Capítulo 3

Polinomios. Cuerpos finitos.

En esta práctica vamos a trabajar con polinomios. Los coeficientes pueden ser tanto racionales, como elementos de $\mathbb{Z}_p$. Por defecto, Maxima trabaja con polinomios con coeficientes racionales. Por ejemplo:

```
(%ixx) p:x^3+3*x^2+5*x-4$ q:x^2-6*x+8$
```

Y ahora podemos operar con ellos:

```
(%ixx) p+q; p-q; p+2*q;
```

(%oxx) $x^3 +4x^2-x+4$

(%oxx) $x^3+2x^2+11x-12$

(%oxx) $x^3+2(x^2-6x+8)+3x^2+5x-4$

Y para que nos muestre el resultado como nos gusta

```
(%ixx) expand(%);
```

(%oxx) $x^3+5x^2-7x+12$

Podemos hacer también multiplicaciones y potencias, y si queremos que nos muestre el resultado desarrollado y simplificado, utilizamos `expand`.

```
(%ixx) expand(p*q); expand(p^3); expand(2*p+q^2);
```

(%oxx) $x^5-3x^4-5x^3-10x^2+64x-32$
(%oxx) $x^9+9x^8+42x^7+105x^6+138x^5-3x^4-187x^3-156x^2+240x-64$
(%oxx) $x^4-10x^3+58x^2-86x+56$

También podemos calcular el cociente y el resto de la división. Tenemos los comandos `quotient` que calcula el cociente, `remainder` el resto, y `divide` que calcula ambos.

```
(%ixx) quotient(p,q); remainder(p,q); divide(p,q);
(%oxx) x+9
(%oxx) 51x-76
(%oxx) [x+9,51x-76]
```

Tal y como hemos definido los polinomios, no podemos considerarlos como funciones, es decir, no podemos evaluarlos. Para ello, podemos definirlos como funciones:

```
(%ixx) p1(x):=x^3+3*x^2+5*x-4$ q1(x):=x^2-6*x+8$
```

Y ahora, igual que antes, podemos calcular sumas, restas, productos, cocientes, restos, y además podemos evaluarlos.

```
(%ixx) p1(3);
(%oxx) 65
```

También podemos emplear el comando `subst`. Este comando tiene 3 argumentos. Por ejemplo, `subst(3,x,p)` sustituye en la expresión `p` la `x` por 3.

```
(%ixx) subst(3,x,p);
(%oxx) 65
```

También podemos, a partir de la expresión almacenada en `p` definir una función con el comando `define`

```
(%ixx) define(p2(x),p)$
(%ixx) p2(3);
(%oxx) 65
```

Para factorizar polinomios tenemos el comando `factor`. La factorización la realiza en $\mathbb{Q}$ (o en $\mathbb{Z}$), aunque los polinomios tengan coeficientes reales, e incluso complejos.

```
(%ixx) factor(p); factor(q);
(%oxx) x^3+3x^2+5x-4
(%oxx) (x-4)(x-2)
```

Pero si le pedimos que nos factorice x^2-2 o x^2+1, cuyas factorizaciones sabemos que son $(x-\sqrt{2})(x+\sqrt{2})$ y $(x+i)(x-i)$, respectivamente, la primera en $\mathbb{R}[x]$ y la segunda en $\mathbb{C}[x]$, Maxima no las hace, aunque sí puede evaluar ambos polinomios por ejemplo en $x=\sqrt{3}$ y $x=i$.

```
(%ixx) factor(x^2-2); factor(x^2+1);
(%oxx) x^2-2
(%oxx) x^2+1
(%ixx) subst(sqrt(3),x,x^2-2); subst(%i,x,x^2+1);
(%oxx) 1
(%oxx) 0
```

De hecho, Maxima es capaz de encontrar raíces reales y complejas de polinomios, con el comando `solve`.

```
(%ixx) p:x^2+1$ q:x^2-2$
(%ixx) solve(p); solve(q);
(%oxx) [x=-%i,x=%i]
(%oxx) [x=-√2,x=√2]
```

