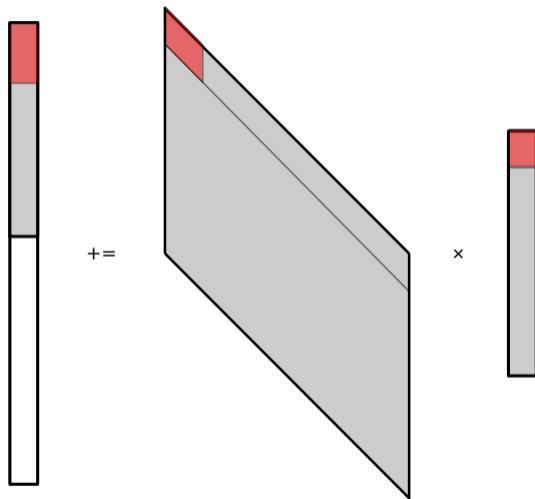
$$(f_0 + X^k f_1) \cdot (g_0 + X^k g_1) = f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



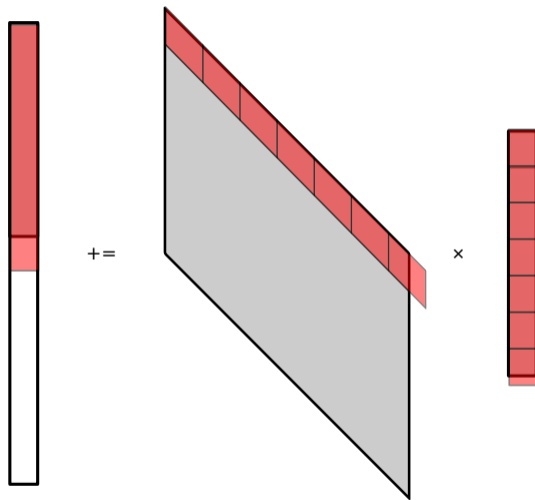
$$(f_0 + X^k f_1) \cdot (g_0 + X^k g_1) = f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



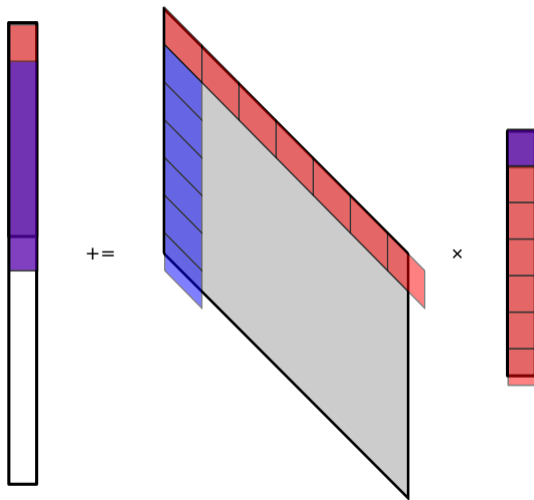
$$(f_0 + X^k f_1) \cdot (g_0 + X^k g_1) = f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



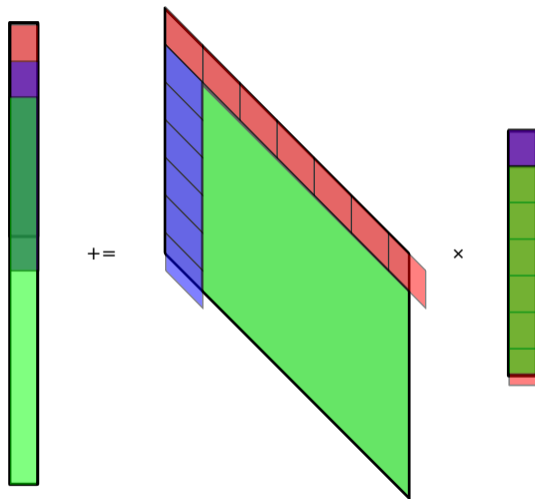
$$(f_0 + X^k f_1) \cdot (g_0 + X^k g_1) = f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



$$(f_0 + X^k f_1) \cdot (g_0 + X^k g_1) = f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



$$(f_0 + X^k f_1) \cdot (g_0 + X^k g_1) = f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Known results for other operations

algorithm	model	time	alg. sp.
Classical algorithms (inversion, division, multipoint evaluation, interpolation)	ro/wo	$O(n^2)$	$O(1)$
Power series inversion	ro/wo	$O(M(n))$	$O(n)$
Power series division	ro/wo	$O(M(n))$	$O(n)$
Euclidean division ¹	ro/wo	$O(M(m+n))$	$O(\max(m, n))$
Evaluation & interpolation	ro/wo	$O(M(n) \log n)$	$O(n \log n)$
			$O(n)$

Strassen (1973),

...

Borodin, Moenck (1974)
von zur Gathen,
Shoup (1992)

¹Inputs of size $(m + n - 1)$ and n

Known results for other operations

algorithm	model	time	alg. sp.	
Classical algorithms (inversion, division, multipoint evaluation, interpolation)	ro/wo	$O(n^2)$	$O(1)$	
Power series inversion	ro/wo	$O(M(n))$	$O(n)$	Strassen (1973), ...
Power series division	ro/wo	$O(M(n))$	$O(n)$	
Euclidean division ¹	ro/wo	$O(M(m+n))$	$O(\max(m, n))$	
Evaluation & interpolation	ro/wo	$O(M(n) \log n)$	$O(n \log n)$	Borodin, Moenck (1974) von zur Gathen, Shoup (1992)
			$O(n)$	

Our results

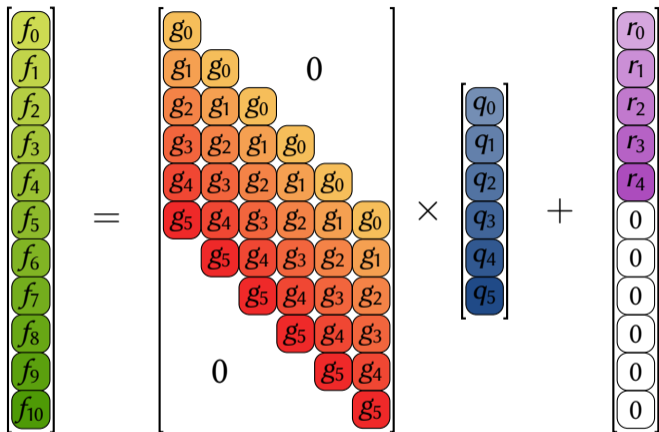
[Giorgi, G., Roche (2020)]

Constant-space algorithms in the ro/rw model, in

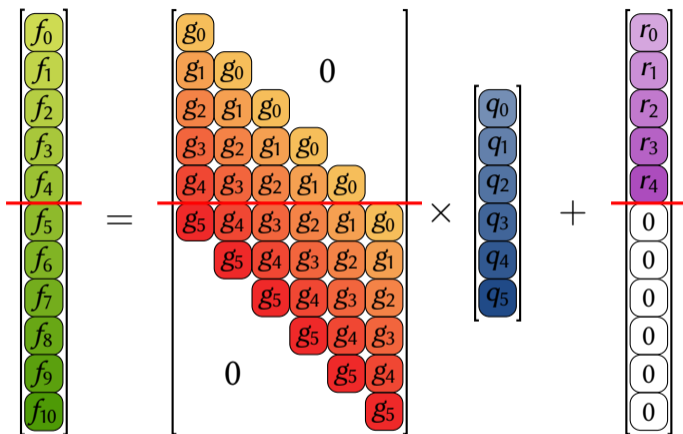
- ▶ same asymptotic time complexity for Euclidean division and evaluation / interpolation
- ▶ $O(M(n) \log n)$ for power series inversion & division

