ELEC70069
Cryptography and Coding Theory Exercise

Wei Dai

November 25, 2024

EEE Department, Imperial College London

Contents

Part I.

Teaching Organisation

1. Teaching Organisation

1.1. Overview

- Lectures
  - Mondays 07/10 - 25/11/2024, 16:00-17:50, Room **407A&B**
  - Mondays 02/12 - 09/12/2024, 16:00-17:50, Room **508A&B**
- Office hours
  - Mondays 07/10 - 02/12/2024, 12:00-12:55, Room 503.
- Communications: Should you have any questions or inquiries:
  - Attend the designated office hours.
  - Post them on the Q&A channel.
  - Refrain from using my college email.
- Assessment
  4 coursework assignments: each counts 25%

1.2. Coursework Assignments

1.2.1. Schedule

Table 1.1.: Coursework Schedule

	Tasks	Due Date/Time
Coursework 1	Assignment	(Week 3) 14/10
	Due	(Week 4) 25/10 11:59
	Feedback	(Week 6) 04/11
Coursework 2	Assignment	(Week 5) 28/10
	Due	(Week 6) 08/11 11:59
	Feedback	(Week 8) 18/11
Coursework 3	Assignment	(Week 7) 11/11
	Due	(Week 8) 22/11 11:59
	Feedback	(Week 10) 02/12
Coursework 4	Assignment	(Week 9) 25/11
	Due	(Week 10) 06/12
	Feedback (Marks only)	(Week 11) 13/12

Coursework Overview:
Individual Data + Team Work + Individual Assessment

The procedure for the coursework assignments is outlined below.

- Each student will receive a unique data file provided by the GTAs.
- Students collaborate in groups while working on the coursework.
  The group information is input in an excel file.

- By the due time, each student must submit their own individual coursework, clearly specifying their contributions to the group effort. Note that submissions cannot be altered after the deadline.
- GTAs will assess individual submissions. Let $M_{i,0}$ denote the raw mark received by the student i. An adjusted mark, calculated using Equation (1.1), will then be provided to the student.
- Some coursework questions will be selected for in-depth discussion. Students chosen to lead these discussions will receive bonus marks to encourage active participation.

1.2.2. Marking

The marking formula for the first three coursework is given by

$$M_i = \min\left(\left(\sum_{k\in\mathcal{G}_l} M_{k,0}\right) * C_i * W_l, M_{i,0}\right), \tag{1.1}$$

where

- M_i is the adjusted mark received by the student i,
- $M_{k,0}$ is the raw mark obtained by the student k,
- $\sum_{k\in\mathcal{G}_l} M_{k,0}$ is the total raw mark obtained by the group $\mathcal{G}_l$,
- C_i is the contribution of the student i in the group,[1]
- W_l is the weighting coefficient for the group $\mathcal{G}_l$, depending on the size of the group,
- and we apply the min function to ensure the adjusted mark does not exceeds the raw mark. [2]

1: Note that
$$\sum_{k\in\mathcal{G}_l} C_k = 1.00 = 100\%.$$

2: It is typical that $M_i = M_{i,0} \times W_l$, i.e., the adjusted mark is simply the raw mark multiplied by the weighting coefficient.

The weighting coefficients for groups are detailed in Table 1.2. There are several considerations behind the design.

- We encourage both individual work and team work within the group.
- The default size of the group is 2 or 3.
- Students from the same group may share the same codes.
- **Cheating offences and plagiarism** are taken very seriously and are dealt with according to the College's Academic Misconduct Policy and Procedure.

Size of the group	1	2	3	4	5	≥ 6
Weighting coefficient W	1.00	0.97	0.94	0.86	0.70	0.50

Table 1.2.: The weighting coefficient W for the group is decided by the size of the group.

Remark 1.2.1 (Bonus marks) GTAs will select students to present their solutions to specific questions in-depth. Students can earn bonus marks by sharing and explaining their coursework solutions during the feedback/discussion sessions.
The bonus marks awarded will be determined by the GTAs and the module leader, based on the quality of the solutions and the clarity of the explanation. The awarded marks will be communicated to the participating student.

1.3. Software for Coursework 1-3

The coursework requires Julia programming. The coursework submission files include a Jupyter notebook file and a data file in JLD2 format. We recommend VSCode as the default editor.