Al igual que con los números enteros, podemos calcular el máximo común divisor de dos polinomios, asi como unos coeficientes de Bezout.

```
(%ixx) gcd(x^5-3*x^4+3*x^3-5*x^2+8*x-4,x^6-5*x^5+10*x^4-11*x^3+x^2+16*x-12);
(%oxx) x^2-3x+2
(%ixx) gcdex(x^5-3*x^4+3*x^3-5*x^2+8*x-4,x^6-5*x^5+10*x^4-11*x^3+x^2+16*x-12);
```

(%oxx)/R/ [- $\frac{5x^3-2x^2+20*x-39}{264}$, $\frac{5x^2+8x+31}{264}$, -x^2+3x-2].

Vimos en la práctica anterior que cambiando el valor de la variable `modulus`, podemos forzar a Maxima a que trabaje módulo un número entero dado. Entonces, lo que hemos hecho aquí para polinomios con coeficientes racionales, podemos hacerlo también para polinomios con coeficientes en $\mathbb{Z}_p$. Por ejemplo:

```
(%ixx) p:x^3+6*x^2-4*x+9$
(%ixx) factor(p);
(%oxx) x^3+6x^2-4x+9
(%ixx) modulus:2$ factor(p);
(%oxx) (x+1)(x^2-x+1)
(%ixx) modulus:3$ factor(p);
(%oxx) (x-1)x(x+1)
```

Recordemos que para Maxima, los enteros módulo p van desde $-\frac{p-1}{2}$ hasta $\frac{p-1}{2}$, no desde 0 hasta $p-1$.

```
(%ixx) gcdex(x^5+2*x^4+x^2+2*x+2,x^5+2*x^3+x^2+x+1);
```

```
(%oxx)/R/ [x-1,-x,-x^2-1]
```

Cuando le pedimos a Maxima que aplique el Algoritmo extendido de Euclides mediante el comando `gcdex`, el polinomio que toma como máximo común divisor no siempre es un polinomio mónico. En el ejemplo anterior, en el que los coeficientes pertenecen a $\mathbb{Z}_3$, el máximo común divisor calculado fué $-x^2-1$, es decir, $2x^2+2$. Así, otro máximo común divisor de los polinomios $x^5+2x^4+x^2+2x+2$ y $x^5+2x^3+x^2+x+1$ es x^2+1. No obstante, si usamos el comando `gcd` para calcular el máximo común divisor, sí nos devuelve un polinomio mónico.

```
(%ixx) gcd(x^5+2*x^4+x^2+2*x+2,x^5+2*x^3+x^2+x+1);
(%oxx) x^2+1
```

También podemos calcular la derivada de un polinomio, la cual, sabemos que nos permite saber si el polinomio tiene raíces múltiples y por tanto factores múltiples. (Repase el Ejercicio 58 de la Relación de problemas del Tema 3.) Por ejemplo:

```
(%ixx) p:x^10+2*x^9+2*x^8+7*x^7+2*x^5+2$
(%ixx) q:diff(p,x);
(%oxx) 10x^9+18x^8+16x^7+7x^6+10x^4.
```

Si queremos que nos lo muestre con los coeficientes reducidos usamos `polymod`.

```
(%ixx) q:polymod(%);
(%oxx) x^9+x^7+x^6+x^4.
(%ixx) d:gcd(p,q);
(%oxx) x^5+x^3+x^2+1
```

Lo que nos dice que el polinomio tiene factores múltiples, pues el máximo común divisor obtenido no es una unidad. El máximo común divisor del polinomio `p` y su derivada podríamos haberlo obtenido también sin más que escribir `d:gcd(p,diff(p,x))`.

Vamos a comprobar que efectivamente `p` tiene factores múltiples.

```
(%ixx) factor(p); factor(q); factor(d);
(%oxx) (x+1)^4(x^2+1)^2(x^2+x-1)
(%oxx) x^4(x+1)^3(x^2+1)
(%oxx) (x+1)^3(x^2+1)
```

Vemos cómo en `d` tenemos los mismos factores que en `p` pero elevados a su exponente correspondiente menos 1.