¹Inputs of size $(m+n-1)$ and n

Fast Euclidean division: $f = g \times q + r$



Fast Euclidean division: $f = g \times q + r$



Fast Euclidean division: $f = g \times q + r$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} g_0 & & & & & \\ g_1 & g_0 & & & & \\ g_2 & g_1 & g_0 & & & \\ g_3 & g_2 & g_1 & g_0 & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & \\ & & & & & 0 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_4 \\ q_5 \end{bmatrix} + \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

$$\begin{bmatrix} f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \end{bmatrix} = \begin{bmatrix} g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \\ & g_5 & g_4 & g_3 & g_2 & g_1 \\ & & g_5 & g_4 & g_3 & g_2 \\ & & & g_5 & g_4 & g_3 \\ & & & & g_5 & g_4 \\ & & & & & g_5 \\ & & & & & & 0 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Fast Euclidean division: $f = g \times q + r$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} g_0 & & & & & \\ g_1 & g_0 & & & & 0 \\ g_2 & g_1 & g_0 & & & \\ g_3 & g_2 & g_1 & g_0 & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

$$\begin{bmatrix} f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \end{bmatrix} = \begin{bmatrix} g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \\ & g_5 & g_4 & g_3 & g_2 & g_1 \\ & & g_5 & g_4 & g_3 & g_2 \\ & & & g_5 & g_4 & g_3 \\ & & & & g_5 & g_4 \\ & & & & & g_5 \\ 0 & & & & & \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Fast Euclidean division: $f = g \times q + r$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} g_0 & & & & & \\ g_1 & g_0 & & & & 0 \\ g_2 & g_1 & g_0 & & & \\ g_3 & g_2 & g_1 & g_0 & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} + \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

$$\begin{bmatrix} f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \end{bmatrix} = \begin{bmatrix} g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \\ & g_5 & g_4 & g_3 & g_2 & g_1 \\ & & g_5 & g_4 & g_3 & g_2 \\ & & & g_5 & g_4 & g_3 \\ & & & & g_5 & g_4 \\ & & & & & g_5 \\ 0 & & & & & & \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix}$$

Fast Euclidean division: $f = g \times q + r$

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} - \begin{bmatrix} g_0 & & & & & \\ g_1 & g_0 & & & & \\ g_2 & g_1 & g_0 & & & \\ g_3 & g_2 & g_1 & g_0 & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & \\ & & & & & 0 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} = \begin{bmatrix} g_5 & g_4 & g_3 & g_2 & g_1 & g_0 \\ & g_5 & g_4 & g_3 & g_2 & g_1 \\ & & g_5 & g_4 & g_3 & g_2 \\ & & & g_5 & g_4 & g_3 \\ & & & & g_5 & g_4 \\ & & & & & g_5 \\ & & & & & & 0 \end{bmatrix}^{-1} \times \begin{bmatrix} f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \end{bmatrix}$$

Fast Euclidean division: $f = g \times q + r$

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} - \begin{bmatrix} g_0 & & & & & \\ g_1 & g_0 & & & & \\ g_2 & g_1 & g_0 & & & \\ g_3 & g_2 & g_1 & g_0 & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & \\ & & & & & 0 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\begin{bmatrix} q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} g_5 & & & & & \\ g_4 & g_5 & & & & \\ g_3 & g_4 & g_5 & & & \\ g_2 & g_3 & g_4 & g_5 & & \\ g_1 & g_2 & g_3 & g_4 & g_5 & \\ g_0 & g_1 & g_2 & g_3 & g_4 & g_5 \end{bmatrix}^{-1} \times \begin{bmatrix} f_{10} \\ f_9 \\ f_8 \\ f_7 \\ f_6 \\ f_5 \end{bmatrix}$$

Fast Euclidean division: $f = g \times q + r$

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} - \begin{bmatrix} g_0 & & & & & \\ g_1 & g_0 & & & & \\ g_2 & g_1 & g_0 & & & \\ g_3 & g_2 & g_1 & g_0 & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\begin{bmatrix} q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} g_5 & & & & & \\ g_4 & g_5 & & & & \\ g_3 & g_4 & g_5 & & & \\ g_2 & g_3 & g_4 & g_5 & & \\ g_1 & g_2 & g_3 & g_4 & g_5 & \\ g_0 & g_1 & g_2 & g_3 & g_4 & g_5 \end{bmatrix}^{-1} \times \begin{bmatrix} f_{10} \\ f_9 \\ f_8 \\ f_7 \\ f_6 \\ f_5 \end{bmatrix}$$

1. $\bar{q} := \bar{f} / \bar{g} \pmod{x^n}$

2. $r := (f - gq) \pmod{x^{n-1}}$

power series division applied on reversed polynomials

short product and subtraction

Power series division

$$\begin{bmatrix} q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix} = \begin{bmatrix} g_5 & & & & & \\ g_4 & g_5 & & & & \\ g_3 & g_4 & g_5 & & & \\ g_2 & g_3 & g_4 & g_5 & & \\ g_1 & g_2 & g_3 & g_4 & g_5 & \\ g_0 & g_1 & g_2 & g_3 & g_4 & g_5 \end{bmatrix}^{-1} \begin{bmatrix} f_{10} \\ f_9 \\ f_8 \\ f_7 \\ f_6 \\ f_5 \end{bmatrix}$$

0

×

Power series division

$$\begin{bmatrix} \vec{q} \end{bmatrix} = \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \vec{f} \end{bmatrix}$$

The diagram shows a blue rounded rectangle containing the vector \vec{q} . This is followed by an equals sign. To the right is a red hatched lower triangular matrix labeled $\sigma_{\alpha, \uparrow}$. A superscript -1 is placed to the top right of the matrix's closing bracket. This is followed by a multiplication sign \times and a green rounded rectangle containing the vector \vec{f} .

Power series division

$$\begin{bmatrix} \vec{q} \end{bmatrix} = \begin{bmatrix} L \\ S \quad L \end{bmatrix}^{-1} \begin{bmatrix} \vec{f} \end{bmatrix}$$

Power series division

$$\begin{bmatrix} \vec{q} \end{bmatrix} = \begin{bmatrix} L^{-1} & & \\ -L^{-1}SL^{-1} & L^{-1} & \\ & & \ddots \end{bmatrix} \times \begin{bmatrix} \vec{f} \end{bmatrix}$$
The diagram illustrates the equation for power series division. On the left is a blue vertical vector labeled \vec{q} . This is equal to a large matrix multiplied by a green vertical vector labeled \vec{f} . The matrix is composed of three red-shaded triangular blocks. The top-left block is a lower triangular matrix labeled L^{-1} . The bottom-left block is a rectangular matrix labeled $-L^{-1}SL^{-1}$. The bottom-right block is a lower triangular matrix labeled L^{-1} .

Power series division

$$\begin{bmatrix} \overleftarrow{q} \end{bmatrix} = \begin{bmatrix} L^{-1} & & \\ -L^{-1}SL^{-1} & L^{-1} & \\ & & \ddots \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f} \end{bmatrix}$$

1. Compute $h = 1/\overleftarrow{g} \bmod x^n$:

1.1 $h^1 := 1/\overleftarrow{g} \bmod x^{n/2}$

1.2 $t := \text{MIDPROD}(\overleftarrow{g}, h^1)$

1.3 $h^0 := -h^1 \times t \bmod x^{n/2}$

L^{-1} (recursive call)

$S \cdot L^{-1}$

$-L^{-1} \cdot (SL^{-1})$

2. Compute \overleftarrow{q} :

2.1 $\overleftarrow{q} := h \times \overleftarrow{f} \bmod x^n$

Power series division

$$\begin{bmatrix} \overleftarrow{q} \end{bmatrix} = \begin{bmatrix} L^{-1} \\ -L^{-1}SL^{-1} & L^{-1} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f} \end{bmatrix}$$

1. Compute $h = 1/\overleftarrow{g} \bmod x^n$:

1.1 $h^1 := 1/\overleftarrow{g} \bmod x^{n/2}$

1.2 $t := \text{MIDPROD}(\overleftarrow{g}, h^1)$

1.3 $h^0 := -h^1 \times t \bmod x^{n/2}$

L^{-1} (recursive call)

$S \cdot L^{-1}$

$-L^{-1} \cdot (SL^{-1})$

$n/2 + s_{\text{INV}}(n/2)$

$n/2 + s_{\text{MP}}(n/2)$

$n/2$

2. Compute \overleftarrow{q} :

2.1 $\overleftarrow{q} := h \times \overleftarrow{f} \bmod x^n$

$O(1)$

Power series division

$$\begin{bmatrix} \overleftarrow{q} \end{bmatrix} = \begin{bmatrix} L^{-1} \\ -L^{-1}SL^{-1} & L^{-1} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f} \end{bmatrix}$$

1. Compute $h = 1/\overleftarrow{g} \bmod x^n$:

1.1 $h^1 := 1/\overleftarrow{g} \bmod x^{n/2}$

L^{-1} (recursive call)