The software that you need to install/have

- Jupyter Notebook
  Download and install the package management software Anaconda. By default, it will install Python and Jupyter Notebook to your system.
- Julia programming language
  - Download and install the current stable version of Julia.
  - Check whether Julia has been installed properly: Run Julia interactive session (a.k.a. REPL) by following the instructions.
- VSCode
  - Download and install VSCode.
  - We need to install some extensions for VSCode. See here for how to install VSCode extensions.
    * Anaconda should have automatically installed VSCode Extensions `Python` and `Pylance` for you. If not, install them.
    * Install extension `Julia` for Julia programming in VSCode. See here and here for more details.

1.3.1. Julia for Programming

- A collection of tutorials.
- Introduction to Scientific Programming and Machine Learning with Julia.
- A tutorial in pdf format.

Your Feedback Matters 2.

2.1. You Said, We Did

Over the past few years, we've listened carefully to your feedback and implemented substantial changes to enhance your learning experience.

2.1.1. Lecture Notes

- **Transition to Book Format**: We've transformed our lecture notes from slides into a comprehensive book format. This allows for a more cohesive and self-contained learning experience.
- **LaTeX Template**: We've adopted the LaTeX template `kaobook`, which gives you more space for note-taking.
- **Increased Examples**: We've incorporated more examples and illustrations to reinforce key concepts and make the material more engaging.

2.1.2. Assessments

- **Shift to Coursework**: We've transitioned from traditional paper-based exams to coursework assignments. This approach allows us to focus more on the practical application of theoretical knowledge.
- **Skill Development**: Coursework assignments are designed to help you hone your analytical and programming skills.
- **Peer Marking Removed**: We've removed peer marking to ensure that your performance is accurately reflected in your final grade.
- **Individual Recognition**: Our coursework marking process is designed to fairly assess and appreciate your unique contributions and efforts.

2.2. Recent Changes

Based on your feedback from the last academic year, we've implemented the following changes:

- **Reduced Coursework Load**: We've reduced the overall workload of coursework assignments by removing a learn-and-tell presentation coursework and spreading the questions across four assignments instead of three.
- **Coursework submission deadlines**. Coursework deadlines are now on Fridays, providing you with a more relaxed weekend and allowing the department to efficiently handle extension requests.
- **Enhanced Feedback**: We've increased the amount of information provided with your coursework marks. In addition to your individual scores, we'll share statistical information about the class marks.

- **In-Lecture Problem Discussion**: To foster deeper understanding and active engagement, we've incorporated in-lecture problem discussions and solving sessions.

2.3. Your Feedback

Please take the time to complete the **MEQ** questionnaires at the end of the term. Your continued feedback is invaluable in helping us improve the learning experience for this module.

2.4. Our Commitment to You

- **Accessibility and Support**: We're dedicated to providing you with the support you need to succeed. We offer regular **office hours** (Mondays, 12:00-12:55, Room 503) where you can discuss any questions or difficulties you may encounter. Additionally, we maintain a **Q&A channel** on Teams for more immediate assistance.
- **Timely Feedback**: We understand the importance of timely feedback. We aim to complete coursework marking within 10 working days of the submission deadline. You'll receive your marked assignments along with statistical information about the class performance.
- **Engaging Learning Experiences**: We're committed to creating an engaging and supportive learning environment. **In-lecture problem discussions and solving sessions** provide opportunities for you to actively participate, ask questions, and deepen your understanding of the material.

Important Notes 3.

3.1. Plagiarism

PLAGIARISM/COLLUSION DECLARATION

Coursework submitted for assessment must be the original work of you and your group. Assignments are subjected to regular checks for plagiarism and/or collusion. Plagiarism is the presentation of another person's thoughts or words (those outside your group) as if they were your own. Collusion involves obtaining help from someone outside your group to complete your work. In preparing your coursework, you should not seek help, or copy from any other person or source, including the Internet, without proper and explicit acknowledgement.

There is a procedure in place for you to declare individual contributions within your group for courswork. You must declare the contributions fairly and accurately.

You must not disclose your solutions or insights related to coursework with anyone else, including future students or the Internet.

By acknowledging the the statements above, you are declaring that both this and all subsequent pieces of coursework are, and will remain, the original work of you and your group.

- Submissions will not be accepted without the aforementioned declaration.
- Members of a group are deemed to have collective responsibility for the integrity for work submitted and are liable for any penalty imposed, proportionate to their contributions.

3.2. Notes for Coursework Submission

- Each registered student will get a data file. The data in the data file can be unique.
- Each registered student needs to submit the solutions related to their own data, no matter whether they are in groups or not.

3.2.1. Handling Solutions

In our coursework, we use the following convention.

