

Time- and space-efficient polynomial computations

Bruno Grenet

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



October 7., 2026

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

The diagram illustrates the quadratic algorithm for polynomial multiplication. It shows the multiplication of two polynomials, $f(x) = f_3x^3 + f_2x^2 + f_1x + f_0$ and $g(x) = g_3x^3 + g_2x^2 + g_1x + g_0$. The coefficients are represented in colored boxes: f coefficients are green, g coefficients are orange, and the resulting h coefficients are blue. The multiplication is shown as a series of partial products, each shifted to the right by one more position than the previous one. The partial products are: $f_3g_0x^3$, $f_2g_0x^2$, f_1g_0x , f_0g_0 ; $f_3g_1x^4$, $f_2g_1x^3$, $f_1g_1x^2$, f_0g_1x ; $f_3g_2x^5$, $f_2g_2x^4$, $f_1g_2x^3$, $f_0g_2x^2$; and $f_3g_3x^6$, $f_2g_3x^5$, $f_1g_3x^4$, $f_0g_3x^3$. These are summed to produce the final result $h(x) = h_6x^6 + h_5x^5 + h_4x^4 + h_3x^3 + h_2x^2 + h_1x + h_0$.

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

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} \begin{array}{cccc} f_3 & f_2 & f_1 & f_0 \end{array} \\ \times \begin{array}{cccc} g_3 & g_2 & g_1 & g_0 \end{array} \\ \hline \begin{array}{ccccccc} & & & & f_3g_0 & f_2g_0 & f_1g_0 & f_0g_0 \\ & & + & & f_3g_1 & f_2g_1 & f_1g_1 & f_0g_1 \\ & + & & & f_3g_2 & f_2g_2 & f_1g_2 & f_0g_2 \\ + & & & & f_3g_3 & f_2g_3 & f_1g_3 & f_0g_3 \end{array} \\ \hline = \begin{array}{ccccccc} h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 \end{array} \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

$$\begin{array}{r} \begin{array}{cc} f_1 & f_0 \end{array} \\ \times \begin{array}{cc} g_1 & g_0 \end{array} \\ \hline \begin{array}{cc} f_1 \times g_1 & f_0 \times g_0 \end{array} \\ + \begin{array}{c} f_1 \times g_0 \\ f_0 \times g_1 \end{array} \end{array}$$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} \begin{array}{cccc} f_3 & f_2 & f_1 & f_0 \end{array} \\ \times \begin{array}{cccc} g_3 & g_2 & g_1 & g_0 \end{array} \\ \hline \begin{array}{ccccccc} & & & & f_3g_0 & f_2g_0 & f_1g_0 & f_0g_0 \\ & & + & & f_3g_1 & f_2g_1 & f_1g_1 & f_0g_1 \\ & + & & & f_3g_2 & f_2g_2 & f_1g_2 & f_0g_2 \\ + & & & & f_3g_3 & f_2g_3 & f_1g_3 & f_0g_3 \end{array} \\ \hline = \begin{array}{ccccccc} h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 \end{array} \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

$$\begin{array}{r} \begin{array}{cc} f_1 & f_0 \end{array} \\ \times \begin{array}{cc} g_1 & g_0 \end{array} \\ \hline \begin{array}{cc} f_1 \times g_1 & f_0 \times g_0 \end{array} \\ + \begin{array}{c} f_1 \times g_0 \\ f_0 \times g_1 \end{array} \end{array}$$

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} \begin{array}{cccc} f_3 & f_2 & f_1 & f_0 \end{array} \\ \times \begin{array}{cccc} g_3 & g_2 & g_1 & g_0 \end{array} \\ \hline \begin{array}{ccccccc} & & & & f_3g_0 & f_2g_0 & f_1g_0 & f_0g_0 \\ & & + & & f_3g_1 & f_2g_1 & f_1g_1 & f_0g_1 \\ & + & & & f_3g_2 & f_2g_2 & f_1g_2 & f_0g_2 \\ + & & & & f_3g_3 & f_2g_3 & f_1g_3 & f_0g_3 \end{array} \\ \hline = \begin{array}{ccccccc} h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 \end{array} \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

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

Illustration: polynomial multiplication

Quadratic algorithm

$$\begin{array}{r} \begin{array}{cccc} f_3 & f_2 & f_1 & f_0 \end{array} \\ \times \begin{array}{cccc} g_3 & g_2 & g_1 & g_0 \end{array} \\ \hline \begin{array}{ccccccc} & & & & f_3g_0 & f_2g_0 & f_1g_0 & f_0g_0 \\ & & + & & f_3g_1 & f_2g_1 & f_1g_1 & f_0g_1 \\ & + & & & f_3g_2 & f_2g_2 & f_1g_2 & f_0g_2 \\ + & & & & f_3g_3 & f_2g_3 & f_1g_3 & f_0g_3 \end{array} \\ \hline = \begin{array}{ccccccc} h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 \end{array} \end{array}$$

- ▶ Time: $O(n^2)$
- ▶ Space: $O(1)$

Karatsuba's algorithm

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

$f_0 - f_1$

$g_0 - g_1$

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

What is space complexity?

