Fast Multiplication with Low Space Complexity

Daniel S. Roche

Ontario Research Centre for Computer Algebra

Symbolic Computation Group
School of Computer Science
University of Waterloo

University of Waterloo

Joint Mathematics Meetings
Washington, D.C.
8 January 2009

A Fun Puzzle

Why care about space complexity?

- Physical restrictions on space; not on time
- Cache misses incur a significant penalty
  in modern architectures
- Specific applications (e.g. embedded devices)
- Theoretical interest

Multiplication Algorithms

(over $\mathbb{Z}$ or $R[x]$)

	Time Complexity	**Space Complexity**
Classical Method	$O(n^2)$	$O(1)$
Divide-and-Conquer Karatsuba/Ofman '63	$O(n^{\log_2 3})$ or $O(n^{1.59})$	$O(n)$
FFT-based Schönhage/Strassen '71 Cantor/Kaltofen '91	$O(n \log n \log \log n)$	$O(n)$

Multiplication Algorithms

(over $\mathbb{Z}$ or R$[x]$)

	Time Complexity	**Space Complexity**
Classical Method	$O(n^2)$	$O(1)$
Divide-and-Conquer Karatsuba/Ofman '63	$O(n^{\log_2 3})$ or $O(n^{1.59})$	$O(n)$
FFT-based Schönhage/Strassen '71 Cantor/Kaltofen '91	$O(n \log n \log \log n)$	$O(n)$

Time-Space Tradeoff: Product of time and space is $\Omega(n^2)$
(Savage & Swamy 1979; Abrahamson 1986)

Standard Space Complexity Model (Papadimitriou)

3-Tape Turing Machine:

- Input tape (read-only)

9	1	1	2	$\times$	7	2	6

- Work tape (read/write)
  Size of this tape determines space complexity

4	$\cdot$	2	$\cdot$	1	8

- Output tape (write only)

$\cdots$	5	3	1	2

Significant improvements not possible in this model

Our Space Complexity Model

3-Tape Turing Machine:

- Input tape (read-only)

9	1	1	2	$\times$	7	2	6

- Work tape (read/write)
  Size of this tape determines space complexity

4	$\cdot$	2	$\cdot$	1	8

- Output tape (read/write)

$\cdots$	5	3	1	2

More realistic model for modern computers

Previous Work

- Monagan 1993: Importance of space efficiency for multiplication over $\mathbb{Z}_p[x]$
- Maeder 1993: Bounds extra space for Karatsuba multiplication so that storage can be preallocated — about $2n$ extra memory cells required.
- Thomé 2002: Karatsuba multiplication for polynomials using n extra memory cells.
- Zimmerman & Brent 2008:
  "The efficiency of an implementation of Karatsuba's algorithm depends heavily on memory usage."

Our Contributions

	Time Complexity	Space Complexity
Classical Method	$O(n^2)$	$O(1)$
Divide-and-Conquer Karatsuba/Ofman '63	$O(n^{\log_2 3})$ or $O(n^{1.59})$	$O(\log n)$
FFT-based Schönhage/Strassen '71 Cantor/Kaltofen '91	$O(n \log n \log \log n)$	$O(2^{\lceil \log_2 n \rceil} - n)$ ($O(1)$ if $n = 2^k$)

Standard Karatsuba Algorithm

Initial Setup

Idea: Reduce one degree-$2k$ multiplication to three of degree k.

Input: $f, g \in \mathrm{R}[x]$ each with degree less than $2k$.

