INF243 - Mandatory 3

By Vegard Berge

Q1 - Basics on Finite Fields

i. For an irreducible polynomial $p(x) = x^4 + x^3 + x^2 + x + 1 \in \mathbb{F}_2[x]$ with root α, use it to create a polynomial representation of $GF(2^4)$. Use this representation of $\mathbb{F}_{2^4}$ to show that α is not a primitive element while $\beta = \alpha + 1$ is primitive. (***A primitive element*** w ***in*** $\mathbb{F}_{2^4}$ ***can represent any non - zero element as*** α^i ***for certain integer*** $0 \leq i \leq 2^4 - 2$ ***.).***
ii. Find the minimal polynomial $q(x)$ of $\beta = \alpha + 1$
iii. Use the minimal polynomial $q(x)$ to generate $\mathbb{F}_{2^4}$, and create a table such as in the textbook (page 206)

Answers:

(i)

First, a primitive element $f(x)$ is an element α in $\mathbb{F}[x]/(f(x))$ that generates all the non - zero elements of the field as powers of α. In other words, every non - zero element of the field can be expresses as α^i for some positive integer $(\mathbb{Z}^+) \in GF(2^4)$. To show, that α is not a primitive element, I've implemented the following in SageMath:

```
# Define polynomial ring R in GF(2) with variable x - in finite field GF(2)
R.<x> = GF(2)[]
# irr. poly. p(x)
p = x^4 + x^3 + x^2 + x + 1
# field extension GF(2^4) by the quot. ring
F.<alpha> = GF(2).extension(p)

# Verify that alpha is indeed a root of p(x) i.e should be zero
p(alpha) # => 0


# Step 5: Check if alpha is a primitive element
for i in range(1, 15):
    alpha_i = alpha^i
    if alpha_i == 0:
        print("alpha^{} = 0".format(i))
        break
    elif alpha_i == 1:
        print("alpha^{} = 1".format(i))
    elif alpha_i == alpha^(i-1):
        print("alpha^{} = alpha^{}".format(i, i-1))
        break
    else:
        print("alpha^{} = {}".format(i, alpha_i))
```

Output:

```
alpha^1 = alpha
alpha^2 = alpha^2
alpha^3 = alpha^3
alpha^4 = alpha^3 + alpha^2 + alpha + 1
alpha^5 = 1
alpha^6 = alpha
alpha^7 = alpha^2
alpha^8 = alpha^3
alpha^9 = alpha^3 + alpha^2 + alpha + 1
alpha^10 = 1
alpha^11 = alpha
alpha^12 = alpha^2
alpha^13 = alpha^3
alpha^14 = alpha^3 + alpha^2 + alpha + 1
```

We see that alpha generates the identity element of the field, at α = 5, and is thus not a primitive element. Now to show that $\beta = \alpha + 1$ is an primitive element:

```
*==========* SAME SETUP AS THE CODE ABOVE *============*
....

# Step 6: Check if beta is a primitive element
beta = alpha + 1
for i in range(1, 15):
    beta_i = beta^i
    if beta_i == 0:
        print("beta^{} = 0".format(i))
        break
    elif beta_i == 1:
        print("beta^{} = 1".format(i))
    else:
        print("beta^{} = {}".format(i, beta_i))
```

output:

```
beta^1 = alpha + 1
beta^2 = alpha^2 + 1
beta^3 = alpha^3 + alpha^2 + alpha + 1
beta^4 = alpha^3 + alpha^2 + alpha
beta^5 = alpha^3 + alpha^2 + 1
beta^6 = alpha^3
beta^7 = alpha^2 + alpha + 1
beta^8 = alpha^3 + 1
beta^9 = alpha^2
beta^10 = alpha^3 + alpha^2
beta^11 = alpha^3 + alpha + 1
beta^12 = alpha
beta^13 = alpha^2 + alpha
beta^14 = alpha^3 + alpha
```

We see that β generates all the non - zero elements $\Rightarrow$ and is thus a primitive element.