Por tanto, si dividimos `p` entre `d`, obtenemos como cociente el producto de los factores irreducibles de `p`, cada uno de ellos con exponente 1.

```
(%ixx) factor(quotient(p,d));
(%oxx) (x+1)(x^2+1)(x^2+x-1)
```

El último polinomio que hemos calculado se denomina la parte libre de cuadrados de `p`. Como otro ejemplo, la parte libre de cuadrados del polinomio x^3 es x. Sin embargo ésto no podemos calcularlo siguiendo el procedimiento anterior, pues la derivada de x^3 en $\mathbb{Z}_3[x]$ es el polinomio cero, con lo cual `gcd(x^3,0)` es igual a `x^3`.

En general, si algún factor de un polinomio $\texttt{p} \in \mathbb{Z}_3[x]$ tiene como exponente un múltiplo de 3, al calcular `gcd(p,diff(p,x))` no disminuye dicho exponente.

Al igual que con los enteros, podemos plantearnos resolver ecuaciones diofánticas de la forma

$$a(x) \cdot p(x) + b(x) \cdot q(x) = c(x).$$

Recordemos que para encontrar una solución en los enteros teníamos la función `diofantica` que habíamos definido como sigue:

```
diofantica(a,b,c):=if mod(c,gcd(a,b))=0 then rest(gcdex(a,b),-1)*c/gcd(a,b)
                                         else "No tiene solución"
```

Para que nos valiera para polinomios, habría que sustituir la función `mod` por `remainder`. Ya hemos comentado que el máximo común divisor calculado con `gcd(a,b)` no tiene por qué ser el mismo que el calculado con `gcdex(a,b)`. (El primero es mónico, el segundo no tiene por qué.) Por tanto, también modificamos eso. Vamos a llamar `d` al resultado de `gcdex(a,b)` y tenemos:

```
(%ixx) diofantica(a,b,c):=block(d:gcdex(a,b),
                                if remainder(c,d[3])=0 then rest(d,-1)*quotient(c,d[3])
                                                       else "No tiene solución")$
```

Recordemos que seguimos trabajando en $\mathbb{Z}_3[x]$.

```
(%ixx) diofantica(x^5+2*x^3+2,x^5+2*x^4+2*x^3+1,x^4+2*x^2+2*x+2);
```

(%oxx)/R/ [x^4+x^3+x^2-1,-x^4+x^3-x+1]

Compobamos el resultado obtenido.

```
(%ixx) rat((x^5+2*x^3+2)*(x^4+x^3+x^2-1)+(x^5+2*x^4+2*x^3+1)*(-x^4+x^3-x+1));
(%oxx)/R/ x^4-x^2-x-1
```

Recordemos que un cuerpo es un anillo conmutativo en el que todo elemento distinto de cero tiene inverso (para el producto). Por ejemplo, son cuerpos $\mathbb{Q}$ (números racionales), $\mathbb{R}$ (números reales), $\mathbb{C}$ (números complejos), y para un número primo p, $\mathbb{Z}_p$ (los enteros módulo p).

Si el número de elementos de un cuerpo $\mathbb{K}$ es finito, entonces se dice que $\mathbb{K}$ es un *cuerpo finito.*

Para cada número primo p, y cada número natural $n \geq 1$, hay un cuerpo con p^n elementos, que denotaremos por $\mathbb{F}_{p^n}$. Este cuerpo puede obtenerse como $\mathbb{Z}_p[x]_{m(x)}$, donde $m(x) \in \mathbb{Z}_p[x]$ es un polinomio de grado n e irreducible.

Dado un número primo p, y un polinomio $m(x) \in \mathbb{Z}_p[x]$, vamos a definir funciones `suma`, `producto`, `inverso`, y `potencia` que nos realicen dichas operaciones en $\mathbb{Z}_p[x]_{m(x)}$.

Para trabajar módulo p, sabemos que tenemos que asignar el valor p a la variable `modulus`. En caso de que introduzcamos un número que no es primo, nos dará un aviso.

