Practica1

November 30, 2021

[ ]:
```
#Jhon Steeven Cabanilla Alvarado
#--------------------------------------------------------------------------------
                                    #EJERCICIO #1
#--------------------------------------------------------------------------------
#Creamos un cuerpo finito de 7 elementos
#Primer comando
C = GF(7)
#Segundo comando
C2 = FiniteField(7)

#Representacion de sus elementos. Seria lo mismo para C2
for i in C:
    print(i)

#Operacion de suma
print("Suma -->", C(3),"+",C(6), "=", C(3) + C(6)) #Accedemos a los distintos␣
 ↪elementos del cuerpo por el indice

#Operacion de multiplicacion
print("Multiplicacion -->", C(3),"*",C(6), "=", C(3) * C(6))

#Creamos un cuerpo finito de 2^5 elementos
#Primer comando
C = GF(2^5, 'a')

#Representacion de sus elementos
for i,x in enumerate(C):
    print("{} {}".format(i, x))

#Suma
s1 = C.random_element()
s2 = C.random_element()
print("Suma -->",s1," + ",s2, "=", s1 + s2 )#Obtenemos un elemento aleatorio del␣
 ↪codigo mediante random_element()

#Multiplicacion
m1 = C.random_element()
```

```
m2 = C.random_element()
print("Multiplicacion -->",s1," * ",s2, "=", s1 * s2 )

#-------------------------------------------------------------------------------
                                    #EJERCICIO #2
#-------------------------------------------------------------------------------
#Definimos un cuerpo finito con 7 elementos
F = GF(7)

#Procedemos a crear una matriz que nos ayudará a definir el codigo lineal
G = matrix(F,[[1,2,1,1],[3,6,5,1]])

##Definimos el codigo##
C = LinearCode(G)

##Parametros##
n = C.length() #longitud
k = C.dimension() #dimension
d = C.minimum_distance() #distancia minima

##Matriz generadora##
G2 = C.generator_matrix()

##Matriz de control##
H = C.parity_check_matrix()

##Polinomio de pesos##
pol_pesos = C.weight_distribution() #coeficientes del polinomio de pesos
W = 0
R.<x> = QQ[]
for i in range(len(pol_pesos)):
    W = W + pol_pesos[i] * x^i

##Descodificacion##
#Comenzamos obteniendo una palabra cualquiera del codigo
c = C.random_element()
#Ahora introducimos un error, por ejemplo, en la primera posicion
e = vector(F,[6,0,0,0])
r = c + e
#Finalmente decodificamos
C.decode_to_code(r)

##Codigo dual##
D = C.dual_code()


#-------------------------------------------------------------------------------
```

```
                    #EJERCICIO #3
                #DECODIFICACION UNICA
#-------------------------------------------------------------------------------
C = GF(11)
##Parametros##
n = 10
k = 5
d = 6
t = math.floor((d-1)/2) #Capacidad correctora del codigo

#Definimos lo y l1 como:
l0 = n - 1 - t
l1 = n - 1 - t - (k-1)

#Creamos una matriz de dimensiones y cuerpo finito adecuados, inicialmente vacía
nn = l0 + 1 + l1 + 1
M = matrix(C, [[0] * nn] * n)
print(M)
print()

#Obtenemos una base. Como 2 es un elemento primitivo de F11, podemos tomar xi = 2^i-1 mod 11
base = []
for i in range(1,11):
    base.append(2^(i-1)%11)
print("base:", base)
print()

r = (5,9,0,9,0,1,0,7,0,5) #Recibimos r
print("Recibimos:",r)
print()

##Procedemos a resolver el sistema de ecuaciones##
#Primera parte de la matriz(hasta l0+1)
for f in range(l0+1):
    for c in range(n):
        M[c,f] = (base[c]^f)%11

#Segunda parte de la matriz(desde l0+1 hasta l0+1+l1+1)
temporal = 0
for f in range(l0+1, nn):
    for c in range(n):
        if f == l0+1:
            M[c, f] = r[c]
        else:
            M[c, f] = ((base[c]^temporal) * r[c])%11
```

```
    temporal += 1

print(M)
print()

V = M.right_kernel()
print(V)
print()


##Procedemos a buscar Q0 y Q1##
#Primero escogemos, por ejemplo A[1]
l = V[1]
Q0 = 0
Q1 = 0
R.<x> = QQ[]
for i in range(l0+1):
    Q0 = Q0 + l[i] * x^i

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

print("Q0:", Q0)
print("Q1:", Q1)

#Ahora calculamos g
g = -Q0/Q1
print("g:", g)
print()

##Comprobaciones finales##
#Primero --> Si g(x) pertenece a  Pk
desc = []
for i in range(n):
    desc.append(g(base[i]))
print("Descodificacion:",desc)
print()

#Segundo --> Comprobamos si se ha superado o no la capacidad correctora del␣
 ↪codigo
dist = 0
fallos = []
for i in range(len(desc)):
    if desc[i] != r[i]:
        dist += 1
        fallos.append(i)
print("distancia:", dist)
```

```
if dist != 0:
    print("Se han corregido",d, "errores en las posiciones correspondiente a:",␣
 ↪"2^",fallos[0],"=",(2^fallos[0])%11,"y 2^",fallos[1],"=",(2^fallos[1])%11)

#--------------------------------------------------------------------------------
                                    #EJERCICIO #3
                                #DECODIFICACION EN LISTA
#--------------------------------------------------------------------------------
##Parametros##
n = 15
k = 3
d = n - k + 1
l = 2 #List size
t = 6 #tau

#Definimos GF y la letra constante a para hacer referencia a alfa.
