MATH 895, Assignment 2, Fall 2021

Instructor: Michael Monagan

Please hand in the assignment by 11pm Saturday October 2nd.
Late Penalty -20% off for up to 36 hours late. Zero after that.
Please export your Maple worksheet(s) containing your Maple code and Maple output to a .pdf file and upload your pdf file(s) to Crowdmark.

Download and read the paper "Sparse Polynomial Arithmetic" by Stephen Johnson. Notice that Johnson's paper assumes univariate polynomials only. One can map a multivariate polynomial $f(x, y, z)$ into a univariate polynomial $g(x)$ by means of the Kronecker substitution: $g := subs(y = x^j, z = x^k, f)$ for sufficiently large j, k in such a way that one can recover $f(x, y, z)$ from $g(x)$. We will use a different monomial to integer encoding.

Let $A, B \in \mathbb{Q}[x, y, z, ...]$ and let $C = A \times B$ and let Q be the quotient of C divided B. 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 encoded as an integer as follows: the monomial $x^i y^j z^k$ with exponent vector $[i, j, k]$ would be represented as the integer $e = (i + j + k)B^3 + iB^2 + jB + k$ where $B = 2^L$ bounds the total degree d of any monomial that appears in the multiplication/division algorithm.

Implement the following Maple procedures where X is a list of variables.

```
A := Maple2SDMP(a,X,B);
a := SDMP2Maple(A,X,B);
```

E.g. `A := Maple2SDMP(a,[x,y,z],B)` converts a Maple polynomial $a(x, y, z)$ into the SDMP data structure and `SDMP2Maple(A,[x,y,z])` converts it back. Note, to convert an integer E to base B in Maple use `convert(E,base,B);` Note, to sort the terms in a polynomial you can use the `sort` command. Now implement the following three algorithms.

1 Classical repeated merging: $f \times g = ((f_1 g + f_2 g) + f_3 g) + \ldots + f_m g$ where $m = \#f$.

2 Johnson's m-way merge using a heap: $f \times g = \sum_{i=1}^{m} f_i \times g$.

3 Divide and conquer multiplication: $f \times g = (f_1 g + \ldots + f_k g) + (f_{k+1} g + \ldots + f_m g)$ where $k = \lfloor m/2 \rfloor$. Use merging for $+$.

Execute your algorithms on the following sparse problems

```
> X := [u,v,w,x,y,z];
> a := randpoly(X,degree=10,terms=5000):
> b := randpoly(X,degree=10,terms=50):
> c := expand(a*b):
> nops(a), nops(b), nops(c);
                                    4977, 49, 127191

> d := degree(a)+degree(b);
                                          20
> B := 1000:
> A := Maple2SDMP(a,X,B):
> B := Maple2SDMP(b,X,B); # show your data structure for this one
> C := Maple2SDMP(c,X,B):
> H := MULTIPLY(A,B): evalb(H=C);
> H := MULTIPLY(B,A): evalb(H=C);
```

You may simply use $B = 2^{10}$ for this experiment.

Compute and print (i) $N =$ the number of monomial comparisons each algorithm makes, (ii) $M =$ the number of coefficient multiplications each algorithm makes and (iii) the quantity $S = N/M$ which measures the monomial comparisons relative to the coefficient arithmetic cost. Now, estimate the theoretical number of monomial comparisons that the three algorithms should make for these inputs. Compare these with the actual values.

For the heap operations you may use Maple's `heap` package. See `?heap`.

To count the number of comparisons done in the heap insertions and extractions, use a global variable like this:

```
> less := proc(a,b) global N; N := N+1; evalb( a[2]<b[2] ) end;
> H := heap[new](less);
> N := 0; # don't forget to initialize it
```

In several places you will need an array for a result with an unknown number of terms. Use a Maple hash table and keep a counter for how many terms are in the table. E.g.

```
> C := table();
> C[1] := 3*x;
> C[2] := 2*x^2;
> n := 2;
> C[2] := C[2]+3*x^2;
> c := [seq(C[i],i=1..n)]; # convert to list
```