Ejercicio9

March 27, 2022

1 Ejercicio 9

Toma n tu número publicado para el ejer_ 2. Escribe en base 2, usa esas cifras para definir un polinomio, f(x), donde tu bit más significativo defina el grado del polinomio n, el siguiente bit va multiplicado por x n-1 y sucesivamente hasta que el bit menos significativo sea el término independiente. El polinomio que obtienes es universal en el sentido de que tiene coeficientes en cualquier anillo.

[1]:
```
n=26505013

from math import gcd
import numpy as np
import sys

def f(x):
    return x*x+1

def rho_de_polard(n,imprime=False):
    x=1
    y=1
    contador=0
    resultado=1

    if imprime:
        print("Iteracion ", contador)
        print("x: ",x," y:",y , " mcd: ", resultado)

    while resultado==1 or resultado == n:
        x=f(x)%n
        y=f(f(y))%n
        resultado=gcd(x-y,n)
        contador+=1

        if imprime:
            print("Iteracion ", contador)
            print("x: ",x," y:",y , " mcd: ", resultado)
        if 1<resultado<n:
            return resultado
```

```
    return "No hay divisores"

def exponenciacion_rapida_izda_dcha (a,exp,m,imprime=False):
    c=0
    num_binario= bin(exp)[2:]
    acu=1
    for i in num_binario:
        c=2*c
        acu=(acu*acu)%m

        if i=='1':
            c+=1
            acu=(acu*a)%m

    return acu

def simbolo_jacobi(a,n):
    t=1
    m=abs(n)
    b=a%m

    while a != 0:
        while a%2==0:
            a=a/2
            if m%8==3 or m%8==5:
                t=-t


        aux=a
        a=m
        m=aux
        if a%4==m%4==3:
            t=-t


        a=a%m



    if m==1:
        return t
    else:
        return 0
```

```python
def comprobacion_lucas_lehmer(a,divisores,n):
    resultado=True

    for i in divisores:
        if exponenciacion_rapida_izda_dcha(a,(n-1)//i,n) == 1:
            resultado=False

    return resultado

def Lucas_Lehmer(n,divisores):
    i=1
    test1=False
    test2=False
    res=0

    while i>0:
        i+=1
        res=exponenciacion_rapida_izda_dcha (i,n-1,n)

        if res==1:
            test1=True

        if comprobacion_lucas_lehmer(i,divisores,n):
            test2=True

        if test1 & test2:
            print("El natural más pequeño cuya clase es primitiva: ", i)
            return True
        else:
            test1=False
            test2=False

def obtener_exponentes_millner_rabin(n):
    valor=n-1
    resultado=[valor]
    while valor%2==0:
        valor=valor//2
        resultado.append(valor)

    return resultado
```

[2]:

```python
num_binario= bin(n)[2:]
print(num_binario)
```

```
110010100011011100110101
```

[3]:
```
def polinomio(coef):
    exp=0
    coeficientes=[]
    polinomio=""
    for i in reversed(coef):
        if (i==1):
            coeficientes.append(exp)
            if (len(coef)-1==exp):
                polinomio+="x^"+str(exp)
            elif (exp==0):
                polinomio+=str(1)+"+"
            else :
                polinomio+="x^"+str(exp)+"+"
        exp=exp+1
    print ("f(x)= ", polinomio)
    return coeficientes
```

[4]:
```
f=polinomio([1,1,0,0,1,0,1,0,0,0,1,1,0,1,1,1,1,0,0,1,1,0,1,0,1])
```

```
f(x)=  1+x^2+x^4+x^5+x^8+x^9+x^10+x^11+x^13+x^14+x^18+x^20+x^23+x^24
```

i) Toma $g(x) = f(x)\ mod\ 2$ y halla el menor cuerpo de característica 2 que contenga a todas las raíces de g. ¿ Qué deduces sobre la irreducibilidad de $g(x)$ en $\mathbb{Z}_2[x]$?.

