Apéndice B
Programa PARI/GP

Nota: este texto será expandido durante el curso.

En nuestro curso aparecen varios ejemplos que, como una gran parte de las matemáticas interesantes en general, provienen de la teoría de números. Uno de los mejores programas para la teoría de números computacional es PARI/GP que se está desarrollando en la universidad de Burdeos (Francia). Invito al lector a descargar el programa de su página oficial

http://pari.math.u-bordeaux.fr/

antes de leer este apéndice.

PARI/GP es un programa muy poderoso y entre otras cosas cuenta con su propio lenguaje de programación. Para más detalles, consulte la documentación. El lector interesado en los algoritmos empleados por PARI/GP puede empezar por el libro de texto [Coh1993].

B.1 Sesión interactiva

Para abrir la calculadora PARI/GP, hay que ejecutar en el terminal el comando `gp`. El símbolo ? al inicio de la línea significa que el programa nos invita a digitar un comando o expresión para evaluar y presionar la tecla Enter. Luego PARI/GP hace los cálculos necesarios e imprime el resultado. Las salidas se etiquetan por `%1`, `%2`, `%3`, etc. con el propósito de usar los resultados de cálculos más adelante. Para usar el último resultado, se puede escribir simplemente `%`.

```
? 10!
%1 = 3628800
? 1/2 + 1/3 + 1/4
%2 = 13/12
? binomial(5,2)
%3 = 10
? %^2
%4 = 100
? %1/(5*6*7*8*9*10)
%5 = 24
```

Las teclas ↑ y ↓ pueden ser usadas para ver los comandos digitados previamente, y la tecla Tab completa los comandos. Por ejemplo, podemos digitar `sqr` y presionar Tab para ver todos los comandos cuyo nombre empieza por "`sqr`".

Para asignar el valor c a una variable x, hay que escribir `x = c`. Para olvidar la variable, hay que ejecutar el comando `kill (x)`. Cuando no nos interesa ver la salida, se puede terminar la expresión por el punto y coma "`;`". En este caso el valor de la expresión no aparecerá.

```
? x = 1;
? x^2 + x + 1
% = 3
? kill (x)
? x^2 + x + 1
% = x^2 + x + 1
```

A veces algo va mal en los cálculos: la expresión contiene un error de digitación o la entrada es incorrecta. Por ejemplo, tratemos de calcular la raíz cuadrada de -1 módulo 11:

```
? sqrt (Mod (-1,11))
  ***   at top-level: sqrt(Mod(-1,11))
  ***                 ^----------------
  *** sqrt: not an n-th power residue in gsqrt: Mod(10, 11).
  ***   Break loop: type 'break' to go back to GP prompt
break>
```

PARI/GP nos señala que hay un error: el número -1 no es un cuadrado módulo 11. El programa entró en un ciclo de interrupción (*break loop*) que sirve principalmente para corregir errores en código del usuario. Para volver al modo normal, hay que ejecutar el comando `break` o simplemente presionar Ctrl + D.

Para salir del programa, use el comando `\q`.

B.2 Ayuda

Un comando muy útil es "`?`". Al ejecutarlo, PARI/GP muestra una lista de temas.

```
? ?
Help topics: for a list of relevant subtopics, type ?n for n in
   0: user-defined functions (aliases, installed and user functions)
   1: PROGRAMMING under GP
   2: Standard monadic or dyadic OPERATORS
   3: CONVERSIONS and similar elementary functions
   4: functions related to COMBINATORICS
   5: NUMBER THEORETICAL functions
   . . . . .
```

Esto quiere decir que, por ejemplo, si queremos consultar las funciones relacionadas con la teoría de números, podemos escribir `?5`. Al hacerlo sale un larga lista de funciones. Allí aparecen por ejemplo `divisors` y `numdiv`. Si queremos saber a qué sirven estas funciones, hay que digitar `?divisors` y `?numdiv` respectivamente.

```
? ?divisors
divisors(x,{flag=0}): gives a vector formed by the divisors of x in increasing order.
If flag = 1, return pairs [d, factor(d)].

? ?numdiv
numdiv(x): number of divisors of x.
```

Entonces, el programa nos dice que `divisors(`x`)` calcula la lista de divisores de x y `numdiv(`x`)` calcula el número de divisores de x.

```
? divisors (20)
% = [1, 2, 4, 5, 10, 20]
? numdiv (20)
% = 6
```

B.3 Números enteros y racionales

Para los números enteros están definidas las operaciones aritméticas habituales, como m+n, m-n, m*n, m^n (el número m^n), $n!$ (el factorial). El número racional $\frac{a}{b}$ se denota por a/b. He aquí algunas funciones útiles.

