MATH 800, Assignment 2, Fall 2023

Instructor: Michael Monagan

Due 11pm Tuesday October 10th

Question 1 (12 marks)

Part (a) Use Maple to compute and simplify the following sums. The first sum is for the number of multiplications of the forward elimination step of Gaussian elimination for an n by m matrix. The second sum is for the number of multiplications of the Gentleman/Johnson algorithm for computing the deterinant of an n by n matrix. Use the `sum(f(k),k=m..n)` command.

$$\sum_{k=1}^{n} \sum_{i=k+1}^{n} \sum_{j=k+1}^{m} 1 \quad \text{and} \quad \sum_{i=2}^{n} \binom{n}{i} i.$$

Part (b) By hand, for moduli $m_1 = 5, m_2 = 6, m_3 = 7$ and images $u_1 = 2, u_2 = 3, u_4 = 2$ find the integer u s.t. $u \equiv u_i \bmod m_i$ for $1 \leq i \leq 3$ and $0 \leq u < m_1 m_2 m_3$. Use the mixed radix representation for u, namely, $u = v_1 + v_2 m_1 + v_3 m_1 m_2$ where $0 \leq v_i < m_i$.

Part (c) Consider the Lagrange representation for the integer u in part (b), namely, $u = v_1 m_2 m_3 + v_2 m_1 m_3 + v_3 m_1 m_2$. Find integers v_1, v_2, v_3 such that $u \equiv u_i \bmod v_i$ and $0 \leq v_i < m_i$ for the problem in part (b).

Part (d) The Lagrange integer u will not always satisfy $0 \leq u < m_1 m_2 m_3$. How big can u be for $m_1 = 5, m_2 = 6, m_3 = 7$? Find u_1, u_2, u_3 that maximize u.

Question 2: The Bareiss/Edmonds Algorithm

Part (a) (8 marks)

For an n by n matrix A with integer entries, implement the Bareiss/Edmonds algorithm as the Maple procedure `FFGE(A,n)` that outputs $\det(A)$.
Letting $A_{00} = 1$, assuming pivoting is not needed, the algorithm is

for $k = 1, 2, 3, ..., n-1$ do
  for $i = k+1, k+2, ..., n$ do
    for $j = k+1, k+2, ..., n$ do
      $A_{ij} := \dfrac{A_{kk} A_{ij} - A_{ik} A_{kj}}{A_{k-1k-1}}$
    end for
  end for
  $A_{ik} := 0$
end for.

Use the `iquo(...)` command for the integer division.

You will need to take care of pivoting: if at any step k, the matrix entry $A_{k,k} = 0$ and $A_{i,k} \neq 0$ for some $k < i \leq n$, then interchange row k with row i before proceeding. Remember, interchanging two rows of a matrix changes the sign of the determinant.

Test the algorithm on the following random integer matrices. Please print out the matrix A after executing the algorithm (the matrix A is updated by the algorithm) and check that $A_{n,n} = \pm\det(A)$.

```
> c := rand(10^4):
> for n from 3 to 4 do
>     A := Matrix(n,n,c);
>     FFGE(A);
>     A;
> od;
```

For parts (b) and (c) we will investigate the intermediate expression swell that occurs if we use the Bareiss/Edmonds algorithm to compute the determinant of a matrix with polynomial entries. For our experiment we will use the generic symmetric matrix $S_n(x)$. Below is the generic 3 by 3 symmetric matrix.

$$S_3 = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_4 & x_5 \\ x_3 & x_5 & x_6 \end{bmatrix}$$

Now $\det S_3(x) = x_1x_4x_6 - x_1x_5^2 - x_2^2x_6 + 2x_2x_3x_5 - x_3^2x_4$ which is a polynomial in $x_1, x_2, ..., x_6$. We will run the Bareiss/Edmonds algorithm in the integral domain $\mathbb{Z}[x_1, x_2, \ldots, x_6]$.

Part (b) (5 marks)

First write a Maple procedure `GenSymMat(n,x)` that creates a generic n by n symmetric matrix $S_n(x)$. Now use Maple's Determinant command from the LinearAlgebra package to compute the number of terms of $\det S_n(x)$ for $n = 3, 4, 5, ..., 9$. Do not print out the determinants $S_n(x)$ as they are very big! To compute $\#f$, the number of terms of a polynomial f, use this command.

```
numterms := proc(f) if f=0 then 0 elif type(f,`+`) then nops(f) else 1 fi end;
```

Part (c) (5 marks)

Modify your implementation of the Bareiss/Edmonds algorithm from part (a) to work for polynomials in $\mathbb{Z}[x_1, x_2, ..., x_m]$. You just need to expand the numerator $a = A_{kk}A_{ij} - A_{ik}A_{kj}$ and, for the division by $b = A_{k-1k-1}$, use the divide command `divide(a,b,'q');` which returns true if $b|a$ and false otherwise. If $b|a$ it assigns q the quotient of $a \div b$. Test your algorithm on $S_3(x)$.

The largest expression swell occurs at the final step when $k = n-1, i = n, j = n$ where we compute $A_{nn} = \pm\det(A)$ using

$$A_{nn} := \frac{A_{n-1n-1}A_{nn} - A_{nn-1}A_{n-1n}}{A_{n-2n-2}}.$$

Let N be the numerator in the fraction. We have

$$N = A_{nn}A_{n-2n-2} = \pm\det(A)A_{n-2n-2}$$

Modify your code to compute the expression swell, that is, how much bigger N is than $\det A$. Compute and print out $\#A_{n-2n-2}$, $\#A_{nn}$, $\#N$ and the expression swell $\#N/\#A_{nn}$. Do this for S_n for $n = 3, 4, 5, 6, 7, 8$. Do not try $n = 9$ unless your computer has 100 gigabytes of RAM!