- If the solution to a question is a unique integer, you need to assign an integer value to your solution variable.
- If the solution to a question does not exist, you need to create a 1-D array of length zero.
- If the solution to a question is not unique, you need to create a 1-D array, of which the length is the number of distinct solutions, and then specify the values in the **ascending** order.

Examples:

```
# The solution is a unique integer
x = 3;

# The solution is not well-defined (does not exist)
y = Array{Int64,1}(undef,0);

# The solution is not unique
z = Array{Int64,1}(undef,3);
z[1] = 3; z[2] = 4; z[3] = 5;
# Or equivalently
z = [3, 4, 5] ;
```

Part II.

Coursework 1

4. Divisibility

Remark 4.0.1 Note that you are not allowed to use any Julia packages except Base and LinearAlgebra.

1. (Congruence equations)
   Using the data in the data file, solve the following congruence equations in the form of $ax \equiv b \pmod{n}$. Find *all* solutions of x such that $0 < x < n$.
   a) Find all solutions of x such that $0 < x < n$. [2]
   b) Find all solutions of x such that $0 < x < n$. [2]
   c) Find all solutions of x such that $0 < x < n$. [2]
   d) Find all solutions of x such that $0 < x < n$. [2]
   e) Find all solutions of x such that $0 < x < n$. [2]
   f) Find all solutions of x such that $0 < x < n$. [2]
2. a) The **Fibonacci numbers** $1, 1, 2, 3, 5, 8, \cdots$ are defined using the relationship $F_1 = 1,\ F_2 = 1,\ F_{n+1} = F_n + F_{n-1}$.
   Find the closed form for $\gcd(F_n, F_{n-1})$ for all $n \geq 1$. [1]
   Prove your result. [3]
   b) Let $a = 111\cdots11$ be formed with F_n many repeated 1's.
   Let $b = 111\cdots11$ be formed with F_{n-1} many repeated 1's.
   Find $\gcd(a, b)$. [1]
   Prove your answer. [3]
3. (Modular Exponentiation) Using the data in the data file, compute the following exponentiation in modular arithmetic.
   a) Compute $b^p \pmod{n}$. [2]
   b) Compute $b^p \pmod{n}$. [2]
   c) Compute $b^p \pmod{n}$. [2]
   d) Compute $b^p \pmod{n}$. [2]
   e) Compute $b^p \pmod{n}$. [2]
4. (Matrix inverse)
   a) Based on the data in the data file, find all values of $b \pmod{26}$ such that
   $$\begin{bmatrix} 1 & a \\ b & 3 \end{bmatrix} \pmod{26}$$
   is invertible. [3]
   b) Find all numbers $3 \leq n \leq 50$ for which
   $$\begin{bmatrix} 1 & a \\ b & 3 \end{bmatrix} \pmod{n}$$
   is invertible. [3]
5. (Matrix inverse) In this question, we will use cofactors to compute the inverse of a matrix. You can refer to this webpage for more details on how to use a cofactor matrix to compute the inverse.
   Write a function to compute the inverse of a matrix in modular arithmetic, i.e., $\boldsymbol{A}^{-1} \pmod{n}$, using its cofactor matrix. The entries

in your $\boldsymbol{A}^{-1}$ must be integers between 0 and $n-1$. You may use the functions `LinearAlgebra.det` and `Base.gcdx`.
Using the provided data file and your programmed function, complete the following tasks:

a) Compute $\boldsymbol{A}^{-1} \pmod{n}$. [3]
b) Compute $\boldsymbol{A}^{-1} \pmod{n}$. [3]
c) Compute $\boldsymbol{A}^{-1} \pmod{n}$. [3]

5. Classical Cryptosystems

Decrypting a ciphertext usually involves trial and error. Often, this process necessitates writing a specific program or function.

The Julia package `FreqTables.jl` can be helpful.

1. (Shift cipher) The ciphertext in the data file was encrypted by a shift cipher. Decrypt it. [2]
2. (Affine cipher) The ciphertext in the data file was encrypted by an affine cipher. Decrypt it. [4]
3. (Vigenère cipher) The ciphertext in the data file was encrypted by a Vigenère cipher, where the key is of length at most 8.
   a) Write a function to calculate the shifted coincidence frequency for $1 \leq l \leq 8$. [2]
   Use this to estimate the length of the key. [1]
   b) Find the key and hence the plaintext. [6]
4. (Block cipher) We consider a block cipher of the form,

$$\boldsymbol{c} = \boldsymbol{x}\boldsymbol{A} + \boldsymbol{b} \mod 26,$$

   where the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ are given in the data file.
   a) Write a function to encrypt the plaintext. Find the corresponding ciphertext of the plaintext in the data file. Note that zero-padding is required in this question if the length of plaintext is not a multiple of the block length. [3]
   b) Find the letter frequency of the ciphertext. Observe the difference between the letter frequency of the ciphertext and that of the English language. [2]

Part III.