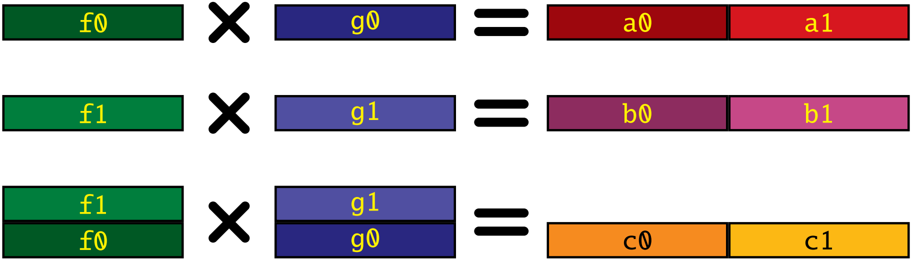
Write $f = f_0 + f_1 x^k$ and $g = g_0 + g_1 x^k$.

Standard Karatsuba Algorithm

Recursive Multiplications

Compute two sums: $f_0 + f_1$ and $g_0 + g_1$,
and three intermediate products:

$$a = f_0 \cdot g_0 \qquad b = f_1 \cdot g_1 \qquad c = (f_0 + f_1) \cdot (g_0 + g_1)$$



Standard Karatsuba Algorithm

Final Additions and Subtractions

Combine the computed products as follows:

$$
\begin{aligned}
a + (c - a - b) \cdot x^k + b \cdot x^{2k} \\
&= f_0 g_0 + (f_0 g_1 + f_1 g_0) \cdot x^k + f_1 g_1 \cdot x^{2k} \\
&= f \cdot g
\end{aligned}
$$

Extra Requirements for Improved Karatsuba

Read-Only Input Space:

f01	f11		g0	g1

Read/Write Output Space:

(empty)	(empty)	(empty)	(empty)

To Compute: $f \cdot g$

Extra Requirements for Improved Karatsuba

- The low-order coefficients of the output are initialized as h, and the product $f \cdot g$ is added to this.

Read-Only Input Space:

f01	f11		g0	g1

Read/Write Output Space:

h0	h1	(empty)	(empty)

To Compute: $f \cdot g + h$

Extra Requirements for Improved Karatsuba

- The low-order coefficients of the output are initialized as h, and the product $f \cdot g$ is added to this.
- The first polynomial f is given as a sum $f^{(0)} + f^{(1)}$.

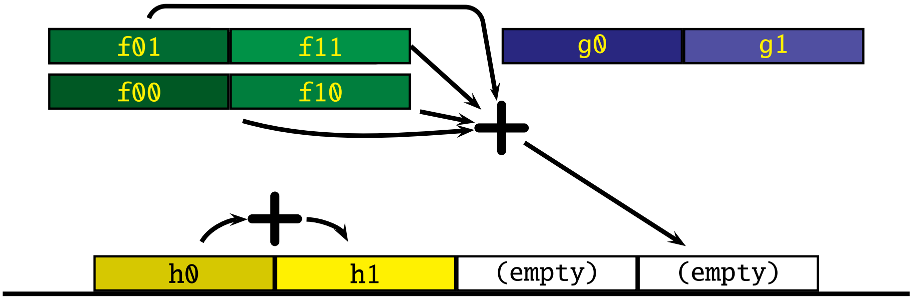
Read-Only Input Space:

Read/Write Output Space:

To Compute: $(f^{(0)} + f^{(1)}) \cdot g + h$

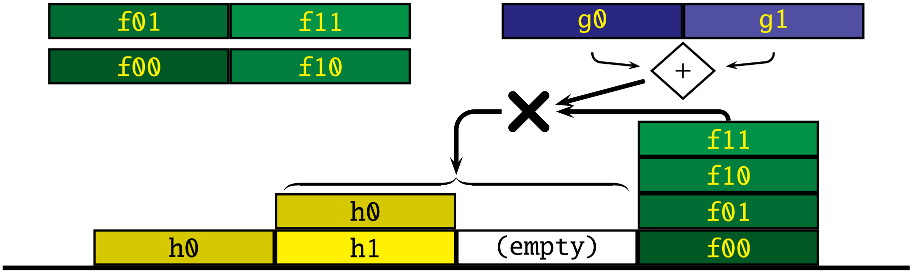
Space-Efficient Karatsuba Algorithm

Step 1: Preparing to Multiply



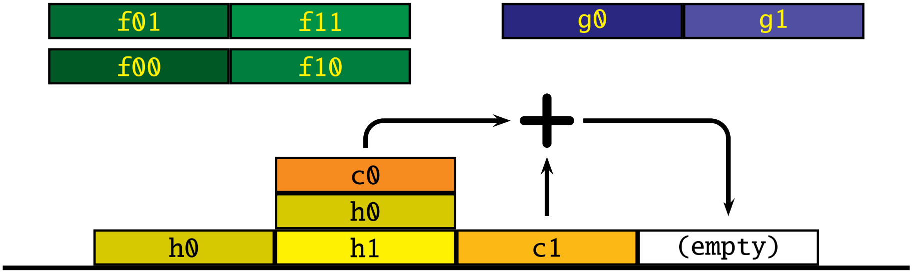
Space-Efficient Karatsuba Algorithm

Step 2: First product c



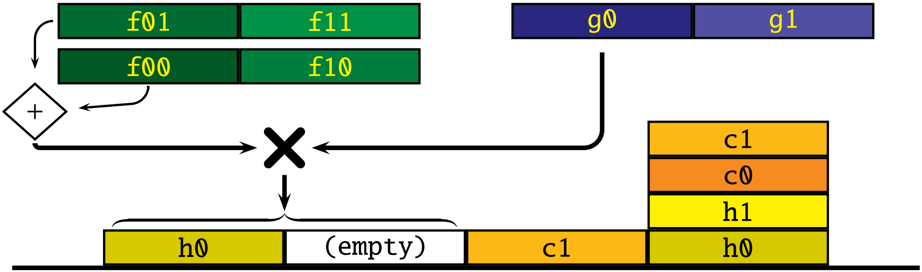
Space-Efficient Karatsuba Algorithm

Step 3: Rearranging



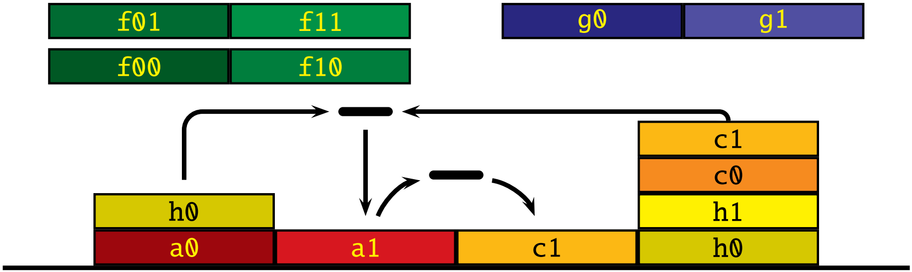
Space-Efficient Karatsuba Algorithm

Step 4: Second product a



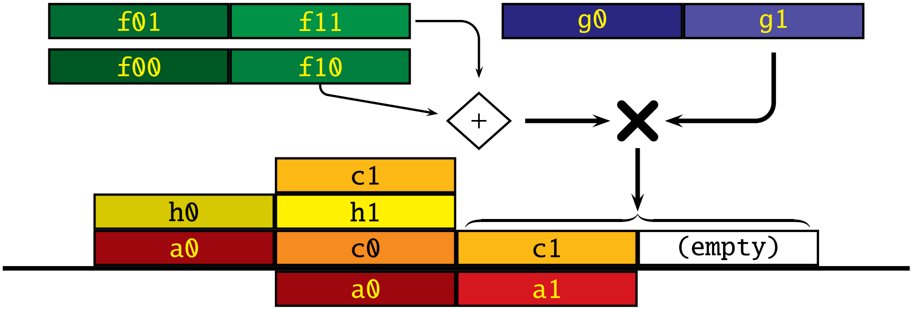
Space-Efficient Karatsuba Algorithm

Step 5: Rearranging



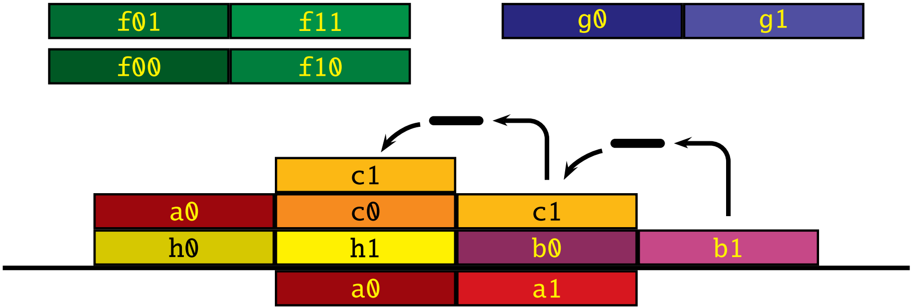
Space-Efficient Karatsuba Algorithm

Step 6: Third product b



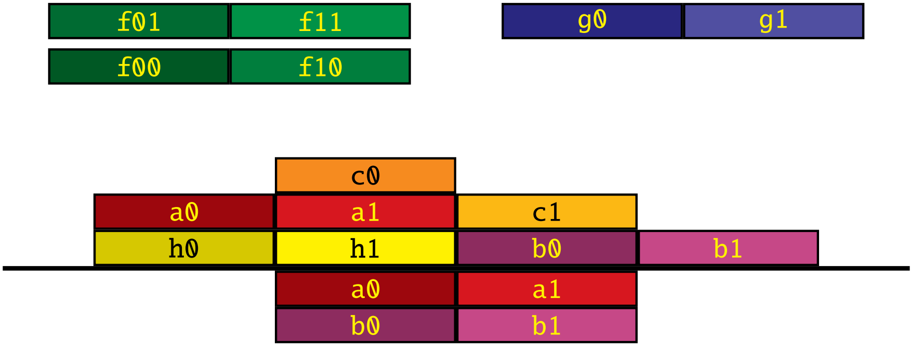
Space-Efficient Karatsuba Algorithm

Step 7: Rearranging



Space-Efficient Karatsuba Algorithm

Final Result



Analysis

- 3 recursive calls on degree-k arguments
  $\Rightarrow O(n^{\log_2 3})$ time complexity
- Constant extra space required at each recursive step
  $\Rightarrow O(\log n)$ space complexity
- At most $9n/2$ additions at each recursive step
  (compared to $4n$ for naïve implementation)