Y ahora, queremos trabajar módulo $m(x)$. Para esto creamos una variable, a la que podemos llamar `contexto`, en la que guardaremos este polinomio.

Por ejemplo, vamos a comenzar en $\mathbb{Z}_5[x]_{x^3+x^2+3x+2}$. Para esto, introducimos:

```
(%ixx) modulus:5$ contexto:x^3+x^2+3*x+2$
```

Definimos las funciones `suma` y `producto`. Para esto, lo que tenemos que hacer es calcular la suma y el producto de los polinomios, y reducirlos módulo $x^3 + x^2 + 3x + 2$.

```
(%ixx) suma(p,q):=remainder(p+q,contexto)$
(%ixx) producto(p,q):=remainder(p*q,contexto)$
```

Y ahora, podemos probar a hacer algunos cálculos.

```
(%ixx) p1:3*x^2+4*x+1$ p2:x^4+2*x^3-3*x^2-2$ p3:2*x^2+x+3$
(%ixx) suma(p1,p2); suma(p1,p3); suma(p1,-p3); producto(p1,p2); producto(p2,2*p3);
```

(%oxx) x^2-x+2

(%oxx) -1

(%oxx) x^2-2x-2

(%oxx) x^2+2x

(%o23) 2x^2+x-2

Recordemos, que para Maxima, los enteros módulo 5 son $\{-2, -1, 0, 1, 2\}$.

Vamos a definir ahora el inverso. El inverso de $q(x)$ en $\mathbb{Z}_p[x]_{m(x)}$ existe si, y sólo si, $\text{mcd}(q(x), m(x))$ es una constante distinta de 0, es decir, en notación de Maxima que `gcd(q(x),m(x))=1`. En caso de que exista, sabemos que ese inverso es un polinomio $u(x)$ que verifica que $q(x) \cdot u(x) + m(x) \cdot v(x) = 1$, y puede calcularse con el comando `gcdex`. El inverso entonces puede ser definido como sigue:

```
(%ixx) inverso(p):= if gcd(p,contexto)=1
                       then block(d:gcdex(p,contexto), d[1]/d[3])
                       else "No existe el inverso"$
```

Sabemos que cuando hacemos `d:gcdex(p,contexto)`, entonces `d[3]` es el máximo común divisor de `p` y `contexto`, que vale 1. Sin embargo, el resultado que nos devuelve con `gcdex` no tiene porqué ser mónico, luego en este caso podría ser cualquier constante. De ahí que dividamos por `d[3]`.

(%ixx) inverso(x);

(%oxx)/R/ 2x^2+2x+1

(%ixx) inverso(x^2+2*x+2);

(%oxx) No existe el inverso

Vamos a definir ahora una función para calcular una potencia. Si quisiéramos calcular, por ejemplo, $p(x)^{18}$, podríamos multiplicar $p(x)$ consigo mismo 18 veces. Podemos definir entonces:

```
(%ixx) potencia(a,m):=block([y], y:1, for i:1 thru m do y:producto(y,a),y)$
```

En primer lugar, inicializamos la variable donde vamos a guardar el resultado a 1. Y ahora, vamos multiplicando `a` consigo mismo tantas veces como nos indica el exponente.

Realizamos algunos cálculos:

(%ixx) contexto:x^20$ potencia(x,5); potencia(x,14); potencia(x,20);

(%oxx) x^5

(%oxx) x^{14}

(%oxx) 0

Y ahora volvemos al ejemplo que teníamos.