$n/2 + s_{\text{INV}}(n/2)$

1.2 $t := \text{MIDPROD}(\overleftarrow{g}, h^1)$

$S \cdot L^{-1}$

$n/2 + s_{\text{MP}}(n/2)$

1.3 $h^0 := -h^1 \times t \bmod x^{n/2}$

$-L^{-1} \cdot (SL^{-1})$

$n/2$

2. Compute \overleftarrow{q} :

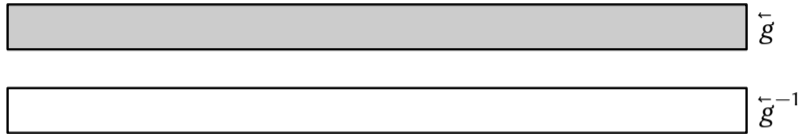
2.1 $\overleftarrow{q} := h \times \overleftarrow{f} \bmod x^n$

$O(1)$

$s_{\text{DIV}}(n) = O(n)$

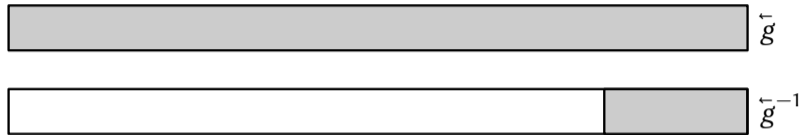
Constant-space divisions

Classical Newton iteration



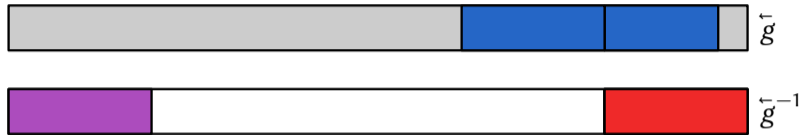
Constant-space divisions

Classical Newton iteration



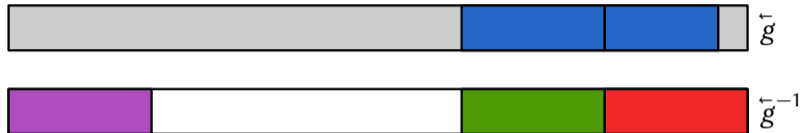
Constant-space divisions

Classical Newton iteration



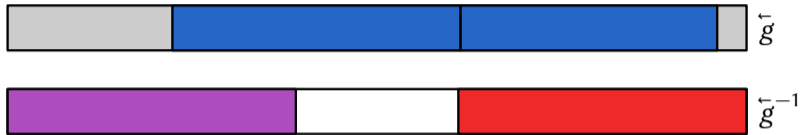
Constant-space divisions

Classical Newton iteration



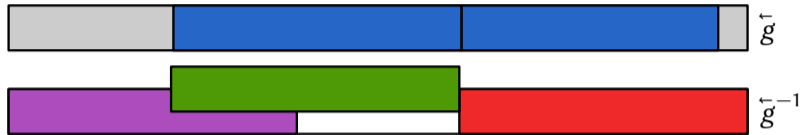
Constant-space divisions

Classical Newton iteration



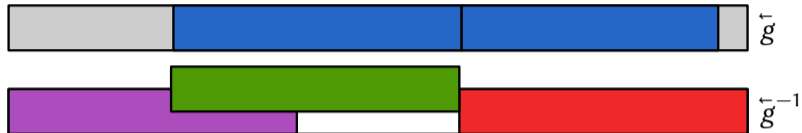
Constant-space divisions

Classical Newton iteration

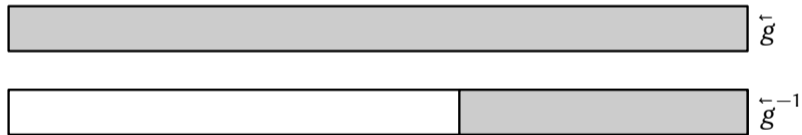


Constant-space divisions

Classical Newton iteration

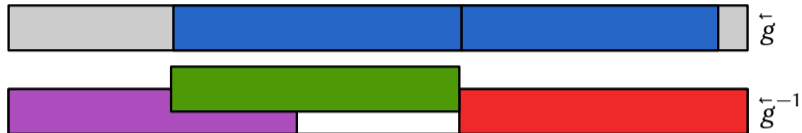


Modified Newton iteration: slow down the computation

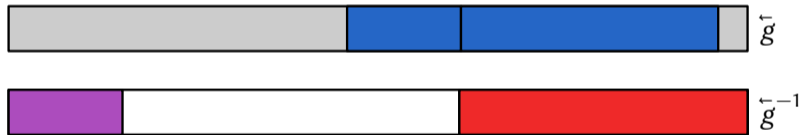


Constant-space divisions

Classical Newton iteration

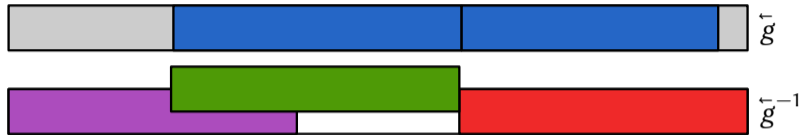


Modified Newton iteration: slow down the computation

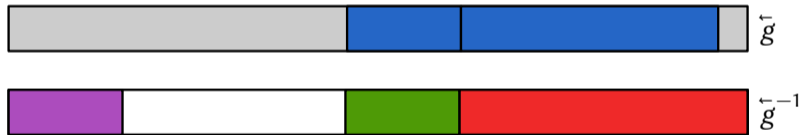


Constant-space divisions

Classical Newton iteration

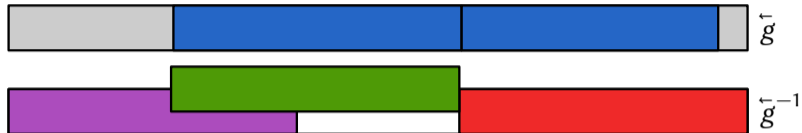


Modified Newton iteration: slow down the computation



Constant-space divisions

Classical Newton iteration

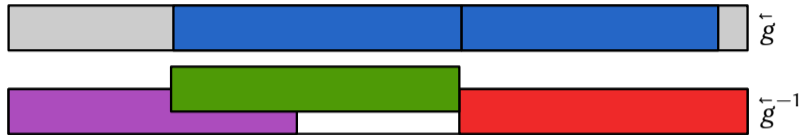


Modified Newton iteration: slow down the computation

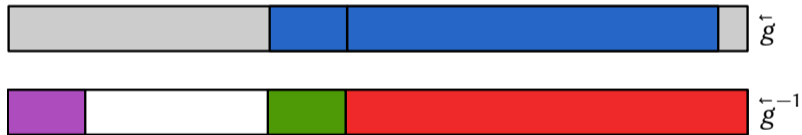


Constant-space divisions

Classical Newton iteration

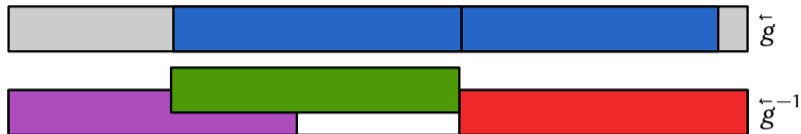


Modified Newton iteration: slow down the computation



Constant-space divisions

Classical Newton iteration



Modified Newton iteration: slow down the computation



Constant-space computation of $\bar{q} := \bar{f}/\bar{g} \bmod x^n$

[Giorgi, G., Roche (2020)]

- ▶ in time $O(M(n) \log n)$
- ▶ in time $O(M(n))$ if f can be erased

Fast Euclidean division in the ro/rw model

$$f = gq + r \text{ with } \deg(r) < \deg(g)$$

$$m = \deg(g), n = \deg(q), k = \max(n, m)$$

Algorithm	Model		QuoRem	Quotient only	Remainder only
Classical	ro/wo	time	$O(mn)$	same result [Monagan'93]	same result [Monagan'93]
		space	$O(1)$		
Fast [Strassen'73, ...]	ro/wo	time	$O(M(k))$	same result	same result
		space	$O(k)$		
Fast & low space [Giorgi G. Roche'20]	ro/rw	time	$O(M(k))$	$O(M(n)\log n)^*$	$O(M(k))$
		space	$O(1)$	$O(1)$	$\min(m, n) + O(1)$

* $O(M(n))$ if $M(n) = \Omega(n^{1+\epsilon})$

Fast Euclidean division in the ro/rw model

$$f = gq + r \text{ with } \deg(r) < \deg(g)$$

$$m = \deg(g), n = \deg(q), k = \max(n, m)$$

Algorithm	Model		QuoRem	Quotient only	Remainder only
Classical	ro/wo	time	$O(mn)$	same result [Monagan'93]	same result [Monagan'93]
		space	$O(1)$		
Fast [Strassen'73, ...]	ro/wo	time	$O(M(k))$	same result	same result
		space	$O(k)$		
Fast & low space [Giorgi G. Roche'20]	ro/rw	time	$O(M(k))$	$O(M(n)\log n)^*$	$O(M(k))$
		space	$O(1)$	$O(1)$	$\min(m, n) + O(1)$

* $O(M(n))$ if $M(n) = \Omega(n^{1+\epsilon})$

Can we compute the remainder only in constant space?

