MT361/MT461/MT5461
Error Correcting Codes

Mark Wildon, mark.wildon@rhul.ac.uk

- Please take a handout, the preliminary problem sheet, and the sheet of challenge problems.
- All handouts and problem sheets will be put on the Moodle page for MT361. Everyone has access to this page. If you are not yet registered for the course, bookmark this link: moodle.rhul.ac.uk/course/view.php?id=381
- Lectures are at: Monday 3pm in MFLEC, Tuesday 3pm in BLT2, Thursday 10am in BLT2.
- There is an additional lecture at Thursday noon for MT4610 and MT5461 in ABLT3. **Correction: in week TWO only, this lecture will be moved to Friday 2pm in C219.**
- Office hours: Tuesday 11am, Thursday 2pm and Friday 11am. (This week as normal.)

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits
- Text on first page of handout: 7144 bits

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits
- Text on first page of handout: 7144 bits
- First 14 pages of handout: 1826816 bits

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits
- Text on first page of handout: 7144 bits
- First 14 pages of handout: 1826816 bits or 0.223 MB

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits
- Text on first page of handout: 7144 bits
- First 14 pages of handout: 1826816 bits or 0.223 MB
- Compact disc of Beethoven 9th: 700 MB

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits
- Text on first page of handout: 7144 bits
- First 14 pages of handout: 1826816 bits or 0.223 MB
- Compact disc of Beethoven 9th: 700 MB
- Bluray disc of 3 hour film: 50 GB

Why Coding Theory?

We want to communicate reliably in the presence of noise. With small amounts of data, we can do this quite easily.

But in the modern world we want to store and process huge amounts of data.

Think of a 'bit' as a single piece of information: on or off, 0 or 1.

- A small QR-code:  441 bits
- Text on first page of handout: 7144 bits
- First 14 pages of handout: 1826816 bits or 0.223 MB
- Compact disc of Beethoven 9th: 700 MB
- Bluray disc of 3 hour film: 50 GB
- Large Hadron Collider data, per second: 300 GB

Learning Objectives

(A) Examples of codes. Error detection and error correction and connection with Hamming distance and Hamming balls. Information rate and the binary symmetric channel.

(B) The Main Coding Theory Problem. Singleton bound and codes based on Latin squares. Plotkin bound and Hadamard codes. Hamming and Gilbert–Varshamov bounds.

(C) Linear codes. Generator matrices and encoding. Cosets and decoding by standard arrays. Parity check matrices and syndrome decoding. Hamming codes. Dual codes.

Prerequisites

- Basic discrete probability.
- Modular arithmetic in $\mathbf{Z}_p$ where p is prime. If you are happy with calculations such as $5 + 4 \equiv 2 \bmod 7$, $5 \times 4 \equiv 6 \bmod 7$ and $5^{-1} \equiv 3 \bmod 7$, that should be enough.
- Some basic linear algebra: matrices, vector spaces, subspaces, row-reduced echelon form. This will be reviewed when we need it in Part C of the course.

Recommended Reading

[1] *Combinatorics: Topics, Techniques, Algorithms.* Peter J. Cameron, CUP, 1994. (Chapter 17 gives a concise account of coding theory.)

[2] *Coding and Information Theory.* Richard W. Hamming, Prentice-Hall, 1980. (Chapters 2, 3 and 11 are relevant to this course.)

[3] *A First Course in Coding Theory.* Raymond Hill, OUP, 1986. (Highly recommended. It is very clear, covers all the 3rd year course, and the library has several copies.)

[4] *Coding Theory: A First Course.* San Ling and Chaoping Xing, CUP, 2004.

[5] *The Theory of Error-Correcting Codes.* F. J. MacWilliams and N. J. A. Sloane, North-Holland, 1977. (Mainly for reference.)

Richard W. Hamming

"Mathematics is not merely an idle art form, it is an essential part of our society."