For parts (d) and (e) let F be a field, $D = F[x]$ and A be an n by n matrix over D. We will compare two algorithms for computing $\det A$, the Bareiss/Edmonds algorithm and an algorithm based on interpolation to see which is fastest.

Part (d) (9 marks)

If we assume $\deg(A_{i,j}) \le d$ and classical quadratic algorithms are used for polynomial multiplication and exact division in $F[x]$, how many multiplications in F does the Bareiss/Edmonds algorithm do in the worst case?

Try to get an exact formula in terms of n and d assuming $\deg(A_{i,j}) = d$. I suggest you do this for a 3x3 matrix first. Note, to divide a polynomial in $F[x]$ of degree d by a polynomial of degree $m \le d$, the classical division algorithm does $\le m(d-m+1)$ multiplications in F.

Use Maple's `sum(...)` command to evaluate any sums that you need.
You should get a polynomial in n and d of degree 6.

Part (e) (6 marks)

Consider the following algorithm for computing $\det A$ where the entries of A are in $F[x]$. Assume again that $\deg(A_{ij}) \le d$. We have $\deg(\det(A)) \le nd$. Assume also that the field F satisfies $|F| > nd$.

1. Pick $nd+1$ distinct points $\alpha_0, \alpha_1, \ldots, \alpha_{nd}$ from F.
2. Compute $B_i = A(x = \alpha_i)$ for $0 \le i \le nd$.
3. Compute $y_i = \det(B_i)$ using Gaussian elimination over F.
4. Interpolate x in $\det A$.

How many multiplications in F does this algorithm do?
Does it do fewer multiplications than the Bareiss/Edmonds algorithm?

Question 3: Solving $Ax = b$ using p-adic lifting and rational reconstruction.

Part (a) (6 marks)

Implement Wang's rational number reconstruction algorithm as the Maple procedure `WangRNR(u,m,N,D)`. Assume $2ND < m$. To do this just modify my Maple code in the handout for the extended Euclidean algorithm (attached). If the gcd condition does not hold then return the value FAIL. Execute Wang's algorithm on the following input

```
> m := 35;
> r := [ seq( WangRNR(u,m,4,4), u=0..m-1 ) ];
```

Observe that all rationals n/d satisfying $|n| \leq 4$ and $0 < d \leq 4$ appear once in r and no other fractions do.

Part (b) (9 marks)

Let $A \in \mathbb{Z}^{n\times n}$ and $b \in \mathbb{Z}^n$. In class we studied an algorithm for solving $Ax = b$ for $x \in \mathbb{Q}^n$ using linear $p-$adic lifting and rational number reconstruction. Implement the algorithm in Maple as the procedure `PadicLinearSolve(A,b)`. Use the prime $p = 2^{31} - 1$. Your procedure should return the solution vector x and also print out the number of lifting steps k that are required. Test your implementation on the following examples. The first has large rationals in the solution vector. The second has very small rationals.

```
> with(LinearAlgebra):
> B := 2^16;
> m := 3;
> U := rand(B^m);
> n := 50;
> A := RandomMatrix(n,n,generator=U);
> b := RandomVector(n,generator=U);
> x := padicLinearSolve(A,b);
> convert( A.x-b, set ); # should be {0}
> y := [1,0,-1/2,2/3,4,3/4,-2,-3,0,-1];
> y := map( op, [y$5] );
> x := Vector(y);
> b := A.x;
> A,b := 12*A,12*b; # clear fractions
> x := padicLinearSolve(A,b);
> convert( A.x-b, set ); # should be {0}
```

To compute A^{-1} mod p use the Maple command `Inverse(A) mod p`.
The `Inverse` command runs Gaussian elimination on the bookkeeping matrix $[A|I]$ over the field $\mathbb{Z}_p$. To multiply A times a vector x over $\mathbb{Z}$ use `A.x` in Maple.

For rational number reconstruction use the Maple command `iratrecon`. Note, if u is a vector of integers modulo m, `iratrecon(u,m)` will automatically apply rational reconstruction to each entry in u separately.

The Extended Euclidean Algorithm

```
>
> EEA := proc(m,u) local s,t,r,q,i;
>   r[0],r[1] := m,u;
>   # s[0],s[1] := 1,0;
>   t[0],t[1] := 0,1;
>   printf("\n");
>   printf("%4s %10s %10s %10s %12s\n","i","r[i]","t[i]","q[i+1]","r[i]/t[i]");
>   for i from 1 while r[i]<>0 do
>     q[i+1] := iquo(r[i-1],r[i]);
>     r[i+1] := r[i-1]-q[i+1]*r[i];
>     # s[i+1] := s[i-1]-q[i+1]*s[i];
>     t[i+1] := t[i-1]-q[i+1]*t[i];
>     printf("%4d %10d %10d %10d %12a\n",i,r[i],t[i],q[i+1],r[i]/t[i]);
>   od:
> end:
>
> m := 10^6-17;
                                  m := 999983

> u := 72/109 mod m;
                                  u := 137613

> EEA(m,u);

   i       r[i]       t[i]     q[i+1]    r[i]/t[i]
   1     137613          1          7       137613
   2      36692         -7          3     -36692/7
   3      27537         22          1     27537/22
   4       9155        -29          3    -9155/29
   5         72        109        127       72/109
   6         11     -13872          6   -11/13872
   7          6      83341          1     6/83341
   8          5     -97213          1    -5/97213
   9          1     180554          5     1/180554
```