Contents

1. Algorithms in the `ro/rw` model

2. Algorithms in the `rw/rw` model

Cumulative and in-place operations

Rewriting view of algebraic problems

- ▶ No distinction input/output: $u \in \mathbb{R}^n$ replaced by $\pi(u) \in \mathbb{R}^n$
 - ▶ Inputs: entries that are not ignored
 - ▶ Outputs: entries that are modified
- ▶ Examples:
 - ▶ $h := f \times g$: $(f, g, h) \mapsto (f, g, f \times g)$
 - ▶ $h += f \times g$: $(f, g, h) \mapsto (f, g, h + f \times g)$
 - ▶ $g *= f$: $(f, g) \mapsto (f, f \times g)$

$\pi(u)$ depends on $u_{[i]}$
 $\pi(u)_{[i]} \neq u_{[i]}$

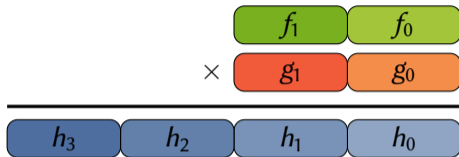
cumulative op.
in-place op.

rw/rw model and catalytic computation

- ▶ Catalytic model: ro/wo + *catalytic space* [Buhrman, Cleve, Koucký, Loff, Speelman (2014)]
 - ▶ catalytic space is initially non-empty, can be used but restored
 - ▶ rw/rw \simeq inputs and outputs as catalytic space
- ▶ Global storage model: [Goldreich (2008)]
 - ▶ *global* storage for inputs, outputs and oracle queries
 - ▶ *local* storage for extra space

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



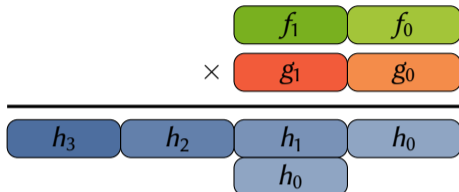
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



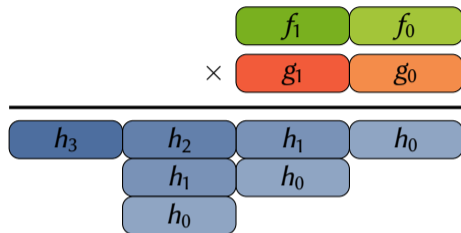
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



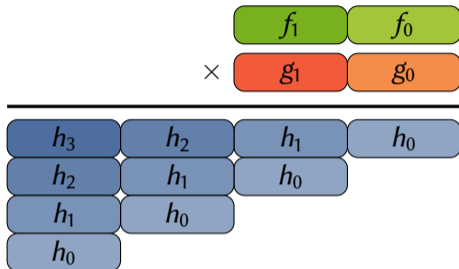
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



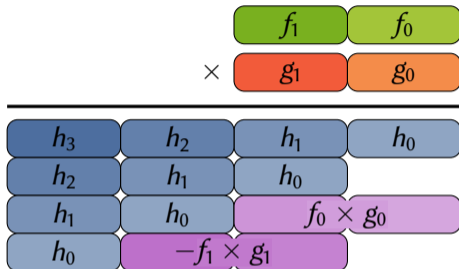
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



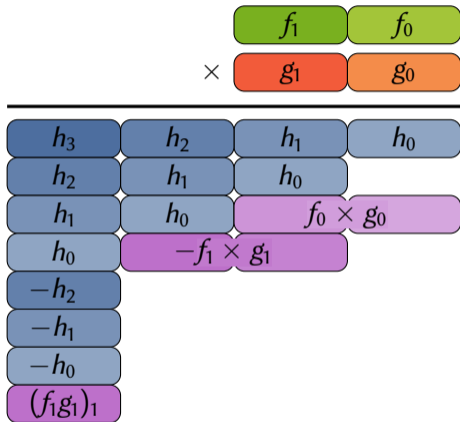
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



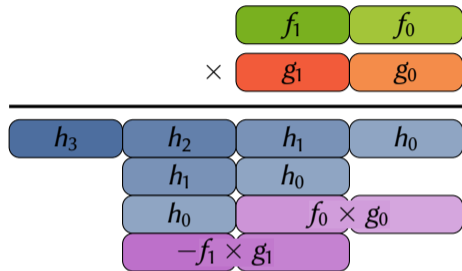
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



Inputs: f, g, h

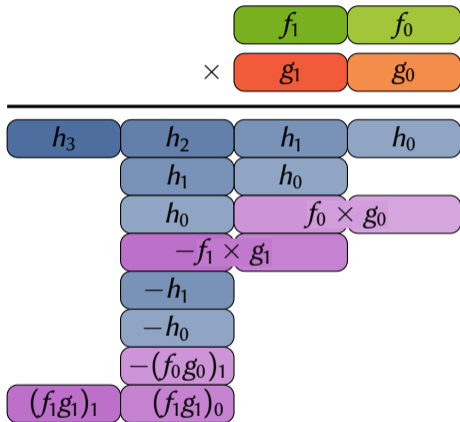
Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

$(f_1g_1)_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



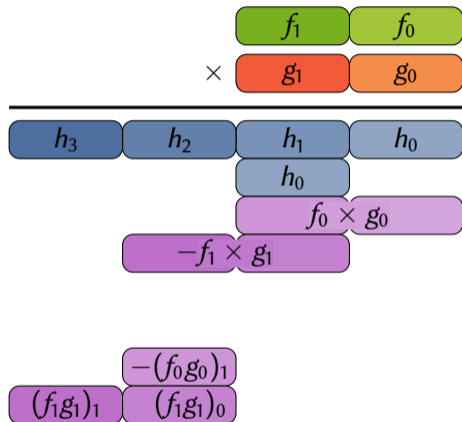
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



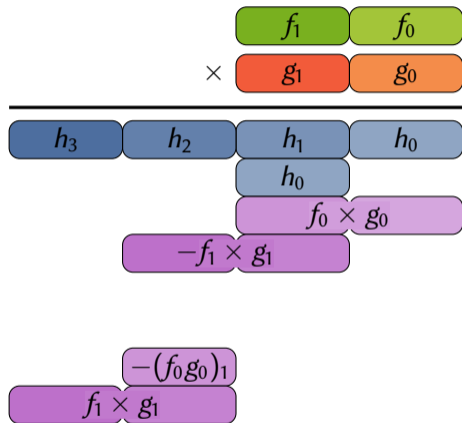
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



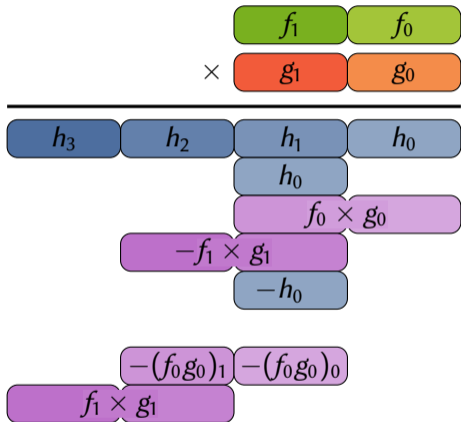
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



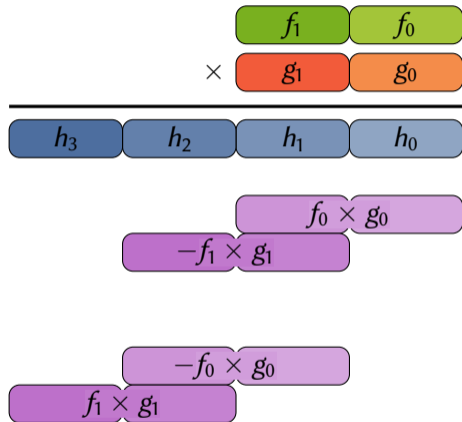
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



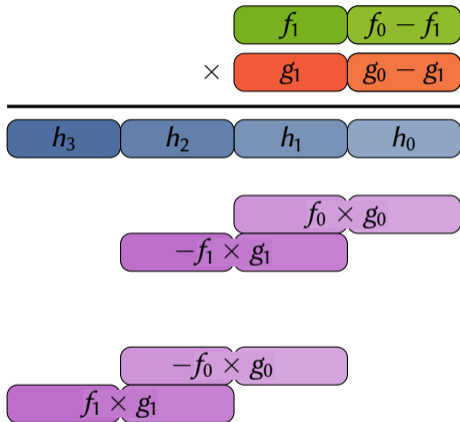
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