- `binomial(`m`,`n`)` devuelve el coeficiente binomial $\binom{m}{n}$. Por ejemplo, `binomial(6,2)` devuelve `15`.
- `isprime(`n`)` verifica si n es primo. Por ejemplo, `isprime(2^521-1)` devuelve `1` porque el número $2^{521}-1$ es primo, mientras que `isprime(2^520-1)` devuelve `0`: el número $2^{520}-1$ no es primo: este es divisible por 3:
$$2^{520}-1 \equiv (-1)^{520}+2 \equiv 0 \pmod 3.$$
- `primes(`n`)` devuelve la lista de los primeros n primos. Por ejemplo, `primes(10)` devuelve
`[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]`
- `divisors(`n`)` devuelve la lista de los divisores de n. Por ejemplo, `divisors(40)` devuelve
`[1,2,4,5,8,10,20,40]`
- `factor(`n`)` devuelve los factores primos de n. Por ejemplo,

```
? factor (-180)
% =
[-1 1]

[ 2 2]

[ 3 2]

[ 5 1]
```

La salida es una matriz de $s \times 2$ donde en cada fila está un factor irreducible y el exponente correspondiente. En la primera fila también puede estar un factor invertible (en el caso de los números enteros, esto significa que allí estará -1 si el número es negativo).

- `valuation(`n`,`p`)` devuelve la valuación p-ádica de n.

```
? valuation (180,2)
% = 2
? valuation (180,3)
% = 2
? valuation (18/25,2)
% = 1
? valuation (18/25,3)
% = 2
? valuation (18/25,5)
% = -2
```

- `eulerphi(`n`)` calcula la función de Euler $\phi(n)$. Por ejemplo,

```
? eulerphi(4)
% = 2
? eulerphi(25)
% = 20
? eulerphi(100)
% = 40
```

B.4 Números reales y complejos

Para trabajar con aproximaciones a los números reales, se usan los números con punto flotante. Para los números complejos, se usa el símbolo `I` que denota la unidad imaginaria. Note que esta es la `I` mayúscula.

Están predefinidas varias funciones como `sqrt(`z`)` (raíz cuadrada), `sin(`z`)` (seno), `cos(`z`)` (coseno), `exp(`z`)` (exponencial), `log(`z`)` (logaritmo natural), etc. La constante `Pi` (con la P mayúscula) corresponde al valor aproximado del número π.

```
? Pi
% = 3.1415926535897932384626433832795028842

? sqrt (-2)
% = 1.4142135623730950488016887242096980786*I

? cos (2.3)^2 + sin (2.3)^2
% = 1.0000000000000000000000000000000000000

? exp (2*Pi*I/3) + exp (4*Pi*I/3)
% = -1.0000000000000000000000000000000000000 + 5.877471754111437540 E-39*I
```

No vamos a enumerar todas las posibles funciones relacionadas con los números reales y complejos; el lector puede revisar la documentación de PARI/GP (en particular la sección “Transcendental functions”).

Para cambiar la precisión, se puede usar el comando `\p`.

```
? \p 10
   realprecision = 19 significant digits (10 digits displayed)
? Pi
% = 3.141592654
? sin (1)
% = 0.8414709848
? \p 50
   realprecision = 57 significant digits (50 digits displayed)
? Pi
% = 3.1415926535897932384626433832795028841971693993751
? sin (1)
% = 0.84147098480789650665250232163029899962256306079837
```

Para convertir un número racional x en un número con punto flotante, podemos escribir `x*1.0`. Por ejemplo,

```
? x = 2/3
% = 2/3
? x*1.0
% = 0.66666666666666666666666666666666666666667
? 2.0/3
% = 0.66666666666666666666666666666666666666667
```

