1 Things to Revise

1.1 Groups and Subgroups

- Add, subtract and invert modulo an integer n.
- The meaning of "order" of a group or of an element, and how to compute it.
- Permuatations - how to compose, invert and convert to/from cycle notation.
- What a subgroup is, and how to check if a subset is one.
- What a generator is, how to find one (or check if it exists) and what the $\langle g \rangle$ notation means where g is an element in some group.
- Simple proofs in group theory, as done in the lectures and in the script (excluding "diamond" sections).

Proofs of the kind discussed in the lectures may appear in the exam. Examples of this include why $x + a = x + b$ implies $a = b$, or showing that something is an equivalence relation.

1.2 Rings and polynomials

- Multiply and (where possible) divide modulo an integer n.
- The meaning of "unit" and "zero-divisor" and how to classify elements in a ring as one of four cases (zero, unit, zero-divisor, neither).
- Compute Euler's totient (ϕ) function.
- Perform exponent arithmetic, e.g. ($a^b \mod n$).
- Divide polynomials with remainder modulo an integer n.
- Define and compute the characteristic of a ring.

Simple proofs may appear in the exam, such as when $ax = ay$ implies $x = y$ and why (it's not the same as for $a + x = a + y$).

1.3 Finite fields

- Perform arithmetic modulo a polynomial, in particular reducing higher-degree polynomials.
- Understand the notation $GF(p^n)$.
- Check if a polynomial of degree 2 or 3 is irreducible; find such polynomials.

- Add, subtract, multiply and divide (except by 0) in $GF(p^n)$, including writing down the operation tables and finding the explicit multiplication formula,
- Run Euclid's algorithm (extended version), compute the gcd of integers or polynomials and solve equations of the form $ax + by = c$ for x, y where a, b, c are all integers or polynomials.
- Compute the Frobenius map and the other automorphisms in a finite field; know how many automorphisms there are in $GF(p^n)$.
- State for which integers n there can be a field with n elements.

1.4 Linear Algebra

- Do arithmetic in vector spaces over finite fields.
- Check if vectors are linearly independent.
- Evaluate matrix-vector products and find the inverse of a given matrix.
- Solve linear equation systems over finite fields.
- Write down Vandermonde matrices for given points and dimensions.
- Interpolate and evaluate polynomials in a finite field.
- Interpret polynomials as vectors and finite fields $GF(p^n)$ as vector spaces over $GF(p)$.
- Convert a linear function on a finite field (such as the Frobenius map) between polynomial and matrix notation.

1.5 Coding Theory

- Understand the difference between a code and an encoding.
- Understand the minimum distance of a code and how it relates to error detection/correction.
- Compute the Hamming distance of two words and of a block code.
- Use the Hamming distance of a linear code to determine how many errors a code can correct or detect.
- Work with repetition codes and checksum codes.
- Write a generator matrix for a checksum code given as a function.
- Encode with a generator matrix.
- Convert a generator matrix to systematic form.

- Compute a parity-check matrix from a systematic generator matrix and use it to check if a word is in a code or not.
- Encode, decode and check for errors with the (7, 4) Hamming code.
- Encode with Reed-Solomon codes, including creating the generator matrix.

The ISBN, UPC (barcode) and Luhn codes will not appear in the exam, nor will the topic on detecting transpositions (as opposed to single symbol errors).

The Singleton bound, Hamming bound and Gilbert-Varshamov bound will not appear in the exam.

Generalised Hamming codes will not feature in the exam.

1.6 Information Theory

- Encode and decode with variable-length codes.
- Draw a decoding tree from a table of an encoding function.
- Identify if a code is prefix-free.
- Compute logarithms to base 2 and the functions $h(x)$ and $h2(x)$.
- Compute the information content of an event and the entropy of a random variable.
- Compute conditional entropies and mutual information between random variables.
- Construct a Huffman code for a given probability distribution.
- Convert descriptions of channels between diagram and conditional probability table forms.
- Compute the output distribution and error probability of a channel given an input distribution and the channel characteristics.
- Compute the joint and conditional distributions for simple channels.
- Possibly in "Part 2": compute the capacity of a channel, where this is possible with the material in this unit.
- Be able to define the binary symmetric and erasure channels (with parameter p).

