MATH 497, MATH 895, CMPT 894. Assignment 2, Summer 2007

Instructor: Michael Monagan

Please hand in the assignment by 2:30pm on June 13th before class starts.
Late Penalty -20% off for each day late.
Please submit a printout of a Maple worksheet containing Maple code and output.

2 The Division Algorithm for $\mathbb{Q}[x, y, z, ...]$

Download and read the paper "Polynomial Division using Dynamic Arrays, Heaps and Packed Exponent Vectors" by Monagan and Pearce from

`http://www.cecm.sfu.ca/personal/monaganm/teaching/TopicsinCA07/`

The goal of this assignment is to study three of the data structures considered in the paper for polynomial division using a sparse distributed representation.

Represent a polynomial as a Maple list of terms sorted in descending *graded lexicographical* order. Represent each term in the form $[c, e]$ where $c \in \mathbb{Q}$ is a coefficient and e, the exponent vector, is represented as a Maple list of integers. You do not need to pack exponent vectors. E.g. the term $5x^2y^3z$ would be represented as [5,[2,3,1]] (or better, as [5,[6,2,3]] where $x^iy^jz^k$ is stored as $[i + j + k, i, j]$). Implement the following Maple procedures where X is a list of variables.

```
SDMP2Maple(a,X)
Maple2SDMP(A,X)
SDMPAdd(A,B)
```

E.g. `A := SDMP2Maple(a,[x,y,z])` converts a Maple polynomial $a(x, y, z)$ into the SDMP data structure and `Maple2SDMP(A,[x,y,z])` converts it back. Note, you can add and subtract lists of integers in Maple using $+$ and $-$ but you cannot compare them using $<$.

Now design and implement the following algorithms for polynomial division.

(i) the simple merging algorithm using two dynamic arrays, p and p',

(ii) the geobucket data structure using fixed size arrays and dynamic buffers p and p', and

(iii) (graduate students only) the heap of divisor pointers data structure.

Write a Maple procedure `DIVIDE(A,B)` that outputs a pair (Q,R), the quotient and remainder of A divided B that satisfy $A = BQ + R$ and $R = 0$ or no term in R is divisible by $LT(B)$. Use graded lexicographical order. You do not have to make your algorithm work for multiple divisors, but you may if you wish.

Note, for (iii), you should implement your own heap operations. Implement the heap in a Maple table H using $H[0]$ as a counter for the number of entries in the heap, and putting entries in $H[1], H[2], ..., H[H[0]]$. Execute your algorithm on the following sparse problem using your distributed data structure for each algorithm.

```
> X := [u,v,w,x,y,z];
> a := randpoly(X,degree=10,terms=5000):
> b := randpoly(X,degree=5,terms=5):
> c := expand(a*b):
> A,B,C := Maple2SDMP(a,X),Maple2SDMP(b,X),Maple2SDMP(c,X);
> Q,R := DIVIDE(C,A): evalb(Q=B); R;
> Q,R := DIVIDE(C,B): evalb(Q=A); R;
```

Compute and print (i) the number of monomial comparisons N each algorithm makes, (ii) the number of monomial multiplications and divisions M each algorithm makes and (iii) the number $S = N/M$ which measures the monomial comparison cost. Repeat the above experiment on the following sparse example.

```
> macro(D=DD);
> X := [u,v,w,x,y,z];
> for i to 4 do f[i] := randpoly(X,degree=5,terms=20); od:
> a := expand( f[1] ); A := Maple2SDMP(a,X):
> b := expand(a*f[2]): B := Maple2SDMP(b,X):
> c := expand(b*f[3]): C := Maple2SDMP(c,X):
> d := expand(c*f[4]): D := Maple2SDMP(d,X):
> Q,R := SMPDivide(D,A): R;
> Q,R := SMPDivide(D,B): R;
> Q,R := SDMDivide(D,C): R;
```

And repeat the above experiment on the following dense example.

```
> c := proc() randpoly(y,dense,degree=19) end;
> a := add( c()*x^i, i=0..19 ):  A := Maple2SDMP(a,[x,y]):
> b := add( c()*x^i, i=0..19 ):
> c := collect(a*b, x, expand):  C := Maple2SDMP(c,[x,y]):
> SMPDivide(C,A);
```