B.5 Anillos $\mathbb{Z}/n\mathbb{Z}$

El resto de a módulo n se denota por `Mod(`a`,`n`)`. Para estos restos están definidas las operaciones aritméticas habituales.

```
? Mod (4,10) + Mod (6,10)
% = Mod(0, 10)
? Mod (6,10)^2
% = Mod(6, 10)
? Mod (3,10)^-1
% = Mod(7, 10)
```

- `issquare(Mod(`a`,`n`))` verifica si a es un cuadrado módulo n.

```
? issquare (Mod (-1,11))
% = 0
? issquare (Mod (-1,13))
% = 1
? sqrt (Mod (-1,13))
% = Mod(5, 13)
```

- `kronecker(a,p)` calcula el símbolo de Legendre

$$\left(\frac{a}{p}\right) = \begin{cases} +1, & p \nmid a \text{ y } a \text{ es un cuadrado módulo } p, \\ -1, & p \nmid a \text{ y } a \text{ no es un cuadrado módulo } p, \\ 0, & p \mid a. \end{cases}$$

(De hecho, como sugiere su nombre, esta función calcula el símbolo de Kronecker que es una generalización del símbolo de Legendre, donde p no es necesariamente primo.)

```
? kronecker (3,5)
% = -1
? kronecker (5,3)
% = -1
```

B.6 Enteros de Gauss $\mathbb{Z}[i]$

Funcionan todas las operaciones aritméticas con los enteros de Gauss que son los números de la forma `a + b*I`, donde a y b son enteros. Al dividir dos enteros de Gauss, se devuelve una expresión de la forma `x + y*I`, donde $x, y \in \mathbb{Q}$. Por ejemplo,

```
? (2+3*I)/(3+2*I)
% = 12/13 + 5/13*I
? (5-5*I)/(1+I)
% = -5*I
```

Para factorizar un entero de Gauss $\alpha \in \mathbb{Z}[i]$, hay que escribir `factor(α,I)`. Por ejemplo,

```
? factor (180,I)
% =
[      I 1]

[      3 2]

[  1 + I 4]

[  2 + I 1]

[1 + 2*I 1]
```

B.7 Polinomios

Un polinomio $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$ se escribe como

$$a_n*x\hat{}n \;+\; a_{n-1}*x\hat{}(n-1) \;+\; \cdots \;+\; a_2*x\hat{}2 \;+\; a_1*x \;+\; a_0$$

Los coeficientes a_i pueden ser números enteros, racionales, reales, complejos, números módulo n, etc. En lugar de x puede estar cualquier otra letra sin valor asignado[*] como a, t etc. o de hecho cualquier palabra.

[*] Recordemos que para olvidar el valor asignado a x, hay que escribir `kill(x)`.

Es nada más una cuestión de estilo, pero PARI/GP usa las letras minúsculas para denotar las variables. Por esto vamos a romper con nuestra tradición de usar la X mayúscula para la variable de un polinomio.

```
? (x+1)^4
% = x^4 + 4*x^3 + 6*x^2 + 4*x + 1
? (x+y)^4
% = y^4 + 4*x*y^3 + 6*x^2*y^2 + 4*x^3*y + x^4
```

Notamos que PARI/GP no está dirigido a cálculos con polinomios en diversas variables. Para este propósito hay otros programas especializados, como por ejemplo Macaulay2 (http://macaulay2.com/).

PARI/GP también trabaja con el cuerpo de fracciones

$$k(x) := \operatorname{Frac} k[x] := \left\{ \frac{f}{g} \;\middle|\; f, g \in k[x],\ g \neq 0 \right\}.$$

Las fracciones de polinomios se simplifican según lo esperado:

```
? (1-x^2)/(1-x)^2
% = (-x - 1)/(x - 1)
```

La función `polcyclo(`n`)` devuelve el polinomio ciclotómico Φ_n en la variable `x`. He aquí un par de ejemplos.

```
? polcyclo(105)
% = x^48 + x^47 + x^46 - x^43 - x^42 - 2*x^41 - x^40 - x^39 + x^36 + x^35
+ x^34 + x^33 + x^32 + x^31 - x^28 - x^26 - x^24 - x^22 - x^20 + x^17
+ x^16 + x^15 + x^14 + x^13 + x^12 - x^9 - x^8 - 2*x^7 - x^6 - x^5
+ x^2 + x + 1

? polcyclo(1)*polcyclo(2)*polcyclo(5)*polcyclo(10)
% = x^10 - 1
```