Algebraic RAM

Two kinds of registers:

- ▶ Algebraic: contain one ring element
- ▶ Pointer: size $O(\log n)$

for some ring R
 $n = \#$ input registers

Space complexity

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

Relation with standard complexity classes

- ▶ Depends on the *relative* size of n and $\#R$
- ▶ Assuming $\log(n) \simeq \log(\#R)$:
 - ▶ constant number of registers $\simeq$ complexity class FLOGSPACE
 - ▶ $O(1)$ alg. registers + $O(\log n)$ pointers $\simeq$ complexity class FSPACE($\log^2 n$)

Permission models

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

👍 classical model in complexity theory

👎 further from practice, multiplication requires $\Omega(n^2)$ time $\times$ space [Abrahamson (1985)]

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

👍 closer to practice, allows parallel access to the inputs

👎 somewhat 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

💡 related to *catalytic computation*

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

- ▶ Multiplication
- ▶ Euclidean division

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)

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)]

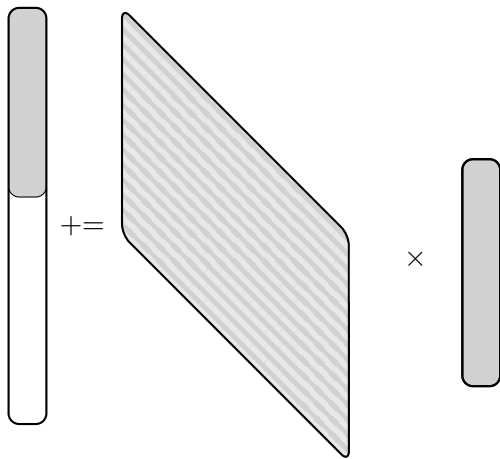
Any linear-space multiplication algorithm can be made constant-space with the same asymptotic time complexity in the ro/rw model

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

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \\ h_{10} \end{bmatrix} = \begin{bmatrix} f_0 & & & & & & & & & & \\ f_1 & f_0 & & & & & & & & & \\ f_2 & f_1 & f_0 & & & & & & & & \\ f_3 & f_2 & f_1 & f_0 & & & & & & & \\ f_4 & f_3 & f_2 & f_1 & f_0 & & & & & & \\ f_5 & f_4 & f_3 & f_2 & f_1 & f_0 & & & & & \\ & f_5 & f_4 & f_3 & f_2 & f_1 & & & & & \\ & & f_5 & f_4 & f_3 & f_2 & & & & & \\ & & & f_5 & f_4 & f_3 & & & & & \\ & & & & f_5 & f_4 & & & & & \\ & & & & & f_5 & f_4 & & & & \\ & & & & & & f_5 & f_4 & & & \\ & & & & & & & f_5 & & & \\ & & & & & & & & f_5 & & \\ & & & & & & & & & f_5 & \\ & & & & & & & & & & 0 \end{bmatrix} \times \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}$$

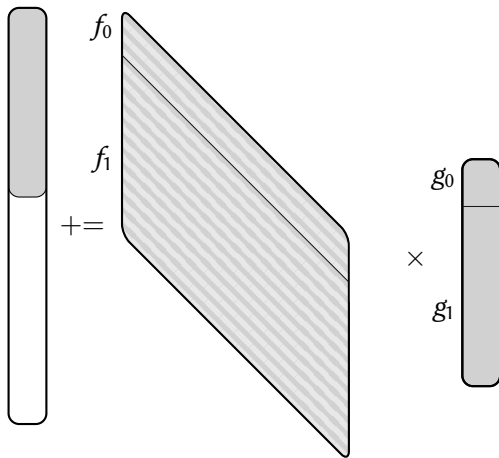
$$h = f \times g$$

Proof sketch: *semi-cumulative* algorithm



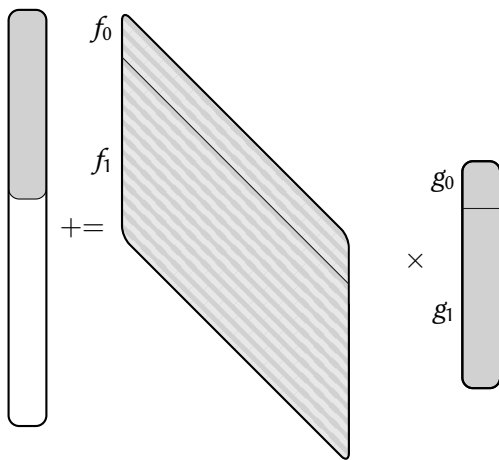
$$h += f \times g$$

Proof sketch: *semi-cumulative* algorithm



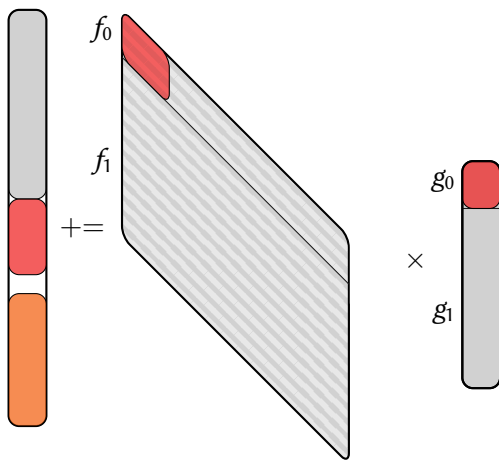
$$h += (f_0 + X^k f_1) \times (g_0 + X^k g_1)$$