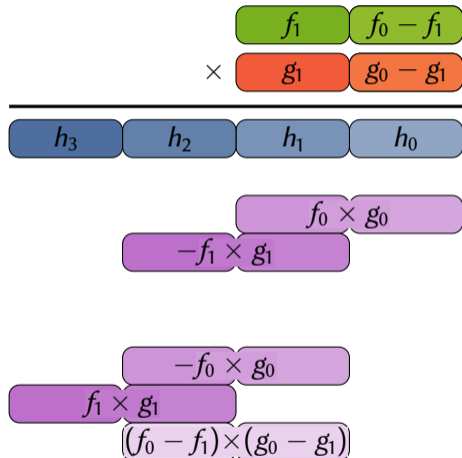
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



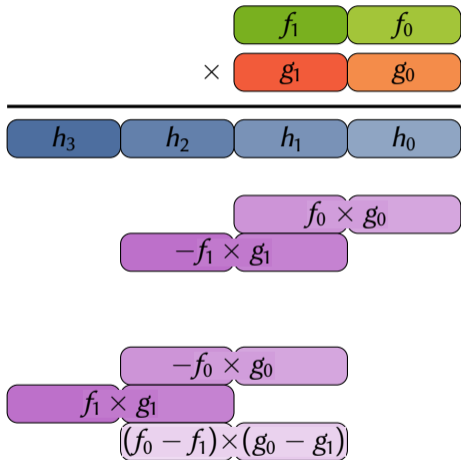
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



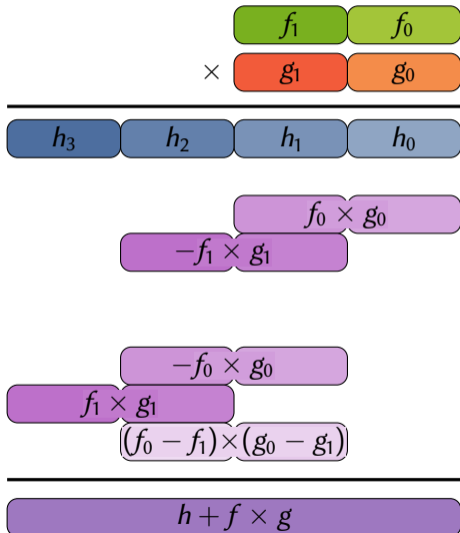
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



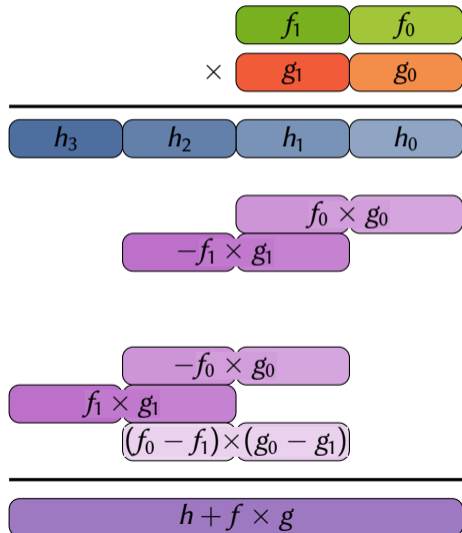
Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

Cumulative multiplication of polynomials

Example of Karatsuba's algorithm



Inputs: f, g, h

Output: $h += f \times g$

1. $h_0 += h_1; h_2 += h_1; h_3 += h_2$
2. $[h_0, h_1] += f_0 \times g_0$ (recursive call)
3. $[h_1, h_2] -= f_1 \times g_1$ (recursive call)
4. $h_3 -= h_2; h_2 -= h_1; h_0 -= h_1$
5. $f_0 -= f_1; g_0 -= g_1$
6. $[h_1, h_2] += f_0 \times g_0$ (recursive call)
7. $f_0 += f_1; g_0 += g_1$

▶ Time: $O(n^{\log 3})$

▶ Space: $O(1)$ alg. + $O(\log n)$ pointers

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \end{array} \text{ mod } x^n$$

Inputs: f, g, h

Output: $h += f \times g \text{ mod } x^n$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \text{ mod } x^{n/2}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \begin{array}{|c|} \hline -h_0 \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \begin{array}{|c|} \hline -h_0 \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0$; $h += f_0 \times g_1$; $h_1 -= h_0$
2. $g_0 -= g_1$; $h += f_0 \times g_0$; $g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \begin{array}{|c|} \hline -h_0 \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \\ \begin{array}{|c|} \hline h_0 \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_0 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0$; $h += f_0 \times g_1$; $h_1 -= h_0$
2. $g_0 -= g_1$; $h += f_0 \times g_0$; $g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0$; $h += f_0 \times g_1$; $h_1 -= h_0$
2. $g_0 -= g_1$; $h += f_0 \times g_0$; $g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 - g_1 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 - g_1 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_0 \times (g_0 - g_1) \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0$; $h += f_0 \times g_1$; $h_1 -= h_0$
2. $g_0 -= g_1$; $h += f_0 \times g_0$; $g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_0 \times (g_0 - g_1) \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \\ \hline \end{array} \pmod{x^n}$$

$$\begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array}$$

$$f_0 \times g_1$$

$$f_0 g_1 \pmod{x^{\frac{n}{2}}}$$

$$f_0 \times (g_0 - g_1)$$

$$f_1 g_1 \pmod{x^{\frac{n}{2}}}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product

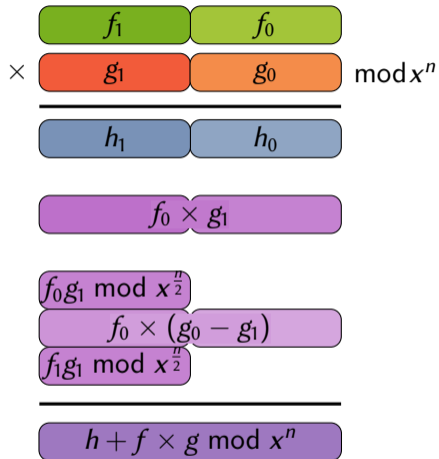
$$\begin{array}{r} \begin{array}{|c|c|} \hline f_1 & f_0 \\ \hline \end{array} \\ \times \begin{array}{|c|c|} \hline g_1 & g_0 \\ \hline \end{array} \pmod{x^n} \\ \hline \begin{array}{|c|c|} \hline h_1 & h_0 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 \times g_1 \\ \hline \end{array} \\ \\ \begin{array}{|c|} \hline f_0 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_0 \times (g_0 - g_1) \\ \hline \end{array} \\ \begin{array}{|c|} \hline f_1 g_1 \pmod{x^{\frac{n}{2}}} \\ \hline \end{array} \\ \hline \begin{array}{|c|} \hline h + f \times g \pmod{x^n} \\ \hline \end{array} \end{array}$$

Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

Cumulative short product



Inputs: f, g, h

Output: $h += f \times g \pmod{x^n}$

1. $h_1 -= h_0; h += f_0 \times g_1; h_1 -= h_0$
2. $g_0 -= g_1; h += f_0 \times g_0; g_0 += g_1$
3. $h_1 += f_1 \times g_1 \pmod{x^{n/2}}$ (tail recursive call)

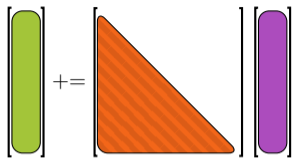
▶ Time: $M(n)$

▶ Space: $O(1)$ (+ space of full product)