Si queremos evaluar un polinomio en variable x en $x = c$, podemos escribir

`subst(`f`,`x`,`c`)`

Por ejemplo, si queremos calcular los coeficientes del polinomio $\Phi_9(x+1)$, podemos hacer lo siguiente.

```
? subst (polcyclo(9),x,x+1)
% = x^6 + 6*x^5 + 15*x^4 + 21*x^3 + 18*x^2 + 9*x + 3
```

B.8 Series formales

Si queremos indicar que no se trata de un polinomio, sino de una serie formal o serie de Laurent con términos hasta $a_{n-1}\,x^{n-1}$, hay que añadir a la expresión "`+ O (x^n)`". Por ejemplo, `1/(1-x)` corresponde a la fracción $\frac{1}{1-x}$, mientras que `1/(1-x) + O (x^10)` nos dará los primeros 10 coeficientes (es decir, $a_0, a_1, \ldots, a_9$) de la serie inversa a $1-x$. He aquí algunos cálculos de este tipo.

```
? 1/(1-x) + O (x^10)
% = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + O(x^10)

? 1/(1-x)^2 + O (x^10)
% = 1 + 2*x + 3*x^2 + 4*x^3 + 5*x^4 + 6*x^5 + 7*x^6 + 8*x^7 + 9*x^8 + 10*x^9 + O(x^10)

? 1/(x^4-2*x^3+x^2) + O (x^10)
% = x^-2 + 2*x^-1 + 3 + 4*x + 5*x^2 + 6*x^3 + 7*x^4 + 8*x^5 + 9*x^6 + 10*x^7
+ 11*x^8 + 12*x^9 + O(x^10)
```

Las series formales como `sin(x)`, `cos(x)`, `tan(x)`, `cotan(x)`, `exp(x)`, `log(1+x)`, etc. están predefinidas. Para cambiar la precisión a n términos, hay que escribir

`default(seriesprecision,`n`)`

He aquí un ejemplo.

```
? log (1+x)
% = x - 1/2*x^2 + 1/3*x^3 - 1/4*x^4 + 1/5*x^5 - 1/6*x^6 + 1/7*x^7 - 1/8*x^8 + 1/9*x^9
- 1/10*x^10 + 1/11*x^11 - 1/12*x^12 + 1/13*x^13 - 1/14*x^14 + 1/15*x^15 + O(x^16)
? default(seriesprecision,10)
? log (1+x)
% = x - 1/2*x^2 + 1/3*x^3 - 1/4*x^4 + 1/5*x^5 - 1/6*x^6 + 1/7*x^7 - 1/8*x^8
+ 1/9*x^9 + O(x^10)
? cos (x)
% = 1 - 1/2*x^2 + 1/24*x^4 - 1/720*x^6 + 1/40320*x^8 - 1/3628800*x^10 + O(x^12)
? sin (x)
% = x - 1/6*x^3 + 1/120*x^5 - 1/5040*x^7 + 1/362880*x^9 + O(x^11)
? cos (x)^2 + sin (x)^2
% = 1 + O(x^12)
```

La función `deriv(`f`,`x`)` devuelve la derivada formal de f respecto a la variable x. Aquí f debe ser un polinomio o una serie formal. El segundo argumento se puede omitir si en f aparece una sola variable. De manera similar, `intformal(`f`,`x`)` calcula la integral formal $\int_0^x f(x)\,dx$.

```
? default(seriesprecision,8)
? log (1+x)
% = x - 1/2*x^2 + 1/3*x^3 - 1/4*x^4 + 1/5*x^5 - 1/6*x^6 + 1/7*x^7 + O(x^8)
? deriv (log (1+x),x)
% = 1 - x + x^2 - x^3 + x^4 - x^5 + x^6 + O(x^7)
? % - 1/(1+x)
% = O(x^7)
? intformal (log(1+x))
% = 1/2*x^2 - 1/6*x^3 + 1/12*x^4 - 1/20*x^5 + 1/30*x^6 - 1/42*x^7 + 1/56*x^8 + O(x^9)
? % - ((1+x)*log(1+x) - x)
% = O(x^8)
```

