How fast can we multiply and divide polynomials?

Cybernet Systems, December 14, 2009.

Michael Monagan
Center for Experimental and Constructive Mathematics,
Simon Fraser University,
Vancouver, British Columbia,
CANADA.

Joint work with Roman Pearce.

The Mathematics Of Computer Algebra and Analysis project.

> factor$(2\,x^3 + x^2y^2 - x^3y + x^2 - 5\,yx - 3\,y^3 + 3\,y^2x - 3\,y - 1)$;

> solve$(\{x^2 + y^2 + z^2 - 4\,,\; x\,y\,z + 2\,,\; x\,y + z^3 - 1\})$;

> Determinant$\left(\begin{bmatrix} t & 1-2\,t & 1 \\ t^2 & t & 1 \\ 1+t & 1 & 1+t+t^2 \end{bmatrix}\right)$;

> $\int x^2\,\ln(x)\mathrm{e}^{-x} + (1-x)\,\ln(x)\mathrm{e}^{-x} - 2x\mathrm{e}^{-x}\,dx$;

The Mathematics Of Computer Algebra and Analysis project.

> $\text{factor}(2\,x^3 + x^2y^2 - x^3y + x^2 - 5\,yx - 3\,y^3 + 3\,y^2x - 3\,y - 1)$;

> $\text{solve}(\{x^2 + y^2 + z^2 - 4\ ,\ x\,y\,z + 2\ ,\ x\,y + z^3 - 1\})$;

> $\text{Determinant}\left(\begin{bmatrix} t & 1-2\,t & 1 \\ t^2 & t & 1 \\ 1+t & 1 & 1+t+t^2 \end{bmatrix}\right)$;

> $\int x^2\,\ln(x)\mathrm{e}^{-x} + (1-x)\,\ln(x)\mathrm{e}^{-x} - 2x\mathrm{e}^{-x}\,\mathrm{d}x$;

Risch

$\mathrm{e}^{-x} \rightarrow \theta_1$
$\ln x \rightarrow \theta_2$

$$\int \overbrace{x^2\theta_2\theta_1 + (1-x)\theta_2\theta_1 - 2x\theta_1}^{\text{a polynomial}}\,\mathrm{d}x \quad \text{where} \quad \begin{array}{l} \theta_1' = -\theta_1 \\ \theta_2' = 1/x. \end{array}$$

Polynomials are the key!

Talk Outline:

- How do CAS represent polynomials?
- How do CAS multiply and divide polynomials?
- Our representation and algorithms.
- How fast we compared with other CAS?
- Immediate Monomial Project (for Maple 15)
- Parallel Multiplication (for Maple 15)

How do CAS *represent* polynomials?

Recursive and distributed polynomial representations.

The distributed represention: monomials $x^i y^j z^k$ are sorted in *lexicographical order* (Magma, Mathematica):

$$f = -6x^3 + 9xy^3z - 8xy^2z + 7y^2z^2 + 5$$

or *graded lex order* (Singular, Maple 15):

$$f = 9xy^3z - 8xy^2z + 7y^2z^2 - 6x^3 + 5.$$

Key property: if X, Y, Z are monomials then $Y > Z \implies XY > XZ$.

Recursive and distributed polynomial representations.

The distributed represention: monomials $x^i y^j z^k$ are sorted in *lexicographical order* (Magma, Mathematica):

$$f = -6x^3 + 9xy^3z - 8xy^2z + 7y^2z^2 + 5$$

or *graded lex order* (Singular, Maple 15):

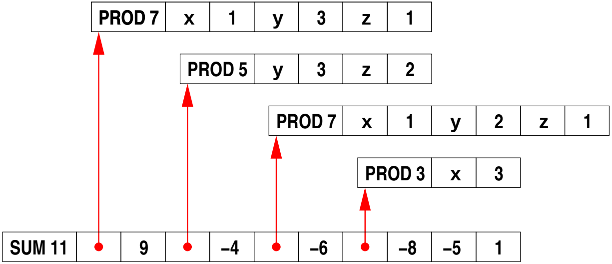
$$f = 9xy^3z - 8xy^2z + 7y^2z^2 - 6x^3 + 5.$$

Key property: if X, Y, Z are monomials then $Y > Z \implies XY > XZ$.

