Códigos y Criptografía

Práctica 1

Alonso Sayalero Blázquez

In [ ]:
```
#*****************************************************************
#                           Ejercicio 1
#*****************************************************************
# Creacion de un cuerpo finito con un numero primo de elementos
cuerpo = GF(13)

# Representacion de los elementos
for i in cuerpo:
    print(i)

# Suma y multiplicacion de elementos del cuerpo
print("Suma: ", cuerpo(9), " + ", cuerpo(11), " = ", cuerpo(9) + cuerpo(11))
print("multiplicacion: ", cuerpo(9), " * ", cuerpo(11), " = ", cuerpo(9) * cuerp
```

In [ ]:
```
# Creacion de un cuerpo finito con un numero potencia de un primo de elementos
cuerpo = GF(2^2, 'x')

# Representacion de los elementos
for i,a in enumerate(cuerpo):
    print("{} {}".format(i, a))

# Suma y multiplicacion de elementos del cuerpo
elemento_1 = cuerpo.random_element()
elemento_2 = cuerpo.random_element()

print("Suma: ", elemento_1, " + ", elemento_2, " = ", elemento_1 + elemento_2)
print("multiplicacion: ", elemento_1, " * ", elemento_2, " = ", elemento_1 * ele
```

In [ ]:
```
#*****************************************************************
#                           Ejercicio 2
#*****************************************************************
cuerpo_finito = GF(5)

# Creacion de la matriz para la creacion del codigo lineal
matriz = matrix(cuerpo_finito, [[1,2,3,4,0],[3,4,0,2,3]])

# Codigo lineal
codigo = LinearCode(matriz)

# Parametros
n = codigo.length()             # Longitud
k = codigo.dimension()          # Dimension
d = codigo.minimum_distance()   # Distancia Minima
print("[", n, ", ", k, ", ", d,"]")

# Matriz generadora
G = codigo.generator_matrix()
```

```
# Matriz de control
H = codigo.parity_check_matrix()

# Polinomio de pesos
coeficientes_polinomio_pesos = codigo.weight_distribution()
polinomio_pesos = 0
R.<x> = QQ[]
for i in range(len(coeficientes_polinomio_pesos)):
    polinomio_pesos = polinomio_pesos + coeficientes_polinomio_pesos[i] * x^i

# Codigo Dual
codigo_dual = codigo.dual_code()

# Introduccion de errores y decodificacion
mensaje = codigo.random_element()
# Introduccion de error en la tercera posicion
error = vector(cuerpo_finito, [0,0,4,0,0])
mensaje_error = mensaje + error
# Decodificacion
mensaje_final = codigo.decode_to_code(mensaje_error)

print(mensaje)
print(mensaje_error)
print(mensaje_final)
```

In [ ]:

```
#*********************************************************************
#                          Ejercicio 3
#                      Decodificacion Unica
#*********************************************************************
tamano_cuerpo = 11
if tamano_cuerpo in Primes():
    cuerpo_finito = GF(tamano_cuerpo)
else:
    cuerpo_finito = GF(tamano_cuerpo, 'x')
n = 10                    # Longitud
k = 5                     # Dimension
d = n - k + 1             # Distancia Minima
t = math.floor((d - 1)/2) # Capacidad Correctora

# Definicion de l0 y l1
l0 = n - 1 - t
l1 = n - 1 - t - (k - 1)

# Creacion de la matriz vacia con el tamaño y cuerpo finito adecuados
largo = l0 + l1 + 2
matriz = matrix(cuerpo_finito, [[0] * largo] * n)

# Obtencion de una base.
elemento_primitivo = primitive_root(tamano_cuerpo)
base = []
for i in range(1, tamano_cuerpo):
    base.append(elemento_primitivo^(i-1) % tamano_cuerpo)

# Mensaje recibido
r = (5, 9, 0, 9, 0, 1, 0, 7, 0, 5)

# Creacion y resolucion del sistema de ecuaciones
for i in range(n):
    for j in range(l0 + 1):
        matriz[i, j] = (base[i]^j) % tamano_cuerpo
```

```
for i in range(n):
    auxiliar = 0
    for j in range(l0 + 1, largo):
        matriz[i, j] = ((base[i]^auxiliar) * r[i]) % tamano_cuerpo
        auxiliar += 1

solucion = matriz.right_kernel()

# Extraccion de Q0 y Q1 de una de las soluciones
solucion_escogida = solucion[1]
Q0 = 0
Q1 = 0
R.<x> = QQ[]

for i in range(l0 + 1):
    Q0 = Q0 + solucion_escogida[i] * x^i

for i in range(l1 + 1):
    Q1 = Q1 + solucion_escogida[l0 + 1 + i] * x^i

# Calculo de g
g = -Q0/Q1

# Comprobacion final y descodificacion
F.<x> = PolynomialRing(cuerpo_finito) # Pk

mensaje_final = []
for i in range(n):
    mensaje_final.append(g(base[i]))

distancia = 0
for i in range(len(mensaje_final)):
    if mensaje_final[i] != r[i]:
        distancia += 1

if g in F and distancia <= t:
    print(mensaje_final)
else:
    print("Error en el mensaje")
```