B.9 Factorización de polinomios

La función `factor(`f`)` factoriza el polinomio f en polinomios irreducibles. Aquí f puede tener coeficientes en $\mathbb{Z}$, $\mathbb{Q}$ o $\mathbb{F}_p$. Si queremos considerar un polinomio f con coeficientes enteros como un polinomio mó-

dulo p, hay que escribir f`*Mod(1,`p`)`. Por ejemplo, factoricemos el octavo polinomio ciclotómico $\Phi_8 = x^4+1$ en $\mathbb{F}_p[x]$ para $p = 2,3$:

```
? factor (polcyclo(8)*Mod(1,2))
% =
[Mod(1, 2)*x + Mod(1, 2) 4]

? factor (polcyclo(8)*Mod(1,3))
% =
[Mod(1, 3)*x^2 + Mod(1, 3)*x + Mod(2, 3) 1]

[Mod(1, 3)*x^2 + Mod(2, 3)*x + Mod(2, 3) 1]
```

Para dar otro ejemplo, factoricemos el polinomio $x^8 - x$ en $\mathbb{F}_2[x]$.

```
? factor ((x^8-x)*Mod(1,2))
% =
[                               Mod(1, 2)*x 1]

[                  Mod(1, 2)*x + Mod(1, 2) 1]

[  Mod(1, 2)*x^3 + Mod(1, 2)*x + Mod(1, 2) 1]

[Mod(1, 2)*x^3 + Mod(1, 2)*x^2 + Mod(1, 2) 1]
```

La función `polisirreducible(`f`)` verifica si f es un polinomio irreducible.

```
? f = x^4+x^3+x^2+x+1;
? polisirreducible (f)
% = 1
? polisirreducible (f*Mod(1,2))
% = 1
? polisirreducible (f*Mod(1,11))
% = 0
? factor (f*Mod(1,11))
% =
[Mod(1, 11)*x + Mod(2, 11) 1]

[Mod(1, 11)*x + Mod(6, 11) 1]

[Mod(1, 11)*x + Mod(7, 11) 1]

[Mod(1, 11)*x + Mod(8, 11) 1]
```

El contenido de un polinomio $f \in \mathbb{Q}[x]$ puede ser calculado mediante `content(`f`)`. Por ejemplo, `content(3*x^2 + 9/2*x + 3/4)` devuelve `3/4`.

B.10 División con resto, MCD y MCM

- `divrem(`a`,`b`)` devuelve `[`q`,`r`]`, donde q es el cociente y r es el resto de división. Esto funciona de la misma manera para números enteros $a, b \in \mathbb{Z}$ y para polinomios $f, g \in k[x]$ (recordemos que $\mathbb{Z}$ y $k[x]$

son dominios euclidianos).

```
? divrem (13, 4)
% = [3, 1]~

? divrem (x^2+x, x-1)
% = [x + 2, 2]~
```

- `gcd(a,b)` y `lcm(a,b)` calculan el mcd y mcm respectivamente. De nuevo, a y b pueden ser números enteros o polinomios.

```
? gcd (12,18)
% = 6
? lcm (12,18)
% = 36

? gcd (x^3-1, x^5-1)
% = x - 1
? lcm (x^3-1, x^5-1)
% = x^7 + x^6 + x^5 - x^2 - x - 1
```

- `gcdext(a,b)` calcula la identidad de Bézout: la salida es $[u,v,d]$, donde $d = \mathrm{mcm}(a,b)$ y u, v son elementos tales que $ua + vb = d$.

```
? gcdext (13, 17)
% = [4, -3, 1]

? gcdext (x^3-1, x^5-1)
% = [x^3 + 1, -x, x - 1]
```

B.11 Anillos cociente $k[x]/(f)$

Un caso muy importante de los anillos cociente son $k[x]/(f)$, donde k es un cuerpo y $f \in k[x]$ es un polinomio. Si f es irreducible, entonces el cociente $k[x]/(f)$ es un cuerpo. En todo caso diferentes elementos del cociente se representan por los polinomios de grado $< n$:

$$a_{n-1}\,x^{n-1} + \cdots + a_1\,x + a_0, \quad a_i \in k,\ n = \deg f.$$

Un polinomio g módulo f se denota en PARI/GP por `Mod(g,f)`.