Para las siguientes secciones usaremos SageMath, primero definimos el anillo de polinomios $\mathbb{Z}_2[x]$ y nuestro polinomio.

Por otro lado, denominaremos m al exponente del cuerpo finito F_{p^m} y llamaremos n al grado de los polinomios

[5]:
```
R.<x> = PolynomialRing(GF(2))
```

[6]:
```
f=R(1+x^2+x^4+x^5+x^8+x^9+x^10+x^11+x^13+x^14+x^18+x^20+x^23+x^24)
```

Calculamos su derivada:

[7]:
```
f_derivada=derivative(f)
f_derivada
```

[7]:
```
x^22 + x^12 + x^10 + x^8 + x^4
```

Comprobamos si es libre de cuadrados calculando el máximo común divisor entre f y su derivada:

[8]:
```
f.gcd(f_derivada)
```

[8]:
```
1
```

Como vemos f es libre de cuadrados, luego de ser reducible, su factorización es única en irreducible distintos de $\mathbb{Z}_2[x]$. vamos a aplicar el algoritmo que nos da el criterio de irreducibilidad:

[9]:
```
def algoritmo_cf(f,R,p):
    g=x^p
    resto=R(g-x)%f
    k=1
    while len(resto.coefficients(sparse=False))>0:
        g=(g^p)%f
        resto=R(g-x)%f
        k+=1

    return k
```

[10]:
```
algoritmo_cf(f,R,2)
```

[10]: 23

Así el menor cuerpo de característica 2 que contiene todas las raices del polinomio sería $F_{2^{23}}$.

Como podemo ver, hemos obtenido que $m = 23 \neq 24 = n$, y por el teorema que nos caracteriza los polinomios irreducibles podemos concluir que nuestro polinomio $f(x)$ es **reducible** en $\mathbb{Z}_2[x]$.

ii) Extrae la parte libre de cuadrados de f(x) y le calculas su matriz de Berlekamp por columnas. Resuelve el s.l. (B-Id)X=0.

iii) Aplica Berlekamp si es necesario recursivamente para hallar la descomposición en irreducibles de $f(x)$ en $\mathbb{Z}_2[x]$.

En nuestro caso, $f(x)$ ya era libre de cuadrados, luego vamos a calcular la matriz de Berlekamp directamente.

Pasamos a calcular las potencias x^{2i} mod $g(x)$ para $0 \leq i \leq n-1 = 23$

[11]:
```
def prepara(coef,p):
    if len(coef)!=p:
        for i in range(p-len(coef)):
            coef.append(0)
    return coef
```

[12]:
```
lista_coef=[]
for i in range(24):
    a=R(x^(2*i))
    mod=a%f
    print()
    print(mod)
    coef=mod.coefficients(sparse=False)
    coef=prepara(coef,24)
    lista_coef.append(coef)
```

1

```
x^2

x^4

x^6

x^8

x^10

x^12

x^14

x^16

x^18

x^20

x^22

x^23 + x^20 + x^18 + x^14 + x^13 + x^11 + x^10 + x^9 + x^8 + x^5 + x^4 + x^2 + 1

x^23 + x^22 + x^21 + x^19 + x^18 + x^16 + x^11 + x^10 + x^8 + x^7 + x^3 + x + 1

x^23 + x^20 + x^19 + x^18 + x^15 + x^14 + x^13 + x^11 + x^6 + x^2 + x

x^23 + x^22 + x^19 + x^18 + x^17 + x^16 + x^12 + x^6 + x^2 + x + 1

x^20 + x^18 + x^15 + x^12 + x^11 + x^10 + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x

x^22 + x^20 + x^17 + x^14 + x^13 + x^12 + x^11 + x^10 + x^8 + x^7 + x^6 + x^4 +
x^3

x^23 + x^22 + x^20 + x^19 + x^18 + x^16 + x^15 + x^12 + x^11 + x^6 + x^4 + x^2 +
1

x^22 + x^20 + x^19 + x^18 + x^17 + x^15 + x^13 + x^12 + x^11 + x^10 + x^9 + x^8
+ x^5 + x^4 + x^3 + x^2 + x

x^23 + x^22 + x^21 + x^19 + x^18 + x^17 + x^15 + x^12 + x^9 + x^8 + x^7 + x^6 +
x^3 + x^2 + 1

x^23 + x^20 + x^17 + x^15 + x^12 + x^8 + x^6 + x^4 + x^3 + x^2 + x

x^23 + x^22 + x^21 + x^20 + x^18 + x^17 + x^15 + x^14 + x^13 + x^12 + x^10 + x^5
+ x^2 + x + 1
```

```
x^23 + x^22 + x^21 + x^20 + x^17 + x^16 + x^11 + x^10 + x^9 + x^7 + x^6 + x^5 +
x^4 + x^2 + x
```