Coursework 2

Number Theory 6.

You are allowed to use functions in the Julia packages `Primes.jl` and `Combinatorics.jl`.

1. Suppose $x \equiv a_1 \bmod n_1$, $x \equiv a_2 \bmod n_2$, and $x \equiv a_3 \bmod n_3$. Find the value of $0 < y < n_1 n_2 n_3$ such that $x \equiv y \bmod n_1 n_2 n_3$. [3]
2. Without the help of the computer, divide 2^m by 101. What is the remainder? Briefly show how you get your solution. [2]
3. (Primitive root) Note that the prime number p can be very large.

   a) Write a function `is_primitive_root` which can check whether a number a is a primitive root of the prime number p.
   Use your function to check whether the given a is a primitive root of the corresponding given p in the data file. [3]
   b) Write a function `next_primitive_root` of which the input is a number a and a prime number p, and the output is a number x such that $x \geq a$ is the smallest primitive root of p.
   Use your function to find the next primitive root using the given a and p in the data file. [3]
   c) Find all primitive roots of the prime number p. [3]

Modern Cryptography 7.

Starting from this chapter, we will no longer use mod-26 arithmetic directly for encryption. Instead, each English letter is mapped to an integer as follows:

$$a \mapsto 1, \quad b \mapsto 2, \quad \cdots$$

To efficiently use the digits available, English letters are grouped to form numeric plaintext. For example:

$$abc \mapsto 010203 = 10203.$$

After this transformation, both the ciphertext and decrypted text will be represented as numerical integers, with no need to convert them back to English text.

When submitting answers, convert your integers to strings using `string(my_int)`.

The final question involves polynomials with integer coefficients. You may use the `Polynomials.jl` package. To obtain the coefficients of a polynomial, use the `coeffs` function from this package. Convert the integer coefficients into a string vector by using `string.(my_int_vector)`. If the answer requires multiple polynomials, format your submission as a vector of vectors of strings, with each inner vector representing the coefficients of one polynomial.

1.
   a) Use the two prime numbers p and q (each 20 digits) given in the data file. Set $n = pq$. Choose e as the smallest integer at least 305407503811 that is co-prime to $(p-1)(q-1)$. [1]
   b) Use (n, e) to encrypt each of the following plaintexts
      i. `bat`
      ii. `cat`
      iii. `hat`

      Note the difference among the ciphertexts that correspond to the three plaintexts which differ in only one letter. [3]
2. We consider RSA cryptosystem. Load the encrypt (public) keys (n, e) and the ciphertext c given in the data file. Note that $n = pq$ where p and q are consecutive prime numbers. Find the corresponding plaintext (as an integer). [4]
   Remark: The reason that you can do decryption is that the choice of p and q are bad.
3. (Diffie-Hellman Key Exchange, ElGamal Cryptosystem, ElGamal Digital Signature)
   Load the prime number p (20 digits) in the data file.
   a) Let g be the smallest primitive root of p that is at least 305407503811. [2]
   b) Suppose that Alice chooses a secret key `apple`, Bob's secret key is `banana`, and Charlie's secret key is `cherry`. Compute the public keys (in integers) of Alice, Bob, and Charlie respectively. [3]

c) We consider Diffie-Hellman key exchange. Compute the secret integers that Alice and Bob agree on, Bob and Charlie agree on, and Alice and Charlie agree on. [3]

d) We consider ElGamal public key cryptography. Suppose that Bob want to send Alice a message `nice`. The random number k that Bob generates is in the data file. Compute the corresponding encrypted message (in integers). [2]

e) Suppose that Alice wants to sign her message `seven` using ElGamal digital signature scheme. She chooses the smallest k that is at least 5000 and satisfies the requirements. Compute the signed message (as integers). [2]

f) Suppose that Bob receives a message given in the data file, which is supposed to be sent from Alice. Check the digital signature to verify whether the message is truly from Alice or not. [2]

g) Suppose that Bob receives a message given in the data file, which is supposed to be sent from Charlie. Check the digital signature to verify whether the message is truly from Charlie or not. [2]

4. a) Using Lagrangian interpolation construct a polynomial $f(x)$ such that

$$f(x_k) \equiv y_k \pmod{p}, \quad k = 1, 2, \cdots, t.$$

Save your polynomials $y_k l_k(x)$, $k = 1, 2, \cdots, t$, and $f(x)$ to your data file. It is required that the coefficients of the polynomial must be integers between 0 and $p - 1$. [3]

b) We consider a Shamir $(4, 7)$ secret sharing scheme. The prime number p and the 7 shares are given in the data file.
One of these shares was incorrectly received. Determine which one is incorrect, and find the shared secret. [3]

You are required to use the Julia package `Polynomials.jl` to represent your polynomials.