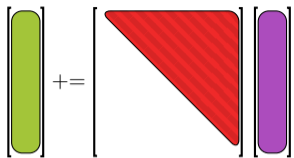
Remark

- ▶ FFT-based full product with $O(1)$ pointers

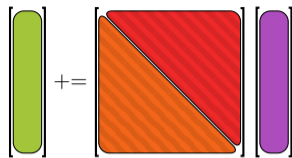
Cumulative short and middle products



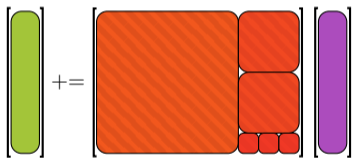
$$h += f \cdot g \bmod x^n$$



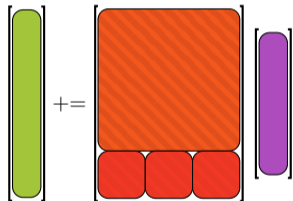
$$h += f \cdot g \operatorname{div} x^n$$



$$h += \operatorname{MIDPROD}(f, g)$$



$$h += \operatorname{MIDPROD}(f, g)$$

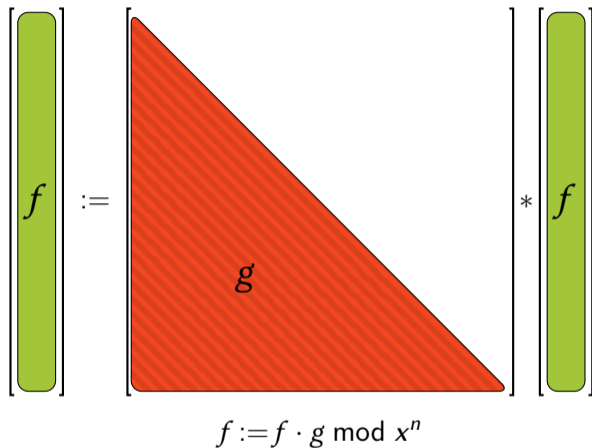


$$h += \operatorname{MIDPROD}(f, g)$$

- ▶ Time: $O(M(n))$
- ▶ Space: $O(1)$

or $O(\frac{n}{m}M(m))$

In-place short product

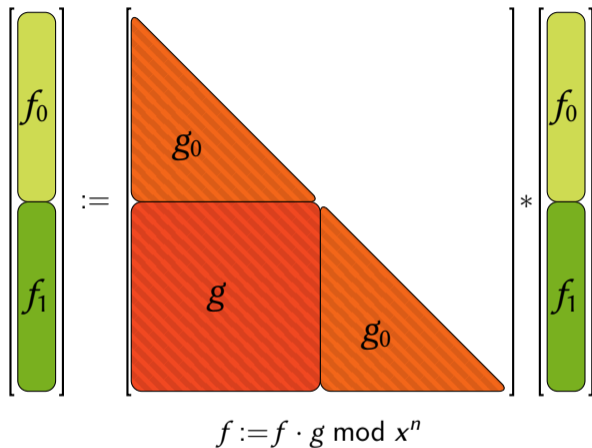


Inputs: f and g

Output: $f := f \cdot g \bmod x^n$

1. $f_1 *= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 += \text{MIDPROD}(f_0, g)$
3. $f_0 *= g_0 \bmod x^{n/2}$ (recursive call)

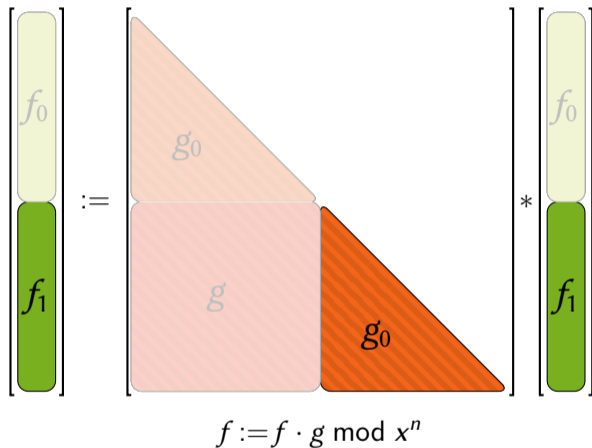
In-place short product



Inputs: f and g
Output: $f := f \cdot g \bmod x^n$

1. $f_1 *= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 += \text{MIDPROD}(f_0, g)$
3. $f_0 *= g_0 \bmod x^{n/2}$ (recursive call)

In-place short product

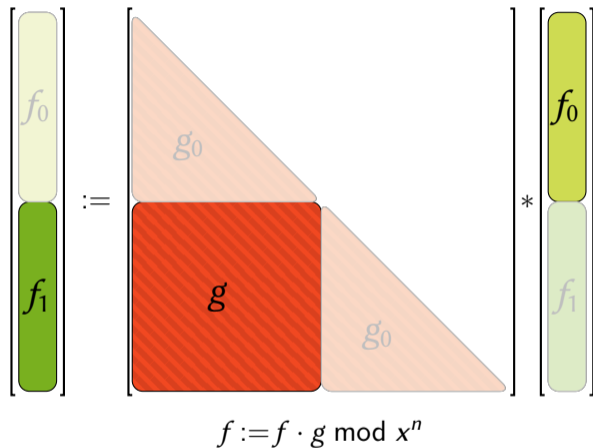


Inputs: f and g

Output: $f := f \cdot g \bmod x^n$

1. $f_1 := g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 += \text{MIDPROD}(f_0, g)$
3. $f_0 := g_0 \bmod x^{n/2}$ (recursive call)

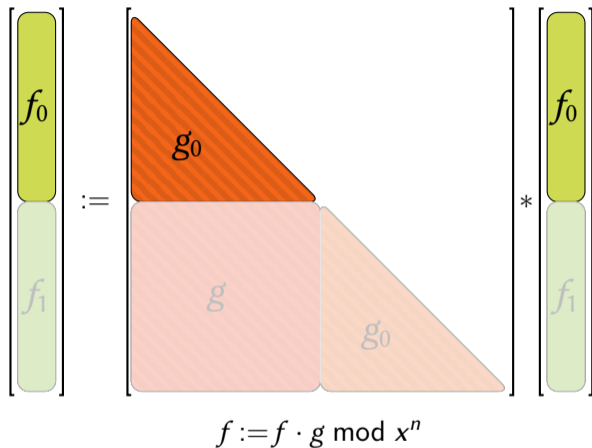
In-place short product



Inputs: f and g
Output: $f := f \cdot g \bmod x^n$

1. $f_1 *= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 += \text{MIDPROD}(f_0, g)$
3. $f_0 *= g_0 \bmod x^{n/2}$ (recursive call)

In-place short product

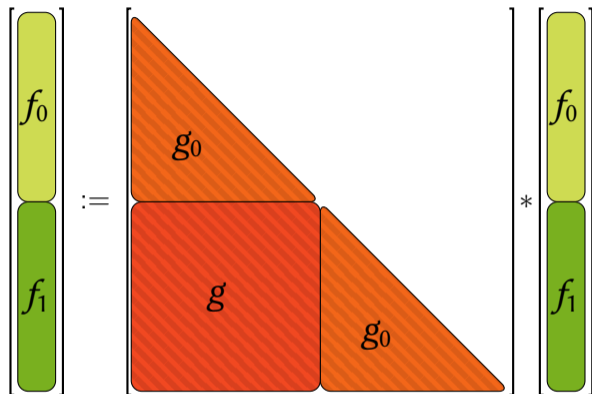


Inputs: f and g

Output: $f := f \cdot g \bmod x^n$

1. $f_1 *= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 += \text{MIDPROD}(f_0, g)$
3. $f_0 *= g_0 \bmod x^{n/2}$ (recursive call)

In-place short product



$$f := f \cdot g \bmod x^n$$

Inputs: f and g
Output: $f := f \cdot g \bmod x^n$

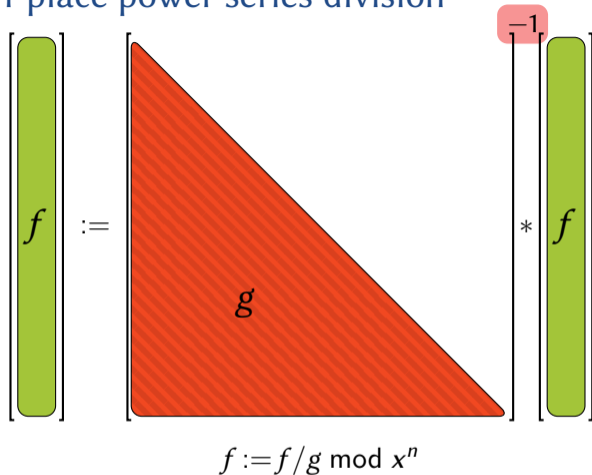
1. $f_1 := g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 += \text{MIDPROD}(f_0, g)$
3. $f_0 := g_0 \bmod x^{n/2}$ (recursive call)

▶ Time: $O(M(n) \log n)^*$

▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)