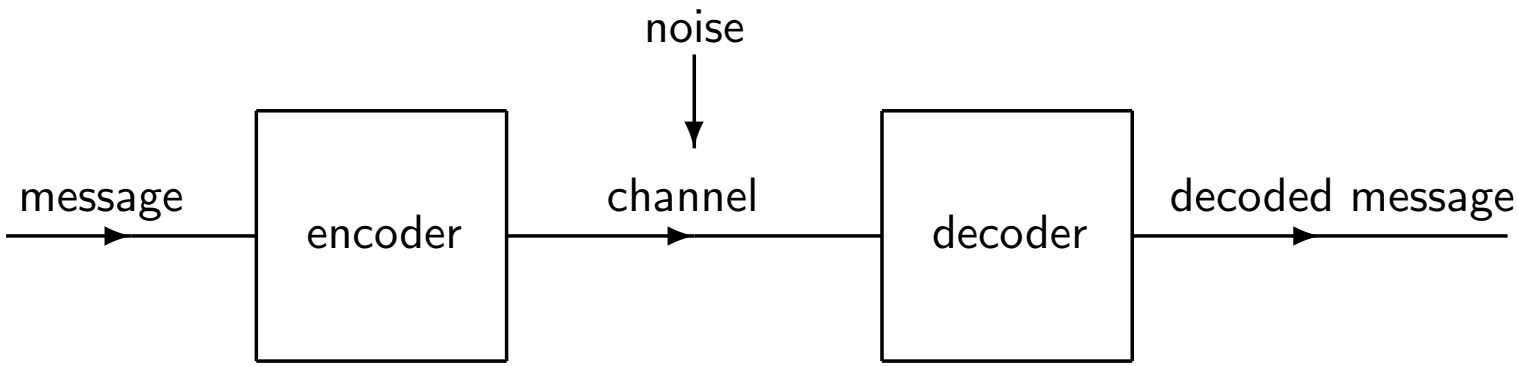
R. W. Hamming, 1998

Hamming's original paper, *Error Detecting and Error Correcting Codes*, Bell Systems Technical Journal, **2** (1950) 147–160, is beautifully written and, for a mathematics paper, very easy to read. It is available from `www.lee.eng.uerj.br/~gil/redesII/hamming.pdf`.

Main Problem.

Problem 1.1

Alice wants to send a message to Bob. She can communicate with him by sending him a word formed from symbols taken from some fixed set. But every time she sends a word, there is a chance that some of its symbols will be corrupted, so the word that Bob receives may not be the word that Alice sent. How can Alice and Bob communicate reliably?



Example 1.2

Alice wants to send the message 'Yes' or 'No' to Bob. The available symbols are 0 and 1.

Scheme 1. The two decide, in advance, that Alice will send

- 00 for 'No',
- 11 for 'Yes'.

If Bob receives 00 or 11 then he will assume this is the word that Alice sent, and decode her message. If he receives 01 or 10 then he knows an error has occurred, but does not know which symbol is wrong.

Example 1.2

Alice wants to send the message 'Yes' or 'No' to Bob. The available symbols are 0 and 1.

Scheme 1. The two decide, in advance, that Alice will send

- 00 for 'No',
- 11 for 'Yes'.

If Bob receives 00 or 11 then he will assume this is the word that Alice sent, and decode her message. If he receives 01 or 10 then he knows an error has occurred, but does not know which symbol is wrong.

Scheme 2. Suppose instead they decide that Alice will send

- 000 for 'No',
- 111 for 'Yes'.

Then Bob can decode Alice's message correctly, provided at most one error occurs, by assuming that the symbol in the majority is correct.

MT361/MT461/MT5461
Error Correcting Codes

Mark Wildon, mark.wildon@rhul.ac.uk

- Fresh copies of handout, the preliminary problem sheet, and the sheet of challenge problems.
- All handouts and problem sheets will be put on the Moodle page for MT361. Everyone has access to this page. If you are not yet registered for the course, bookmark this link: moodle.rhul.ac.uk/course/view.php?id=381
- There is an additional lecture at Thursday noon for MT4610 and MT5461 in ABLT3. **Correction: in week TWO only, this lecture will be moved to Friday 2pm in C219.**
- Office hours: Tuesday 11am, Thursday 2pm and Friday 11am. (This week as normal.)

Definitions

Definition 1.3

Let $q \in \mathbf{N}$. A q-ary alphabet is a set of q different elements, called *symbols*. A *word* of *length* n over an alphabet A is a sequence $(x_1, x_2, \ldots, x_n)$ where $x_i \in A$ for each i.

Definitions

Definition 1.3

Let $q \in \mathbf{N}$. A q-ary alphabet is a set of q different elements, called *symbols*. A *word* of *length* n over an alphabet A is a sequence $(x_1, x_2, \ldots, x_n)$ where $x_i \in A$ for each i.

Definition 1.4

Let A be an alphabet and let $n \in \mathbf{N}$. A *code over* A of *length* n is a subset C of A^n containing at least two words. The elements of C are called *codewords*. The *size* of C is $|C|$.

Definitions

Definition 1.3

Let $q \in \mathbf{N}$. A q-ary alphabet is a set of q different elements, called *symbols*. A *word* of *length* n over an alphabet A is a sequence $(x_1, x_2, \ldots, x_n)$ where $x_i \in A$ for each i.

Definition 1.4

Let A be an alphabet and let $n \in \mathbf{N}$. A *code over* A of *length* n is a subset C of A^n containing at least two words. The elements of C are called *codewords*. The *size* of C is $|C|$.

Definition 1.5

The *binary alphabet* of **bi***nary digi***ts**, or *bits*, is $\{0, 1\}$. A *binary code* is a code over $\{0, 1\}$.

Alice and Bob revisited

When writing words we will often omit the brackets and commas. So e.g. $(1, 1, 1)$ can be written as 111, and so on.

Example 1.2 (continued)

In *Scheme 1*, Alice and Bob use the binary code

$$C = \{00, 11\}$$

which has length 2 and size 2 and in *Scheme 2* they use

$$D = \{000, 111\}$$

which has length 3 and size 2.

Alice and Bob revisited

Example 1.2 (concluded)

Suppose that whenever a bit 0 or 1 is sent down the channel used by Alice and Bob, there is a probability p that it flips, so a 0 becomes a 1, and a 1 becomes a 0.