Part IV.

Coursework 3

8. Finite Field

Starting from this chapter, you are permitted to use the Julia packages `Nemo.jl` and `AbstractAlgebra.jl`.

A polynomial can be represented by its vector of coefficients. A polynomial of degree d, given by $p(x) = a_0 + a_1 x + \cdots + a_d x^d$, is represented by a vector of length-$d+1$: $[a_0, a_1, \cdots, a_k]^\mathsf{T}$. For example, the polynomial $1 + x^2 + x^3$ is represented by $[1, 0, 1, 1]^\mathsf{T}$.

In some exercises, you will be asked to sort multiple polynomials lexicographically. To do this, treat the polynomial's coefficients $a_k \in \mathbb{F}_q$ as the digits of a base-q number n, i.e., $n = \sum_{k=0}^{d} a_k q^k$. Each polynomial thus corresponds to a number n in base q. The polynomials should then be sorted by the corresponding number n in the *increasing* order. Finally, save the vector representations $[a_0, a_1, \cdots, a_d]^\mathsf{T} \in \{0, 1, \cdots, q-1\}^{d+1}$ of the sorted polynomials to the data file as your answer.

Find the following example for `Nemo.jl`

```
using Nemo;

F3 = GF(3);

# Create a polynomial ring,
#   and then a polynomial in that ring
polyRingF3,u = polynomial_ring( F3,"u" );
mypoly = polyRingF3( [2,2,1] );

# Mypoly is actually irreducible.
# We use it to create a finite field.
# Then we create an element in that field.
F9,x = finite_field( mypoly,"x" );
myelem = F9([0,1]);

# Now we extract the coefficients of myelem
polyZZ,w = polynomial_ring( ZZ,"w" );
mypolyZZ = lift( polyZZ,myelem );
myelem_coeff = Int.(
    collect(coefficients(mypolyZZ)) );
```

1. (Polynomials)
   a) Write a function that generates all possible coefficient vectors (as integer vectors) of polynomials in $\mathbb{F}_p[u]$ with degree at most d. This function should take p and d as input parameters and output the coefficient vectors in lexicographical order. The function `base.digits` (with no zero padding) can be useful.
   Test your function with $p = 2$ and $d = 5$. [2]
   b) Write a function that generates all possible coefficient vectors (as integer vectors) of polynomials in $\mathbb{F}_p[u]$ with degree exactly d. This function should take p and d as input parameters and output the coefficient vectors in lexicographical order.
   Test your function with $p = 3$ and $d = 4$. [2]

2. (Irreducible polynomials) Read from the data file the coefficients of your polynomial $p(x)$. Decide whether the polynomial is irreducible or not.
   a) Let $p(x) \in \mathbb{F}_2[x]$. [2]
   b) Let $p(x) \in \mathbb{F}_3[x]$. [2]
   c) Let $p(x) \in \mathbb{F}_5[x]$. [2]

3. (Congruence Equations for Polynomials) In this question, we consider linear equations of the form

$$a(x)p(x) \equiv b(x) \mod m(x),$$

   where $a(x), b(x), m(x), p(x) \in \mathbb{F}_3[x]$.
   The polynomials $a(x), b(x), m(x)$ are provided in the data file. Using theses given polynomials, determine the solution(s) for $p(x)$ such that $0 \leq \deg(p(x)) < \deg(m(x))$.
   If there are multiple solutions, list all of them in lexicographical order.
   a) Solve for $p(x)$. [2]
   b) Solve for $p(x)$. [2]
   c) Solve for $p(x)$. [4]

4. (Primitive Elements for a Polynomial Field)
   a) Consider the finite field $\mathbb{F}_2[u]/f_1(u)$, where $f_1(u) \in \mathbb{F}_2[u]$ is irreducible and given in the data file. Find all the primitive element in this finite field. Sort them according to the lexicographical order. [3]
   Inspect your result to see whether the polynomial u is a primitive element or not.
   b) Consider the finite field $\mathbb{F}_2[v]/f_2(v)$, where $f_2(v) \in \mathbb{F}_2[v]$ is irreducible and given in the data file. Find all the primitive element in this finite field. Sort them according to the lexicographical order. [3]
   Inspect your result to see whether the polynomial v is a primitive element or not.