(%ixx) contexto:x^3+x^2+3*x+2$

(%ixx) potencia(x,3);

(%oxx) -x^2+2x-2

El método expuesto para calcular potencias de polinomios es muy lento, pues requiere muchas operaciones. Ésto puede ser comprobarlo escribiendo, por ejemplo:

```
(%ixx) potencia(x,111111);
```

Para resolver ésto, vamos a ver otra forma para calcular una potencia. Supongamos que queremos calcular $p(x)^{18}$, para algún polinomio $p(x)$. Entonces, procedemos como sigue:

$$\begin{aligned}
&p_1(x) = p(x)^2,\\
&p_2(x) = p(x)^4 = p_1(x) \cdot p_1(x),\\
&p_3(x) = p(x)^8 = p_2(x) \cdot p_2(x),\\
&p_4(x) = p(x)^{16} = p_3(x) \cdot p_3(x).
\end{aligned}$$

Y una vez llegados aquí, hacer $p(x)^{18} = p(x)^{16} \cdot p(x)^2 = p_4(x) \cdot p_1(x)$.

En definitiva, lo que hacemos es expresar el exponente como suma de potencias de dos, o lo que es lo mismo, calculamos su expresión en binario. Calculamos $p(x)$ elevado a cada una de las potencias de dos, y luego multiplicamos las que nos interesan (que se corresponden con los *unos* en la expresión binaria del exponente).

Recordemos que la expresión binaria de un número se obtiene dividiendo sucesivamente el número por dos, y quedándonos con el resto. Lo que vamos a hacer es tomar el exponente y dividirlo por dos.

Simultáneamente, vamos elevando el polinomio al cuadrado. Cuando el resto sea uno, tomamos esa potencia del polinomio. Cuando sea cero, la dejamos. Y así, hasta que el exponente llegue hasta cero.

Para decirle que vaya realizando estas operaciones mientras el exponente sea mayor que cero, usaremos el comando `while do`.

Definimos entonces:

```
(%ixx) potencia(p,m):=block(y:1, while m>0 do(
                                   if mod(m,2)=1 then y:producto(y,p),
                                   p:producto(p,p), m:quotient(m,2)),
                         y)$
```

Vamos a comprobar que lo hemos hecho bien:

```
(%ixx) contexto:x^20$ potencia(x,5); potencia(x,14); potencia(x,20);
```

(%oxx) x^5

(%oxx) x^{14}

(%oxx) 0

Y ahora volvemos al ejemplo que teníamos.

```
(%ixx) contexto:x^3+x^2+3*x+2$
(%ixx) potencia(x,3);
```

(%oxx) $-x^2+2x-2$

Sabemos que si m es negativo, entonces p^m es el inverso de p^{-m}. Podemos aprovechar ésto para modificar nuestra función potencia, y que también nos valga para exponentes negativos.

```
(%ixx) potencia(a,m):=block([y,n], n:abs(m), y:1,
                         while n>0 do(
                         if mod(n,2)=1 then y:producto(y,a),
                         a:producto(a,a), n:quotient(n,2)),
                         if m>0 then y else inverso(y))$
```

Lo que hace es, en primer lugar calcula a elevado al valor absoluto de m, y luego, según sea m positivo o negativo, lo deja como está o calcula el inverso.

```
(%ixx) potencia(x+1,-2);
```

(%oxx) $-x^2+x+1$

Podría ocurrir que alguna potencia con exponente negativo no existiera.

```
(%ixx) potencia(x^2+4*x+3,-3);
(%oxx) No existe el inverso
```

Al principio del bloque, hemos introducido `[y,n]`. Esto lo que hace es tratar las variables `y` y `n` como variables locales dentro del bloque. De esta forma, si tenían un valor antes de ejecutar el bloque, tras la ejecución de éste, tales valores no serán modificados. Por ejemplo:

```
(%ixx) y:37$ n:35$ potencia(x,4); y; n;
```

(%oxx) $-2x^2+x+2$.

(%oxx) 37

(%oxx) 35

Ya tenemos los ingredientes necesarios para la aritmética en anillos de la forma $\mathbb{Z}_p[x]_{m(x)}$. Vamos, por ejemplo, a trabajar en $\mathbb{Z}_3[x]_{x^3+x^2+x+1}$. Para ésto, definimos las constantes apropiadas.

```
(%ixx) modulus:3$ contexto:x^3+x^2+x+1$
```