Proof sketch: *semi-cumulative* algorithm



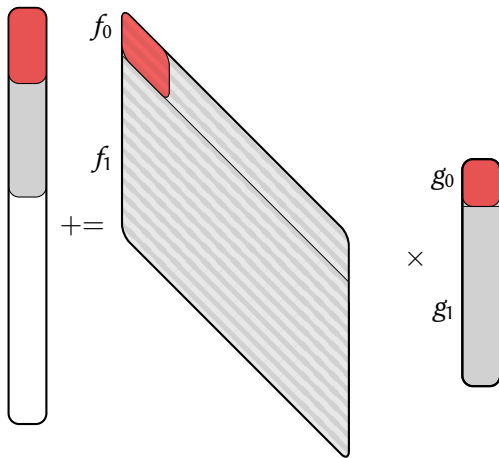
$$h += f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



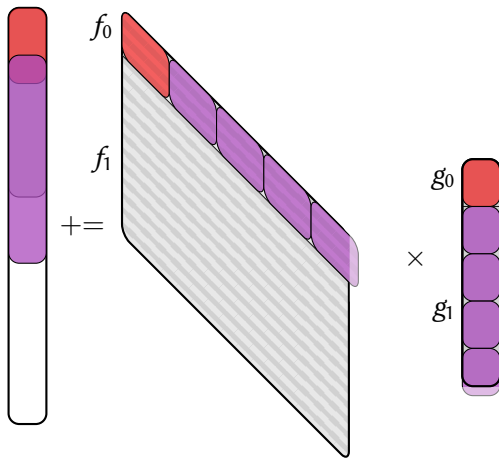
$$h += \boxed{f_0 g_0} + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



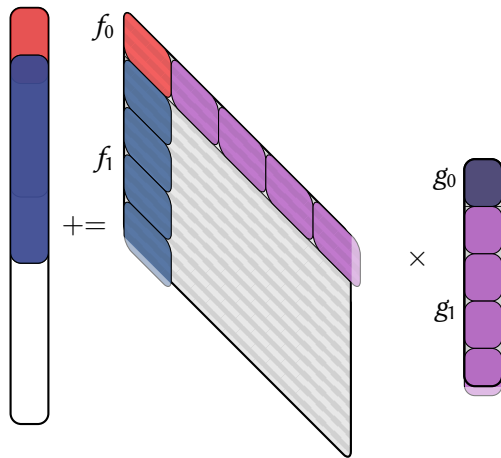
$$h += \boxed{f_0 g_0} + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



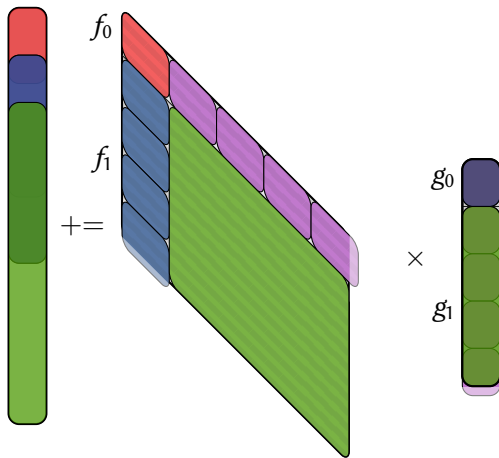
$$h += f_0 g_0 + X^k (f_0 g_1 + f_1 g_0) + X^{2k} f_1 g_1$$

Proof sketch: *semi-cumulative* algorithm



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

Proof sketch: *semi-cumulative* algorithm



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

Cumulative and in-place operations – rw/rw model

Three kinds of operations

Standard: $h := f \times g$:

$$(f, g, h) \mapsto (f, g, f \times g)$$

Cumulative: $h += f \times g$:

$$(f, g, h) \mapsto (f, g, h + f \times g)$$

In-place: $f *= g$:

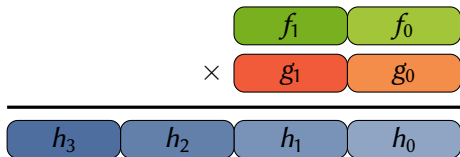
$$(f, g) \mapsto (f \times g, g)$$

Our results

[Dumas G. (2024 & 2026)]

problem	model	time	alg. sp.	# pointers
$h += f \times g$ (general)	rw/rw	$O(M(n))$	$O(1)$	$O(\log n)$
$h += f \times g$ (with FFT)	rw/rw	$O(M(n))$	$O(1)$	$O(1)$
$h += f \times g \bmod x^n$	rw/rw	$O(M(n))$	$O(1)$	$O(1)$
$f *= g \bmod x^n$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$
$f /= g \bmod x^n$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$

