Memory-efficient polynomial arithmetic

Pascal Giorgi[1] **Bruno Grenet**[1] Daniel S. Roche[2]

LIP, ÉNS de Lyon — 11 apr. 2019

[1] LIRMM, Université de Montpellier

[2] CS Department, US Naval Academy

Polynomial arithmetic

- Multiplication: $\mathrm{M}(n)$
  - Naïve: $2n^2 + 2n - 1$
  - Karatsuba: $< 6.5n^{\log_2 3}$
  - Toom-3: $< 18.75n^{\log_3 5}$
  - FFT-based: $4.5n \log n + O(n)$ or $O(n \log n \log \log n)$

Polynomial arithmetic

- Multiplication: $\mathrm{M}(n)$
  - Naïve: $2n^2 + 2n - 1$
  - Karatsuba: $< 6.5n^{\log_2 3}$
  - Toom-3: $< 18.75n^{\log_3 5}$
  - FFT-based: $4.5n \log n + O(n)$ or $O(n \log n \log \log n)$
- Other tasks:
  - Euclidean division: $5\mathrm{M}(n) + o(\mathrm{M}(n))$
  - GCD: $O(\mathrm{M}(n) \log n)$
  - Evaluation & interpolation: $O(\mathrm{M}(n) \log n)$
  - Power series computations: $O(\mathrm{M}(n))$ or $O(\mathrm{M}(n) \log n)$
  - ...

Polynomial arithmetic

- Multiplication: $\mathsf{M}(n)$
  - Naïve: $2n^2 + 2n - 1$
  - Karatsuba: $< 6.5n^{\log_2 3}$
  - Toom-3: $< 18.75n^{\log_3 5}$
  - FFT-based: $4.5n \log n + O(n)$ or $O(n \log n \log \log n)$
- Other tasks:
  - Euclidean division: $5\mathsf{M}(n) + o(\mathsf{M}(n))$
  - GCD: $O(\mathsf{M}(n) \log n)$
  - Evaluation & interpolation: $O(\mathsf{M}(n) \log n)$
  - Power series computations: $O(\mathsf{M}(n))$ or $O(\mathsf{M}(n) \log n)$
  - ...

What about space complexity?

Space complexity of polynomial arithmetic

- Quadratic multiplication algorithm: $O(1)$ [1]
- Karatsuba, Toom-3, FFT: $O(n)$
- Other tasks: often $O(n)$

1. Models to be defined later.

Space complexity of polynomial arithmetic

- Quadratic multiplication algorithm: $O(1)$ [1]
- Karatsuba, Toom-3, FFT: $O(n)$
- Other tasks: often $O(n)$

- Improvements on Karatsuba's algorithm:
  - Thomé (2002): $n + O(\log n)$
  - Roche (2009): $O(\log n)$

  $\rightarrow$ time complexity multiplied by a constant

1. Models to be defined later.

Space complexity of polynomial arithmetic

- Quadratic multiplication algorithm: $O(1)$ [1]
- Karatsuba, Toom-3, FFT: $O(n)$
- Other tasks: often $O(n)$

- Improvements on Karatsuba's algorithm:
  - Thomé (2002): $n + O(\log n)$
  - Roche (2009): $O(\log n)$

  $\rightarrow$ time complexity multiplied by a constant

- Improvements on FFT-based algorithms:
  - Roche (2009): $O(1)$ if $n = 2^k$
  - Harvey & Roche (2010): $O(1)$

  $\rightarrow$ time complexity multiplied by a constant

1. Models to be defined later.

Space-complexity models

Algebraic-RAM machine:

- $\rightarrow$ *Standard* registers of size $O(\log n)$
- $\rightarrow$ *Algebraic* registers containing one coefficient

Space-complexity models

Algebraic-RAM machine:

$\rightarrow$ *Standard* registers of size $O(\log n)$

$\rightarrow$ *Algebraic* registers containing one coefficient

- Read-only input / write-only output
  - (Close to) classical complexity theory
  - Lower bound $\Omega(n^2)$ on time $\times$ space for multiplication

Space-complexity models

Algebraic-RAM machine:

$\rightarrow$ *Standard* registers of size $O(\log n)$

$\rightarrow$ *Algebraic* registers containing one coefficient

- Read-only input / write-only output
  - (Close to) classical complexity theory
  - Lower bound $\Omega(n^2)$ on time $\times$ space for multiplication
- Read-only input / read-write output
  - Thomé (2002), Roche (2009) and Harvey & Roche (2010)
  - *Reasonable* from a programmer's viewpoint

Space-complexity models

Algebraic-RAM machine:

$\rightarrow$ *Standard* registers of size $O(\log n)$

$\rightarrow$ *Algebraic* registers containing one coefficient

- Read-only input / write-only output
  - (Close to) classical complexity theory
  - Lower bound $\Omega(n^2)$ on time $\times$ space for multiplication
- Read-only input / read-write output
  - Thomé (2002), Roche (2009) and Harvey & Roche (2010)
  - *Reasonable* from a programmer's viewpoint
- Read-write input and output
  - Too permissive in general
  - Variant: inputs must be restored at the end

Space-complexity models

Algebraic-RAM machine:

$\rightarrow$ *Standard* registers of size $O(\log n)$

$\rightarrow$ *Algebraic* registers containing one coefficient

- Read-only input / write-only output
  - (Close to) classical complexity theory
  - Lower bound $\Omega(n^2)$ on time $\times$ space for multiplication
- ✓ Read-only input / read-write output
  - Thomé (2002), Roche (2009) and Harvey & Roche (2010)
  - *Reasonable* from a programmer's viewpoint
- Read-write input and output
  - Too permissive in general
  - Variant: inputs must be restored at the end

Previous results

- Karatsuba's algorithm:
  - Divide-and-Conquer: $\left(f_0 + X^{\frac{n}{2}} f_1\right) \cdot \left(g_0 + X^{\frac{n}{2}} g_1\right)$
    $= f_0 g_0 + ((f_0 + f_1)(g_0 + g_1) - f_0 g_0 - f_1 g_1)X^{\frac{n}{2}} + f_1 g_1 X^n$

Previous results

- Karatsuba's algorithm:
  - Divide-and-Conquer: $\left(f_0 + X^{\frac{n}{2}} f_1\right) \cdot \left(g_0 + X^{\frac{n}{2}} g_1\right)$
    $= f_0 g_0 + ((f_0 + f_1)(g_0 + g_1) - f_0 g_0 - f_1 g_1) X^{\frac{n}{2}} + f_1 g_1 X^n$
  - Thomé'02: Careful use of n temp. registers + $O(\log n)$ stack
  - Roche'09: *half-additive* version $\rightsquigarrow$ only $O(\log n)$ stack
    ($h_\ell \leftarrow h_\ell + fg$ where $\deg(h_\ell) < \deg(f), \deg(g)$)