En primer lugar, vamos a construir un conjunto con todos sus elementos:

```
(%ixx) Z3:{0,1,-1}$ A27:makeset(a*x^2+b*x+c,[a,b,c],cartesian_product(Z3,Z3,Z3));
```

```
(%oxx) {-1,0,1,-x-1,1-x,x-1,-x,x,x+1,-x^2-1,1-x^2,-x^2-x-1,-x^2-x,-x^2-x+1,-x^2+x
-1,x-x^2,-x^2+x+1,x^2-1,-x^2,x^2,x^2+1,x^2-x-1,x^2-x,x^2-x+1,x^2+x-1,x^2+x,x^2+x+1}
```

Ahora vamos a ver cuáles son unidades y cuáles divisores de cero. Tenemos que $q(x)$ es unidad en $\mathbb{Z}_p[x]_{m(x)}$ si, y sólo si, $\mathrm{mcd}(p(x), m(x)) = 1$, y es divisor de cero si, y sólo si, $\mathrm{mcd}(p(x), m(x)) \neq 1$. Entonces:

```
(%ixx) UA27:subset(A27,lambda([a],is(gcd(a,contexto)=1)));
(%oxx) {-1,1,1-x,x-1,-x,x,-x^2-x-1,-x^2 -x+1,x-x^2,-x^2+x+1,-x^2,x^2,x^2-x-1,x^2-x,x^2
+x-1,x^2+x+1}
```

En vista de éste resultado, deducimos que el anillo $\mathbb{Z}_3[x]_{x^3+x^2+x+1}$ no es un cuerpo, pues hay elementos distintos del cero que no tienen inverso, como por ejemplo $x+1$.

Tenemos que UA27 es un grupo respecto de la operación producto. Su cardinal es 16, y por tanto cualquiera de sus elementos elevado a 16 da de resultado 1.

```
(%ixx) ua27:listify(UA27)$ makelist(potencia(ua27[i],16),i,1,16);
(%oxx) [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
```

De hecho, en este caso, se tiene que elevados todos a 8 sale 1, mientras que elevados a la cuarta potencia, los resultados son, la mitad 1 y la otra mitad x^2.

Como hemos ya mencionado, el conjunto de las unidades del anillo $\mathbb{Z}_3[x]_{x^3+x^2+x+1}$ es cerrado para el producto. Eso significa que si multiplicamos dos elementos de ese conjunto, nos vuelve a dar un elemento del conjunto. Por ejemplo, lo comprobamos en tres casos particulares:

```
(%ixx) producto(ua27[3],ua27[11]); producto(ua27[8],ua27[14]); producto(ua27[2],ua27[4]);

(%oxx) x^2-x-1
(%oxx) -x^2-x-1
(%oxx) x-1
```

Y podemos ver que los tres pertenecen al conjunto UA27. Recuerde el comando `elementp(a,A)` que estudiamos en la Práctica del Tema 1.

Ahora calculamos los inversos de todos los elementos del conjunto UA27. Para ello usamos la lista ya definida ua27:

```
(%ixx) Inversos:makelist(inverso(ua27[i]),i,1,16);
(%oxx) /R/ [-1,1,x^2-x,-x^2+x,x^2+x+1,-x^2-x-1,x,x^2-x-1,x-1,x^2 +x-1,-x^2,x^2,-x^2-x+1
,-x+1,-x^2+x+1,-x]
```

Vemos que contiene exactamente a los 16 elementos de UA27. También podríamos haber obtenido esta lista mediante `makelist(potencia(ua27[i],7),i,1,16);`

En el caso de que el polinomio $m(x)$ sea irreducible, entonces todos los elementos de $\mathbb{Z}_p[x]_{m(x)}$, salvo el cero son unidades, luego $\mathbb{Z}_p[x]_{m(x)}$ es un cuerpo. Por ejemplo:

```
(%ixx) factor(x^3+x^2+2);
(%oxx) x^3+x^2+2
```

Lo que nos dice que $\mathbb{Z}_3[x]_{x^3+x^2+2}$ es un cuerpo que tiene 27 elementos. Vamos a llamar a este cuerpo F27, y UF27 al conjunto de las unidades.

```
(%ixx) contexto:x^3+x^2+2$ F27:A27$ UF27:setdifference(F27,{0})$
```

