MATH 895, Assignment 4, Fall 2023

Instructor: Michael Monagan

Please hand in the assignment by 11pm Tuesday October 31st.
Late Penalty -20% off for up to 24 hours late. Zero after that.

Question 1 (20 marks): Term Orderings and Division

Do this question by hand.

(a) Let $\alpha, \beta, \gamma \in \mathbb{Z}^n_{\geq 0}$. Let $>$ be the lexicographical monomial ordering on $\mathbb{Z}^n_{\geq 0}$. From the definition of lexicographical order prove that $\alpha > \beta \Longrightarrow \alpha + \gamma > \beta + \gamma$.

(b) Consider the following total ordering in $\mathbb{Q}[x]$.

$$1 > x > x^2 > x^3 > \cdots$$

Prove or disprove that this ordering is a monomial ordering.

(c) Consider the polynomials

$$a = 3y^7 + 6x^2y^4 + 14x\,y^5 + 10x^3y^2 + 15x^2y^3 + 7x\,y^4 + 14x^3y + 21x^2y^2$$

$$b = 3xy + y^3 + 2x^2$$

Divide a by b over $\mathbb{Z}$. The remainder is 0. Do the division first using lexicographical order with $x > y$ then using graded lexicographical order with $x > y$.

(d) Let $<$ be a monomial ordering. In the division algorithm for $a \div b$ we tested if $LT(b)$ divides $LT(a)$ and if so we computed $t = LT(a)/LT(b)$ then $a - tb$. We claimed either $a - tb = 0$ or $LM(a - tb) < LM(a)$. Prove this.

Question 2 Polynomial Multiplication Algorithms (30 marks)

We will represent a polynomial in $\mathbb{Q}[x_1, x_2, \ldots, x_n]$ as a Maple list of terms sorted in descending *graded lexicographical* order. Each term is a Maple list $[c, e]$ where $c \in \mathbb{Q}$ is the coefficient and e, the exponent vector, is a list of integers.

The code in the Appendix for Maple2SDMP and SDMP2Maple converts a Maple polynomial to a list of terms in this representation. There is also a procedure `Greater` for comparing the exponent vectors of two monomials that you can use. `Greater(u,v)` outputs true if $u > v$ and false otherwise. Use these as follows

```
> a := 3*x^2*y^2*z+2*y^3*z-5*x*y*z;
```

$$a := 3x^2y^2z + 2y^3z - 5xyz$$

```
> A := Maple2SDMP(a,[x,y,z]);
```

$$[[3,[2,2,1]],[2,[0,3,1]],[-5,[1,1,1]]]$$

```
> SDMP2Maple(A,[x,y,z]);
```

$$a := 3x^2y^2z + 2y^3z - 5xyz$$

Implement the following three multiplication algorithms for $f \times g$ in Maple for $\mathbb{Q}[x_1, x_2, ..., x_n]$.

1 Classical repeated merging: $f \times g = ((f_1 g + f_2 g) + f_3 g) + \ldots + f_m g$ where $m = \#f$ where each addition is done by a merge. So you must first program addition using the merge algorithm. To do this you may modify the merge code that I wrote for one variable (see handout).

2 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 $+$. The algorithm must be recursive.

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

Execute your three 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): # the answer
> nops(a), nops(b), nops(c);
                                  4977, 49, 127191

> d := degree(a)+degree(b);
                                        20
> A := Maple2SDMP(a,X):
> B := Maple2SDMP(b,X):
> C := Maple2SDMP(c,X):
> H := MULTIPLY(A,B): evalb(H=C);
> H := MULTIPLY(B,A): evalb(H=C);
```

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. You should find that the heap algorithm and divide-and-conquer algorithm do fewer comparisons than the repeated merging algorithm.

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:

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

In several places you will need an array for an intermediate 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;
```

$$C_1 := 3\,x$$

```
> C[2] := 2*x^2;
```

$$C_2 := 2\,x^2$$

```
> C[2] := C[2]+3*x^2;
```

$$C_2 := 5\,x^2$$

```
> n := 2: # number of terms
> [seq(C[i],i=1..n)]; # convert to list
```