Previous results

- Karatsuba's algorithm:
  - Divide-and-Conquer: $\left(f_0 + X^{\frac{n}{2}} f_1\right) \cdot \left(g_0 + X^{\frac{n}{2}} g_1\right)$
    $= f_0 g_0 + ((f_0 + f_1)(g_0 + g_1) - f_0 g_0 - f_1 g_1)X^{\frac{n}{2}} + f_1 g_1 X^n$
  - Thomé'02: Careful use of n temp. registers + $O(\log n)$ stack
  - Roche'09: *half-additive* version $\rightsquigarrow$ only $O(\log n)$ stack
    ($h_\ell \leftarrow h_\ell + fg$ where $\deg(h_\ell) < \deg(f), \deg(g)$)
- FFT-based algorithms:
  - $(F, G) \to (F(\omega^i), G(\omega^i))_i \to FG(\omega^i)_i \to FG$

Previous results

- Karatsuba's algorithm:
  - Divide-and-Conquer: $\left(f_0 + X^{\frac{n}{2}} f_1\right) \cdot \left(g_0 + X^{\frac{n}{2}} g_1\right)$
    $= f_0 g_0 + ((f_0 + f_1)(g_0 + g_1) - f_0 g_0 - f_1 g_1) X^{\frac{n}{2}} + f_1 g_1 X^n$
  - Thomé'02: Careful use of n temp. registers + $O(\log n)$ stack
  - Roche'09: *half-additive* version $\rightsquigarrow$ only $O(\log n)$ stack
    ($h_\ell \leftarrow h_\ell + fg$ where $\deg(h_\ell) < \deg(f), \deg(g)$)
- FFT-based algorithms:
  - $(F, G) \to (F(\omega^i), G(\omega^i))_i \to FG(\omega^i)_i \to FG$
  - Every $\to$ is in-place (overwriting) but # points is $1 + \deg(FG)$
  - $\rightsquigarrow$ $\mathrm{size}((F(\omega^i), G(\omega^i))_i) = 2\,\mathrm{size}(FG)$

Previous results

- Karatsuba's algorithm:
  - Divide-and-Conquer: $\left(f_0 + X^{\frac{n}{2}} f_1\right) \cdot \left(g_0 + X^{\frac{n}{2}} g_1\right)$
    $= f_0 g_0 + ((f_0 + f_1)(g_0 + g_1) - f_0 g_0 - f_1 g_1) X^{\frac{n}{2}} + f_1 g_1 X^n$
  - Thomé'02: Careful use of n temp. registers + $O(\log n)$ stack
  - Roche'09: *half-additive* version $\rightsquigarrow$ only $O(\log n)$ stack
    $(h_\ell \leftarrow h_\ell + fg$ where $\deg(h_\ell) < \deg(f), \deg(g))$
- FFT-based algorithms:
  - $(F, G) \to (F(\omega^i), G(\omega^i))_i \to FG(\omega^i)_i \to FG$
  - Every $\to$ is in-place (overwriting) but # points is $1 + \deg(FG)$
  - $\rightsquigarrow$ $\mathrm{size}((F(\omega^i), G(\omega^i))_i) = 2\,\mathrm{size}(FG)$
  - Roche'09: Compute half of the result + recurse
  - Harvey-Roche'10: same with TFT (vdH'04)

Our problematic

Can *every* polynomial multiplication algorithm be performed without extra memory?

Our problematic

Can *every* polynomial multiplication algorithm be performed without extra memory?

- $O(1)$-space Karatsuba's algorithm?
- What about Toom-Cook algorithm?

Our problematic

Can *every* polynomial multiplication algorithm be performed without extra memory?

- $O(1)$-space Karatsuba's algorithm?
- What about Toom-Cook algorithm?
- What about other products (short and middle)?

Our problematic

Can *every* polynomial multiplication algorithm be performed without extra memory?

- $O(1)$-space Karatsuba's algorithm?
- What about Toom-Cook algorithm?
- What about other products (short and middle)?

Results:

- Yes!
- Almost (for other products)

Outline

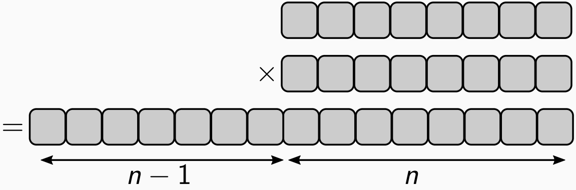
Polynomial products and linear maps

Space-preserving reductions

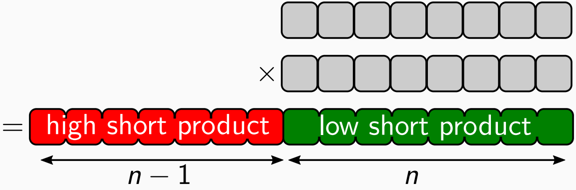
In-place algorithms from out-of-place algorithms

Polynomial products and linear maps

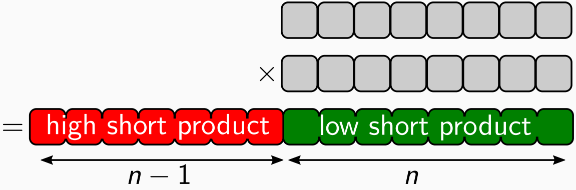
Short product



Short product

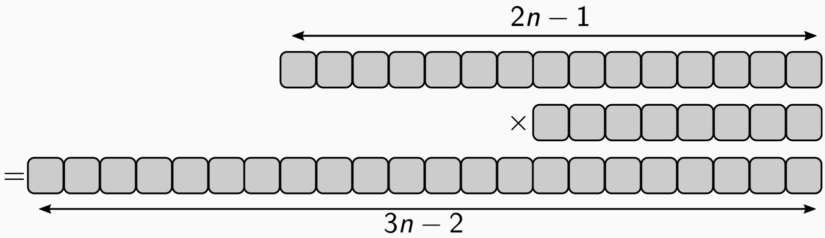


Short product

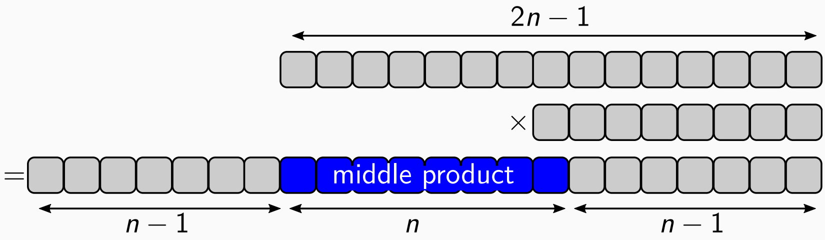


- Low short product: product of truncated power series
- Useful in other algorithms
- Time complexity: $\mathrm{M}(n)$
- Space complexity: $O(n)$