Exercise: Why is it reasonable to assume that $p < 1/2$?

For definiteness we shall suppose that Alice sends 'Yes' to Bob: you should be able to check that we get the same behaviour if Alice sends 'No'. Using Scheme 2, Alice sends 111 and Bob decodes wrongly if and only if he receives 000, 001, 010 or 100. This event has probability

$$p^3 + 3p^2(1-p).$$

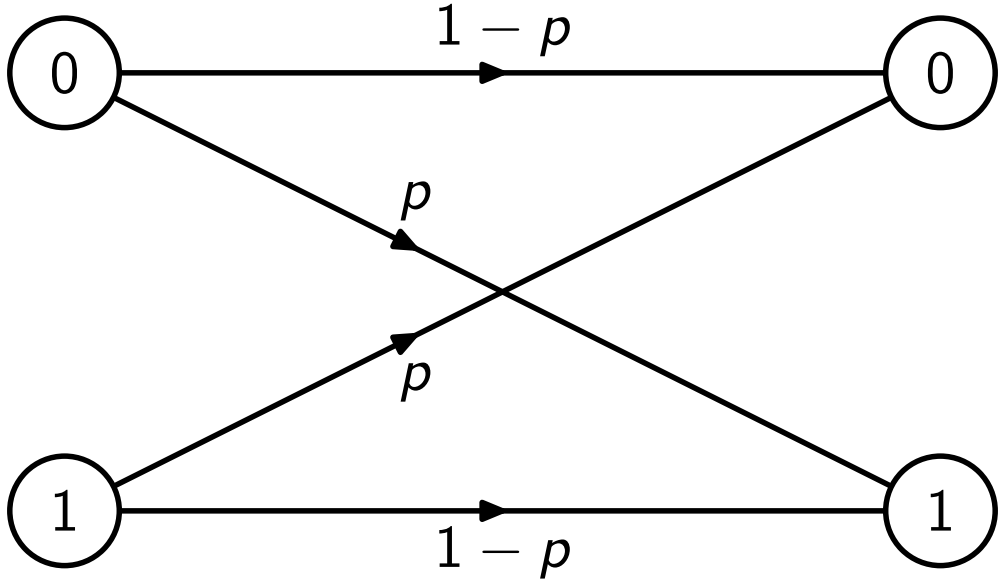
See the preliminary problem sheet for an analysis of Scheme 1.

Binary Symmetric Channel

Definition 1.6

The binary channel in which each transmitted bit flips, so a 0 becomes a 1 and a 1 becomes a 0, independently with probability p, is called the *binary symmetric channel* with *cross-over probability* p.

The transition probabilities for the binary symmetric channel are shown in the diagram below.



Remarks on the Definition of Codes

Remarks 1.7

The following remarks on Definition 1.4 should be noted.

(1) By Definition 1.4, all codewords in a code have the same length.

(2) We assume that all our codes have size ≥ 2: if a code has no codewords, or only one, then it's useless for communication.

(3) It is very important to realise that the codes in this course **are not secret codes**. The set of codewords, and how Alice and Bob plan to use the code to communicate, should be assumed to be known to everyone.

(4) The definition of a code does not mention the encoder or decoder. This is deliberate: the same code might be used for different sets of messages, and with different decoding strategies: see Example 1.8.

Using the Same Code in Different Ways

Recall Example 1.2, where in Scheme 2, Alice sent a 'Yes' / 'No' message to Bob using the code $D = \{000, 111\}$. Bob decoded a received word by assuming that the majority symbol was correct.

Example 1.8

Suppose ALICE wants to send BOB one of the messages 'Launch nukes' or 'Stand-down'. They decide to use the binary code $D = \{000, 111\}$ from Example 1.2, with the encoder

$$\text{'Stand-down'} \overset{\text{encoded as}}{\longmapsto} 000$$

$$\text{'Launch nukes'} \overset{\text{encoded as}}{\longmapsto} 111.$$

Erring on the side of safety, they decide that if Bob receives a non-codeword (i.e. one of $001, 010, 100, 110, 101, 011$), then he will request retransmission. So the same code is used, but with a different encoder and a different decoding strategy.

Guessing and Liar Games

Exercise: Alice thinks of a number between 0 and 15. Playing the role of Bob, how many yes/no questions do you need to ask Alice to find out her number?

Guessing and Liar Games

Exercise: Alice thinks of a number between 0 and 15. Playing the role of Bob, how many yes/no questions do you need to ask Alice to find out her number?

Exercise: Now suppose that Alice is allowed to tell *at most* one lie when she answers Bob's questions. (Or, corresponding more closely to noise in the channel, suppose that Alice can choose to mumble in one of her answers so that Bob mishears her answer.) Repeat the game in the previous exercise. How many questions do you need?

Guessing and Liar Games

Exercise: Alice thinks of a number between 0 and 15. Playing the role of Bob, how many yes/no questions do you need to ask Alice to find out her number?