The recursive representation (Macsyma, REDUCE, Derive, Pari):

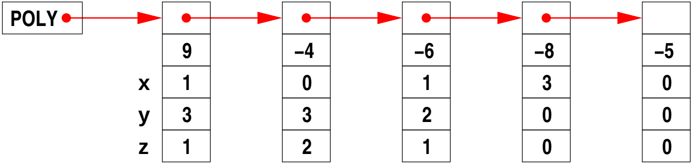
$$f = (-6)x^3 + ((9z)y^3 + (-8z)y^2)x^1 + ((7z^2)y^2 + 5y^0)x^0.$$

Maple's sum of products representation.

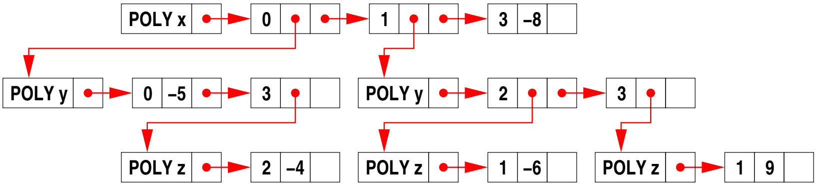


$$9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$$

Singular's distributed representation.

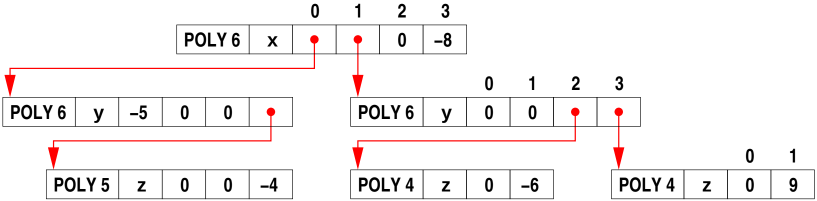


Trip's recursive sparse representation.



$$(-5y - 4z^2y^3) + (-6zy^2 + 9zy^3)x - 8x^3$$

Pari's recursive dense representation.



Our representation uses packed monomials.

Packing for $x^i y^j z^k$ in graded lex order with $x > y > z$:

$$\text{One 64 bit word}: \quad \underbrace{\boxed{i+j+k \;|\; i \;|\; j \;|\; k}}_{(i+j+k)2^{48}+2^{32}i+2^{16}j+k.}.$$

Why?

Our representation uses packed monomials.

Packing for $x^i y^j z^k$ in graded lex order with $x > y > z$:

$$\text{One 64 bit word}: \quad \underbrace{\boxed{i+j+k \;|\; i \;|\; j \;|\; k}}_{(i+j+k)2^{48} + 2^{32} i + 2^{16} j + k.}.$$

Why? Because monomial $>$ and $\times$ are one machine instruction.

Our representation uses packed monomials.

Packing for $x^i y^j z^k$ in graded lex order with $x > y > z$:

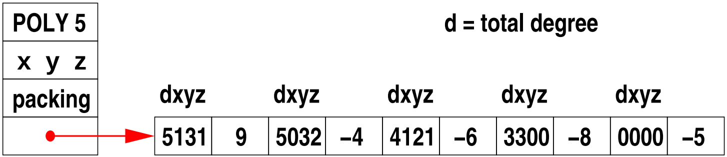
One 64 bit word :

$i+j+k$	i	j	k

$$\underbrace{\quad}_{(i+j+k)2^{48} + 2^{32} i + 2^{16} j + k.}$$

Why? Because monomial $>$ and $\times$ are one machine instruction.

Our packed array for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$.



Why graded lex order?

Our representation uses packed monomials.