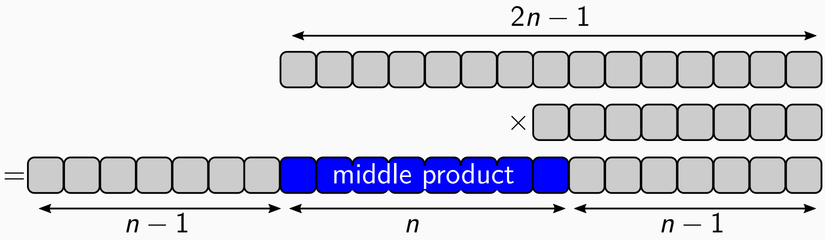
Middle product



Middle product

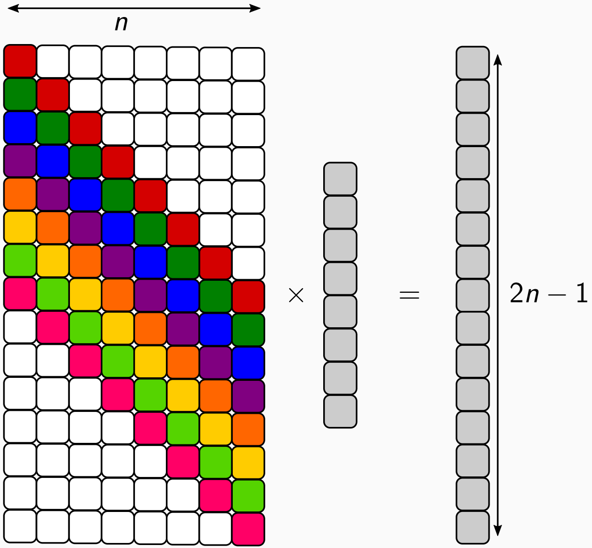


Middle product

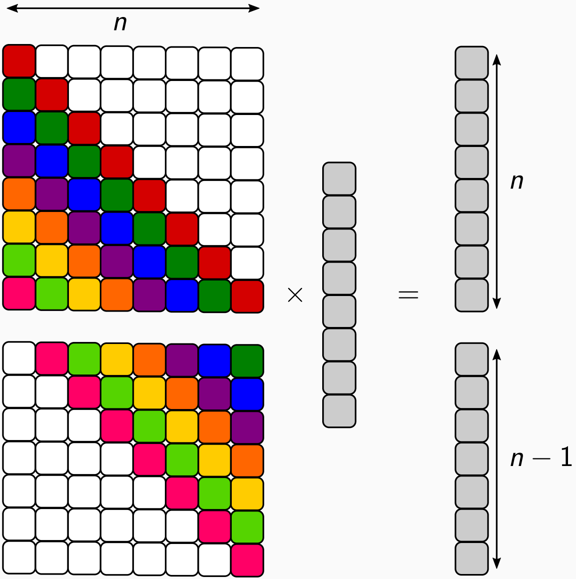


- Useful for Newton iteration
  - $G \leftarrow G(1-GF) \bmod X^{2n}$ with $GF = 1 + X^n H$
  - division, square root, ...
- Time complexity: $\mathsf{M}(n) \rightarrow$ Tellegen's transposition
- Space complexity: $O(n)$
- $O(1)$ space in the most permissive model via transposition of Harvey-Roche algorithm (Bostan-Lecerf-Schost'03)

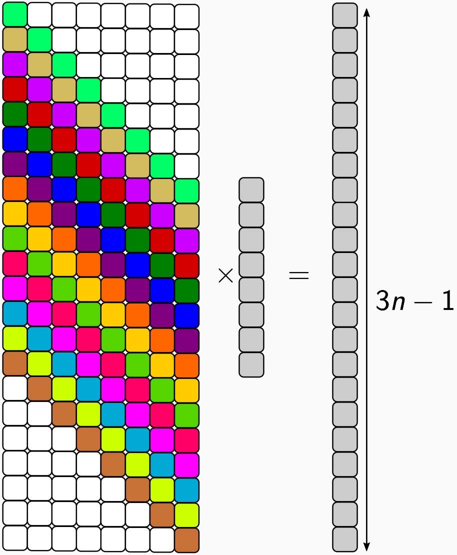
Multiplications as linear maps



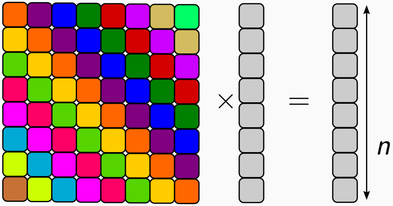
Multiplications as linear maps



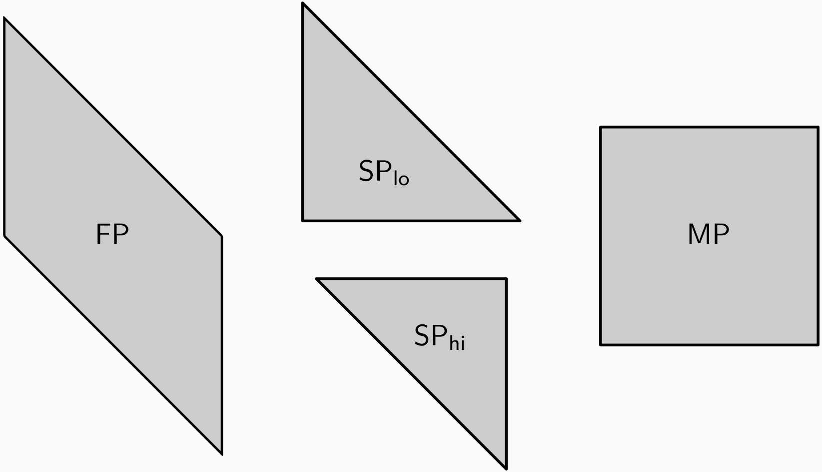
Multiplications as linear maps



Multiplications as linear maps



Multiplications as linear maps



Full product | Short products | Middle product

Space-preserving reductions

Relative difficulties of products

- Without space restrictions:
    - $\mathsf{SP} \leq \mathsf{FP}$ and $\mathsf{FP} \leq \mathsf{SP}_{\mathsf{lo}} + \mathsf{SP}_{\mathsf{hi}}$
    - $\mathsf{MP} \equiv \mathsf{FP}$ (transposition)
    - $\mathsf{MP} \leq \mathsf{SP}_{\mathsf{lo}} + \mathsf{SP}_{\mathsf{hi}} + (n-1)$ additions

Relative difficulties of products

- Without space restrictions:
  - SP $\leq$ FP and FP $\leq \mathrm{SP}_{\mathrm{lo}} + \mathrm{SP}_{\mathrm{hi}}$
  - MP $\equiv$ FP (transposition)
  - MP $\leq \mathrm{SP}_{\mathrm{lo}} + \mathrm{SP}_{\mathrm{hi}} + (n-1)$ additions
- Size of inputs and outputs:
  - FP : $n \times n \to 2n-1$
  - $\mathrm{SP}_{\mathrm{lo}}$: $n \times n \to n$
  - $\mathrm{SP}_{\mathrm{hi}}$: $n-1 \times n-1 \to n-1$;
  - MP : $2n-1 \times n \to n$

Relative difficulties of products

- Without space restrictions:
  - $\mathsf{SP} \leq \mathsf{FP}$ and $\mathsf{FP} \leq \mathsf{SP}_{\mathsf{lo}} + \mathsf{SP}_{\mathsf{hi}}$
  - $\mathsf{MP} \equiv \mathsf{FP}$ (transposition)
  - $\mathsf{MP} \leq \mathsf{SP}_{\mathsf{lo}} + \mathsf{SP}_{\mathsf{hi}} + (n-1)$ additions
- Size of inputs and outputs:
  - $\mathsf{FP} : n \times n \to 2n-1$
  - $\mathsf{SP}_{\mathsf{lo}} : n \times n \to n$
  - $\mathsf{SP}_{\mathsf{hi}} : n-1 \times n-1 \to n-1;$
  - $\mathsf{MP} : 2n-1 \times n \to n$

Reductions unusable in space-restricted settings!