In [ ]:

```
#*****************************************************************
#                          Ejercicio 3
#                    Decodificacion en Lista
#*****************************************************************
tamano_cuerpo = 16
if tamano_cuerpo in Primes():
    cuerpo_finito = GF(tamano_cuerpo)
else:
    cuerpo_finito = GF(tamano_cuerpo, 'a')
n = 15                    # Longitud
k = 3                     # Dimension
d = n - k + 1             # Distancia Minima
tamano_lista = 2          # Tamaño de la Lista
tau = 6                   # Tau

# Base
a = cuerpo_finito.0
base = []
for i in range(n):
```

```
    base.append(a^i)

# Mensaje recibido
r = (2, 0, 11, 0, 0, 2, 3, 8, 7, 0, 1, 14, 15, 2, 4)

# Sumatorio de sistemas de ecuaciones
x,y = PolynomialRing(cuerpo_finito, 2, ['x', 'y']).gens()
Qxy = 0
for j in range(tamano_lista + 1):
    matriz_diagonal = matrix(cuerpo_finito, [[0] * n] * n)
    for i in range(n):
        matriz_diagonal[i, i] = r[i]^j

    lj = n - tau - 1 - j * (k - 1)
    matriz = matrix(cuerpo_finito, [[0] * lj] * n)
    for i in range(n):
        for k in range(lj):
            matriz[i, j] = base[i] ^ j

    matriz_multiplicada = matriz_diagonal * matriz

    soluciones = matriz_multiplicada.right_kernel()

    if len(soluciones) > 1:
        Qj = 0
        solucion = soluciones[1]
        for i in range(lj):
            Qj = Qj + solucion[i] * x^i

        Qxy = Qxy + Qj * y^j

# Factores de Q(x, y) de la forma (y - f(x)) con grado de f(x) < k
if Qxy != 0:
    factor_qxy = factor(Qxy)

lista_factores = list(factor_qxy)
lista_candidatos = []
for i in range(len(lista_factores)):
    grados = lista_factores[i][0].degrees()
    if grados[1] == 1 and grados[0] < k:
        lista_candidatos.append((lista_factores[i][0] - y))


polinomio = (a^2 + 1) * x^2 + (a^3 + a)*x + 1

factores = matrix(cuerpo_finito,[[0] * n] * len(lista_candidatos))
print(factores)
print()

for i in range(len(lista_candidatos)):
    polinomio = lista_candidatos[i]
    for j in range(n):
        factores[i,j] = polinomio(x=base[j])

#Salida
output = []

for i in range(len(lista_candidatos)):
    distancia = 0
    polinomio = factores[i]
```

```
    for j in range(n):
        if polinomio[j] != r[j]:
            distancia += 1
    if distancia <= t:
        output.append(polinomio)

print(output)
```