(ii)

To find the minimal polynomial $q(x)$ of $\beta = \alpha + 1$, recall that the **minimal polynomial** $q(x)$ of β is the poly. s.t.

- $q(\beta) = 0$
- If $f(\beta) = 0, \ then \, q(x) \mid f(x)$

To find the minimal polynomial $q(x)$ of β, we express β^n for all $n \geq 1$ that satisfies:

$$\beta = \alpha + 1, \alpha = \beta + 1$$

Find cyclotomic coset

$$c_i = \{i * q^j mod(q^m - 1)\}, i = 1, q = 2, j = (0, 1, \ldots, m-1) \\ = \{1, 2, 4, 8\}$$

Then solve for:

$$min.poly = (x - (\beta))(x - (\beta)^2)(x - (\beta)^4)(x - (\beta)^8) \\ = (x - (\alpha + 1))(x - (\alpha + 1)^2)(x - (\alpha + 1)^4)(x - (\alpha + 1)^8)$$

Now, I solved this expression in SageMath. *(In two ways, to show both the formula, and the built-in function in sagemath)*

```
R.<x> = PolynomialRing(GF(2))
p = x^4 + x^3 + x^2 + x + 1
F.<a> = GF(2^4).extension(p) # FiniteField(2^4)
# a.minimal_polynomial()
beta = F("a+1")
```

```
beta.minpoly()
```

```
a^4 + a^3 + 1
```

(iii)

Use the minimal polynomial $q(x)$ to generate $\mathbb{F}_{2^4}$, and create a table such as in the textbook (page 206)

$$\alpha^4 + \alpha^3 + 1$$

i	$Polynomial\ repr.$	$Vector Repr.$	α^n	$Logarithm\ n$	$ZechLogarithm$
1.	α	0010	α	1	12
2.	α^2	0100	α^2	2	9
3.	α^3	1000	α^3	3	4
4.	$\alpha^3 + 1$	1001	α^4	4	3
5.	$\alpha^3 + \alpha + 1$	1011	α^5	5	10
6.	$\alpha^3 + \alpha^2 + \alpha + 1$	1111	α^6	6	8
7.	$\alpha^2 + \alpha + 1$	0111	α^7	7	13
8.	$\alpha^3 + \alpha^2 + \alpha$	1110	α^8	8	6
9.	$\alpha^2 + 1$	0101	α^9	9	2
10.	$\alpha^3 + \alpha$	1010	α^{10}	10	5
11.	$\alpha^3 + \alpha^2 + 1$	1101	α^{11}	11	14
12.	$\alpha + 1$	0011	α^{12}	12	1
13.	$\alpha^2 + \alpha$	0110	α^{13}	13	7
14.	$\alpha^3 + \alpha^2$	1100	α^{14}	14	11

Q2 - Basics on factorization

Partition the set $\{1, 2, \ldots, 2^m - 2\}$ into cyclotomic cosets modulo $2^m - 1$ for $m = 3, 4, 5, 6$. Suppose α is a primitive element of $GF(2^5)$ generated by $x^5 + x^2 + 1$. Use your cyclotomic cosets for $m = 5$ to factorize $x^{31} - 1$ into a product of polynomials over $\mathbb{F}_2$.

(i)

Partition into cyclotomic cosets:

$$\begin{gathered}
c_3 = \{1,2,4,5,7,8\},\{3,6\} \\
c_4 = \{0,1,2,4,8\},\{0,3,6,8,12\},\{0,4,5,8,10\},\{0,7,8,12,14\},\{0,2,4,8,9\}, \\
\{0,6,8,11,12\},\{0,4,8,10,13\}, \\
c_5 = \{0,1,2,4,8,16\},\{0,3,6,12,16,24\},\{0,5,8,10,16,20\},\{0,7,14,16,24,28\}, \\
\{0,4,8,9,16,18\},\{0,11,12,16,22,24\}, \\
\{0,8,13,16,20,26\},\{0,15,16,24,28,30\}, \\
\{0,2,4,8,16,17\},\{0,6,12,16,19,24\},\{0,8,10,16,20,21]\{0,14,16,23,24,28\}, \\
\{0,4,8,16,18,25\},\{0,12,16,22,24,27\},\{0,8,16,20,26,29\} \\
c_6 = \{1,2,4,8,16,32\},\{3,6,12,24,34,48\},\{5,10,18,20,36,40\},\{7,14,28,38,50,56\}, \\
\{9,10,18,20,36,40\},\{11,22,26,42,44,52\},\{13,22,26,42,44,52\},\{15,30,46,54,58,60\},\{6,12,17,24,34,48\}, \\
\{14,19,28,38,50,56\},\{21,22,26,42,44,52\},\{23,30,46,54,58,60\},\{14,25,28,38,50,56\}, \\
\{27,30,46,54,58,60\},\{29,30,46,54,58,60\}, \\
\{0,31\},\{2,4,8,16,32,33\},\{2,4,8,16,32,35\},\{6,12,24,34,37,48\}, \\
\{2,4,8,16,32,39\},\{10,18,20,36,40,41\},\{6,12,24,34,43,48\},\{14,28,38,45,50,56\}, \\
\{2,4,8,16,32,47\},\{10,18,20,36,40,49\},\{10,18,20,36,40,51\},\{22,26,42,44,52,53\}, \\
\{6,12,24,34,48,55\},\{22,26,42,44,52,57\},\{14,28,38,50,56,59\},\{30,46,54,58,60,61]\},
\end{gathered}$$

Solved in Sagemath:

```
m = 5
# compute modulus for the field GF(2^m)
p = 2^m -1
# Define the field GF(2^m) with primitive element alpha
GF = GF(2^m, 'a', modulus=x^5+x^2+1)
# compute a primitive element alpha in GF(2^m)
alpha = GF.multiplicative_generator()
# compute the cyclotomic cosets modulo p
cosets = []
for i in range(1, p):
    # check if i is in a new coset iff. add to list
    if all(pow(alpha, i, p) != pow(alpha, j, p) for j in range(i)):
        cosets.append([j for j in range(p) if pow(alpha, j, p)== pow(alpha, i, p)])
# Factorize x^31 - 1 over F_2 in the cyclotomic cosets
R.<x> = GF[]
f = x^31 - 1
for coset in cosets:
    # Check if the coset has more than one element
    if len(coset) > 1:
        # iff. construct the polynomial corresponding to the coset
        g = R(1)
        for i in coset:
            g *= x - alpha^i
        # Divide out the polynomial from x^31 -1
        f = f.quo_rem(g)[0] # tuple rem. (r, div)


# factors
faq = f.factor()
for i, f in enumerate(faq):
    print("%d.  %s" % (i+1, f)) # hate to do it, but index from 1..
```

Output

```
1.  (x + 1, 1)
2.  (x + a, 1)
3.  (x + a + 1, 1)
4.  (x + a^2, 1)
5.  (x + a^2 + 1, 1)
6.  (x + a^2 + a, 1)
7.  (x + a^2 + a + 1, 1)
8.  (x + a^3, 1)
9.  (x + a^3 + 1, 1)
10.  (x + a^3 + a, 1)
11.  (x + a^3 + a + 1, 1)
12.  (x + a^3 + a^2, 1)
13.  (x + a^3 + a^2 + 1, 1)
14.  (x + a^3 + a^2 + a, 1)
15.  (x + a^3 + a^2 + a + 1, 1)
16.  (x + a^4, 1)
17.  (x + a^4 + 1, 1)
18.  (x + a^4 + a, 1)
19.  (x + a^4 + a + 1, 1)
20.  (x + a^4 + a^2, 1)
21.  (x + a^4 + a^2 + 1, 1)
22.  (x + a^4 + a^2 + a, 1)
23.  (x + a^4 + a^2 + a + 1, 1)
24.  (x + a^4 + a^3, 1)
25.  (x + a^4 + a^3 + 1, 1)
26.  (x + a^4 + a^3 + a, 1)
27.  (x + a^4 + a^3 + a + 1, 1)
28.  (x + a^4 + a^3 + a^2, 1)
29.  (x + a^4 + a^3 + a^2 + 1, 1)
30.  (x + a^4 + a^3 + a^2 + a, 1)
31.  (x + a^4 + a^3 + a^2 + a + 1, 1)
```