A relevant notion of reduction

Definitions

- $\mathrm{TISP}(t(n), s(n))$: computable in time $t(n)$ and space $s(n)$
- $A \leq_c B$: A computable with oracle B and
  - constant number c of calls to oracle
  - negligible extra time
  - without extra space ($O(1)$)
- $A \equiv_c B$: $A \leq_c B$ and $B \leq_c A$

A relevant notion of reduction

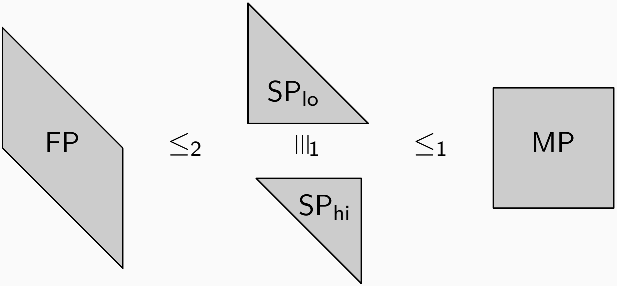
Definitions

- $\mathrm{TISP}(t(n), s(n))$: computable in time $t(n)$ and space $s(n)$
- $A \leq_c B$: A computable with oracle B and
  - constant number c of calls to oracle
  - negligible extra time
  - without extra space ($O(1)$)
- $A \equiv_c B$: $A \leq_c B$ and $B \leq_c A$

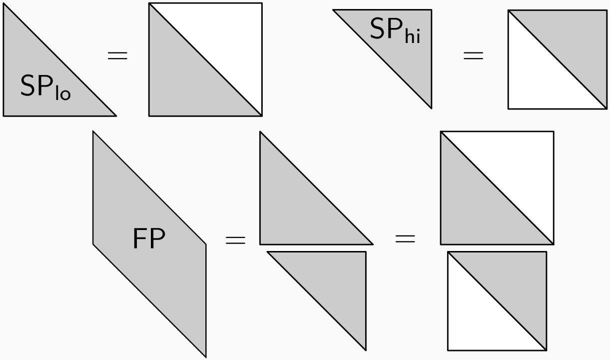
Proposition
If $B \in \mathrm{TISP}(t(n), s(n))$ and $A \leq_c B$, then

$$A \in \mathrm{TISP}(c\,t(n) + o(t(n)), s(n) + O(1))$$

Theorem

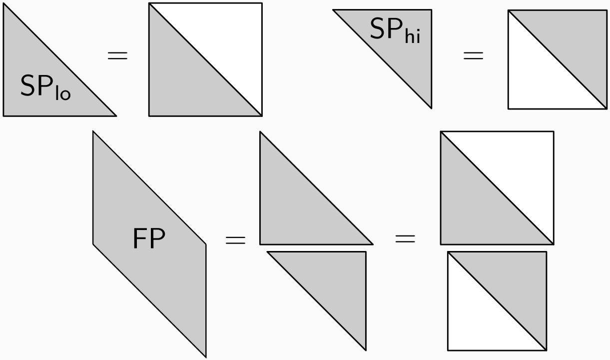


Visual proof



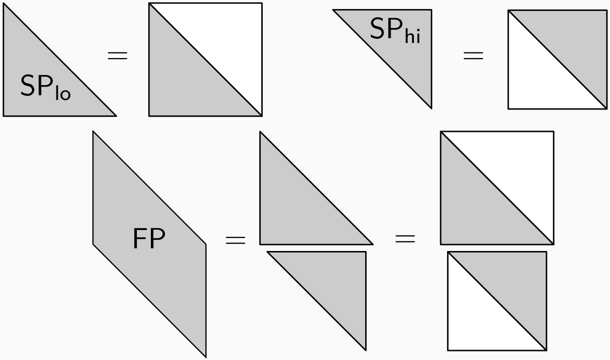
- Use of *fake padding* (in input, **not** in output!)

Visual proof



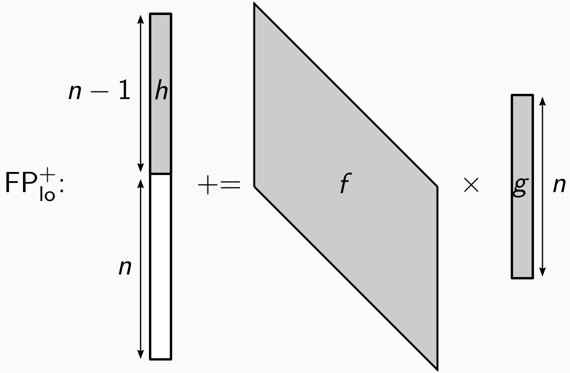
- Use of *fake padding* (in input, **not** in output!)
- $\mathrm{SP}_{\mathrm{lo}}(n) \leq \mathrm{MP}(n)$; $\mathrm{SP}_{\mathrm{hi}}(n) \leq \mathrm{MP}(n-1)$

Visual proof

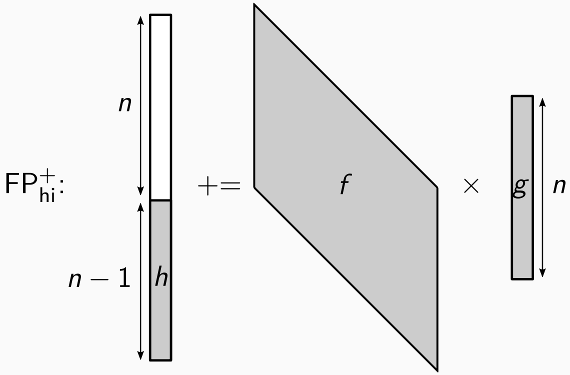


- Use of *fake padding* (in input, **not** in output!)
- $\mathrm{SP_{lo}}(n) \leq \mathrm{MP}(n)$; $\mathrm{SP_{hi}}(n) \leq \mathrm{MP}(n-1)$
- $\mathrm{FP}(n) \leq \mathrm{SP_{hi}}(n) + \mathrm{SP_{lo}}(n) \leq \mathrm{MP}(n) + \mathrm{MP}(n-1)$

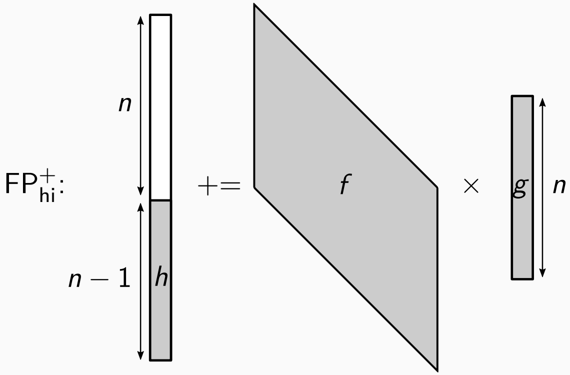
Half-additive full product: $h \leftarrow h + f \cdot g$