Packing for $x^i y^j z^k$ in graded lex order with $x > y > z$:

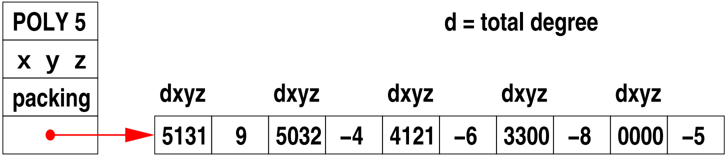
One 64 bit word :

$i+j+k$	i	j	k

$$\underbrace{\;}_{(i+j+k)2^{48} + 2^{32}i + 2^{16}j + k.}$$

Why? Because monomial $>$ and $\times$ are one machine instruction.

Our packed array for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$.



Why graded lex order? No exponent overflow in division.

How do CAS multiply and divide polynomials?

Let $f = f_1 + f_2 + \cdots + f_n$ and $g = g_1 + g_2 + \cdots + g_m$ where $f_1 > f_2 > \cdots > f_n$ and $g_1 > g_2 > \cdots > g_m$.

Using

$$h = f \times g = ((f_1 g + f_2 g) + f_3 g) + \cdots + f_n g \quad \text{and}$$
$$h \div g = f \;\; : \;\; (((h - f_1 g) - f_2 g) - f_3 g) - \cdots - f_n g$$

Let $f = f_1 + f_2 + \cdots + f_n$ and $g = g_1 + g_2 + \cdots + g_m$
where $f_1 > f_2 > \cdots > f_n$ and $g_1 > g_2 > \cdots > g_m$.

Using

$$h = f \times g = ((f_1 g + f_2 g) + f_3 g) + \cdots + f_n g \quad \text{and}$$
$$h \div g = f \;\; : \;\; (((h - f_1 g) - f_2 g) - f_3 g) - \cdots - f_n g$$

takes $O(n^2 m)$ comparisons of monomials
and $O(nm)$ multiplications of coefficient and monomials.

Example:
$f = x^n + x^{n-1} + \ldots + x$ and $g = y^n + y^{n-1} + \ldots + y$.

Our algorithms for multiplication and division use heaps.

Heaps

A binary heap H with n entries is a partially ordered array satisfying

$$H_i \geq H_{2i} \quad \text{and} \quad H_i \geq H_{2i+1}.$$

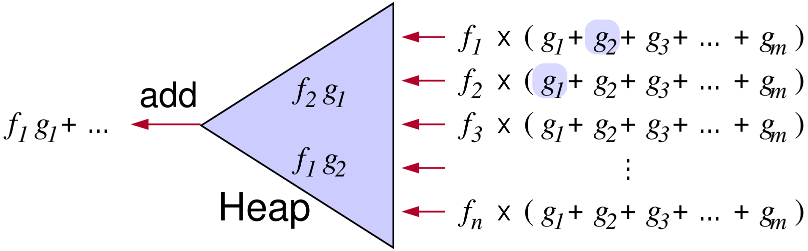
H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
10	9	8	3	5	4	7	–

- H_1 is the biggest entry in a heap.
- We can extract the maximum entry in $O(\log_2 n)$ comparisons.
- We can insert a new entry in $O(\log_2 n)$ comparisons.

Multiplication using a binary heap.

Johnson, 1974, a simultaneous n-ary merge:

$$\begin{aligned} f &= a_1X_1 + a_2X_2 + \cdots + a_nX_n \\ g &= b_1Y_1 + b_2Y_2 + \cdots + b_mY_m \end{aligned} \quad \text{(sorted)}$$



- $O(nm \log n)$ comparisons.
- Space for $\leq n$ monomials in the heap.
- Can pick $n \leq m$.

Division using a heap.

Johnson's quotient heap algorithm.

Dividing $f \div g = q$ compute $$f - \sum_{i=1}^{\#q} q_i \times g$$

- $O(\#f + \#q\#g \log \#q)$ comparisons
- $O(\#q)$ working memory

Our divisor heap algorithm.