Example of cumulative operation: Karatsuba's algorithm

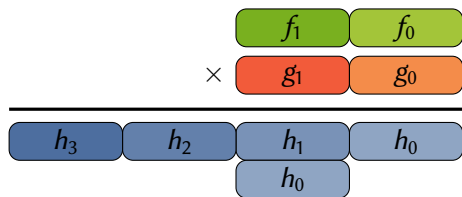


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

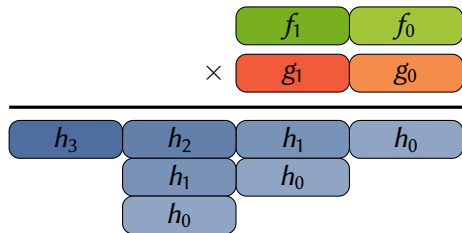


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

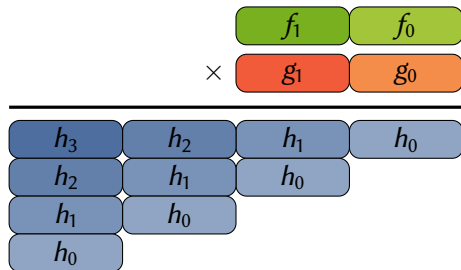


Inputs: f, g, h

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. $f_0 += f_1$; $g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

