

Polynomial Division using Dynamic Arrays, Heaps, and Packed Exponent Vectors

Roman Pearce

CECM, Simon Fraser University

September 2007

Joint work with Dr. Michael Monagan, Simon Fraser University

Why Sparse Polynomials ?

- The number of monomials in n variables with degree $\leq d$ is

$$\frac{(n+d)!}{n!d!}$$

- For $n = 8$ and $d = 50$, this is 1916797311. A dense polynomial with 64 bit coefficients would be 14.28 GB.

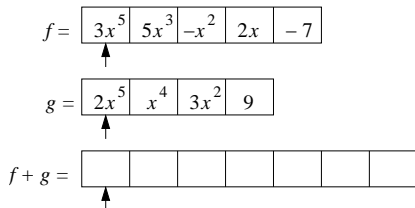
Polynomials that arise in practice are sparse.

- We will use a distributed representation. The monomials and coefficients are stored explicitly in a sorted structure.

$$-56x^2z^3 - 94xyz^2 + 87y^3z - 62yz^2 + 34$$

Sparse Distributed Polynomial Arithmetic

- We add or subtract polynomials using a *merge*. The pointers are incremented as terms are copied or added together:



- Linear time and space: $O(\#f + \#g)$ comparisons/terms.
- The result is also sorted.

The Division Algorithm

- The polynomials are sorted with respect to a *monomial order*.
- $LT(f)$ denotes the *leading term* (the first term) of f .

Algorithm

Input: $f, g \in F[x_1, \dots, x_n]$, F a field.

Output: $q, r \in F[x_1, \dots, x_n]$ satisfying $f = qg + r$.

1: $(p, q, r) := (f, 0, 0)$.

2: while $p \neq 0$ do

3: if $LT(g) | LT(p)$ then

4: $(q, p) := (q + t, p - tg)$ where $t = LT(p)/LT(g)$.

5: else

6: $(r, p) := (r + LT(p), p - LT(p))$.

7: return (q, r) .

$p - tg$ is $O(\#p + \#g)$ comparisons, everything else is $O(1)$.

Intermediate Blowup

Let $f = qg + r$ with $\#q = n$ and $\#g = m$.

- Each division step may add $m - 2$ more terms to p .
- After step i : $\#p \leq \#f + i(m - 2)$ terms.
- Each merge is $O(\#p + \#g)$ comparisons.

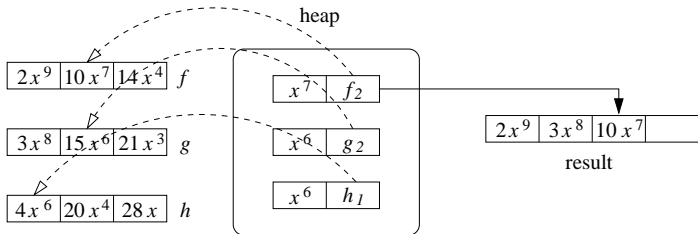
Cost: $\sim \sum_{i=1}^n i(m - 2) \in O(n^2m)$ comparisons, $O(nm)$ space.

One Solution: Geobucket Data Structure

4	8	16	32	64 terms	...
---	---	----	----	----------	-----

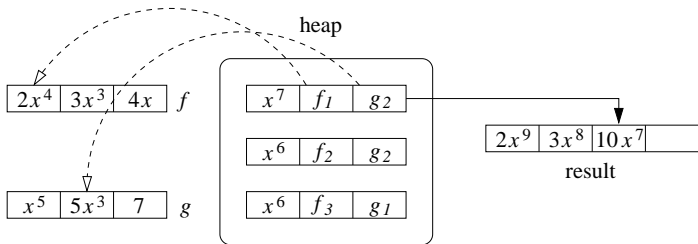
- Merge polynomials into the smallest possible bucket.
- “Divide-and-conquer” complexity: $O(nm \log n)$ comparisons.
- $O(nm)$ space, with a bad memory access pattern.

Addition With a Heap of Pointers



- Do a simultaneous n -ary merge using a heap (priority queue).
- We use the heap to add up terms with the same (largest) monomial, and we write out the terms of the result in order.
- After merging each term, we insert its successor into the heap.
- $O(\log n)$ comparisons to insert and extract terms.
- The polynomials are all accessed sequentially.
- **Runs in cache for sufficiently small n (at least up to $\sim 2^{16}$).**

Multiplication With a Heap of Pointers

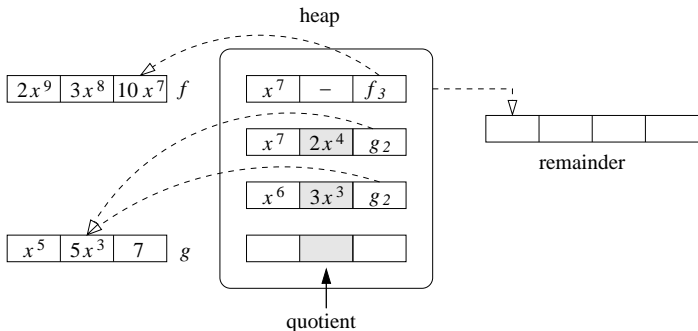


- Avoid constructing the partial products $f_i g_j$.
- Use two pointers: one into f , the second increments along g .
- Each product $f_i g_j$ is constructed “on-the-fly”, when it is inserted into the heap.