Q3 - BCH codes

Let α be a root of the polynomial $f(x)x^6 + x^4 + x^3 + x + 1$ in $\mathbb{F}_2[x]$, which is used to generate the finite field $\mathbb{F}_{2^6}$.

Suppose a binary ***BCH*** code C of length 63 is defined by the generator polynomial $g(x)$ that has roots

$$\alpha, \alpha^3, \alpha^5, \alpha^6, \alpha^7$$

i. What is the BCH bound on the minimum distance of the code C

ii. Suppose a message **m** has a binary representation as

$$m = m_0 m_1 \dots m_{38}$$
$$= 000001111100000111110000011111000001010$$

Encode this message in the systematic way.

Answer (i)

BCH Bound:

The roots of $g(x)$ are a, a^3, a^5, a^6, a^7 . I find the cosets from these roots by:

$$c_i = \{i * q^j mod(q^m - 1)\}, i = 1, q = 2, j = (0, 1, \dots, m-1)$$
$$c_1 = \{1, 2, 4, 8, 16, 32\}$$
$$c_3 = \{3, 6, 12, 24, 48, 33\}$$
$$c_5 = \{5, 10, 20, 40, 17, 34\}$$
$$c_7 = \{7, 14, 28, 56, 49, 35\}$$

Note: $c_6 = c_3$

We list the powers:

$\{1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 17, 20, 24, 28, 32, 33, 34, 35, 40, 48, 49, 56\}$

We have eight consecutive powers, t is at least 4.

Thus, BCH bound of the code $bound\ d \geq 4 * 2 + 1$ = $d \geq 9$

The minimum distance of the code is at least 9.

I then find the generator polynomial by, taking the LCM of its root´s min. poly:

(in sagemath:)

```
p = 2
m = 6
R.<x> = PolynomialRing(GF(p))
Fm.<a> = FiniteField(p^m)
# g(x) = lcm(m_1(x),...,m_d-1(x))
# g(x) with coefficiants in GF(q) and divides x^n -1

a_1 = a.minimal_polynomial()
a_3 = (a^3).minimal_polynomial()
a_5 = (a^5).minimal_polynomial()
a_6 = (a^6).minimal_polynomial()
a_7 = (a^7).minimal_polynomial()

print(f"a_1: {a_1}")
print(f"a_3: {a_3}")
print(f"a_5: {a_5}")
print(f"a_6: {a_6}")
print(f"a_7: {a_7}")

# a_3 == a_6

a = lcm(a_1, a_3)
b = lcm(a_5, a_7)
g = lcm(a, b)
g
```

```
a_1: x^6 + x^4 + x^3 + x + 1
a_3: x^6 + x^5 + x^4 + x^2 + 1
a_5: x^6 + x + 1
a_6: x^6 + x^5 + x^4 + x^2 + 1
a_7: x^6 + x^3 + 1
```

```
g
```

```
x^24 + x^23 + x^21 + x^19 + x^18 + x^16 + x^15 + x^14 + x^12 + x^11 + x^10 + x^8 + x^7 + x^
5 + 1
```