Half-additive full product: $h \leftarrow h + f \cdot g$



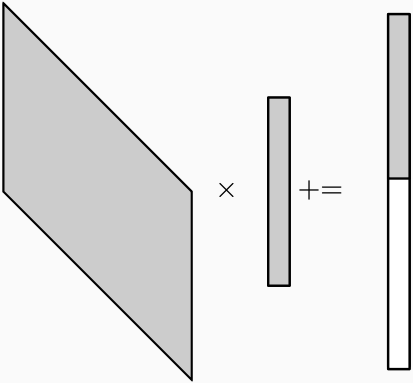
Half-additive full product: $h \leftarrow h + f \cdot g$



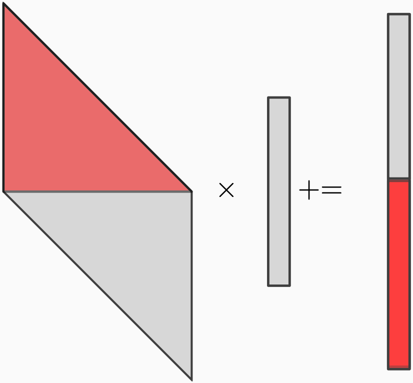
Remark $\mathrm{FP}^+_{\mathrm{lo}} \equiv_1 \mathrm{FP}^+_{\mathrm{hi}}$

Theorem $\mathrm{FP}^+ \leq_{3/2} \mathrm{SP}$ and $\mathrm{SP} \leq_2 \mathrm{FP}^+$

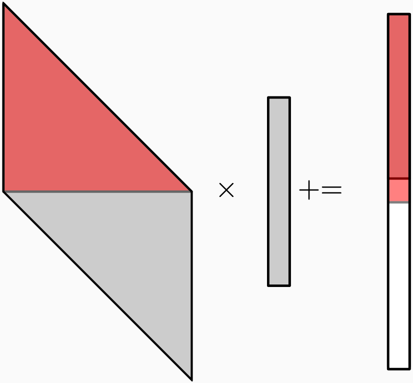
From SP to FP$^+$



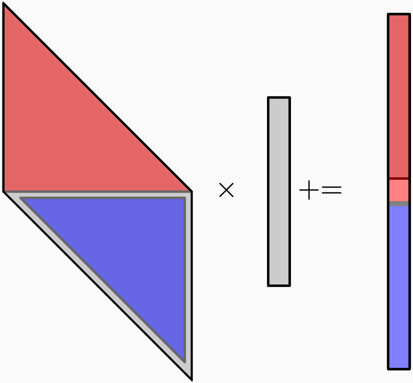
From SP to FP$^+$



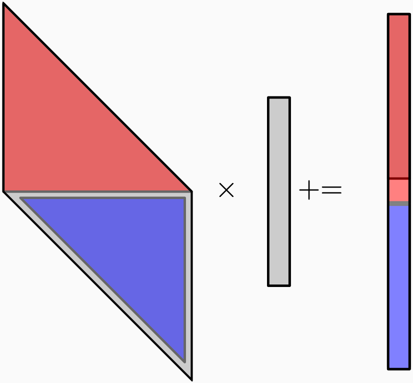
From SP to FP^{+}



From SP to FP$^+$



From SP to FP$^+$



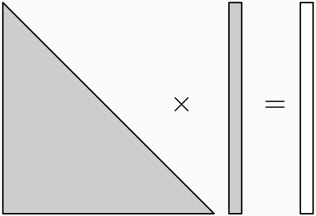
$$\mathsf{FP}^+_{\mathsf{lo}}(n) \leq \mathsf{SP}_{\mathsf{lo}}(n) + \mathsf{SP}_{\mathsf{hi}}(n) + n - 1$$

From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil}(f_0 g_1 + f_1 g_0) \mod X^n$$

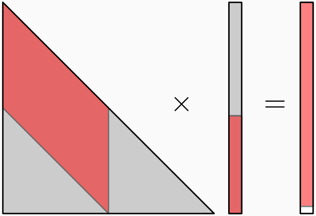
From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil}(f_0 g_1 + f_1 g_0) \mod X^n$$



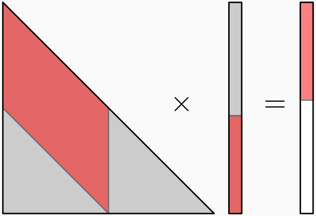
From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil} (f_0 g_1 + f_1 g_0) \mod X^n$$



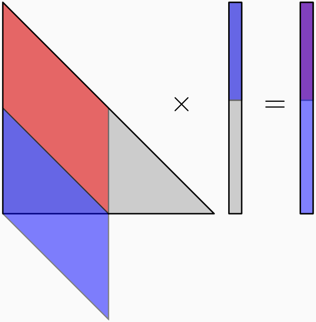
From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil} (f_0 g_1 + f_1 g_0) \mod X^n$$



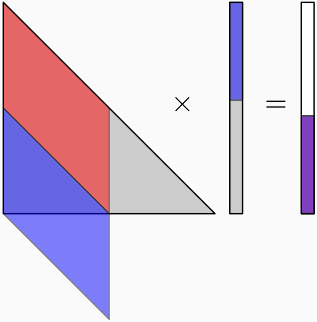
From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil}(f_0 g_1 + f_1 g_0) \mod X^n$$



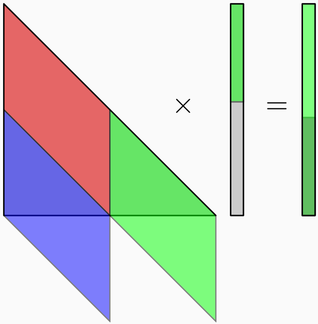
From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil} (f_0 g_1 + f_1 g_0) \mod X^n$$



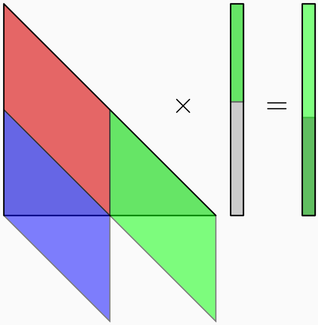
From FP$^+$ to SP

$$\left(f_0 + X^{\lceil n/2 \rceil} f_1\right) \cdot \left(g_0 + X^{\lceil n/2 \rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2 \rceil} (f_0 g_1 + f_1 g_0) \mod X^n$$



From FP^+ to SP

$$\left(f_0 + X^{\lceil n/2\rceil} f_1\right)\cdot\left(g_0 + X^{\lceil n/2\rceil} g_1\right) = f_0 g_0 + X^{\lceil n/2\rceil}(f_0 g_1 + f_1 g_0) \mod X^n$$



$$\mathrm{SP}_{\mathrm{lo}}(n) \le \mathrm{FP}(\lfloor n/2\rfloor) + \mathrm{FP}^+_{\mathrm{lo}}(\lfloor n/2\rfloor) + \mathrm{FP}^+_{\mathrm{hi}}(\lceil n/2\rceil)$$

Converse directions?