5. (Mapping) Consider the two fields $\mathbb{F}_2[u]/f_1(u)$ and $\mathbb{F}_2[v]/f_2(v)$ where $f_1(u)$ and $f_2(v)$ are two irreducible polynomials of the same degree. The two irreducible polynomials $f_1(u)$ and $f_2(v)$ are given in the data file.
   Consider the map

$$\begin{aligned} \phi : \mathbb{F}_2[u]/f_1(u) &\to \mathbb{F}_2[v]/f_2(v) \\ u &\mapsto g(v). \end{aligned}$$

   In the following sub-questions, we will have distinct $g(v)$.
   The data file also gives $p_1(u), p_2(u), p_3(u) \in \mathbb{F}_2[u]/f_1(u)$. We define

$$\begin{aligned} q_1(v) &= \phi(p_1(u)), \\ q_2(v) &= \phi(p_2(u)), \\ q_3(v) &= \phi(p_3(u)), \\ q_4(v) &= \phi((p_1(u) + p_2(u))\, p_3(u)), \\ q_5(v) &= \left(\phi(p_1(x)) + \phi(p_2(x))\right) \phi(p_3(u)). \end{aligned}$$

   a) Based on the $g_1(v)$ given in the data file, compute $q_n(v)$,

$1 \leq n \leq 5$. [5]
Inspect your result to see whether $q_4 = q_5$.

b) Based on the $g_2(v)$ given in the data file, compute $q_n(v)$, $1 \leq n \leq 5$. [5]
Inspect your result to see whether $q_4 = q_5$.

9. Linear Codes

Consider the binary Hamming code $\mathscr{C}[7,4,3] \subset \mathbb{F}_2^n$ of which the parity check matrix is given by

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{9.1}$$

Note that the columns of $\boldsymbol{H}$ are arranged in lexicographical order. Specifically, $[h_{1,i}, h_{2,i}, h_{3,i}]^\mathsf{T}$ is the binary representation of the integer $h_{1,i} + 2 \cdot h_{2,i} + 4 \cdot h_{3,i}$, which proceeds the integer represented by $[h_{1,i+1}, h_{2,i+1}, h_{3,i+1}]^\mathsf{T}$.

The systematic structure of $\boldsymbol{H}$ in (9.1) is not very straightforward. However, the sub-matrix $\boldsymbol{H}_{:,[1,2,4]}$ is an identity matrix. Apply the same idea mentioned in Example 9.1.6 in the lecture notes. The corresponding generator matrix $\boldsymbol{G}$ can be found with negligible computational cost.

1. (Binary Hamming Code) Consider the binary Hamming code $\mathscr{H}(r, p = 2)$, where r is given in your data file. It is clear that the corresponding parity-check matrix $\boldsymbol{H}$ of the binary Hamming code $\mathscr{H}(r, 2)$ has r many rows. Assume that the columns of $\boldsymbol{H}$ are arranged in lexicographical order (see (9.1) for an example).
   a) Find the parity-check matrix $\boldsymbol{H}$. [2]
   b) Find the corresponding generator matrix $\boldsymbol{G}$. [2]
   c) A codeword of this binary Hamming code is transmitted through a Binary Symmetric Channel (BSC) with crossover probability $p_{e,1}$ (given in the data file). A nearest codeword (in Hamming distance) decoder $\mathscr{D}$ is applied to the received word. Compute the decoding error probability $P_{\text{err},1}$ of $\mathscr{D}$. [2]
   d) Suppose that the received word is the $\boldsymbol{y}_1$ given in the data file. Suppose that the channel introduces one symbol error. Perform the decoding in the following steps.
      i. Find the syndrome vector $\boldsymbol{s}_1$. [1]
      ii. Find the transmitted codeword $\boldsymbol{c}_1$. [1]
      iii. Find the message $\boldsymbol{m}_1$ that generates the codeword $\boldsymbol{c}_1$. [1]
   e) A codeword of this binary Hamming code is transmitted through a Binary Erasure Channel (BEC) with crossover probability $p_{e,2}$ (given in the data file). A nearest codeword (in Hamming distance) decoder $\mathscr{D}$ is applied to the received word. Compute the decoding error probability $P_{\text{err},2}$ of $\mathscr{D}$. [2]
   f) Suppose that the received word is the $\boldsymbol{y}_2$ given in the data file, where the symbols ? means that the corresponding symbol gets erased by the channel. Perform the decoding in the following steps.
      i. Find the syndrome vector $\boldsymbol{s}_2$ by setting the symbols ? to zero. [1]
      ii. Find the transmitted codeword $\boldsymbol{c}_2$. [1]
      iii. Find the message $\boldsymbol{m}_2$ that generates the codeword $\boldsymbol{c}_2$. [1]
2. (q-ary Hamming codes)