Exercise: Now suppose that Alice is allowed to tell *at most* one lie when she answers Bob's questions. (Or, corresponding more closely to noise in the channel, suppose that Alice can choose to mumble in one of her answers so that Bob mishears her answer.) Repeat the game in the previous exercise. How many questions do you need?

Variant 1: only questions about the number are allowed. (E.g. is your number odd?, is your number in the set $\{2,3,6,7\}$?, etc.)

Variant 2: any question with a yes/no answer is allowed.

Guessing and Liar Games

Exercise: Alice thinks of a number between 0 and 15. Playing the role of Bob, how many yes/no questions do you need to ask Alice to find out her number?

Exercise: Now suppose that Alice is allowed to tell *at most* one lie when she answers Bob's questions. (Or, corresponding more closely to noise in the channel, suppose that Alice can choose to mumble in one of her answers so that Bob mishears her answer.) Repeat the game in the previous exercise. How many questions do you need?

Variant 1: only questions about the number are allowed. (E.g. is your number odd?, is your number in the set $\{2,3,6,7\}$?, etc.)

Variant 2: any question with a yes/no answer is allowed.

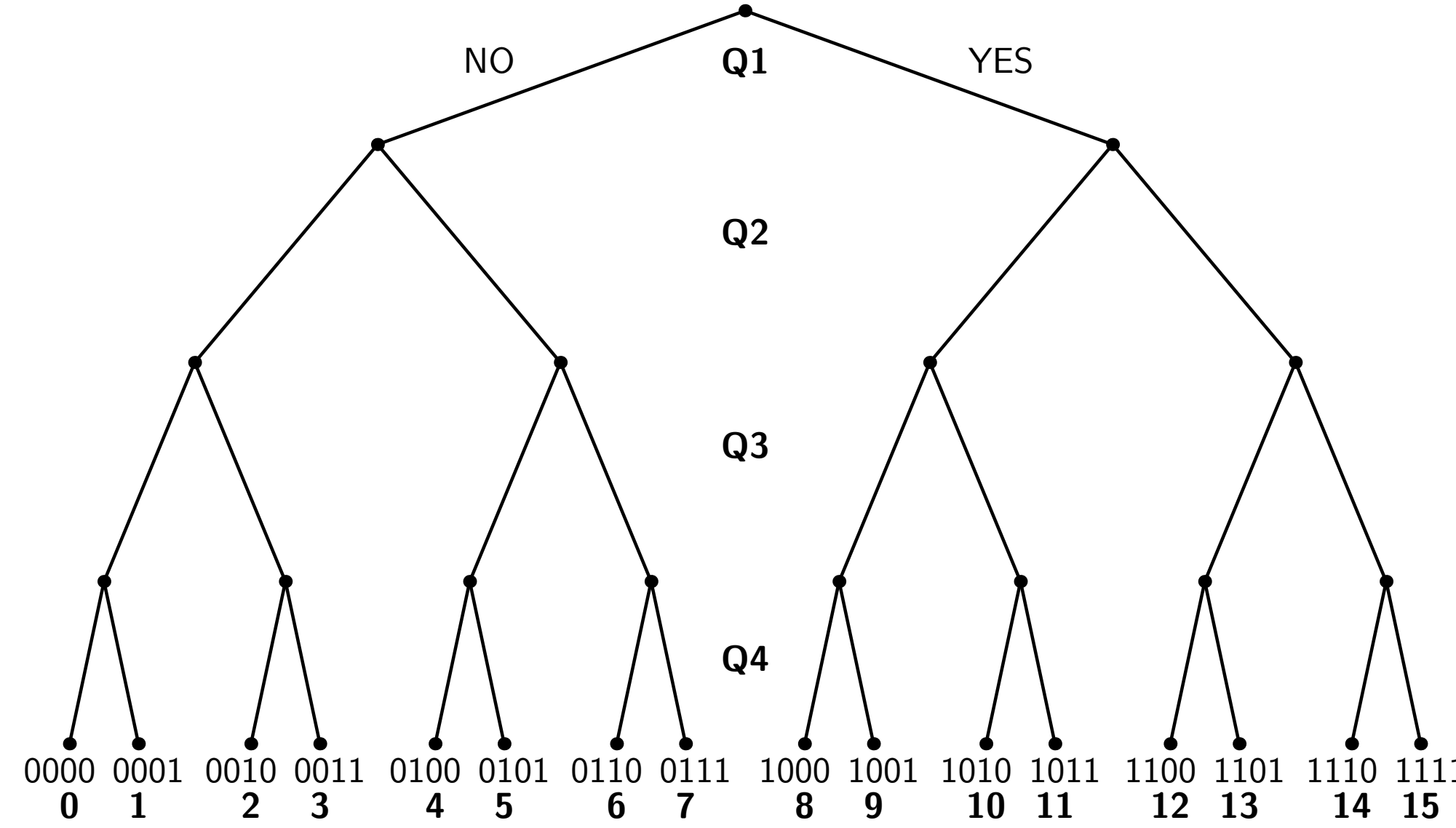
Example 1.9

We will convert some of the possible questioning strategies for Bob into binary codes of size 16.