For $\#f = n$, $\#g = m$:

- $O(nm \log n)$ comparisons, $O(n + \#result)$ space.

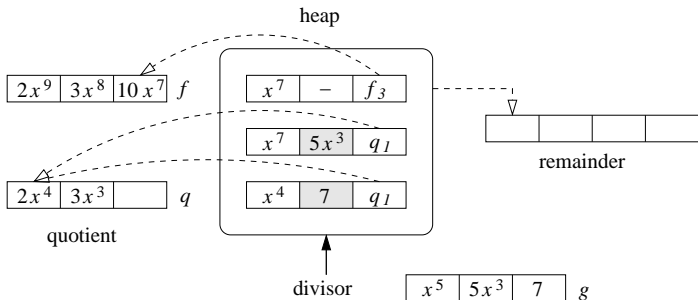
Division With a “Quotient Heap”



To divide $f = qg + r$, with $\#q = n$, $\#g = m$:

- Merge f with each $-q_i g$ in a heap of size $\#q + 1$.
- Pointers increment along the divisor g .
- $O(nm \log n)$ comparisons, $O(n + \#r)$ space.
- Runs in cache for n up to $\sim 2^{16}$.

Division With a “Divisor Heap”



To divide $f = qg + r$, with $\#q = n$, $\#g = m$:

- Merge f with each $-g_i q$ in a **fixed heap of size $\#g$** .
- Pointers increment along the quotient q .
- Terms of q are constructed “just-in-time”.
- $O(nm \log m)$ comparisons, $O(n + m + \#r)$ space.
- $O(m)$ random access: **fast** for big quotient/small divisor.

Applications of “Divisor Heap” Division

1. Let $g = \gcd(A, B)$. Compute the cofactors A/g and B/g .

Divisions over $\mathbb{Z}_{32003}[x, y, z, t, u, v, w]$

$\deg(f_i) = 10$, $\#f_i = 50$, $\#(f_1 f_2) = 2492$, $\#(f_1 f_2 f_3 f_4) = 4523085$.

	$(f_1 f_2 f_3 f_4)/(f_1 f_2)$ $q = 39$ KB	$(f_1 f_2 f_3 f_4)/f_1$ $q = 8.3$ MB
direct merge	74.540 s (207 MB)	3526.750 s (207 MB)
geobuckets	2.040 s (321 MB)	2.210 s (305 MB)
divisor heap	1.620 s (0.25 MB)	1.080 s (0.06 MB)

2. Sparse arithmetic with algebraic extensions.

Divisions over $\mathbb{Z}_{32003}[x, y, z, \alpha, \beta, s, t] \pmod{\{\alpha^2 - 3 = 0, \beta^2 + st - 1 = 0\}}$

$\deg(f_i) = 10$, $96 \leq \#f_i \leq 106$, $\#(f_1 f_2) = 8934$, $\#(f_1 f_2 f_3 f_4) = 1663235$.

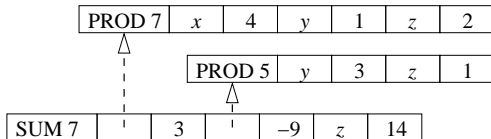
	$(f_1 f_2 f_3 f_4)/(f_1 f_2)$ $q = 140$ KB	$(f_1 f_2 f_3 f_4)/f_1$ $q = 3.9$ MB
geobuckets	35.520 s (1205 MB)	13.140 s (1039 MB)
divisor heap	35.040 s (82 MB)	5.980 s (113 MB)

High Performance Data Structure

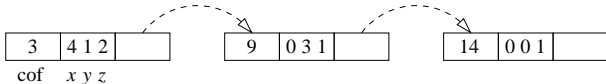
- The algorithms are designed to access terms in order.
- Memory references are expensive, and may miss the cache.

$$3x^4yz^2 - 9y^3z + 14z$$

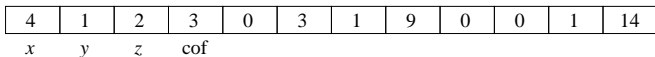
Maple:



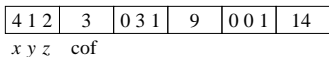
Singular:



Array:



Packed:



Current and Future Work

- Routines for big integer (GMP) coefficients.
- Division in $O(nm \log(\min(n, m)))$ comparisons (optimal).
- Simultaneous multiplication and division using a heap.
- Maple integration.

Thank you!