* $O(M(n))$ if $M(n) = \Omega(n^{1+\epsilon})$

In-place power series division

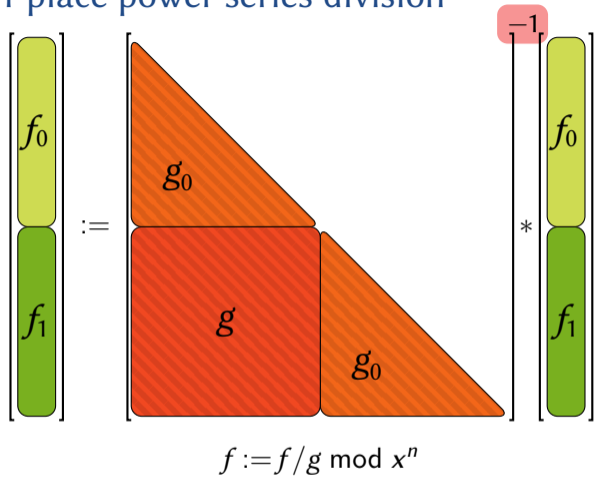


Inputs: f and g

Output: $f := f/g \bmod x^n$

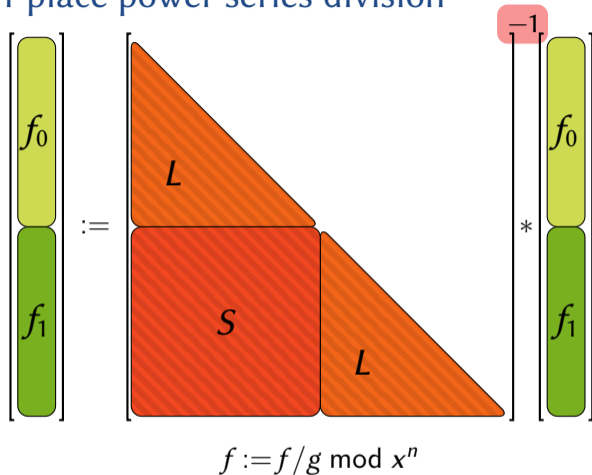
1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 -= \text{MIDPROD}(f_0, g)$
3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

In-place power series division



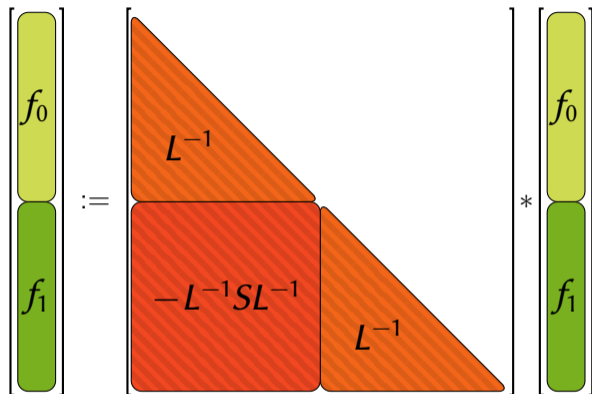
- Inputs:** f and g
Output: $f := f/g \bmod x^n$
1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
 2. $f_1 -= \text{MIDPROD}(f_0, g)$
 3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

In-place power series division



- Inputs:** f and g
Output: $f := f/g \bmod x^n$
1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
 2. $f_1 -= \text{MIDPROD}(f_0, g)$
 3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

In-place power series division

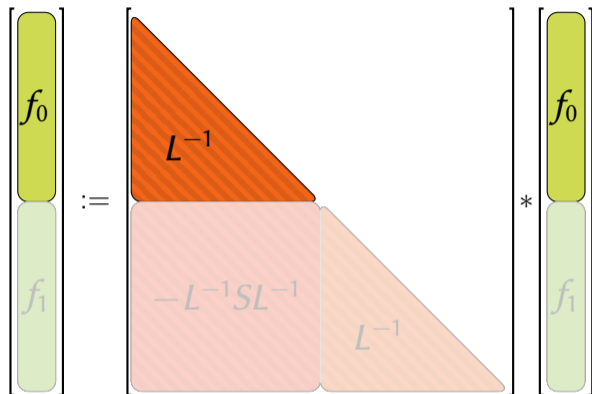


$$f := f/g \bmod x^n$$

Inputs: f and g
Output: $f := f/g \bmod x^n$

1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 -= \text{MIDPROD}(f_0, g)$
3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

In-place power series division

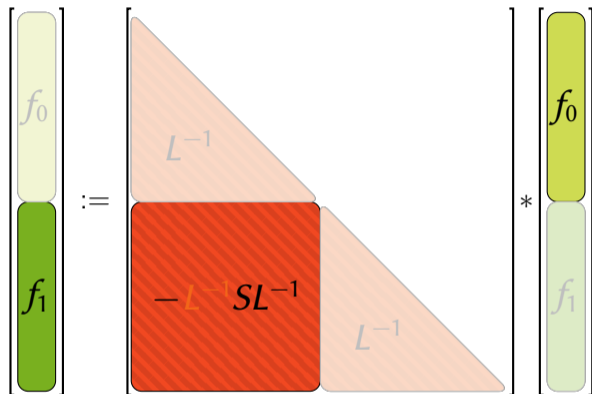


$$f := f/g \bmod x^n$$

Inputs: f and g
Output: $f := f/g \bmod x^n$

1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 -= \text{MIDPROD}(f_0, g)$
3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

In-place power series division

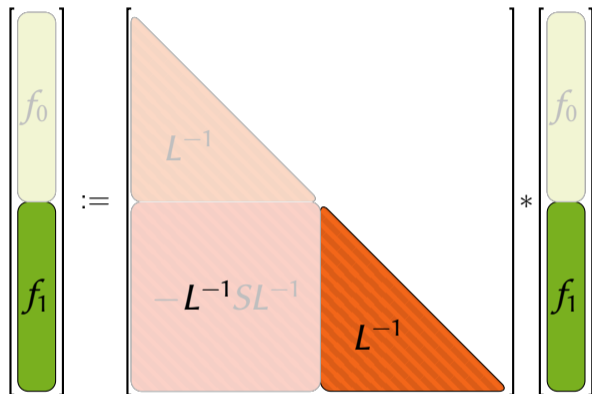


$$f := f/g \bmod x^n$$

Inputs: f and g
Output: $f := f/g \bmod x^n$

1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 -= \text{MIDPROD}(f_0, g)$
3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

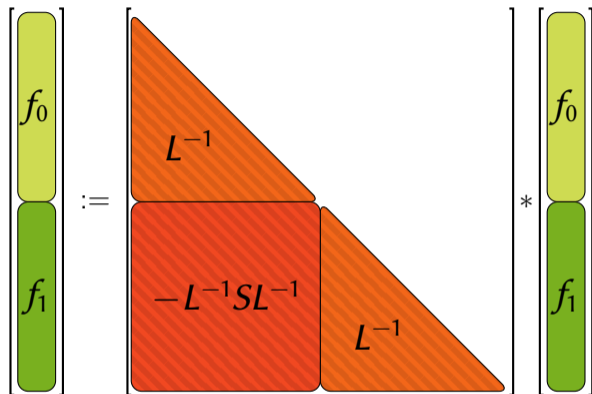
In-place power series division



$$f := f/g \bmod x^n$$

- Inputs:** f and g
Output: $f := f/g \bmod x^n$
1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
 2. $f_1 -= \text{MIDPROD}(f_0, g)$
 3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

In-place power series division



$$f := f/g \bmod x^n$$

Inputs: f and g
Output: $f := f/g \bmod x^n$

1. $f_0 /= g_0 \bmod x^{n/2}$ (recursive call)
2. $f_1 -= \text{MIDPROD}(f_0, g)$
3. $f_1 /= g_0 \bmod x^{n/2}$ (recursive call)

- ▶ Time: $O(M(n) \log n)^*$
- ▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)

* $O(M(n))$ if $M(n) = \Omega(n^{1+\epsilon})$

Back to fast Euclidean division

$$1. \begin{bmatrix} \bar{q} \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \bar{f}_1 \end{bmatrix} \quad 2. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$$

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} q \end{bmatrix}$$

Quotient and remainder: $(q, r) := (f \operatorname{div} g, f \operatorname{mod} g)$

[Giorgi, G., Roche (2020)]

- ▶ Time: $O(M(n))$
- ▶ Space: $O(1)$ alg. reg. + $O(1)$ pointers
- ▶ Model: ro/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \bar{q} \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \bar{f}_1 \end{bmatrix} \quad 2. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$$

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} q \end{bmatrix}$$

Remainder only: $r := f \bmod g$

[Giorgi, G., Roche (2020)]

- ▶ Time: $O(M(n))$
- ▶ Space: $O(n)$ alg. reg. + $O(1)$ pointers
- ▶ Model: ro/rw

Back to fast Euclidean division

1. $\begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$ 2. $\begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$