F = GF(16,'a')
a = F.0

base = [a^0,a^1,a^2,a^3,a^4,a^5,a^6,a^7,a^8,a^9,a^10,a^11,a^12,a^13,a^14]
r = [1,1,1,1,0,0,0,0,0,0,0,0,0,0,0] #Palabra recibida

#Matriz generadora
G = matrix(F, [[0] * n] * n)
for i in range(n):
    for j in range(n):
        G[i,j] = base[i]^j

#Definimos el codigo
C = LinearCode(G)

x,y = PolynomialRing(F,2,['x','y']).gens()
Qxy = 0

#Implementacion sumatorio
for j in range(l+1):

    ##Matriz diagonal##
    D = matrix(F, [[0] * n] * n)
    for i in range(n):
        D[i,i] = r[i]^j
    print(D)
    print()

    ##Seguna Matriz##
    lj = n-t-1-j*(k-1)
```

```
    M = matrix(F, [[0] * lj] * n)
    for i in range(lj):
        for j in range(n):
            M[j,i] = base[j]^i
    print(M)
    print()

    ##1) Encontramos una solucion NO NULA del sistema de ecuaciones lineales##
    P = D * M
    print(P)
    print()
    A = P.right_kernel()
    print(A)
    print()

    ##2)Encontramos Qj(x) y Q(x,y) ##
    if (len(A) > 1):
        #Qj
        #Qj = PolynomialRing(F, 2, 'ax')
        Qj = 0
        var = (A[1])
        for k in range(lj):
            Qj = Qj + var[k] * (x^k)

        #Qxy
        Qxy = Qxy + Qj * y^j

##fin for

print("Q(x,y):", Qxy)
print()

##3)Procedemos a encontrar los factores de Q(x,y) de la forma (y-f(x)) con␣
 ↪gradof(x) < k##
if (Qxy != 0):
    Factor = factor(Qxy)

print(Factor)
print()

lista_factores = list(Factor)
lista_candid = []

for i in range(len(lista_factores)):
        grados = lista_factores[i][0].degrees()
        if grados[1] == 1 and grados[0] < k:
```

```
            lista_candid.append((lista_factores[i][0]-y)) #Multiplicar por el␣
 ↪coeficiente de y * -1


print("lista candidatos:", lista_candid)

polino = (a^2 + 1) * x^2 + (a^3 + a)*x + 1
print(polino(x = 1))
print()

Factores = matrix(F,[[0] * n] * len(lista_candid))
print(Factores)
print()

for i in range(len(lista_candid)):
    polino = lista_candid[i]
    for j in range(n):
        Factores[i,j] = polino(x=base[j])

#Como output tendremos la lista formada por todos los fatores anteriores que␣
 ↪verifiquen --> d( (f(x1),...f(xn)),(r1,...,rn))<=tau
output = []

for i in range(len(lista_candid)):
    dist = 0
    polino = Factores[i]
    for j in range(n):
        if polino[j] != r[j]:
            dist += 1
    if dist <= t:
        output.append(polino)

print("Lista de palabras con distancia d < tau:", output)


#--------------------------------------------------------------------------------
                                    #EJERCICIO #4
#--------------------------------------------------------------------------------
import numpy as np
#Definimos un cuerpo finito con 7 elementos
F = GF(2)

#Procedemos a crear una matriz que nos ayudará a definir el codigo lineal
G = matrix(F,[[1,0,0,1,0,1],[0,1,0,1,1,0],[0,1,0,1,1,0]])

##Definimos el codigo C##
C = LinearCode(G)
```

```
##Parametros##
n = C.length() #longitud
k = C.dimension() #dimension
d = C.minimum_distance() #distancia minima

#Matriz de control
H = C.parity_check_matrix()

#Probabilidad de error en cada bit
p = 0.5

#a) Probabilidad de tener un error no detectable

#Calculamos el polinomio de pesos
pol_pesos = C.weight_distribution() #coeficientes del polinomio de pesos
W = 0
R.<x> = QQ[]
for i in range(len(pol_pesos)):
    W = W + pol_pesos[i] * x^i
print("Polinomio de pesos:", W)

#La probabilidad de enviar una palabra del codigo y recibir una palabra del␣
↪codigo diferente(error no detectable) para F2 es:
#(1-p)^n * (W(p/(1-p)) -1)

prob = (1-p)^n * (W(p/(1-p))-1)
print("Probabilidad de error no detectable:", prob)
print()

#b)
def error(c, p):
    for i in range(len(c)):
        if random() < p:
            #Como estamos con codigos binarios, al introducir un error cambiamos␣
↪el bit 0 por el 1 y viceversa.
            if c[i] == 0:
                c[i] = 1
            else:
                c[i] = 0
    #fin for
    return c

#c)
iteraciones = 1000000
cont = 0
```

```
for i in range(iteraciones):
    igual = False
    while(not igual):
        c1 = C.random_element()
        c = copy(c1)
        comp = vector((np.transpose(c) * H)[0])
        if comp.is_zero():
            igual = True

    r = error(c, p)
    comp2 = vector((np.transpose(r) * H)[0])
    if comp2.is_zero():
        cont += 1

#Ahora dividimos cont, con el numero de errores, por las iteraciones para␣
 ↪obtener la probabilidad final
prob2 = cont/iteraciones
print("Probabilidad de error no detectable, de forma experimental:", prob2)
print()
```

```
[ ]:
```