Comprobamos ahora que todo elemento de UF27, elevado a 26, vale 1.

```
(%ixx) uf27:listify(UF27)$
(%ixx) makelist(potencia(uf27[i],26),i,1,16);
(%oxx) [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
```

Si $\mathbb{K}$ es un cuerpo finito tal que $|\mathbb{K}| = q$, entonces se dice un elemento $\alpha \in \mathbb{K}$ es primitivo, si

$$\left\{\alpha^j \ : \ j = 1, 2, \ldots, q-1\right\} = \mathbb{K} \setminus \{0\}.$$

Comprobamos que en el cuerpo `F27` que hemos definido, el elemento $2x+1$ no es primitivo, pero $2x+2$ sí lo es. Para ello obtenemos las potencias de $2x+1$ y $2x+2$.

```
(%ixx) makelist(potencia(2*x+1,i),i,1,26);
(%oxx) [1-x,x^2+x+1,x^2,-x^2-1,x^2+x,x^2 +x-1,x^2-x+1,x,x-x^2,x+1,1-x^2,x^2-x-1,1,1-x,
x^2+x+1,x^2,-x^2-1,x^2+x,x^2 +x-1,x^2-x+1,x,x-x^2,x+1,1-x^2,x^2-x-1,1]
(%ixx) cardinality(setify(%));
(%oxx) 13
(%ixx) makelist(potencia(2*x+2,i),i,1,26);
(%oxx) [-x-1,x^2-x+1,x^2+1,1-x,x^2-1,x,-x^2-x,x^2+x+1,-x^2 +x+1,x-x^2,-x^2-x+1,x^2,-1,
x+1,-x^2+x-1,-x^2-1,x-1,1-x^2,-x,x^2+x,-x^2-x-1,x^2-x-1,x^2 -x,x^2+x-1,-x^2,1]
(%ixx) cardinality(setify(%));
(%oxx) 26
```

Vea cómo en el segundo caso hemos obtenido todos los elementos de `F27` salvo el cero. Para comprobar que $2x+2$ es un elemento primitivo, no es necesario calcular todas las potencias $(2x+2)^i$ para $i = 1, 2, \cdots, 26$. Bastaría haber calculado $(2x+2)^{\frac{26}{13}} = (2x+2)^2$ y $(2x+2)^{\frac{26}{2}}(2x+2)^{13}$ y comprobar que son distintos de uno.

Notemos finalmente, que al tener todos los elementos de `F27` representados como potencias de un elemento, por ejemplo del elemento primitivo $2x+2 = -x-1$, ello nos permite de forma sencilla calcular el producto de dos elementos cualesquiera. Por ejemplo:

$$(x^2+2) \cdot (2x^2+x+2) = (2x+2)^5(2x+2)^{15} = (2x+2)^{20} = x^2+x$$

```
(%ixx) producto(x^2+2,2*x^2+x+2);
(%oxx) x^2+x
```

Ejercicio propuesto 1: Resuelva, con la ayuda de Maxima, los siguientes ejercicios de la Relación de problemas del Tema 3: 52–56, 61, 62, 65–71; compruebe la propiedad establecida en el Ejercicio 59 para los primos $p = 2, 3, 5, 7, 11, 13, 17, 19$.

Ejercicio propuesto 2: En el anillo $Z_{11}[x]_{x^7+3x^6+2x^4+3x^2+3x+3}$, obtenga una solución particular para cada una de la ecuaciones diofánticas siguientes:

1. $(2x^6+4x^5+10x+1) \cdot \alpha(x) + (x^5+7x^4+2x^3+x+8) \cdot \beta(x) = 3x^3+7x^2+10x+6.$
2. $(2x^6+4x^5+10x+1) \cdot \alpha(x) + (x^5+7x^4+2x^3+x+8) \cdot \beta(x) = x^3+x^2+8x+1.$

Ejercicio propuesto 3: Construya un cuerpo de ocho elementos y otro de nueve elementos.