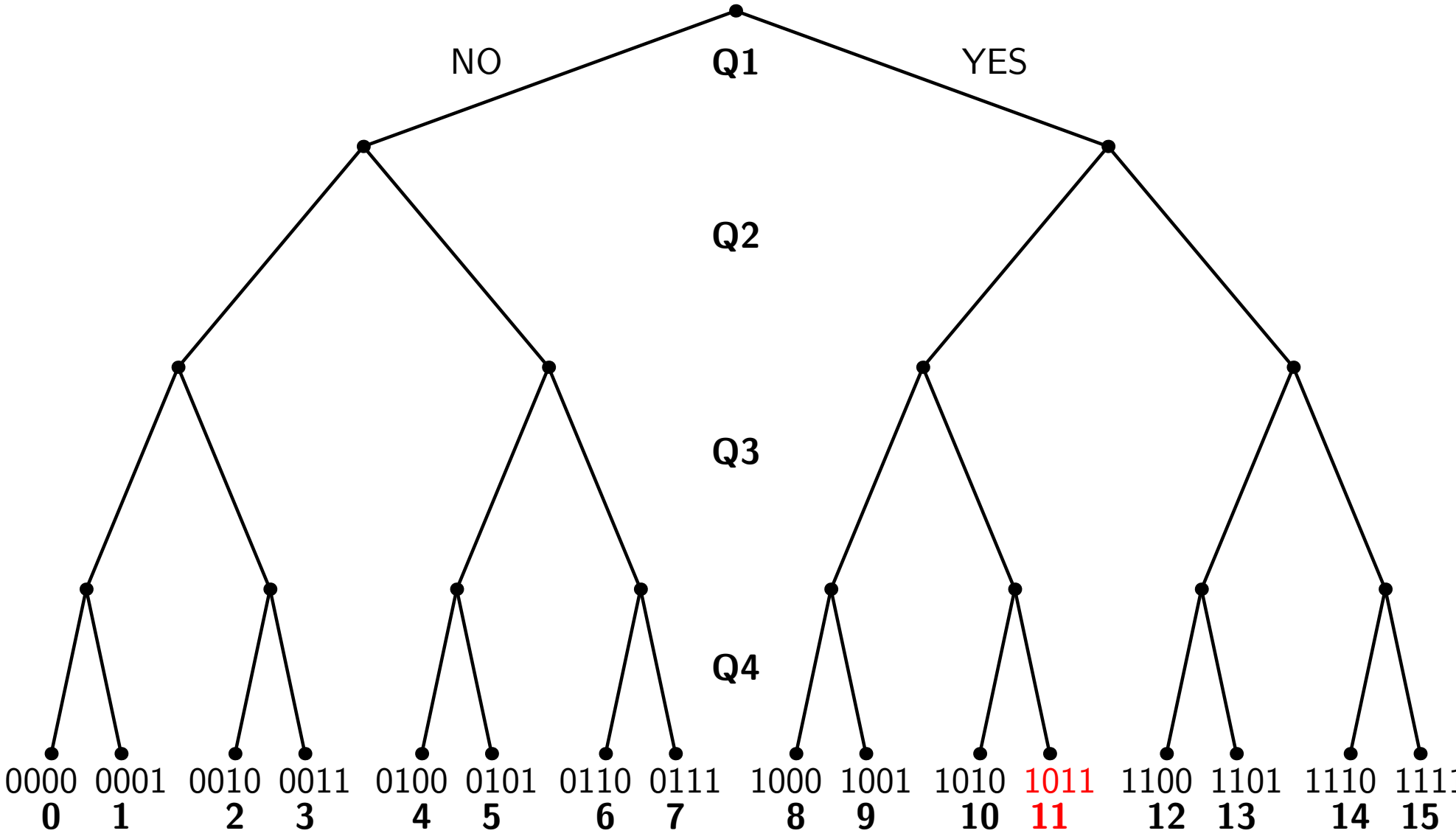
Administration

- From **this Tuesday** the Tuesday lecture at 3pm will be in QBLT (Queen's Building Lecture Theatre).
- Solutions to the preliminary problem sheet will appear on Moodle by 5pm.