Ahora construimos la matriz de Berlekamp:

[13]:
```
B=Matrix(lista_coef)
```

[14]:
```
print(B)
```

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
[1 0 1 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1]
[1 1 0 1 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 1]
[0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 1 0 0 1]
[1 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1]
[0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 0 0]
[0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0]
[1 0 1 0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 1 1 0 1 1]
[0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 0]
[1 0 1 1 0 0 1 1 1 1 0 0 1 0 0 1 0 1 1 1 0 1 1 1]
[0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1]
[1 1 1 0 0 1 0 0 0 0 1 0 1 1 1 1 0 1 1 0 1 1 1 1]
[0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1]
```

Calculamos la diferencia con la identidad:

[15]:
```
I=matrix.identity(24)
D=B-I
print(D)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
```

```
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0]
[1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1]
[1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 0 1 1 0 1 1 1]
[0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 1 1 0 0 1]
[1 1 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1]
[0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 1 0 1 0 1 0 0 0]
[0 0 0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 0]
[1 0 1 0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 1 1]
[0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1 0]
[1 0 1 1 0 0 1 1 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1]
[0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 1 0 1]
[1 1 1 0 0 1 0 0 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 1]
[0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 0]
```

Calculamos el rango de la matriz:

[16]:
```
print(24-D.rank())
```