Por ejemplo, si queremos calcular diferentes potencias $(1+\sqrt{2})^n$ en el cuerpo

$$\mathbb{Q}(\sqrt{2}) \xrightarrow{\cong} \mathbb{Q}[x]/(x^2-2),$$
$$a + b\sqrt{2} \mapsto bx + a,$$

podemos hacer lo siguiente:

```
? a = Mod (1+x, x^2 - 2);
? a^2
% = Mod(2*x + 3, x^2 - 2)
? a^3
% = Mod(5*x + 7, x^2 - 2)
? a^4
% = Mod(12*x + 17, x^2 - 2)
```

Esto significa que

$$(1+\sqrt{2})^2 = 3+2\sqrt{2}, \quad (1+\sqrt{2})^3 = 7+5\sqrt{2}, \quad (1+\sqrt{2})^4 = 17+12\sqrt{2}.$$

Para dar otro ejemplo, el polinomio x^3+x+1 es irreducible en $\mathbb{F}_2[x]$ y el cociente $\mathbb{F}_2[x]/(x^3+x+1)$ es un cuerpo de 8 elementos. Calculemos cuál es el inverso de $x+1$ en este cuerpo:

```
? Mod (x+1, (x^3+x+1)*Mod(1,2))^-1
% = Mod(Mod(1, 2)*x^2 + Mod(1, 2)*x, Mod(1, 2)*x^3 + Mod(1, 2)*x + Mod(1, 2))
```

Si queremos pasar de `Mod(`g`,`f`)` al polinomio g, podemos usar la función `liftpol`. Si además g es un polinomio con coeficientes en $\mathbb{F}_p$ y nos gustaría pasar al polinomio correspondiente con coeficientes enteros que representa a g en $\mathbb{F}_p[x] \cong \mathbb{Z}[x]/(p)$, podemos usar `liftall`.

```
? f = Mod (x, (x^3+x+1)*Mod(1,2))^-1
% = Mod(Mod(1, 2)*x^2 + Mod(1, 2), Mod(1, 2)*x^3 + Mod(1, 2)*x + Mod(1, 2))
? liftpol(f)
% = Mod(1, 2)*x^2 + Mod(1, 2)
? liftall(f)
% = x^2 + 1
```

La función `liftall` hace los resultados más legibles. Calculemos por ejemplo las potencias de x en el cuerpo cociente $\mathbb{F}_2[x]/(x^3+x+1)$.

```
? f = Mod(x, (x^3+x+1)*Mod(1,2));
? liftall (vector (7,i,f^i))
% = [x, x^2, x + 1, x^2 + x, x^2 + x + 1, x^2 + 1, 1]
```

Aquí la función `vector(`n`,`i`,`x_i`)` construye el vector $[x_1,\ldots,x_n]$.

En general, para cualquier primo p y $n = 1,2,3,\ldots$ en el cuerpo $\mathbb{F}_p[x]$ existe un polinomio mónico irreducible de grado n[*]. La función `ffinit(`p`,`n`)` devuelve uno de estos polinomios f. Esto sirve para construir un cuerpo finito de p^n elementos como el cociente $\mathbb{F}_p[x]/(f)$.

[*] Lo probaremos en el segundo semestre cuando vamos a estudiar los cuerpos finitos de manera sistemática.

```
? ffinit (2,2)
% = Mod(1, 2)*x^2 + Mod(1, 2)*x + Mod(1, 2)
? ffinit (2,3)
% = Mod(1, 2)*x^3 + Mod(1, 2)*x^2 + Mod(1, 2)
? ffinit (3,2)
% = Mod(1, 3)*x^2 + Mod(1, 3)*x + Mod(2, 3)
? ffinit (3,3)
% = Mod(1, 3)*x^3 + Mod(1, 3)*x^2 + Mod(1, 3)*x + Mod(2, 3)
```

B.12 Matrices

La matriz

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

se denota por

$$[a_{11}, \ldots, a_{1n}; \ \ldots; \ a_{m1}, \ldots, a_{mn}]$$

Es decir, las entradas de la misma fila se separan por la coma "," y las diferentes filas se separan por el punto y coma ";". Las entradas de matrices pueden ser números enteros, racionales, números módulo n, polinomios, etc.