- From FP to SP:
    - problem with the output size
    - without space restriction: is $\mathsf{SP}(n) \simeq \mathsf{FP}(n/2)$?

Converse directions?

- From FP to SP:
  - problem with the output size
  - without space restriction: is $\mathrm{SP}(n) \simeq \mathrm{FP}(n/2)$?
- From SP to MP:
  - partial result:
    - up to $\log(n)$ increase in time complexity
    - techniques from next part
  - without space restriction or in a permissive model
    - FP to MP through Tellegen's transposition principle

In-place algorithms from out-of-place algorithms

Framework

- In-place algorithms parametrized by out-of-place algorithm
  - Out-of-place: Uses cn extra space
  - Constant c known to the algorithm

Framework

- In-place algorithms parametrized by out-of-place algorithm
  - Out-of-place: Uses cn extra space
  - Constant c known to the algorithm
- Goal:
  - Space complexity: $O(1)$
  - Time complexity: closest to the out-of-place algorithm

Framework

- In-place algorithms parametrized by out-of-place algorithm
  - Out-of-place: Uses cn extra space
  - Constant c known to the algorithm
- Goal:
  - Space complexity: $O(1)$
  - Time complexity: closest to the out-of-place algorithm
- Technique:
  - Oracle calls in smaller size
  - Tail recursive call
  - *Fake* padding

Tail recursion and fake padding

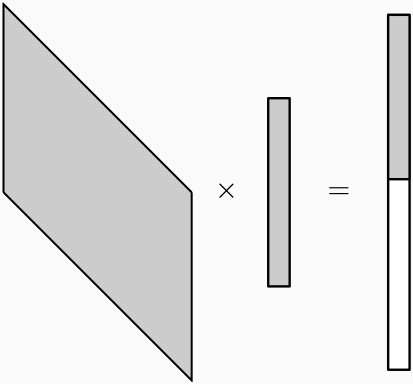
- Tail recursion:
  - Only one recursive call + last (or first) instruction
  - No need of recursive stack $\rightsquigarrow$ avoid $O(\log n)$ extra space

Tail recursion and fake padding

- Tail recursion:
  - Only one recursive call + last (or first) instruction
  - No need of recursive stack $\rightsquigarrow$ avoid $O(\log n)$ extra space
- *Fake* padding:
  - Pretend to pad inputs with zeroes
  - Make the data structure responsible for it
    - $O(1)$ increase in memory
    - *Cf.* strides in dense linear algebra
  - OK in inputs, not in outputs!

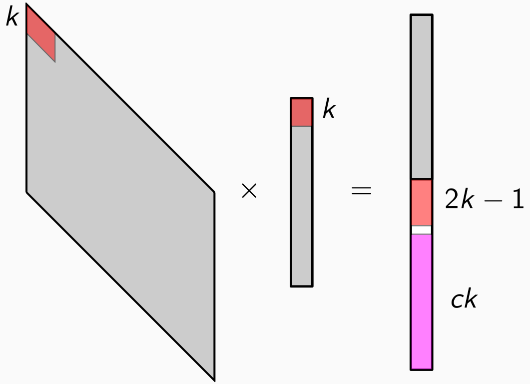
In-place FP$^+$ from out-of-place FP

$$(f_0 + X^k \hat{f}) \cdot (g_0 + X^k \hat{g}) = f_0 g_0 + X^k (f_0 \hat{g} + \hat{f} g_0) + X^{2k} \hat{f} \hat{g}$$



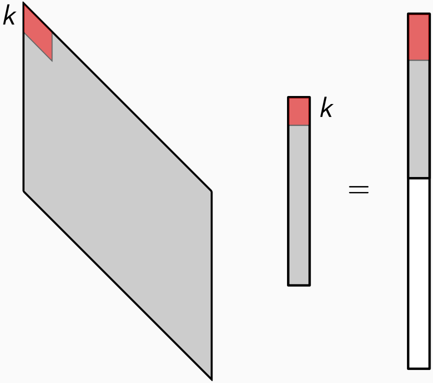
In-place FP$^+$ from out-of-place FP

$$(f_0 + X^k \hat{f}) \cdot (g_0 + X^k \hat{g}) = {\color{red} f_0 g_0} + X^k(f_0 \hat{g} + \hat{f} g_0) + X^{2k} \hat{f} \hat{g}$$



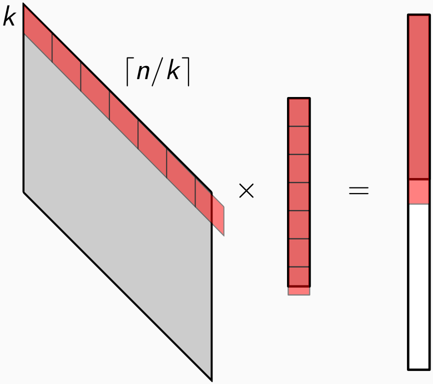
In-place FP$^+$ from out-of-place FP

$$(f_0 + X^k\hat{f}) \cdot (g_0 + X^k\hat{g}) = f_0 g_0 + X^k(f_0\hat{g} + \hat{f}g_0) + X^{2k}\hat{f}\hat{g}$$



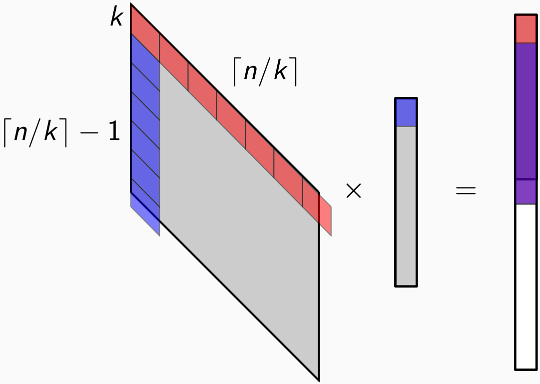
In-place FP$^+$ from out-of-place FP

$$(f_0 + X^k \hat{f}) \cdot (g_0 + X^k \hat{g}) = f_0 g_0 + X^k(f_0 \hat{g} + \hat{f} g_0) + X^{2k} \hat{f} \hat{g}$$



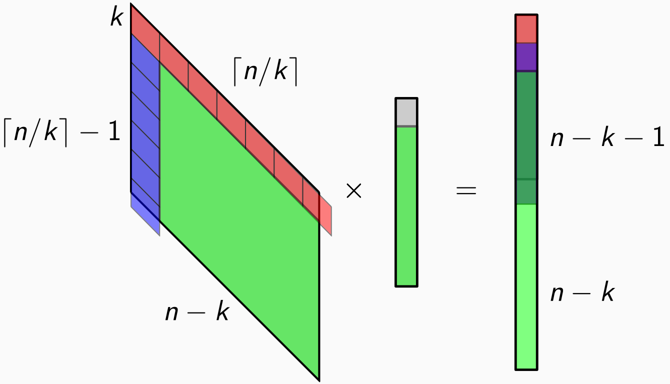
In-place FP$^+$ from out-of-place FP

$$(f_0 + X^k \hat{f}) \cdot (g_0 + X^k \hat{g}) = {\color{red} f_0 g_0} + X^k({\color{red} f_0 \hat{g}} + {\color{blue} \hat{f} g_0}) + X^{2k} \hat{f} \hat{g}$$