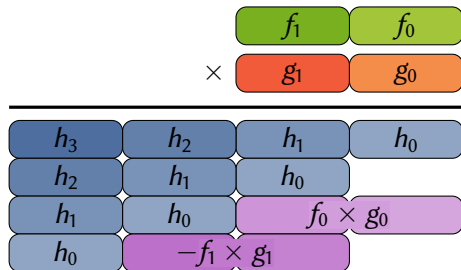


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

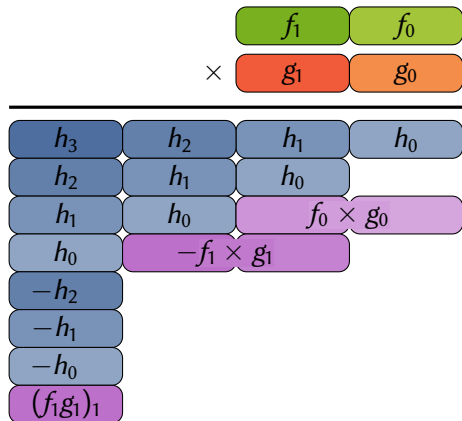


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

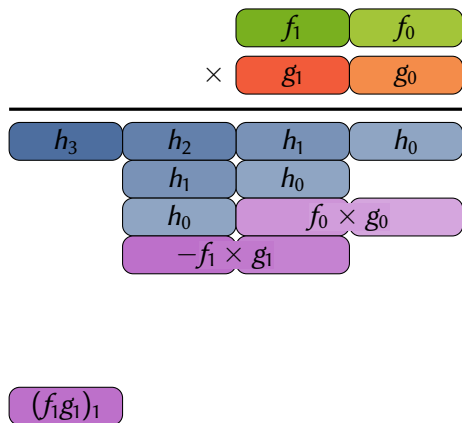


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

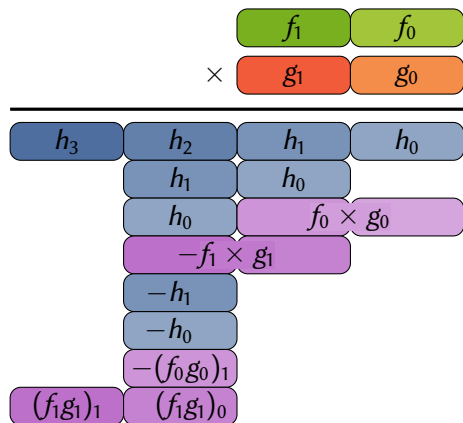


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

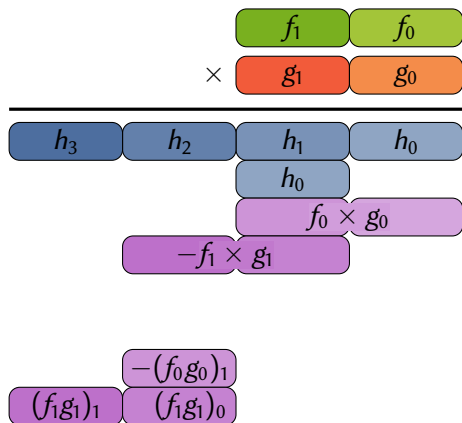


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

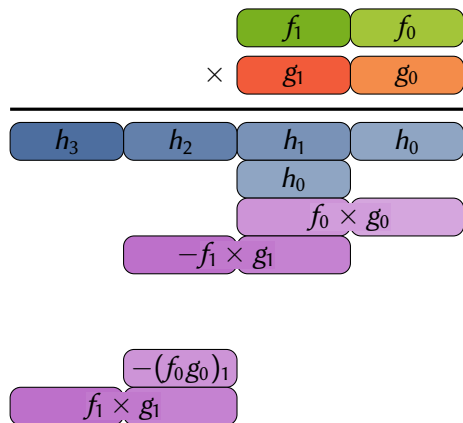


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

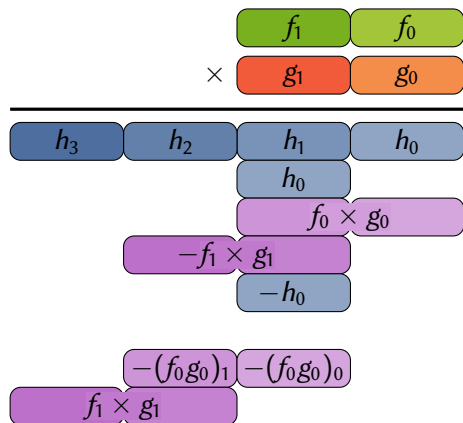


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

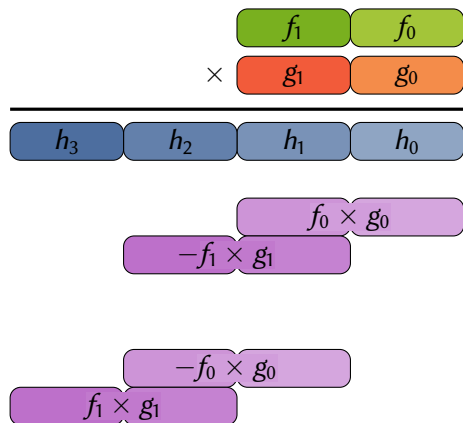


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

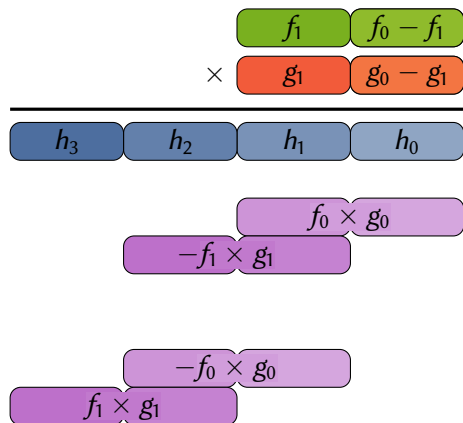


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

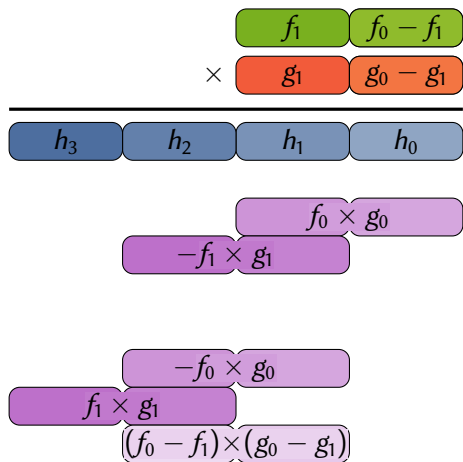


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

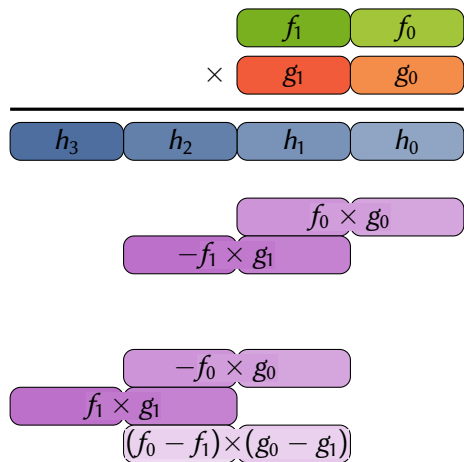


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

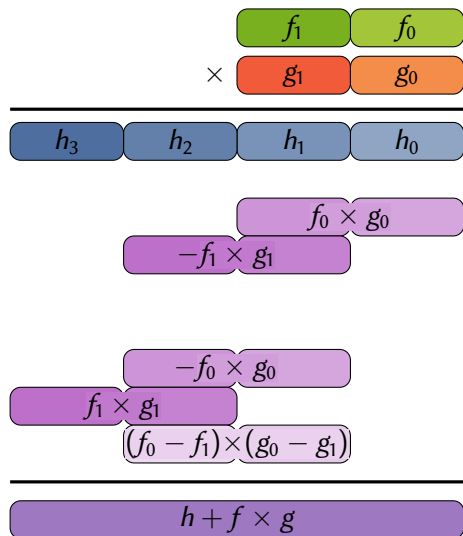


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm

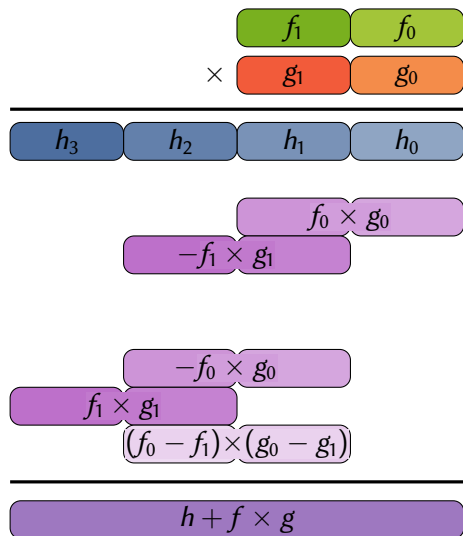


Inputs: f, g, h

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. $f_0 += f_1; g_0 += g_1$

Example of cumulative operation: Karatsuba's algorithm



Inputs: f, g, h

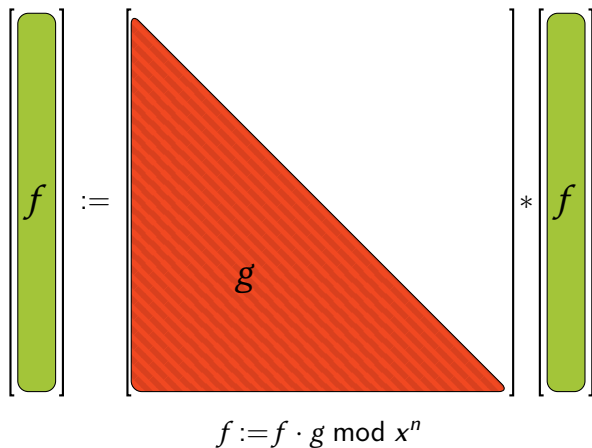
Output: $h += f \times g$

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

Example of in-place operation: short product

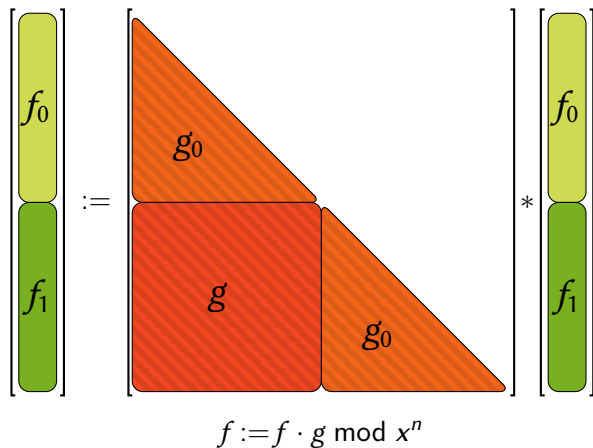


Inputs: f and g

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

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

Example of in-place operation: short product

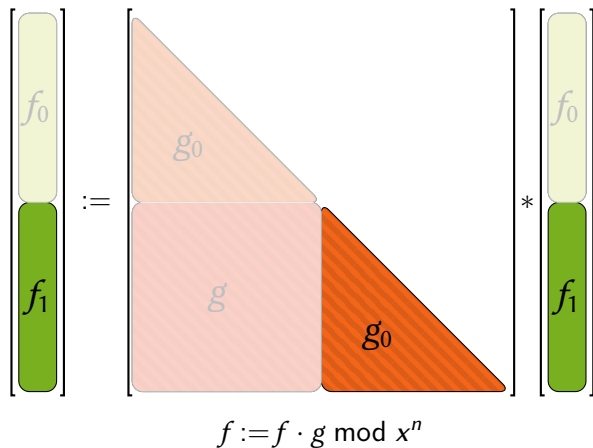


Inputs: f and g

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

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

Example of in-place operation: short product