First multiplication algorithm with $o(n^2)$ time $\times$ space

Initial Recursive Calls

Call the algorithm discussed above Algorithm A

Algorithm B

- Neither operand is given as a sum
- $7n/2$ additions
- 2 recursive calls to B and one to A

Algorithm C

- Neither operand is given as a sum, and output is uninitialized
- $7n/2$ additions
- 2 recursive calls to C and one to B

Algorithm C is the top-level call.

Implementation Details

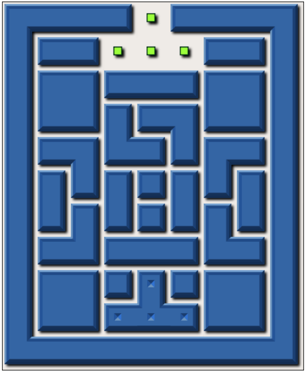
With some slight modifications, we can handle:

1. Odd-length operands
2. Different-length operands (Standard blocking method is used)

Proof-of-concept implementation in NTL

- $\approx 40\%$ slower than NTL Karatsuba
- Versions which destroy the input only $\approx 5\%$ slower
- NTL only allocates space once — not thread-safe!
- Victor Shoup is a better programmer than me

Open Problems



- More efficient implementation for univariate polynomials
- Implementation over $\mathbb{Z}$ (GMP)
- Similar results for Toom-Cook 3-way or k-way
- Better results for FFT-based multiplication
- Is completely in-place (overwriting input) possible?