```
2
```

Por lo tanto el sistema tiene 2 soluciones. Vamos a resulver el sistema de ecuaciones encontrando una base del espacio vectorial que define la matriz.

[17]:
```
D.kernel()
```

[17]:
```
Vector space of degree 24 and dimension 2 over Finite Field of size 2
Basis matrix:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1]
```

Como hacemos siempre, descartamos el polinomio $h(x) = 1$ pues no es f-reductor, por lo que solo el segundo nos dará la factorización de f.

[18]:
```
h=R([0,1, 0, 0, 1 ,0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0 ,0 ,0, 1, 1, 0 ,0, 0, 1])
h
```

[18]:
```
x^23 + x^19 + x^18 + x^13 + x^10 + x^8 + x^4 + x
```

Ahora calculamos el mcd del plinomio $f(x)$ con $h(x)$ y $h(x) - 1$, lo que nos dará la factorización de f

[19]:
```
f.gcd(h)
```

[19]:
```
x + 1
```

[20]:
```
f.gcd(h-1)
```

[20]: x^23 + x^19 + x^18 + x^13 + x^10 + x^8 + x^4 + x + 1

Obtenemos así que $f(x) = (x+1) \cdot (x^{23}+x^{19}+x^{18}+x^{13}+x^{10}+x^8+x^4+x+1)$, vamos a comprobar si dichos factores son primos. Vamos a llamar $f_1(x) = x+1$ y $f_2(x) = x^{23}+x^{19}+x^{18}+x^{13}+x^{10}+x^8+x^4+x+1$.

No es necesario comprobar la irreducibilidad de los dos factores porque el algoritmo inicial lo asegura.

iv) Haz lo mismo para hallar la descomposición en irreducibles de f(x) mod 3.

[21]:
```
R.<x> = PolynomialRing(GF(3))
```

[22]:
```
f=R(1+x^2+x^4+x^5+x^8+x^9+x^10+x^11+x^13+x^14+x^18+x^20+x^23+x^24)
```

Calculamos su derivada:

[23]:
```
f_derivada=derivative(f)
f_derivada
```

[23]: 2*x^22 + 2*x^19 + 2*x^13 + x^12 + 2*x^10 + x^9 + 2*x^7 + 2*x^4 + x^3 + 2*x

Comprobamos si es libre de cuadrados calculando el máximo común divisor entre f y su derivada:

[24]:
```
f.gcd(f_derivada)
```

[24]: 1

Como vemos f es libre de cuadrados, luego de ser reducible, su factorización es única en irreducible distintos de $\mathbb{Z}_3[x]$. vamos a aplicar el algoritmo que nos da el criterio de irreducibilidad:

[25]:
```
algoritmo_cf(f,R,3)
```

[25]: 308

Así el menor cuerpo de característica 3 que contiene todas las raices del polinomio sería $F_{3^{308}}$.

Como podemo ver, hemos obtenido que $m = 308 \neq 24 = n$, y por el teorema que nos caracteriza los polinomios irreducibles podemos concluir que nuestro polinomio $f(x)$ es **reducible** en $\mathbb{Z}_3[x]$.

En nuestro caso, $f(x)$ ya era libre de cuadrados, luego vamos a calcular la matriz de Berlekamp directamente.

Pasamos a calcular las potencias x^{2i} mod $g(x)$ para $0 \leq i \leq n-1 = 23$

[26]:
```
lista_coef=[]
for i in range(24):
    a=R(x^(3*i))
    mod=a%f
    print()
    print(mod)
```

```
    coef=mod.coefficients(sparse=False)
    coef=prepara(coef,24)
    lista_coef.append(coef)
```

```
1

x^3

x^6

x^9

x^12

x^15

x^18

x^21

2*x^23 + 2*x^20 + 2*x^18 + 2*x^14 + 2*x^13 + 2*x^11 + 2*x^10 + 2*x^9 + 2*x^8 +
2*x^5 + 2*x^4 + 2*x^2 + 2

x^22 + x^21 + 2*x^20 + 2*x^19 + x^18 + 2*x^17 + 2*x^14 + x^13 + 2*x^12 + x^10 +
2*x^5 + 2*x^4 + x^3 + 2*x^2 + 2*x + 1

2*x^23 + 2*x^22 + 2*x^20 + 2*x^19 + 2*x^17 + x^16 + x^15 + 2*x^14 + x^13 +
2*x^12 + 2*x^11 + 2*x^10 + 2*x^9 + 2*x^8 + 2*x^7 + x^5 + 2*x^4 + 2*x

2*x^23 + x^19 + x^18 + 2*x^17 + 2*x^16 + 2*x^14 + x^8 + x^6 + x^2

x^23 + 2*x^22 + x^20 + x^19 + x^18 + 2*x^17 + x^16 + 2*x^13 + 2*x^11 + x^10 +
x^9 + x^8 + x^7 + x^5 + 2*x^4 + 2*x^3 + 2*x^2 + 2*x + 1

2*x^23 + 2*x^20 + x^18 + x^16 + x^15 + 2*x^14 + x^13 + 2*x^12 + 2*x^8 + x^7 +
2*x^5 + 2*x^4 + 2*x + 1

x^22 + 2*x^20 + 2*x^18 + 2*x^17 + 2*x^16 + 2*x^15 + 2*x^13 + 2*x^10 + x^4 +
2*x^2 + 2*x + 1

x^21 + x^19 + 2*x^16 + 2*x^15 + 2*x^12 + x^8 + x^7 + 2*x^6 + 2*x^5 + x^2 + 2*x +
1