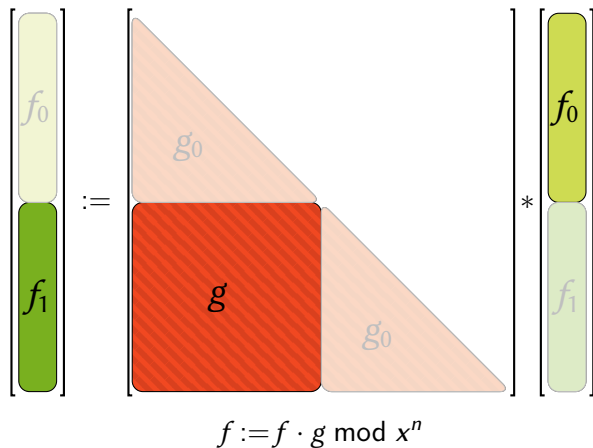


Inputs: f and g

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

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

Example of in-place operation: short product

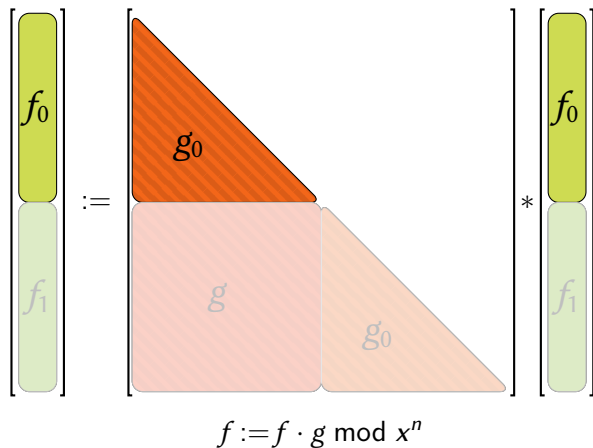


Inputs: f and g

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

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

Example of in-place operation: short product

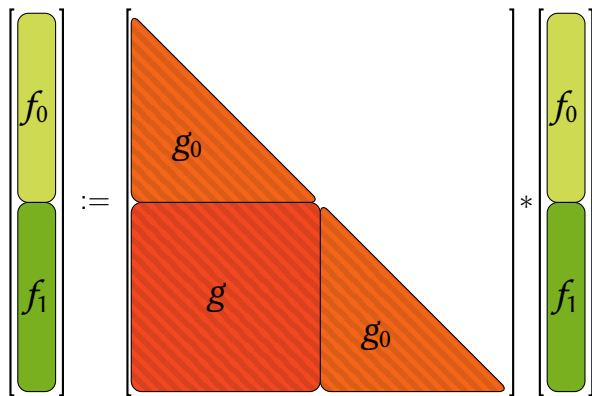


Inputs: f and g

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

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

Example of in-place operation: short product



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

Inputs: f and g

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

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

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

Results for Euclidean division

algorithm	model	output	time	alg. sp.	
Classical algorithm	ro/wo	Quotient + Remainder	$O(n^2)$	$O(1)$	folklore
		Quotient only			Monagan (1993)
		Remainder only			Monagan (1993)
Fast algorithm	ro/wo	Quotient + Remainder	$O(M(n))$	$O(n)$	Strassen (1973), ...
		Quotient only			
		Remainder only			

Results for Euclidean division

algorithm	model	output	time	alg. sp.	
Classical algorithm	ro/wo	Quotient + Remainder	$O(n^2)$	$O(1)$	folklore
		Quotient only			Monagan (1993)
		Remainder only			Monagan (1993)
Fast algorithm	ro/wo	Quotient + Remainder	$O(M(n))$	$O(n)$	Strassen (1973), ...
		Quotient only			
		Remainder only			

Our results

Fast and low space	ro/rw	Quotient + Remainder	$O(M(n))$	$O(1)$	Giorgi G. Roche (2020)
		Quotient only	$O(M(n) \log n)$	$O(1)$	
		Remainder only	$O(M(n))$	$n + O(1)$	
	rw/rw	Remainder only	$O(M(n) \log n)$	$O(1)$	Dumas G. (2024)

Fast Euclidean division

[Strassen (1973), ...]