ii) Encode message systematic way:

```
# use the g found in (i)
# converted 000001111100000111110000011111000001010 to m(x)
# m = x^5 + x^6 + x^7 + x^8 + x^9 + x^15 + x^16 + x^17 + x^18 + x^19 + x^25 + x^26 + x^27 + x^28 + x^29 + x^35 + x^37
m = x^37 + x^35 + x^29 + x^28 + x^27 + x^26 + x^25 + x^19 + x^18 + x^17 +
    x^16 + x^15 + x^9 + x^8 +x^7 + x^6 + x^5

# want to compute r(x) = x^24(m(x)) mod(g(x))
# then add r(x) to m

R = (m*x^24).mod(g)
R
# x^23 + x^21 + x^20 + x^19 + x^16 + x^13 + x^12 + x^11 + x^10 + x^8 + x^7 +
  # x^5 + x^4 + x^3 + x^2 + x + 1

#convert to binary strings for easy comp.
r = ''.join(str(c) for c in R.coefficients(sparse=False)[::-1]) #to binary form
r
# 101110010011110110111111

m = ''.join(str(c) for c in m.coefficients(sparse=False)[::1])
m
# 00000111110000011111000001111100000101

# reversed for consistency
c = m + r

c
# 00000111110000011111000001111100000101101110010011110110111111
```

My result is thus:

$$00000111110000011111000001111100000101101110010011110110111111$$

Q4 - BCH Decoder (Implementation)

Build a decoder for narrow - sense binary BCH codes, which uses

- Peterson's algorithm to obtain error-locator polynomial $\Lambda(x)$
- Chien search to find the roots of $\Lambda(x)$

i. Test your decoder on the binary $(15, 5)$ BCH code with generator polynomial

$$g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$$

Your decoder should correct all received words with errors of Hamming weight up to 3.

(ii). For the BCH code defined in **Q3**, suppose a codeword $c \in C$ is transmitted and the following words is received:

$$r = r_0 r_1 \ldots, r_{62}$$
$$= 010000000000011111000001011011111010101011000001111011010110111$$

Assume that this word can be uniquely decoded. Use the syndrome decoding to obtain the codeword.

In this implementation g is a list representing the coefficients of the poly. $g(x)$ in dec. order of degree. alpha is the size of $\mathbb{F}_\alpha$. The list returns a list of roots $g(x)$ over $\mathbb{F}_\alpha$.

```
def chien_search(g, alpha):
    """
    Performs Chien search to find the roots of g(x) over the field F_alpha.
    Returns a list of roots of g(x).
    """
    roots = []
    n = len(g) - 1
    for i in range(1, alpha**n):
        eval_poly = 0
        for j in range(n+1):
            eval_poly += g[j] * (i ** j)
        if eval_poly == 0:
            roots.append(i)
    return roots

def peterson_algorithm(S):
    """
    Given the syndrome polynomial S(x), computes the error locator polynomial sigma(x)
    using Peterson's algorithm. Returns a polynomial object representing sigma(x).
    """
    n = len(S) - 1  # length of received message
    sigma = [1]
    old_sigma = [1]
    gamma = [0]
    old_gamma = [0]

    for i in range(1, n+1):
        feedback = S[i]
        for j in range(1, i):
            feedback += S[i-j] * old_sigma[j]
        gamma.append(feedback)
        if feedback == 0:
            sigma.append(old_sigma)
        else:
            new_sigma = [0] * len(old_sigma)
            gamma_over_old_gamma = [GF(2)(feedback / old_gamma[-1]) * c for c in old_sigma]
            new_sigma += gamma_over_old_gamma
            sigma.append(new_sigma)

        old_gamma = gamma
        old_sigma = sigma[-1]

    return GF(2)(sigma[-1])


def err_gen(n, t):
  lst = [0] * n
  for i in random.sample(range(n), t):
      lst[i] = 1
  return lst



###########
# variables for code
n = 15 # 2^4 -1
q = 2  # GF(2)
d = 5  # min. dist

#
m = multiplicative_order(mod(q, n))
Fq = GF(q)
R.<x> = PolynomialRing(Fq)

F.<a> = Fq.extension(m)
b = a^((q^m-1)//m) # normally nth root of unity
```

I failed to understand the basics behind BCH in time to make this implementation work.