A q-ary linear code, whose parity-check matrix $\boldsymbol{H}$ has the property that its columns are made up of precisely one vector from each vector subspaces of dimension 1 of $\mathbb{F}_q^r$, is called a q-ary Hamming code, often denoted as $\mathscr{H}(r, q)$.
In the data file, you are given r, and a prime number $q = p$.

a) Construct the corresponding $\boldsymbol{H}$.
   Note that the columns of $\boldsymbol{H}$ should be arranged according to the lexicographical order.
   There might be multiple nonzero vectors span the same vector subspace of dimension 1. The columns of $\boldsymbol{H}$ should be the column vectors that are lexicographically the first. [3]
b) Find the corresponding generator matrix $\boldsymbol{G}$. [2]
c) A codeword of this q-ary Hamming code is transmitted through a q-ary Symmetric Channel with crossover probability $p_{e,1}$ (given in the data file). A nearest codeword (in Hamming distance) decoder $\mathscr{D}$ is applied to the received word. Compute the decoding error probability $P_{\text{err},1}$ of $\mathscr{D}$. [2]
d) Suppose that the received word is the $\boldsymbol{y}_1$ given in the data file. Suppose that the channel introduces one symbol error. Perform the decoding in the following steps.
   i. Find the syndrome vector $\boldsymbol{s}_1$. [1]
   ii. Find the transmitted codeword $\boldsymbol{c}_1$. [1]
   iii. Find the message $\boldsymbol{m}_1$ that generates the codeword $\boldsymbol{c}_1$. [1]
e) A codeword of this q-ary Hamming code is transmitted through a q-ary Erasure Channel with crossover probability $p_{e,2}$ (given in the data file). A nearest codeword (in Hamming distance) decoder $\mathscr{D}$ is applied to the received word. Compute the decoding error probability $P_{\text{err},2}$ of $\mathscr{D}$. [2]
f) Suppose that the received word is the $\boldsymbol{y}_2$ given in the data file, where the symbols ? means that the corresponding symbol gets erased by the channel. Perform the decoding in the following steps.
   i. Find the syndrome vector $\boldsymbol{s}_2$ by setting the symbols ? to zero. [1]
   ii. Find the transmitted codeword $\boldsymbol{c}_2$. [1]
   iii. Find the message $\boldsymbol{m}_2$ that generates the codeword $\boldsymbol{c}_2$. [1]

Part V.

Coursework 4

10. Reed-Solomon Codes

10.1. RS Codes on $\mathbb{F}_p$

Consider standard Reed-Solomon codes

- $\mathscr{C}_1[n, k_1, d_1] \subseteq \mathbb{F}_p^n$,
- $\mathscr{C}_2[n, k_2, d_2] \subseteq \mathbb{F}_p^n$

constructed using primitive element a, where p, a, d_1, and d_2 are given in the data file, and $n = p - 1$.

Define the following codes:

- $\mathscr{C}_3 = \mathscr{C}_1 \bigcup \mathscr{C}_2$.
- $\mathscr{C}_4 = \mathscr{C}_1 \bigcap \mathscr{C}_2$.
- $\mathscr{C}_5 = \mathscr{C}_1 + \mathscr{C}_2 = \{\boldsymbol{c}_1 + \boldsymbol{c}_2 : \boldsymbol{c}_1 \in \mathscr{C}_1 \text{ and } \boldsymbol{c}_2 \in \mathscr{C}_2\}$.
- $\mathscr{C}_6 = \{[\boldsymbol{c}_1 \mid \boldsymbol{c}_2] : \boldsymbol{c}_1 \in \mathscr{C}_1 \text{ and } \boldsymbol{c}_2 \in \mathscr{C}_2\}$ where the symbol $[\cdot \mid \cdot]$ stands for concatenation of words.
- $\mathscr{C}_7$ is constructed by appending each codeword $\boldsymbol{c} \in \mathscr{C}_1$ with a symbol equal to the summation of all symbols in $\boldsymbol{c}$, i.e.,

$$\mathscr{C}_7 = \left\{ \left[\boldsymbol{c} \mid \sum_{k=1}^{p-1} c_k \right] : \boldsymbol{c}_1 \in \mathscr{C}_1 \right\}.$$