2*x^23 + x^22 + 2*x^20 + 2*x^19 + x^18 + 2*x^15 + 2*x^14 + 2*x^13 + x^9 + x^8 +
x^4 + x^3 + 2*x^2 + 2
```

```
x^23 + 2*x^21 + x^19 + x^18 + 2*x^17 + 2*x^15 + 2*x^11 + x^10 + 2*x^8 + 2*x^7 +
2*x^5 + x + 2

2*x^21 + x^20 + x^19 + 2*x^18 + 2*x^16 + 2*x^11 + 2*x^10 + x^9 + 2*x^8 + 2*x^7 +
x^5 + 2*x^2 + x

2*x^23 + x^22 + 2*x^21 + x^20 + 2*x^19 + x^18 + x^12 + x^9 + 2*x^8 + 2*x^4 + x^2
+ 1

x^23 + 2*x^21 + x^20 + x^19 + x^16 + x^14 + x^13 + x^11 + 2*x^10 + x^9 + 2*x^6 +
2*x^5 + x^4 + 2*x^3 + x^2 + x

x^23 + x^21 + 2*x^20 + 2*x^19 + x^17 + 2*x^14 + x^13 + x^12 + 2*x^8 + 2*x^6 +
2*x^5 + x^3 + 2*x^2 + x

x^22 + x^21 + x^20 + x^19 + x^18 + 2*x^17 + x^15 + 2*x^14 + x^10 + x^9 + 2*x^7 +
x^6 + x^5 + x^4 + x^3 + x + 1

x^23 + x^22 + 2*x^20 + 2*x^19 + x^18 + 2*x^17 + 2*x^15 + 2*x^14 + x^13 + 2*x^11
+ x^10 + x^8 + x^7 + 2*x^5 + x^4 + 2*x
```

Ahora construimos la matriz de Berlekamp:

[27]:
```
B=Matrix(lista_coef)
```

[28]:
```
print(B)
```

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
[2 0 2 0 2 2 0 0 2 2 2 2 0 2 2 0 0 0 2 0 2 0 0 2]
[1 2 2 1 2 2 0 0 0 0 1 0 2 1 2 0 0 2 1 2 2 1 1 0]
[0 2 0 0 2 1 0 2 2 2 2 2 2 1 2 1 1 2 0 2 2 0 2 2]
[0 0 1 0 0 0 1 0 1 0 0 0 0 0 2 0 2 2 1 1 0 0 0 2]
[1 2 2 2 2 1 0 1 1 1 1 2 0 2 0 0 1 2 1 1 1 0 2 1]
[1 2 0 0 2 2 0 1 2 0 0 0 2 1 2 1 1 0 1 0 2 0 0 2]
[1 2 2 0 1 0 0 0 0 0 2 0 0 2 0 2 2 2 2 0 2 0 1 0]
[1 2 1 0 0 2 2 1 1 0 0 0 2 0 0 2 2 0 0 1 0 1 0 0]
[2 0 2 1 1 0 0 0 1 1 0 0 0 2 2 2 0 0 1 2 2 0 1 2]
[2 1 0 0 0 2 0 2 2 0 1 2 0 0 0 2 0 2 1 1 0 2 0 1]
[0 1 2 0 0 1 0 2 2 1 2 2 0 0 0 0 2 0 2 1 1 2 0 0]
[1 0 1 0 2 0 0 0 2 1 0 0 1 0 0 0 0 0 1 2 1 2 1 2]
[0 1 1 2 1 2 2 0 0 1 2 1 0 1 1 0 1 0 0 1 1 2 0 1]
```

```
[0 1 2 1 0 2 2 0 2 0 0 0 1 1 2 0 0 1 0 2 2 1 0 1]
[1 1 0 1 1 1 1 2 0 1 1 0 0 0 2 1 0 2 1 1 1 1 1 0]
[0 2 0 0 1 2 0 1 1 0 1 2 0 1 2 2 0 2 1 2 2 0 1 1]
```