Dividing $f \div g = q$ compute $$f - \sum_{i=2}^{\#g} g_i \times q$$

- $O(\#f + \#q\#g \log \#g)$ comparisons
- $O(\#g)$ working memory

Minimal heap division (Monagan & Pearce, 2008)

Problem: we don't know if $\#q > \#g$ when starting a division.
E.g. $(x^7 - y^7) \div (x - y) = x^6 + yx^5 + y^2x^4 + \cdots + y^6$.

Start with quotient heap, switch to divisor heap when $\#q = \#g$.

$$f \quad - \underbrace{\sum_{i=1}^{\min(\#q,\#g)} q_i \times g}_{\text{quotient heap}} \quad - \underbrace{\sum_{i=2}^{\#g} g_i \times (q_{\#g+1} + \cdots)}_{\text{divisor heap}}$$

- $O(\#f + \#q\#g \log \min(\#q, \#g))$ comparisons
- $O(\min(\#q, \#g))$ working memory

Which CAS is fastest?

Benchmark 1: A dense Fateman problem.

$$f = (1 + x + y + z + t)^{20} \quad g = f + 1$$

- f and g have 39 bit coefficients and 10,626 terms
- $h = f \cdot g$ has 83 bit coefficients and 135,751 terms

Intel Core2 3.0 GHz	multiply $p = f \times g$	divide $q = p/f$
Maple 12	289.23 s	187.72 s
Maple 13	187.35 s	159.12 s
Singular 3-0-4	62.00 s	20.00 s
Magma V2.14-7	23.02 s	22.76 s
Pari 2.3.3 (w/ GMP)	32.43 s	14.76 s
Trip v0.99	5.93 s	-
sdmp (unpacked)	5.15 s	5.44 s
sdmp (packed)	2.26 s	2.77 s
Maple 14	3.33 s	4.46s

Benchmark 2: A sparse 10 variable problem.

$$\begin{aligned} f &= (x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_8 + x_8x_9 + x_9x_{10} + x_1x_{10} \\ &\quad + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + 1)^4 \\ g &= (x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2 + x_{10}^2 \\ &\quad + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + 1)^4 \end{aligned}$$

$6,746 \times 8,361 =$ 3,157,883 terms	multiply $p = f \times g$ seconds	divide $q = p/f$ secs
Maple 12	305.76s	280.65s
Maple 13	293.74s	312.29s
Singular 3-0-4	31.00s	18.00s
Magma V2.14-7	17.43s	197.72s
Pari 2.3.3 (w/ GMP)	7.06s	7.05s
Trip v0.99 (rationals)	8.13s	–
sdmp (unpacked)	11.12s	10.37s
sdmp (packed)	2.46s	2.61s
Maple 14	11.74s	14.45s

Benchmark 3: Factorization speedup in Maple 14.

In Maple 13,

```
> h := expand(f*g);
> divide(h,f,'q');
```

call `` `expand/bigprod` ``(f,g) and `` `expand/bigdiv` ``(h,f,q) for large inputs.

In Maple 14, we reprogrammed `` `expand/bigprod` `` and `` `expand/bigdiv` `` to convert to SDMP, multiply (divide) in SDMP, then convert back to Maple.

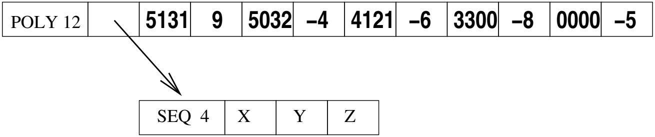
Benchmark	factor(h) where $h=(a+1)(a+2)$			
$a=$	Maple13	Magma	Maple14	Speedup
$(x+y+z)^{30}$	368.88	4.47	18.61	19.8 x
$(1+x+y+z)^{20}$	38.38	10.95	4.01	9.6 x
$(1+x+y+z)^{30}$	679.01	400.4	23.38	29.0 x
$(1+x+y+z+t)^{20}$	5390.32	1286.8	99.00	54.4 x

Table: Factorization Benchmark Timings (in CPU seconds)

The Immediate Monomial Project.

A joint MITACS project with Maplesoft.

A new data structure being implemented by Paul de Marco.