- La suma y producto de matrices a y b se calculan mediante a+b y a*b. La matriz a^n se calcula mediante a^n, y en particular la matriz inversa se calcula mediante a^-1. He aquí un pequeño ejemplo:

```
? a = [0,1,1; 0,0,1; 0,0,0];
% =
[0 1 1]

[0 0 1]

[0 0 0]

? a^2
% =
[0 0 1]

[0 0 0]

[0 0 0]

? a^3
% =
[0 0 0]

[0 0 0]

[0 0 0]

? [1,1,1; 0,1,1; 0,0,1]^-1
% =
```

```
[1 -1  0]

[0  1 -1]

[0  0  1]
```

- `matdet(`a`)` calcula el determinante de a.
- `trace(`a`)` calcula la traza de a.
- `matsize(`a`)` devuelve $[m,n]$ si a es una matriz de $m \times n$.
- a`[`i`,`j`]` devuelve el elemento a_{ij} de la matriz (las filas y columnas se numeran a partir de 1).

Hay muchas más funciones relacionadas con matrices; la mayoría tienen nombre que empieza por "`mat`". El lector interesado puede consultar la documentación de PARI/GP (sección "Vectors, matrices, linear algebra and sets").

B.13 Sumas y productos

Para calcular sumas y productos, pueden servir las siguientes funciones.

- `sum(`i`=`m`,`n`,`a_i`)` calcula la suma $\sum_{m \le i \le n} a_i$, donde a_i es una expresión que depende de i.
- `sumdiv(`n`,`d`,`a_d`)` calcula la suma $\sum_{d \mid n} a_d$ indexada por los divisores $d \mid n$, donde a_d es una expresión que depende de d.
- `prod(`i`=`m`,`n`,`a_i`)` calcula el producto $\prod_{m \le i \le n} a_i$, donde a_i es una expresión que depende de i.

He aquí algunos ejemplos.

```
? sum (i=1,4,i^2)
% = 30
? sum (i=1,16, kronecker(i,17))
% = 0
? sumdiv (100,d, eulerphi(d))
% = 100
? prod (i=1,10,i)
% = 3628800
```

B.14 Ciclos y expresiones condicionales

Para escribir pequeños programas, pueden ser útiles ciclos y expresiones condicionales. He aquí algunas de las funciones relevantes.

- `for(`i`=`m`,`n`,` $expr_1$`;` $expr_2$`;` ...`)` ejecuta las expresiones $expr_1, expr_2, \ldots$ para $m \le i \le n$. Las diferentes expresiones que tienen que ejecutarse se separan por el punto y coma "`;`".

  Cuando el ciclo toma demasiado tiempo y perdimos la paciencia, podemos interrumpir la ejecución presionando Ctrl+D.

- `forprime(`p`=`m`,`n`,` $expr_1$`;` $expr_2$`;` ...`)` hace lo mismo, pero para los números primos $m \le p \le n$.
- `if(condición,` $expr$`,` $expr'$`)` ejecuta la expresión $expr$ si la condición es verdadera y $expr'$ en el caso contrario. El argumento $expr'$ puede ser omitido.

  Las condiciones lógicas normalmente se construyen usando los operadores `!` (negación), `&&` (conjunción), `||` (disyunción) y las comparaciones `==` (igual*), `!=` (no igual), `<`, `<=`, `>`, `>=`.
- En fin, la función `print(`x`)` sirve para imprimir el valor de x. Véase también la documentación sobre la función `printf`.

Ejemplo elaborado

Para dar un ejemplo interesante de experimentos numéricos en PARI/GP, analicemos cómo el polinomio

$$f := x^3 + x^2 + 1$$

puede factorizarse en $\mathbb{F}_p[x]$. Primero escribamos un programa para encontrar los números primos $p \le 100$ tales que f es irreducible en $\mathbb{F}_p[x]$.

```
? f = x^3 + x^2 + 1;
? forprime (p=2,100, if (polisirreducible(f*Mod(1,p)), print (p)));
2
5
7
19
41
59
71
97
```

He aquí otro programa que calcula la proporción de los primos p tales que f es irreducible en $\mathbb{F}_p[x]$. En este caso podemos considerar $2 \le p \le 10^6$. De hecho, hay 78498 primos $\le 10^6$, lo que será suficiente para nuestros propósitos.

```
? n=0; m=0;
? forprime (p=2,10^6, n++; if (polisirreducible(f*Mod(1,p)), m++));
? m/n*1.0
% = 0.33352442100435679889933501490483833983
```

Entonces, aproximadamente $\frac{1}{3}$ de los primos tienen esta propiedad.