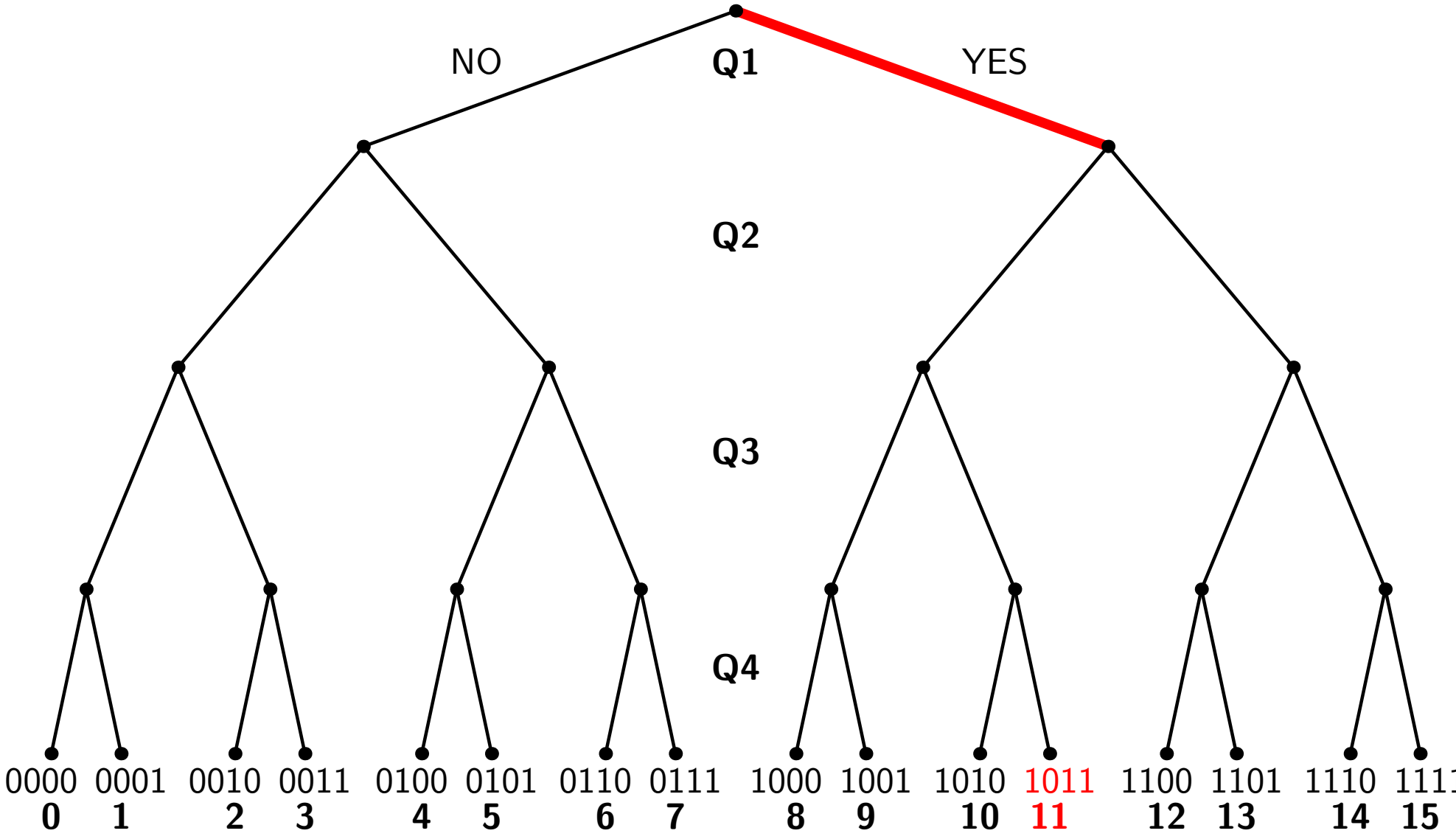
Example 1.9: A Decision Tree for the Guessing Game