Calculamos la diferencia con la identidad:

[29]:
```
I=matrix.identity(24)
D=B-I
print(D)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 2 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
[0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
[2 0 2 0 2 2 0 0 1 2 2 2 0 2 2 0 0 0 2 0 2 0 0 2]
[1 2 2 1 2 2 0 0 0 2 1 0 2 1 2 0 0 2 1 2 2 1 1 0]
[0 2 0 0 2 1 0 2 2 2 1 2 2 1 2 1 1 2 0 2 2 0 2 2]
[0 0 1 0 0 0 1 0 1 0 0 2 0 0 2 0 2 2 1 1 0 0 0 2]
[1 2 2 2 2 1 0 1 1 1 1 2 2 2 0 0 1 2 1 1 1 0 2 1]
[1 2 0 0 2 2 0 1 2 0 0 0 2 0 2 1 1 0 1 0 2 0 0 2]
[1 2 2 0 1 0 0 0 0 0 2 0 0 2 2 2 2 2 2 0 2 0 1 0]
[1 2 1 0 0 2 2 1 1 0 0 0 2 0 0 1 2 0 0 1 0 1 0 0]
[2 0 2 1 1 0 0 0 1 1 0 0 0 2 2 2 2 0 1 2 2 0 1 2]
[2 1 0 0 0 2 0 2 2 0 1 2 0 0 0 2 0 1 1 1 0 2 0 1]
[0 1 2 0 0 1 0 2 2 1 2 2 0 0 0 0 2 0 1 1 1 2 0 0]
[1 0 1 0 2 0 0 0 2 1 0 0 1 0 0 0 0 0 1 1 1 2 1 2]
[0 1 1 2 1 2 2 0 0 1 2 1 0 1 1 0 1 0 0 1 0 2 0 1]
[0 1 2 1 0 2 2 0 2 0 0 0 1 1 2 0 0 1 0 2 2 0 0 1]
[1 1 0 1 1 1 1 2 0 1 1 0 0 0 2 1 0 2 1 1 1 1 0 0]
[0 2 0 0 1 2 0 1 1 0 1 2 0 1 2 2 0 2 1 2 2 0 1 0]
```

Calculamos el rango de la matriz:

[30]:
```
print(24-D.rank())
```

```
4
```

Por lo tanto el sistema tiene 2 soluciones. Vamos a resulver el sistema de ecuaciones encontrando una base del espacio vectorial que define la matriz.

[31]:
```
D.kernel()
```

[31]:
```
Vector space of degree 24 and dimension 4 over Finite Field of size 3
Basis matrix:
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 1 0 0 2 0 1 1 1 1 0 1 1 0 1 2 2 2 0 0 2 2 0 0]
```

```
[0 0 1 0 1 1 2 1 2 1 0 0 2 2 2 0 0 2 0 1 1 0 2 1]
[0 0 0 1 2 0 1 2 0 0 0 2 0 0 1 2 1 2 2 2 2 1 1 2]
```

Como hacemos siempre, descartamos el polinomio $h(x) = 1$ pues no es f-reductor, por lo que solo el segundo, tercero y cuarto nos dará la factorización de f en $\mathbb{Z}_3[x]$.