Siendo un polinomio cúbico, si f se vuelve reducible módulo p, este tiene por lo menos una raíz. ¿Cuándo hay raíces múltiples? Notamos que si $\alpha \in \mathbb{F}_p$ es una raíz múltiple, entonces $f'(\alpha) = 0$. Pero $f' = 3x^2 + 2x$ tiene raíces 0 y $-\frac{2}{3}$ para $p \neq 3$. Ahora 0 no es una raíz de f, mientras que

$$\left(-\frac{2}{3}\right)^3 + \left(-\frac{2}{3}\right)^2 + 1 = \frac{31}{3^3} \equiv 0 \pmod{p} \quad \text{si y solamente si } p = 31.$$

Entonces, con la única excepción de $p = 31$, las raíces de f son distintas, y si f es reducible, este puede factorizarse de dos maneras:

$$f = (x-\alpha_1)(x-\alpha_2)(x-\alpha_3) \quad \text{o} \quad f = (x-\alpha)\,g \quad \text{en } \mathbb{F}_p[x],\ p \neq 31,$$

*No confunda x==y (verifica si x es igual a y) con x=y (asigna a x el valor de y).

donde g es un polinomio cuadrático irreducible. El caso cuando f es un producto de tres distintos polinomios lineales ocurre raramente: los primeros primos con esta propiedad son 47 y 67.

```
? forprime (p=2,10^2, if (matsize (factor (f*Mod(1,p)))[1] == 3, print(p)));
47
67
```

Contemos cuál es la proporción de los primos con esta propiedad.

```
? n = 0; m = 0;
? forprime (p=2,10^6, n++; if (matsize (factor (f*Mod(1,p)))[1] == 3, m++));
? m/n*1.0
% = 0.16564753242120818364799103161863996535
```

Entonces, f es un producto de tres polinomios lineales en $\mathbb{F}_p[x]$ para aproximadamente $\frac{1}{6}$ de los primos p. Nos quedan $1-\frac{1}{3}-\frac{1}{6}=\frac{1}{2}$ de los primos, y estos darán factorizaciones de la forma $(x-\alpha)\,g$, donde g es cuadrático irreducible. Podemos concluir nuestras investigaciones de la siguiente manera.

tipo de factorización de f mód p	**proporción de primos p**
$(x-\alpha_1)(x-\alpha_2)(x-\alpha_3)$	$\frac{1}{6}$
$(x-\alpha)\,g$, donde $\deg g=2$	$\frac{1}{2}$
f (irreducible)	$\frac{1}{3}$
$(x-\alpha_1)(x-\alpha_2)^2$	solo $p=31$ (excepcional)

La aparición de los números $\frac{1}{3}$, $\frac{1}{6}$, $\frac{1}{2}$ se explica* por el **teorema de Frobenius** y el **teorema de densidad de Chebotarëv**. Un buen artículo sobre el tema es [LS1996].

*La explicación que podremos entender mucho más adelante: el grupo de Galois del polinomio $x^3+x^2+1\in\mathbb{Q}[x]$ es el grupo simétrico S_3 que tiene $3!=6$ elementos que se descomponen en tres clases de conjungación:

$$\{\mathrm{id}\}\sqcup\{(1\ 2),\ (1\ 3),\ (2\ 3)\}\sqcup\{(1\ 2\ 3),\ (1\ 3\ 2)\}.$$

De aquí salen los números $\frac{1}{6}$, $\frac{3}{6}$, $\frac{2}{6}$.

Bibliografía

[Coh1993] Henri Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138, Springer-Verlag Berlin Heidelberg, 1993.
https://doi.org/10.1007/978-3-662-02945-9

[LS1996] Hendrik Willem Lenstra and Peter Stevenhagen, *Chebotarëv and his density theorem*, The Mathematical Intelligencer (1996), no. 18, 26–37. MR1395088
http://www.math.leidenuniv.nl/~hwl/papers/cheb.pdf