3. $\begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} q \end{bmatrix}$

Remainder only: $r := f \bmod g$

[Giorgi, G., Roche (2020)]

- ▶ Time: $O(M(n))$
- ▶ Space: $O(n)$ alg. reg. + $O(1)$ pointers
- ▶ Model: ro/rw

Back to fast Euclidean division

1. $\begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$ 2. $\begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$

3. $\begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$

Remainder only: $r := f \bmod g$

[Giorgi, G., Roche (2020)]

- ▶ Time: $O(M(n))$
- ▶ Space: $O(n)$ alg. reg. + $O(1)$ pointers
- ▶ Model: ro/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

$$2. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$$

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha, \uparrow} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

Remainder only: $r := f \bmod g$

[Giorgi, G., Roche (2020)]

- ▶ Time: $O(M(n))$
- ▶ Space: $O(n)$ alg. reg. + $O(1)$ pointers
- ▶ Model: ro/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

$$2. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$$

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

Remainder only: $r := f \bmod g$

[Dumas, G. (2024)]

- ▶ Time: $O(M(n)\log(n))^*$ * $O(M(m))$ if $M(m) = \Omega(m^{1+\epsilon})$
- ▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)
- ▶ Model: rw/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

$$2. r \ += f_0$$

$$3. r \ -= \begin{bmatrix} \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

Remainder only: $r \ += f \bmod g$ (cumulative)

[Dumas, G. (2024)]

- ▶ Time: $O(M(n)\log(n))^*$ * $O(M(m))$ if $M(m) = \Omega(m^{1+\epsilon})$
- ▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)
- ▶ Model: rw/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

$$2. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$$

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \uparrow} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

Remainder only: $r += f \bmod g$ (cumulative)

[Dumas, G. (2024)]

- ▶ Time: $O(M(n)\log(n))^*$ * $O(M(m))$ if $M(m) = \Omega(m^{1+\epsilon})$
- ▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)
- ▶ Model: rw/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \tau} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

$$2. r := f_0$$

$$3. f_0 := \begin{bmatrix} g \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \sigma_{\alpha, \tau} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

Remainder only: $r += f \bmod g$ (cumulative)

[Dumas, G. (2024)]

- ▶ Time: $O(M(n)\log(n))^*$ * $O(M(m))$ if $M(m) = \Omega(m^{1+\epsilon})$
- ▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)
- ▶ Model: rw/rw

Back to fast Euclidean division

$$1. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha, \uparrow} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

$$2. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} f_0 \end{bmatrix}$$

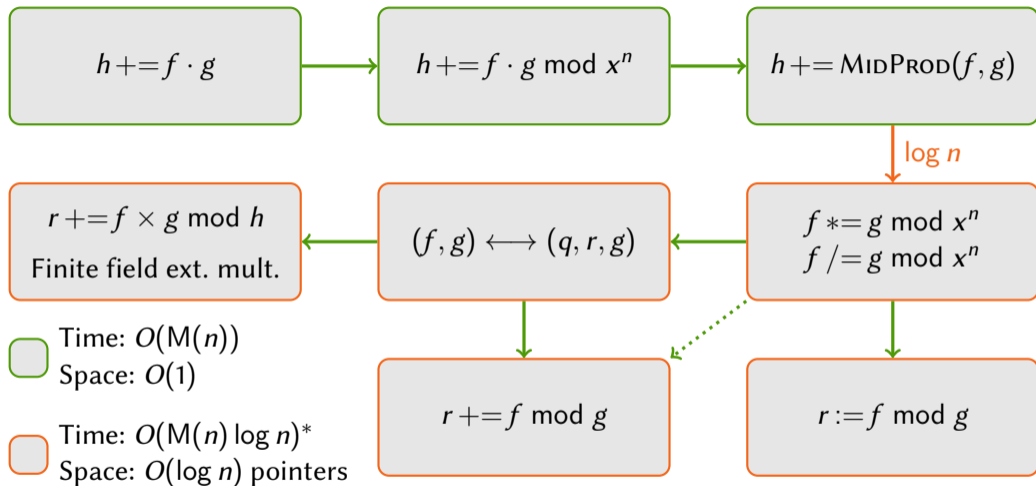
$$3. \begin{bmatrix} f_0 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha} \end{bmatrix} \times \begin{bmatrix} f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \sigma_{\alpha, \uparrow} \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \end{bmatrix}$$

In-place quotient and remainder: $(f, g) \leftrightarrow (q, r, g)$ (reversible) [Dumas, G. (2024)]

- ▶ Time: $O(M(n)\log(n))^*$ * $O(M(m))$ if $M(m) = \Omega(m^{1+\epsilon})$
- ▶ Space: $O(1)$ alg. reg. + $O(\log n)$ pointers (call stack)
- ▶ Model: rw/rw

Summary of reductions in the rw/rw model



* $O(M(n))$ if $M(n) = \Omega(n^{1+\epsilon})$

Generalization and automatization

Bilinear computations

$$\vec{w} += C \cdot \left[(A \cdot \vec{u}) \odot (B \cdot \vec{v}) \right]$$

- ▶ Includes many polynomial computations
- ▶ Matrix multiplication, system solving, ...

Our result (informally)

[Dumas, G. (2024)]

Given a *bilinear algorithm* (A, B, C) , implement it with constant algebraic space and same time complexity in the rw/rw model

Consequences

- ▶ Many polynomial algorithms can be obtained automatically
 - ▶ Caveat: recursive algorithm with *a priori* a call stack of $O(\log n)$ pointers
- ▶ Strassen's matrix multiplication algorithm in constant algebraic space
- ▶ Fast in-place linear algebra
 - ▶ triangular matrix multiplication and system solving, LU decomposition, ...

Conclusion

Time- and space-efficient polynomial computations

- ▶ Many fast algorithms can be made
 - ▶ (almost) constant-space
 - ▶ with (almost) the same asymptotic time complexity
- ▶ Requires to move away from the classical ro/wo model

Automatic derivations

- ▶ Make bilinear algorithms constant-space automatically
- ▶ Application to matrix computations

And in practice?

- ▶ Some implementations:
 - ▶ fewer allocations compensate the slight increase in time complexity
 - ▶ competitive timings

Main results

		model	time	alg. sp.	# pointers
product	$h := f \times g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
	$h += f \times g$	rw/rw	$O(M(n))$	$O(1)$	$O(\log n)$
	$h += f \times g$ (with FFT)	rw/rw	$O(M(n))$	$O(1)$	$O(1)$
short product	$h := f \times g \bmod x^n$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
	$h += f \times g \bmod x^n$	rw/rw	$O(M(n))$	$O(1)$	$O(1)$
	$f *= g \bmod x^n$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$
middle product	$h := \text{MIDPROD}(f, g)$	ro/rw	$O(M(n) \log n)$	$O(1)$	$O(1)$
	$h += \text{MIDPROD}(f, g)$	rw/rw	$O(M(n))$	$O(1)$	$O(1)$
power series division	$h := f/g \bmod x^n$	ro/rw	$O(M(n) \log n)$	$O(1)$	$O(1)$
	$f /= g \bmod x^n$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$
Euclidean division	$(q, r) := \text{QUOREM}(f, g)$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
	$r += f \bmod g$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$
	$(f, g) \leftrightarrow (q, r, g)$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$
	$r += f \times g \bmod h$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$
eval./interp.	$f \leftrightarrow (f(\alpha_i))_i$	ro/rw	$O(M(n) \log n)$	$O(1)$	$O(1)$

Open problems

Polynomial computations

- ▶ Other operations: gcd, power series composition, ...
- ▶ Non-optimal complexities: remove log factors, remove call stacks, ...
- ▶ From rw/rw to ro/rw ?
- ▶ Extension to integer arithmetic

Models

- ▶ Right models for time-space complexity of functions?
- ▶ Comparisons between the models
- ▶ Links with catalytic computations
- ▶ Time-space lower bounds in models ro/rw and rw/rw ?

One of my favorite open problems

Can you replace   by  without extra space?

[Roche'09]

Open problems

Polynomial computations

- ▶ Other operations: gcd, power series composition, ...
- ▶ Non-optimal complexities: remove log factors, remove call stacks, ...
- ▶ From rw/rw to ro/rw ?
- ▶ Extension to integer arithmetic

Models

- ▶ Right models for time-space complexity of functions?
- ▶ Comparisons between the models
- ▶ Links with catalytic computations
- ▶ Time-space lower bounds in models ro/rw and rw/rw ?

One of my favorite open problems

Can you replace   by  without extra space?

[Roche'09]

Thank you!