[32]:
```
h2=R([0, 1, 0, 0, 2, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 2 ,2 ,2, 0, 0, 2, 2, 0, 0])
h3=R([0, 0, 1, 0, 1, 1, 2, 1, 2, 1, 0, 0, 2, 2, 2, 0, 0, 2, 0, 1, 1, 0, 2, 1])
h4=R([0, 0, 0, 1, 2, 0, 1, 2, 0, 0, 0, 2, 0, 0, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2])

h2
```

[32]:
```
2*x^21 + 2*x^20 + 2*x^17 + 2*x^16 + 2*x^15 + x^14 + x^12 + x^11 + x^9 + x^8 +
x^7 + x^6 + 2*x^4 + x
```

[33]:
```
h3
```

[33]:
```
x^23 + 2*x^22 + x^20 + x^19 + 2*x^17 + 2*x^14 + 2*x^13 + 2*x^12 + x^9 + 2*x^8 +
x^7 + 2*x^6 + x^5 + x^4 + x^2
```

[34]:
```
h4
```

[34]:
```
2*x^23 + x^22 + x^21 + 2*x^20 + 2*x^19 + 2*x^18 + 2*x^17 + x^16 + 2*x^15 + x^14
+ 2*x^11 + 2*x^7 + x^6 + 2*x^4 + x^3
```

Ahora calculamos los siguientes mcd del plinomio $f(x)$ con $h_2(x)$,$h_3(x)$ y $h_4(x)$, lo que nos dará la factorización de f:

Como vemos a continuación obtenemos algunos polinomios repetidos:

[35]:
```
f1=f.gcd(h2)
```

[36]:
```
f2=f.gcd(h2-1)
```

[37]:
```
f3=f.gcd(h2-2)
```

[38]:
```
f4=f.gcd(h3)
```

[39]:
```
f5=f.gcd(h3-1)
```

El siguiente coincide con $f_3(x)$.

[40]:
```
f.gcd(h3-2)
```

[40]:
```
x^11 + x^10 + x^9 + 2*x^8 + x^7 + x^6 + 2*x^5 + x^4 + x^2 + x + 1
```

[41]:
```
f6=f.gcd(h4)
```

El siguiente coincide con $f_3(x)$.

[42]:
```
f.gcd(h4-1)
```

[42]: x^11 + x^10 + x^9 + 2*x^8 + x^7 + x^6 + 2*x^5 + x^4 + x^2 + x + 1

[43]:
```
f7=f.gcd(h4-2)
```

Por otro lado, tenemos que $f_2(x) \cdot f_7(x) = f_4(x)$:

[44]:
```
f2*f7==f4
```

[44]: True

También tenemos que $f_2(x) \cdot f_5(x) = f_6(x)$:

[45]:
```
f2*f5==f6
```

[45]: True

También tenemos que $f_5(x) \cdot f_7(x) = f_1(x)$:

[46]:
```
f5*f7==f1
```

[46]: True

Por lo que podemos quitar $f_1(x)$,$f_4(x)$ y $f_6(x)$ si demostramos que los polinomios $f_2, f_3, f_5 y f_7$ son irreducibles. Por lo tanto tenemos que ver si son irreducibles $f_2, f_3, f_5 y f_7$.

[47]:
```
f2
```

[47]: x^2 + x + 2

[48]:
```
f3
```

[48]: x^11 + x^10 + x^9 + 2*x^8 + x^7 + x^6 + 2*x^5 + x^4 + x^2 + x + 1

[49]:
```
f5
```

[49]: x^7 + x^6 + 2*x^5 + 2*x^4 + x^2 + x + 2

[50]:
```
f7
```

[50]: x^4 + x^3 + x^2 + x + 1

Vamos a centrarnos en primer lugar en $f_2(x) = x^2 + x + 2$. En primer lugar vemos si es libre de cuadrados:

[51]:
```
f2_derivada=derivative(f2)
```

[52]:
```
f2.gcd(f2_derivada)
```

[52]: 1

Como vemos es libre de cuadrados, por lo que vamos a comprobar el cardinal del menor cuerpo finito que contiene todas las soluciones de $f_2(x)$.

[53]:
```
algoritmo_cf(f2,R,3)
```

[53]: 2

Así el menor cuerpo de característica 3 que contiene todas las raices del polinomio sería F_{3^2}.

Como podemos ver, hemos obtenido que $m = 2 = n$, y por el teorema que nos caracteriza los polinomios irreducibles podemos concluir que nuestro polinomio $f_2(x)$ es **irreducible** en $\mathbb{Z}_3[x]$.