2 Groups

2.1 Group laws

A group $\mathbb{G} = (G, +)$ consists of a set G and an operation $+ : G \times G \to G$ that has the following properties:

- **associative**: For any elements g, h, k of G we have $(g + h) + k = g + (h + k)$
- **neutral element**: There is an element e such that for all elements of $g \in G$, $e + g = g + e = g$.
- **inverses**: For any element $g \in G$ there is an element h of G such that $g + h = h + g = e$

2.2 Group orders

The order of a group G is the magnitude of G. The order of and element g in group G is the size of the subset made by the group $\langle g \rangle$. This means $\{g, g + g, g + g + g\}$ and so on. (The sign might be a different sign depending on the definition of the group)

3 Rings

3.1 Ring Laws

A ring is a structure $(R, +, \times)$ where R is a set and $+, \times$ are two operations $R \times R \to R$ is a ring if the following hold.

- **additive group**: The structure $(R, +)$ is an *Abelian group*. We call its neutral element the zero of R and write it with the symbol 0 (or sometimes 0_R).
- **multiplication**: The structure $(R, \times)$ is associative and has a neutral element which we call the **one** of the ring and we write it as 1 or 1_R.
- **distributive law**: For any elements a, b, c in R we have $(a+b) \times c = a \times c + b \times c$ and $c \times (a+b) = c \times a + c \times b$

An element of a ring can be classified as follows:

- The neutral element of a addition is the **zero** of the ring and written with the symbol 0.
- An element a in the ring that has a multiplicative inverse b ($a \times b = b \times a = 1$) is called a **unit**.
- An element $a \neq 0$ for which there is some $b \neq 0$ such that $a \times b = b \times a = 0$ is called a **zero divisor**.
- Or, it can be none of the above.

3.2 Euler's totient function

The group $\mathbb{Z}_n^\times$ is a subset of $\mathbb{Z}_n$ without all elements that are zero divisors.

To calculate how many elements are in the group, we use Euler's totient function $\Phi(n) = |\mathbb{Z}_n^\times$ for $n > 0 \in \mathbb{N}$. We then end up with:

$$\Phi(n) = \prod_{i=1}^{k} p_i^{a_i - 1}(p_i - 1)$$

Which is just the product of all of the primes that make up the number (where a is the power of said prime number). If n is a prime p, then $\Phi(p) = p - 1$. Also, $\Phi(a \times b) = \Phi(a) \times \Phi(b)$. This is why we can end up with the above result.

If we have a modulo of some big number, we can do:

$$a^k = (a \mod n)^{(k \mod \Phi(n))} \quad (\mod n)$$

4 Fields

4.1 Characteristics of rings and fields

Characteristic of a ring is how many times we have to add 1_R to itself to get 0_R. If it can never happen, then the characteristic is 0. The **characteristic of a field** is either 0 or a prime number.

4.2 Euclid's algorithm

Inverting x mod y is the same as writing $ya + xb = 1$. For example, if $x = 17$ and $y = 256$, we would do the following ($256a + 17b = 1$):

q	r	a	b
0	256	1	0
0	17	0	1
15	1	1	-15
17	0	-17	256

For any row, we let q', r', a', b' be the values from the last row, and double dash that for the second to last row. Then, just:

- Divide r'' by r' with remainder. Put the quotient as q and remainder as r.
- Then, $a := a'' - q \times a'$ and $b := b'' - q \times b'$.

5 Finite Fields

5.1 The Frobenius Automorphism

In a finite field of characteristic *p*, the function $\Phi(a) = a^p$ is called the **Frobenius automorphism**. It has the properties $\Phi(0) = 0, \Phi(1) = 1, \Phi(a \times b) = \Phi(a) \times \Phi(b)$ and $\Phi(a + b) = \Phi(a) + \Phi(b)$.

5.2 The Frobenius Map

In a finite field, the $\Phi(a) = a^p$ must be bijective.

6 Linear Algebra

6.1 Vector Spaces

A vector space over $\mathbb{F}$ is a structure $(V, +, \times)$ where $+ : V \times V \to V$ and $x : \mathbb{F} \times V \to V$ satisfying these laws:

1. $(V, +)$ is an Abelian group.
2. Field and scalar multiplication associate: for any field elements f, g and any vector $\vec{a}$, we have $(fg) \times \vec{a} = f \times (g \times \vec{a})$. Here, fg is multiplication in $\mathbb{F}$.
3. Field multiplication distributes over vector addition: for any vectors $\vec{a}, \vec{b} \in V$ and any field elements $f, g \in \mathbb{F}$ we have $f \times (\vec{a} + \vec{b}) = f \times \vec{a} + f \times \vec{b}$ and $(f +_{\mathbb{F}} g) \times \vec{a} = f \times \vec{a} + g \times \vec{a}$. Field addition is marked $+_{\mathbb{F}}$