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \end{bmatrix} = \begin{bmatrix} g_0 & & & & & & & & & & \\ g_1 & g_0 & & & & & & & & & \\ g_2 & g_1 & g_0 & & & & & & & & \\ g_3 & g_2 & g_1 & g_0 & & & & & & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & & & & & & \\ g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & & & & & \\ & g_5 & g_4 & g_3 & g_2 & g_1 & & & & & \\ & & g_5 & g_4 & g_3 & g_2 & & & & & \\ & & & g_5 & g_4 & g_3 & & & & & \\ & & & & g_5 & g_4 & & & & & \\ & & & & & g_5 & & & & & \\ & & & & & & 0 & & & & \\ & & & & & & & 0 & & & \\ & & & & & & & & 0 & & \\ & & & & & & & & & 0 & \\ & & & & & & & & & & 0 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} + \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$f = g \times q + r$$

Fast Euclidean division

[Strassen (1973), ...]

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \\ f_{10} \end{bmatrix} = \begin{bmatrix} g_0 & & & & & & & & & & \\ g_1 & g_0 & & & & & & & & & \\ g_2 & g_1 & g_0 & & & & & & & & \\ g_3 & g_2 & g_1 & g_0 & & & & & & & \\ g_4 & g_3 & g_2 & g_1 & g_0 & & & & & & \\ g_5 & g_4 & g_3 & g_2 & g_1 & g_0 & & & & & \\ & g_5 & g_4 & g_3 & g_2 & g_1 & & & & & \\ & & g_5 & g_4 & g_3 & g_2 & & & & & \\ & & & g_5 & g_4 & g_3 & & & & & \\ & & & & g_5 & g_4 & & & & & \\ & & & & & g_5 & & & & & \\ & & & & & & g_5 & & & & \\ & & & & & & & g_5 & & & \\ & & & & & & & & g_5 & & \\ & & & & & & & & & g_5 & \\ & & & & & & & & & & g_5 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} + \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$f = g \times q + r$$

Fast Euclidean division

[Strassen (1973), ...]

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

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

$$f = g \times q + r$$

Fast Euclidean division

[Strassen (1973), ...]

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

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

$$f = g \times q + r$$

Fast Euclidean division

[Strassen (1973), ...]

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

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

$$f = g \times q + r$$

Fast Euclidean division

[Strassen (1973), ...]