Ahora comprobamos si $f_3(x) = x^{11} + x^{10} + x^9 + 2 * x^8 + x^7 + x^6 + 2 * x^5 + x^4 + x^2 + x + 1$ es irreducible.

[54]:
```
f3_derivada=derivative(f3)
```

[55]:
```
f3.gcd(f3_derivada)
```

[55]: 1

Como vemos es libre de cuadrados, por lo que vamos a comprobar el cardinal del menor cuerpo finito que contiene todas las soluciones de $f_3(x)$.

[56]:
```
algoritmo_cf(f3,R,3)
```

[56]: 11

Así el menor cuerpo de característica 3 que contiene todas las raices del polinomio sería $F_{3^{11}}$.

Como podemos ver, hemos obtenido que $m = 11 = n$, y por el teorema que nos caracteriza los polinomios irreducibles podemos concluir que nuestro polinomio $f_3(x)$ es **irreducible** en $\mathbb{Z}_3[x]$.

Ahora comprobamos con $f_5(x) = x^7 + x^6 + 2 * x^5 + 2 * x^4 + x^2 + x + 2$ es irreducible.

[57]:
```
f5_derivada=derivative(f5)
```

[58]:
```
f5.gcd(f5_derivada)
```

[58]: 1

Como vemos es libre de cuadrados, por lo que vamos a comprobar el cardinal del menor cuerpo finito que contiene todas las soluciones de $f_5(x)$.

[59]:
```
algoritmo_cf(f5,R,3)
```

[59]: 7

Así el menor cuerpo de característica 3 que contiene todas las raices del polinomio sería F_{3^7}.

Como podemos ver, hemos obtenido que $m = 7 = n$, y por el teorema que nos caracteriza los polinomios irreducibles podemos concluir que nuestro polinomio $f_5(x)$ es **irreducible** en $\mathbb{Z}_3[x]$.

Finalmente vamos a comprobar si $f_7(x) = x^4 + x^3 + x^2 + x + 1$ es irreducible.

[60]:
```
f7_derivada=derivative(f7)
```

[61]:
```
f7.gcd(f7_derivada)
```

[61]: 1

Como vemos es libre de cuadrados, por lo que vamos a comprobar el cardinal del menor cuerpo finito que contiene todas las soluciones de $f_7(x)$.

[62]:
```
algoritmo_cf(f7,R,3)
```

[62]: 4

Así el menor cuerpo de característica 3 que contiene todas las raices del polinomio sería F_{3^4}.

Como podemos ver, hemos obtenido que $m = 4 = n$, y por el teorema que nos caracteriza los polinomios irreducibles podemos concluir que nuestro polinomio $f_7(x)$ es **irreducible** en $\mathbb{Z}_3[x]$.

De esta forma podemos concluir que la descomposición en factores irreducibles de $f(x)$ sería la siguiente:

$$f(x) = f_2(x) \cdot f_3(x) \cdot f_5(x) \cdot f_7(x)$$

Como se puede comprobar a continuación.

[63]:
```
f==f2*f3*f5*f7
```

[63]: True

v) ¿Qué deduces sobre la reducibilidad de $f(x)$ en $\mathbb{Z}[x]$?.

Sabemos que si $f(x)$ factoriza sobre $\mathbb{Z}[x]$ entonces sus factores tienen el mismo grado módulo cualquier primo. Pero como hemos visto, en nuestro caso $f(x)$ factoriza en polinomios irreducibles de grado 1 y 23 en el caso de $\mathbb{Z}_2$ y en polinomios irreducibles de grado 2,4,7 y 11 en el caso de $\mathbb{Z}_3$ como podemos observar los grados son incompatibles y por lo tanto podemos afirmar que $f(x)$ **es irreducible** en $\mathbb{Z}[x]$.