$$[3\,x,\ 5\,x^2]$$

Appendix

Code and examples for Maple2SDMP and SDMP2Maple

```
Greater := proc(a::list(nonnegint),b::list(nonnegint))
# compare two monomials a and b input as exponent vectors in graded monomial ordering
local dega,degb,n,i,z;
   n := nops(a);
   if nops(b)<>n then error "exponent vectors of different sizes" fi;
   dega := add(z,z in a); # = deg(a)
   degb := add(z,z in b); # = deg(b)
   if dega>degb then return true elif dega<degb then return false fi;
   for i to nops(a) do # compare with lexicographical order
       if a[i]>b[i] then return true;
       elif a[i]<b[i] then return false;
       fi;
   od;
   false;
end:

u := [3,1,2];
v := [3,2,1];
Greater(u,v);

Maple2SDMP := proc(a,X::list(name))
local C,M,n,t,A,E,i,m;
   if not type(a,polynom(rational,X)) then error "bad input" fi;
   C := [coeffs(a,X,'M')];
   M := [M];
   n := nops(X);
   t := nops(M);
   E := [seq( [seq(degree(m,X[i]),i=1..n)], m in M)]; # exponent vectors
   A := [seq([C[i],E[i]],i=1..t)];
   sort(A,proc(x,y) Greater(x[2],y[2]) end);
end:
SDMP2Maple := proc(A::list([rational,list(nonnegint)]),X::list(name))
local t,i,n;
   n := nops(X);
   add(t[1]*mul(X[i]^t[2][i],i=1..n),t in A);
end:

a := 3*x^2*y^2*z+2*y^3*z-5*x*y*z;
A := Maple2SDMP(a,[x,y,z]);
SDMP2Maple(A,[x,y,z]);
b := 3/2*x^2+5/2*x*y^2-7/2*y^4;
B := Maple2SDMP(b,[x,y]);
SDMP2Maple(B,[x,y]);
```

Output from executing the previous code in Maple

```
    |\^/|     Maple 2022 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2022
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.

> Greater := proc(a::list(nonnegint),b::list(nonnegint))
# compare two monomials a and b input as exponent vectors
# in the graded monomial ordering
> local dega,degb,n,i,z;
>    n := nops(a);
>    if nops(b)<>n then error "exponent vectors of different sizes" fi;
>    dega := add(z,z in a); # = deg(a)
>    degb := add(z,z in b); # = deg(b)
>    if dega>degb then return true elif dega<degb then return false fi;
>    for i to nops(a) do # compare with lexicographical order
>        if a[i]>b[i] then return true;
>        elif a[i]<b[i] then return false;
>        fi;
>    od;
>    false;
> end:

> u := [3,1,2];
                                u := [3, 1, 2]

> v := [3,2,1];
                                v := [3, 2, 1]

> Greater(u,v);
                                    false


> Maple2SDMP := proc(a,X::list(name))
> local C,M,n,t,A,E,i,m;
>    if not type(a,polynom(rational,X)) then error "bad input" fi;
>    C := [coeffs(a,X,'M')];
>    M := [M];
>    n := nops(X);
>    t := nops(M);
>    E := [seq( [seq(degree(m,X[i]),i=1..n)], m in M)]; # exponent vectors
>    A := [seq([C[i],E[i]],i=1..t)];
>    sort(A,proc(x,y) Greater(x[2],y[2]) end);
> end:

> SDMP2Maple := proc(A::list([rational,list(nonnegint)]),X::list(name))
```

```
> local t,i,n;
>    n := nops(X);
>    add(t[1]*mul(X[i]^t[2][i],i=1..n),t in A);
> end:

> a := 3*x^2*y^2*z+2*y^3*z-5*x*y*z;
                                2  2        3
                      a := 3 x  y  z + 2 y  z - 5 x y z

> A := Maple2SDMP(a,[x,y,z]);
            A := [[3, [2, 2, 1]], [2, [0, 3, 1]], [-5, [1, 1, 1]]]

> SDMP2Maple(A,[x,y,z]);
                             2  2        3
                        3 x  y  z + 2 y  z - 5 x y z

> b := 3/2*x^2+5/2*x*y^2-7/2*y^4;
                                  2          2        4
                       b := 3/2 x  + 5/2 x y  - 7/2 y

> B := Maple2SDMP(b,[x,y]);
              B := [[-7/2, [0, 4]], [5/2, [1, 2]], [3/2, [2, 0]]]

> SDMP2Maple(B,[x,y]);
                               2          2        4
                          3/2 x  + 5/2 x y  - 7/2 y

> quit
memory used=1.0MB, alloc=8.3MB, time=0.05
```