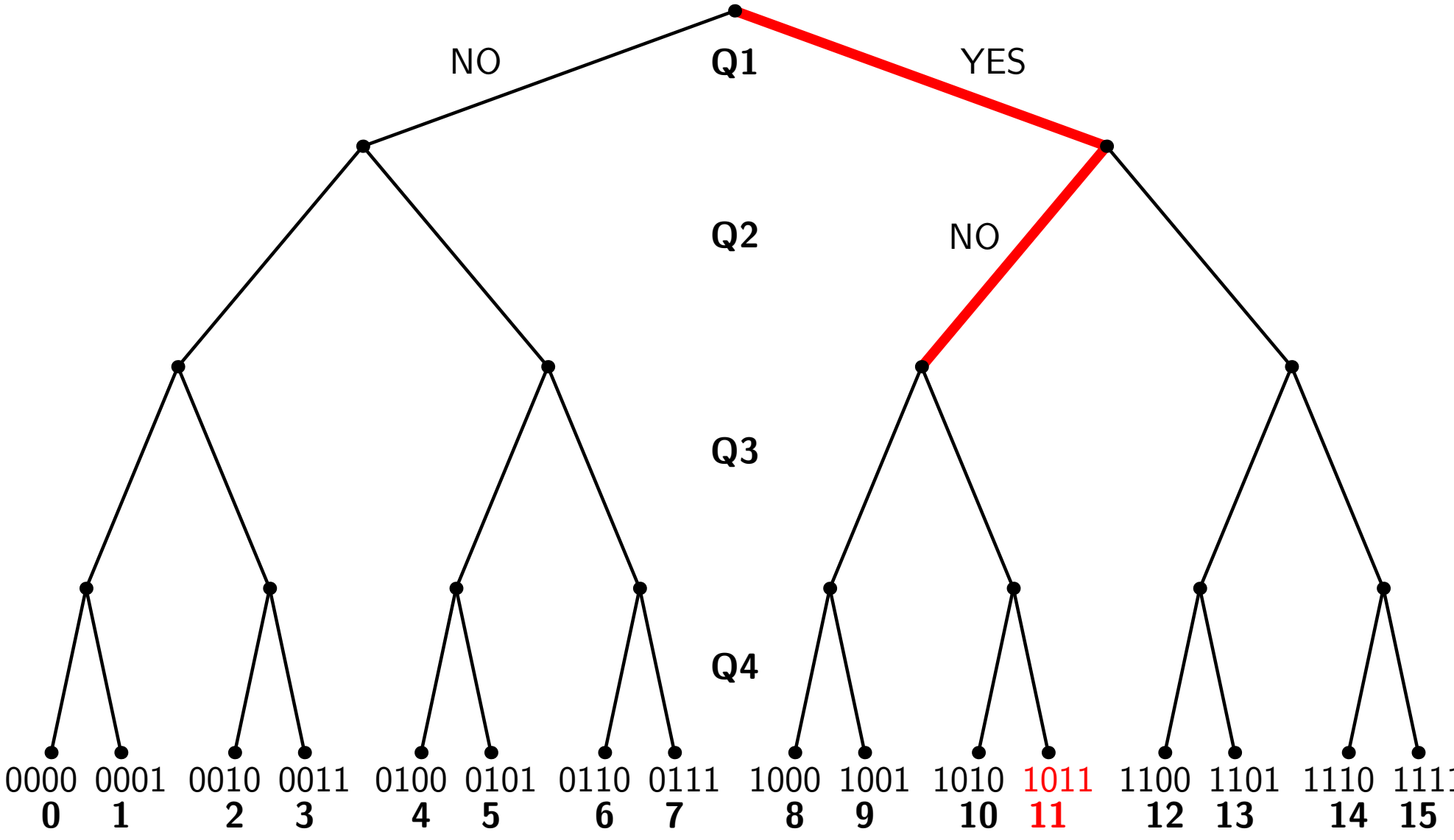
Example 1.9: A Decision Tree for the Guessing Game



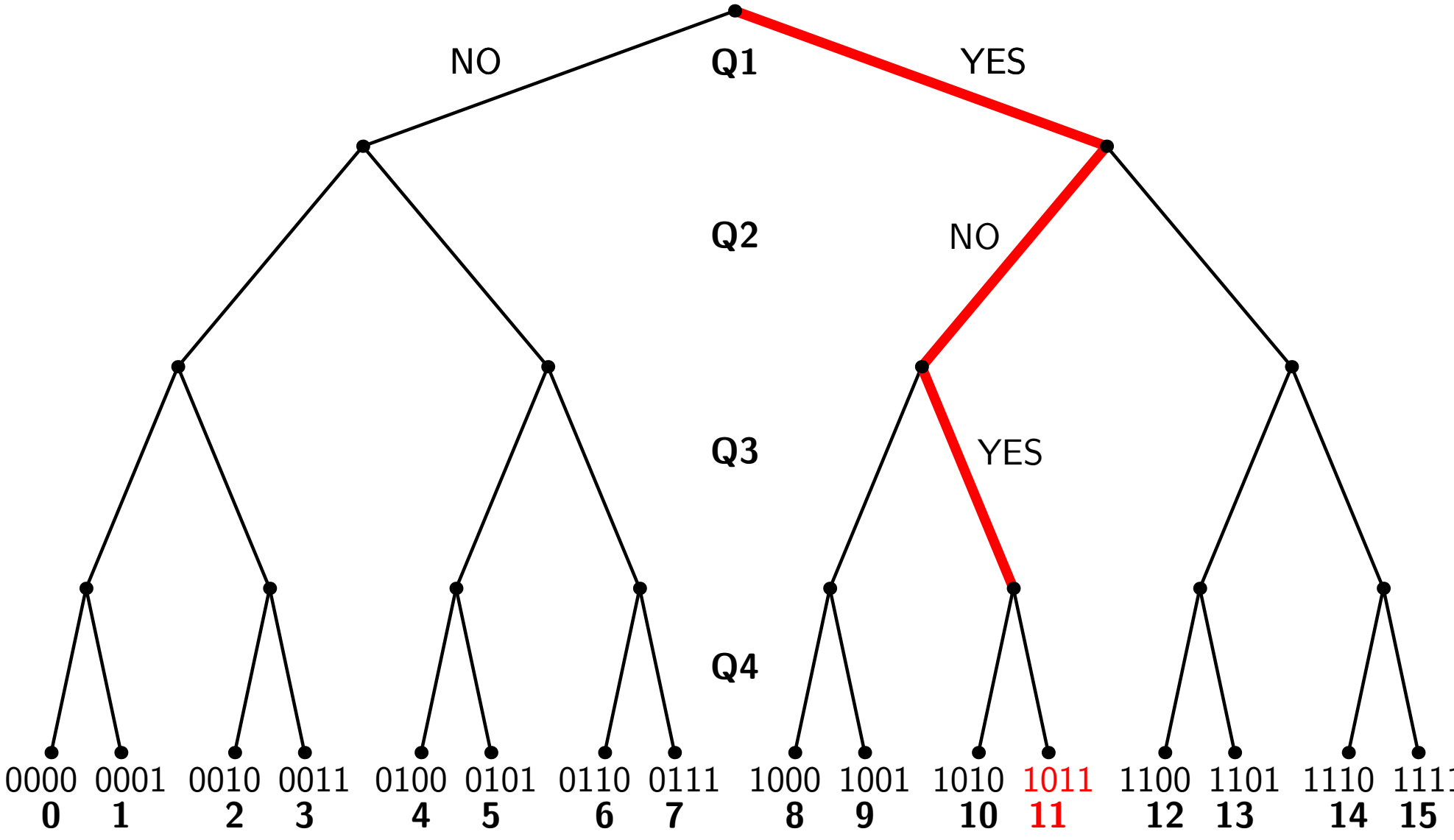
Example 1.9: A Decision Tree for the Guessing Game