- $\mathscr{C}_8 = \{\boldsymbol{c} \otimes \boldsymbol{d} : \boldsymbol{c} \in \mathscr{C}_1 \text{ and } \boldsymbol{d} \in \mathscr{C}_2\}$, where $\boldsymbol{c} \otimes \boldsymbol{d} = [c_1\boldsymbol{d}, \cdots, c_n\boldsymbol{d}]$ denotes Kronecker product.

1. Find the generator matrix $\boldsymbol{G}_1$ and parity check matrix $\boldsymbol{H}_1$ of $\mathscr{C}_1$ according to Theorem 11.2.1 in the lecture notes. [2]
2. Find the generator matrix $\boldsymbol{G}_2$ and parity check matrix $\boldsymbol{H}_2$ of $\mathscr{C}_2$ according to Theorem 11.2.1 in the lecture notes. [2]
3. Find the generator matrix $\boldsymbol{G}_3$ and parity check matrix $\boldsymbol{H}_3$ of $\mathscr{C}_3$. [2]
4. Find the generator matrix $\boldsymbol{G}_4$ and parity check matrix $\boldsymbol{H}_4$ of $\mathscr{C}_4$. [2]
5. Find the generator matrix $\boldsymbol{G}_5$ and parity check matrix $\boldsymbol{H}_5$ of $\mathscr{C}_5$. [2]
6. Find the generator matrix $\boldsymbol{G}_6$ and parity check matrix $\boldsymbol{H}_6$ of $\mathscr{C}_6$. [2]
7. Find the generator matrix $\boldsymbol{G}_7$ [2]
   and parity check matrix $\boldsymbol{H}_7$ of $\mathscr{C}_7$. [2]
8. Find the generator matrix $\boldsymbol{G}_8$ [2]
   and parity check matrix $\boldsymbol{H}_8$ of $\mathscr{C}_7$. [4]
9. Suppose that $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_1$ is transmitted and the received word $\boldsymbol{y} \in (\boldsymbol{F}_p \cup \{?\})^n$ is given in the data file.
   Find the transmitted message $\boldsymbol{m}$. [4]

Remark 10.1.1 Note that the elements in your $\boldsymbol{G}$ and $\boldsymbol{H}$ matrices must be integer values between 0 and $p - 1$.

10.2. RS Codes on $\mathbb{F}_q$

The data file gives an irreducible polynomial to generate $\mathbb{F}_{32}$, and a primitive element α of $\mathbb{F}_{32}$.

1. Consider a Reed-Solomon code $\mathscr{C}[31, 21, 11]$ with the generator matrix $\boldsymbol{G}$ and parity check matrix $\boldsymbol{H}$ given in Theorem 11.2.1. Write $\boldsymbol{G}$ and $\boldsymbol{H}$ explicitly and save then into the data file. [2]
2. Let the $\boldsymbol{y}$ given in the data file be the received word. Compute the syndrome vector $\boldsymbol{s}$. [2]
   The syndrome vector defines the corresponding syndrome polynomial $S(x)$, which is used to find the error locator and evaluator polynomials in the steps below.
3. Find the error locator polynomial, denoted by $L(x)$, and the error evaluator polynomial, denoted by $E(x)$. [8]
4. Use the error locator polynomial $L(x)$ to find the error locations. [2]
   Use both the error locator polynomial $L(x)$ and the error evaluator polynomial $E(x)$ to find the error vector $\boldsymbol{e}$. [4]
   Find the transmitted codeword $\hat{\boldsymbol{x}}$. [1]
5. Find out the message $\boldsymbol{m} = [m_1, m_2, \cdots, m_{21}]$. [3]

Remark 10.2.1
1. The elements in $\mathbb{F}_{32}$ are polynomials and represented by its integer coefficient vector.
2. The entries in $\boldsymbol{G}$, $\boldsymbol{H}$, $\boldsymbol{y}$, and $\boldsymbol{m}$ are elements in $\mathbb{F}_{32}$.
3. The coefficients of $S(x)$, $L(x)$, and $E(x)$ are elements in $\mathbb{F}_{32}$.