6.2 Inverting a Matrix

To invert a matrix, attach the identity matrix, and then swap rows until you end up with the identity matrix on the left.

$$\left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 & 1 & 0 \\ 0 & 2 & 1 & 0 & 0 & 1 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & 3 & 0 \\ 0 & 1 & 0 & 6 & 4 & 0 \\ 0 & 0 & 1 & 2 & 6 & 1 \end{array}\right)$$

6.3 Vandermonde Matrices

To form a Vandermonde matrix, we have some set of numbers (a, b, c) and then (for a 2×2 matrix):

$$\begin{pmatrix} a^0 & b^0 & c^0 \\ a^1 & b^1 & c^1 \\ a^2 & b^2 & c^2 \end{pmatrix}$$

Then, to interpolate a polynomial, we invert the matrix and multiply the coordinates by the matrix to return the polynomial in the form (a, b, c) where $a + bX + cX^2$

6.4 Lagrange Interpolation

Lagrange interpolation has the following formula (where the polynomial $p(X)$ is defined as):

$$p(X) = \sum_{i=0}^{n} y_i \times L_i(X) \text{ where } L_{i,n}(X) = \prod_{k \neq i} \frac{X - x_k}{X_i - x_k}$$

7 Coding theory

7.1 Hamilton Codes

The $(7, 4)$ Hamming code encodes 4 bits of data as a 7-bit codeword (or 8 bits for the SECDED) version. This idea can be expanded to any power of 2:

Def. For any $m \geq 1$ the $(2^m - 1, 2^m - m - 1)$ generalised Hamming code encodes up to $2^m - m - 1$ data bits with m parity bits for a total of $2^m - 1$ bits according to the following rules:

- The minimal distance is $d = 3$ (4 for SECDED) for any value m.
- The i-th bit is a parity bit if i is a power of 2, otherwise a data bit.
- The parity bits are set such that the sum of all bits whose index i has a 1 at the j-th position when written out in binary equals zero.
- For SECDED, an additional initial bit 0 is set such that the sum of all bits in the codeword is zero.

Here's an example:

bit	000	001	010	011	100	101	110	111
type	(parity)	parity	parity	data	parity	data	data	data

7.2 Maximum Distance Separable

MDS (maximum distance separable) codes are codes whose minimum distance is as high as the *Singleton bound* ($d \leq n - k + 1$) for a linear code that turns k-bit messages into n-bit codewords.

7.3 Reed-Solomon Code

We pick a finite field $\mathbb{F}$ of size at least n. Call this size q. Pick n distinct points $x_1, \ldots, x_n$ in $\mathbb{F}$. To encode a message $m = m_0 m_1 \ldots m_{k-1}$, view it as a polynomial

$$M = m_0 + m_1 X + m_2 X^2 + \ldots m_{k-1} X^{k-1}$$

and compute the encoding

$$E(m) = (M(x_1), M(x_2), \ldots, M(x_n))$$

The generator matrix of the Reed-Solomon code is a Vandermonde matrix, transposed to have k rows and n columns.

8 Probability

8.1 Entropy

$$\begin{aligned} H(X) &= \sum_{x \in X} p_x(x) \log_2 \frac{1}{p_x(x)} \\ &= -\sum_{x \in X} p_x(x) \log_2 (p_x(x)) \end{aligned}$$

8.2 Conditional Entropy

Conditional entropy of a random variable X with range $\vec{X}$ conditioned on the event $Y = y$ is:

$$H(X|Y = y) = -\sum_{x \in \vec{X}} p_{X|Y}(x, y) \log_2 (p_{X|Y}(x, y))$$

Conditional entropy of a random variable X with range $\vec{X}$ conditioned on the random variable Y with range $\vec{Y}$ is:

$$H(X|Y) = \sum_{y \in \vec{Y}} p_Y(y) H(X|Y = y)$$

8.3 Mutual Information

To compute the mutual information between two random variables X, Y with ranges $\vec{X}, \vec{Y}$, we get:

$$|(X,Y) = \sum_{x \in \vec{X}} \sum_{y \in \vec{Y}} p_{XY}(x,y) \log_2 \frac{p_{XY}(x,y)}{p_X(x) \times p_Y(y)}$$

It can be quicker to use:

$$|(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$