How will we pack monomials? E.g. $x^i y^j z^k$ on a 64 bit computer.

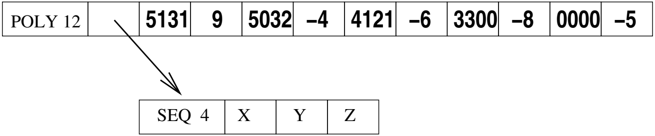
Always try to pack all monomials into one word.

If $i+j+k < 2^{16}$ pack | $i+j+k$ | i | j | k | in one word.

If $i+j+k \geq 2^{16}$ use Maple's existing representation.

So the number of variables determines the packing.

The Immediate Monomial Project



Let $f(x_1, x_2, \cdots, x_n) = f_1 + f_2 + \cdots + f_m$.

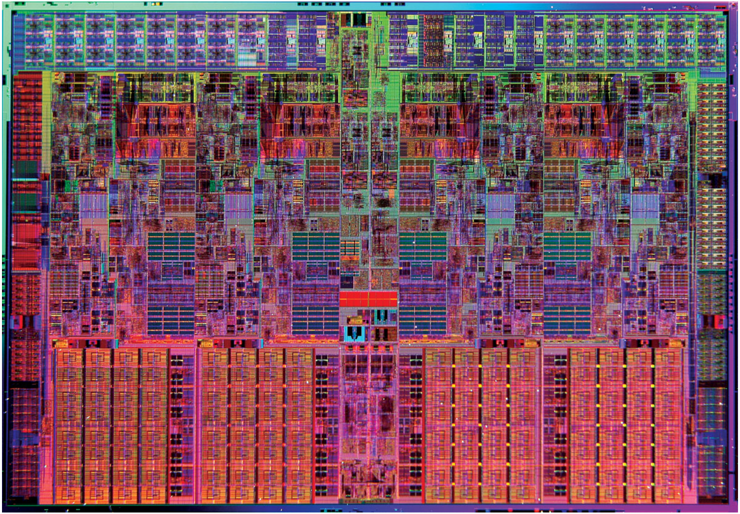
$O(nm) \Rightarrow O(1)$: `lcoeff(f);  degree(f);  indets(f);`

$O(nm) \Rightarrow O(n)$: `f;  has(f,x);  type(f,polynom(integer));`

$O(nm) \Rightarrow O(m)$: `degree(f,x);  diff(f,x);  coeffs(f,x);`

A $10 - 20\%$ gain in overall efficiency gain for Maple 14 ?

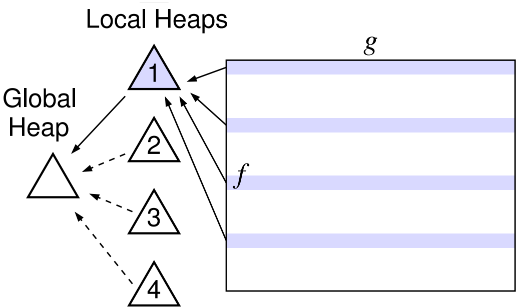
Parrallelizing Multiplication Using Heaps



Intel Core i7.

Parallel Algorithm

One heap per core, merge results in a global heap.

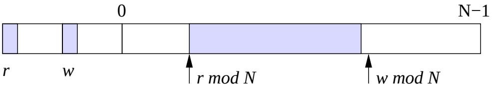


Don't waste real or cpu time:

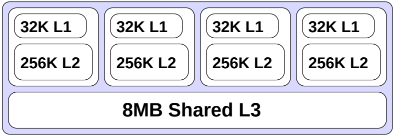
- partition terms
- transfer data
- balance load

Transferring Data

Threads write to a finite circular buffer.