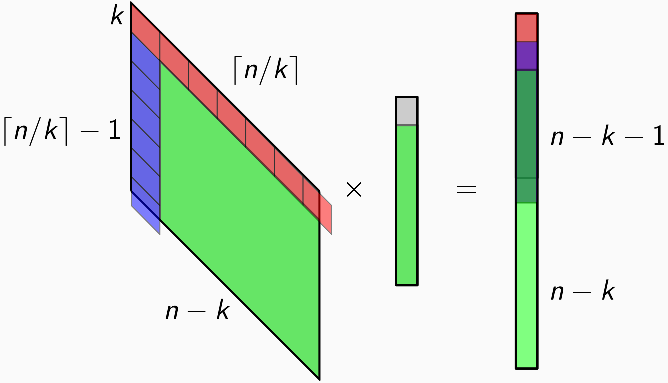


In-place FP$^+$ from out-of-place FP

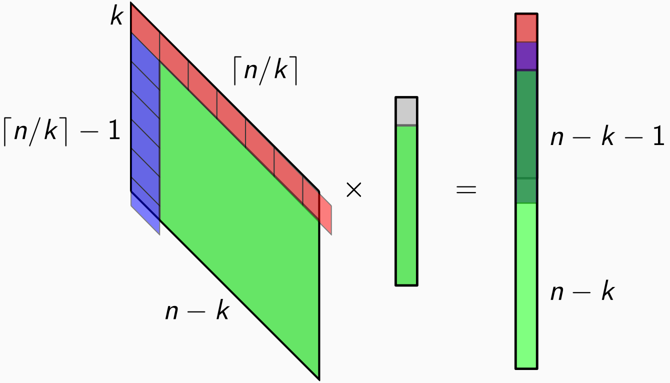
$$(f_0 + X^k \hat{f}) \cdot (g_0 + X^k \hat{g}) = f_0 g_0 + X^k (f_0 \hat{g} + \hat{f} g_0) + X^{2k} \hat{f} \hat{g}$$



Analysis

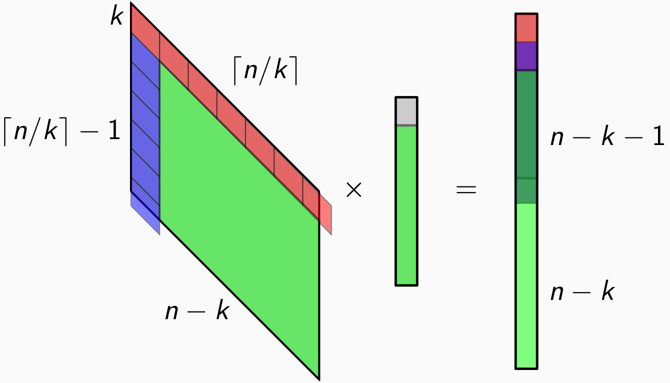


Analysis



- $ck + 2k - 1 \leq n - k \implies k \leq \frac{n+1}{c+3}$
- $T(n) = (2\lceil n/k \rceil - 1)(\mathrm{M}(k) + 2k - 1) + T(n-k)$

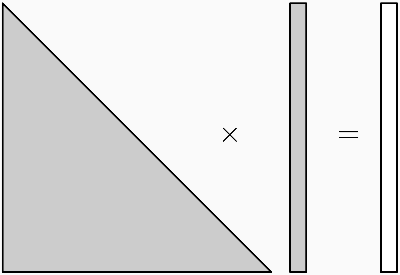
Analysis



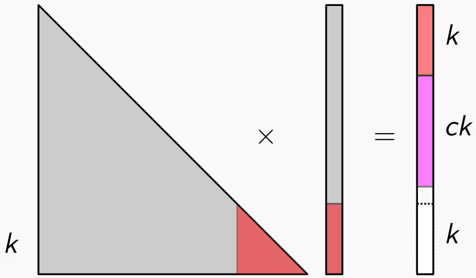
- $ck + 2k - 1 \leq n - k \implies k \leq \frac{n+1}{c+3}$
- $T(n) = (2\lceil n/k \rceil - 1)(\mathrm{M}(k) + 2k - 1) + T(n - k)$

$$T(n) \leq (2c + 7)\mathrm{M}(n) + o(\mathrm{M}(n))$$

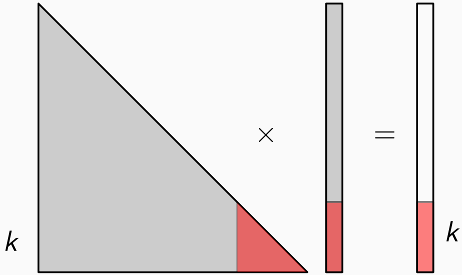
In-place short product



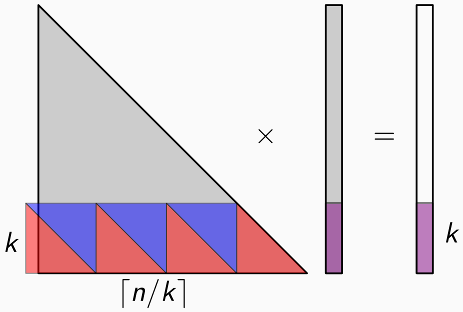
In-place short product



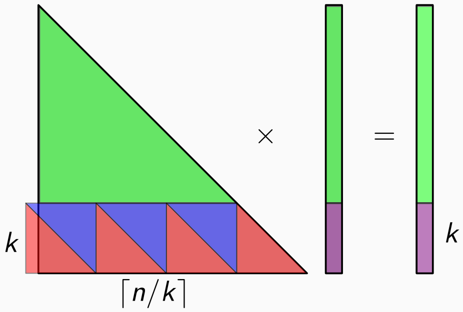
In-place short product



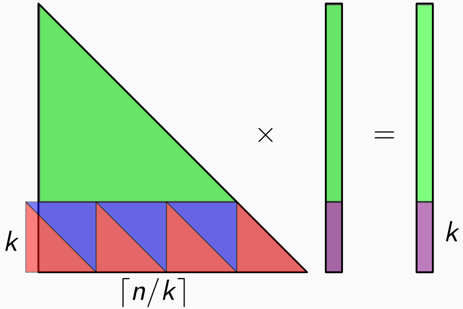
In-place short product



In-place short product

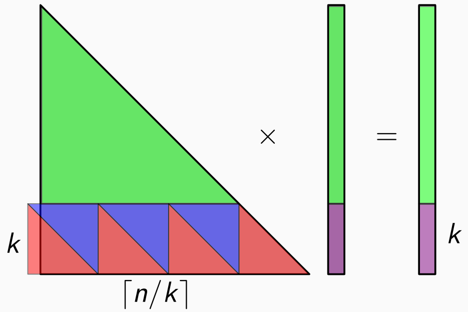


In-place short product



- $k \leq n/(c+2)$
- $T(n) = \lceil n/k \rceil \mathrm{M}(k) + (\lceil n/k \rceil - 1)\mathrm{M}(k-1) + 2k(\lceil n/k \rceil - 1) + T(n-k)$