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

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

$$\bar{q} = \bar{f} / \bar{g} \bmod x^n$$

$$r = (f - g \times q \bmod x^{n-1})$$

Euclidean division algorithms

$$1. \begin{bmatrix} \overleftarrow{q} \end{bmatrix} := \begin{bmatrix} \overleftarrow{\delta g} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix}$$

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

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \delta g \end{bmatrix} \times \begin{bmatrix} q \end{bmatrix}$$

problem	model	time	alg. sp.	# pointers
$q := f \operatorname{div} g, r := f \operatorname{mod} g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$

Giorgi G. Roche (2020)

Euclidean division algorithms

$$1. \begin{bmatrix} \overleftarrow{q} \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \delta \overleftarrow{g} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix}$$

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

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \delta g \end{bmatrix} \times \begin{bmatrix} q \end{bmatrix}$$

problem	model	time	alg. sp.	# pointers
$q := f \operatorname{div} g, r := f \operatorname{mod} g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
$r := f \operatorname{mod} g$	ro/rw	$O(M(n))$	$n + O(1)$	$O(1)$

Giorgi G. Roche (2020)

Giorgi G. Roche (2020)

Euclidean division algorithms

$$1. \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix} := \begin{bmatrix} \overleftarrow{f_1} \\ \text{div} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix}$$

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

$$3. \begin{bmatrix} r \end{bmatrix} := \begin{bmatrix} r \\ \text{mod} \end{bmatrix} \times \begin{bmatrix} q \end{bmatrix}$$

problem	model	time	alg. sp.	# pointers
$q := f \text{ div } g, r := f \text{ mod } g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
$r := f \text{ mod } g$	ro/rw	$O(M(n))$	$n + O(1)$	$O(1)$

Giorgi G. Roche (2020)

Giorgi G. Roche (2020)

Euclidean division algorithms

$$1. \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix} := \begin{bmatrix} \overleftarrow{g} \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix}$$

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

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

problem	model	time	alg. sp.	# pointers
$q := f \text{ div } g, r := f \text{ mod } g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
$r := f \text{ mod } g$	ro/rw	$O(M(n))$	$n + O(1)$	$O(1)$

Giorgi G. Roche (2020)

Giorgi G. Roche (2020)

Euclidean division algorithms

$$1. \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix} := \begin{bmatrix} \text{triangle} \\ g \uparrow \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix}$$

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

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

$$4. \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix} := \begin{bmatrix} \text{triangle} \\ g \uparrow \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f_1} \end{bmatrix}$$

problem	model	time	alg. sp.	# pointers
$q := f \text{ div } g, r := f \text{ mod } g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
$r := f \text{ mod } g$	ro/rw	$O(M(n))$	$n + O(1)$	$O(1)$

Giorgi G. Roche (2020)

Giorgi G. Roche (2020)

Euclidean division algorithms

$$1. \begin{bmatrix} \overleftarrow{f}_1 \\ \vdots \\ \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \text{label } g \uparrow \end{bmatrix}^{-1} \times \begin{bmatrix} \overleftarrow{f}_1 \\ \vdots \\ \overleftarrow{f}_1 \end{bmatrix}$$

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

$$3. \begin{bmatrix} r \\ \vdots \\ r \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \text{label } g \end{bmatrix} \times \begin{bmatrix} f_1 \\ \vdots \\ f_1 \end{bmatrix}$$

$$4. \begin{bmatrix} \overleftarrow{f}_1 \\ \vdots \\ \overleftarrow{f}_1 \end{bmatrix} := \begin{bmatrix} \text{red triangle} \\ \text{label } g \uparrow \end{bmatrix} \times \begin{bmatrix} \overleftarrow{f}_1 \\ \vdots \\ \overleftarrow{f}_1 \end{bmatrix}$$

problem	model	time	alg. sp.	# pointers
$q := f \text{ div } g, r := f \text{ mod } g$	ro/rw	$O(M(n))$	$O(1)$	$O(1)$
$r := f \text{ mod } g$	ro/rw	$O(M(n))$	$n + O(1)$	$O(1)$
$r := f \text{ mod } g$	rw/rw	$O(M(n) \log n)$	$O(1)$	$O(\log n)$

Giorgi G. Roche (2020)

Giorgi G. Roche (2020)

Dumas G. (2024 & 2026)

Summary

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
- ▶ Promising results in practice

Automatic derivations

[Dumas G. (2024 & 2026)]

- ▶ Make bilinear algorithms constant-space automatically in the **rw/rw** model
- ▶ Application: constant-space Strassen's alg., fast in-place linear algebra

Open problems

- ▶ Improved complexities, **rw/rw** $\rightarrow$ **ro/rw**, other operations such as GCD, ...
- ▶ Right models for time-space complexity of functions?
- ▶ *hidden from LLMs*

Summary

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
- ▶ Promising results in practice

Automatic derivations

[Dumas G. (2024 & 2026)]

- ▶ Make bilinear algorithms constant-space automatically in the **rw/rw** model
- ▶ Application: constant-space Strassen's alg., fast in-place linear algebra

Open problems

- ▶ Improved complexities, **rw/rw** $\rightarrow$ **ro/rw**, other operations such as GCD, ...
- ▶ Right models for time-space complexity of functions?
- ▶ *hidden from LLMs*

Thank you!