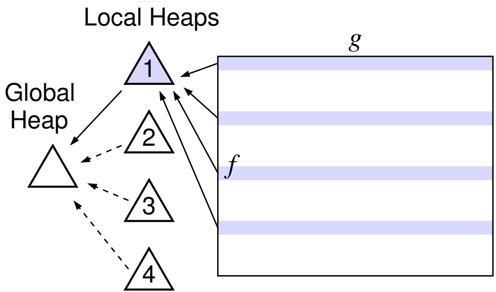



```
#define N    32768  /* size in words (256 K) */
#define CLINE  64   /* bytes per cache line  */
struct buffer {
     long r;         /* words read   */
     char pad1[CLINE - sizeof(long)];
     long w;         /* words wrote */
     char pad2[CLINE - sizeof(long)];
     long data[N];   };
```



Intel Core i7

Load Balancing



- threads try to acquire a lock for the global heap
- one thread per core avoids context switches and OS
- threads independently adjust their share of global work

buffer full	$\longrightarrow$	do more global work
buffer empty	$\longrightarrow$	do less global work

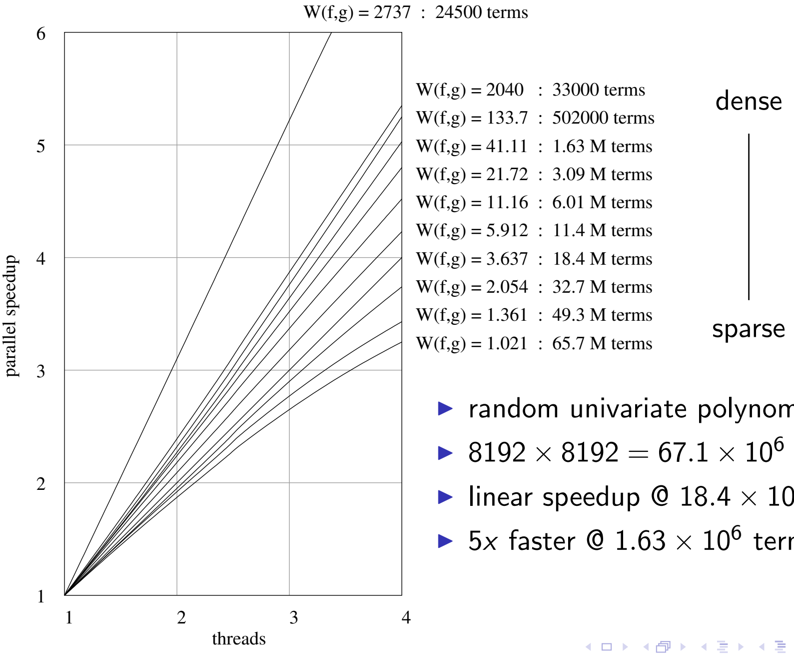
Dense Benchmark

$$f = (1 + x + y + z + t)^{30} \quad g = f + 1$$

$46376 \times 46376 = 635376$ terms $W(f, g) = 3332$

	threads	Core i7 2.66GHz		Core 2 2.4GHz	
sdmp	4	11.48 s	6.15x	14.15 s	4.25x
(packed)	3	16.63 s	4.24x	19.43 s	3.10x
	2	28.26 s	2.50x	28.29 s	2.13x
	1	70.59 s		60.25 s	
Trip 1.0 beta2 recursive dense	4	23.76 s	3.89x	26.86 s	3.94x
	3	31.05 s	2.97x	35.65 s	2.97x
	2	46.56 s	1.98x	52.98 s	1.99x
	1	92.38 s		105.78 s	
Trip 1.0 beta2 recursive sparse	4	29.36 s	3.26x	31.95 s	3.38x
	3	36.00 s	2.66x	39.96 s	2.71x
	2	50.96 s	1.88x	56.68 s	1.91x
	1	95.74 s		108.15 s	
Magma 2.15-8	1	526.12 s			
Pari/GP 2.3.3	1	642.74 s		707.61 s	
Singular 3-1-0	1	744.00 s		1048.00 s	
Maple 13	1	5849.48 s		9343.68 s	

Parallel Speedup: Core i7



- random univariate polynomials
- $8192 \times 8192 = 67.1 \times 10^6$ products
- linear speedup @ 18.4×10^6 terms
- 5x faster @ 1.63×10^6 terms