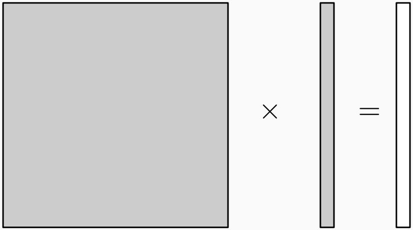
In-place short product



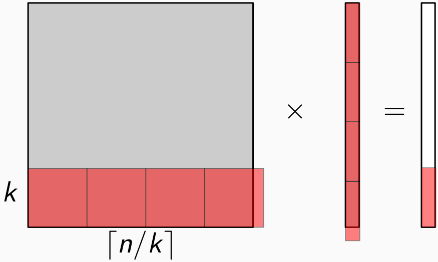
- $k \le n/(c+2)$
- $T(n) = \lceil n/k \rceil \mathsf{M}(k) + (\lceil n/k \rceil - 1)\mathsf{M}(k-1) + 2k(\lceil n/k \rceil - 1) + T(n-k)$

$$T(n) \le (2c+5)\mathsf{M}(n) + o(\mathsf{M}(n))$$

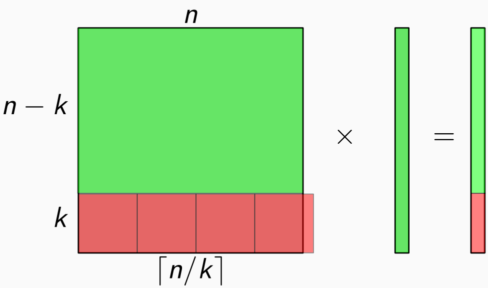
In-place middle product



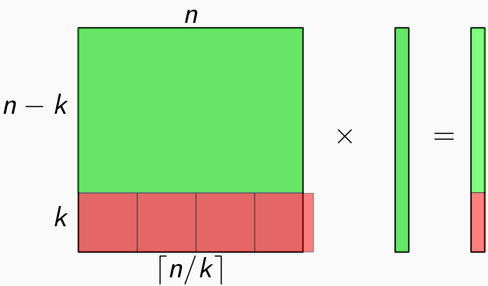
In-place middle product



In-place middle product

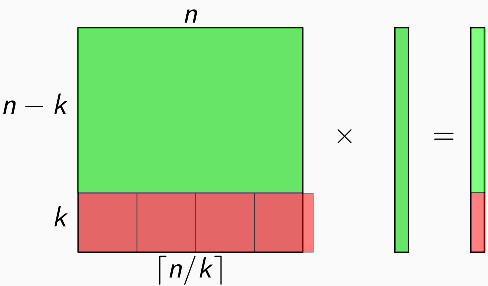


In-place middle product



- Recursive call on part of f... but on full g!
- $T(n,m) = \lceil n/k \rceil \mathsf{M}(k) + T(n, m-k)$

In-place middle product



- Recursive call on part of f... but on full g!
- $T(n,m) = \lceil n/k \rceil \mathsf{M}(k) + T(n, m-k)$

$$T(n,n) \leq \begin{cases} \mathsf{M}(n) \log_{\frac{c+2}{c+1}}(n) + o(\mathsf{M}(n)\log n) & \text{if } \mathsf{M}(n) \text{ is quasi-linear} \\ O(\mathsf{M}(n)) & \text{otherwise} \end{cases}$$

Other operations

Work in progress!

Other operations

Work in progress!

- Use our in-place algorithms as building blocks
  - Newton iteration: division, square root, ...
  - Evaluation & interpolation

$\rightarrow$ (at most) $\log(n)$ increase in complexity

Work in progress!

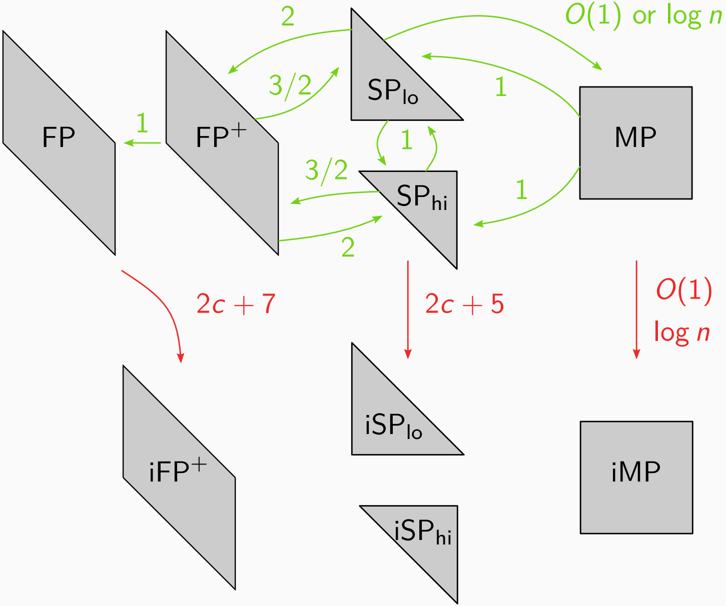
- Use our in-place algorithms as building blocks
  - Newton iteration: division, square root, . . .
  - Evaluation & interpolation

$\rightarrow$ (at most) $\log(n)$ increase in complexity

Remark

- In place: division with remainder
- Only quotient or only remainder: not clear
- Main difficulty: size of the output

Summary



2
$O(1)$ or $\log n$
SP_{lo}
$3/2$
1
FP
1
FP^{+}
1
MP
$3/2$
SP_{hi}
1
2
$2c+7$
$2c+5$
$O(1)$
$\log n$
iSP_{lo}
iFP^{+}
iMP
iSP_{hi}

Conclusion

- TISP-reductions between polynomial products
- Self-reductions to obtain in-place algorithms

Conclusion

- TISP-reductions between polynomial products
- Self-reductions to obtain in-place algorithms

Comparisons

- Better use specialized in-place algorithms…
- … when they exist!

Conclusion

- TISP-reductions between polynomial products
- Self-reductions to obtain in-place algorithms

Comparisons

- Better use specialized in-place algorithms. . .
- . . . when they exist!

Main open problems

- Remove the $\log(n)$ for middle product or prove a lower bound
- General result on Tellegen's transposition principle
- What about integer multiplication?

Conclusion

- TISP-reductions between polynomial products
- Self-reductions to obtain in-place algorithms

Comparisons

- Better use specialized in-place algorithms...
- ... when they exist!

Main open problems

- Remove the $\log(n)$ for middle product or prove a lower bound
- General result on Tellegen's transposition principle
- What about integer multiplication?

Thank you!