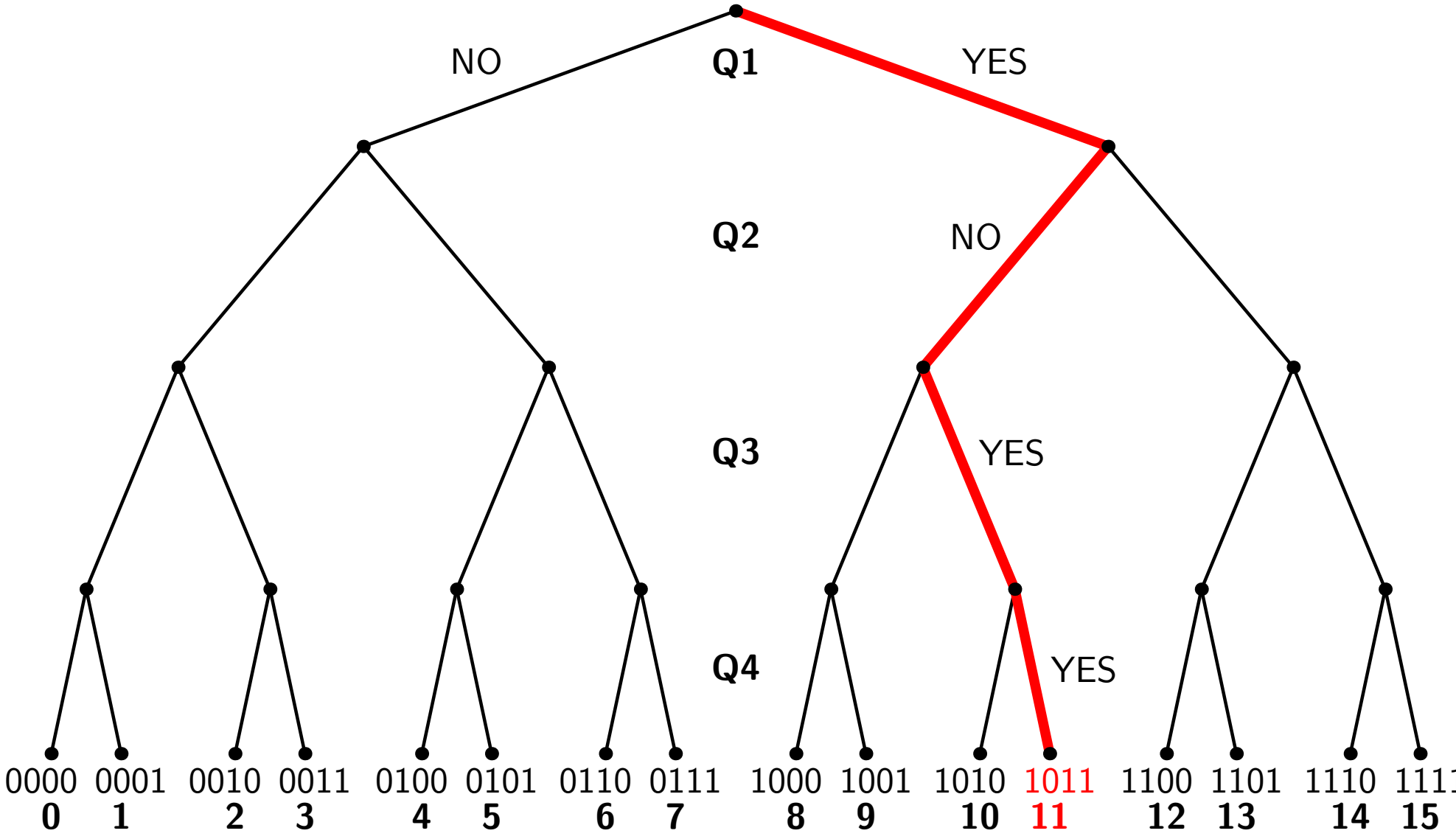
Example 1.9: A Decision Tree for the Guessing Game



Example 1.9: A Decision Tree for the Guessing Game



Example 1.9: A Decision Tree for the Guessing Game



Example 1.9: Codes from the Guessing and Liar Games

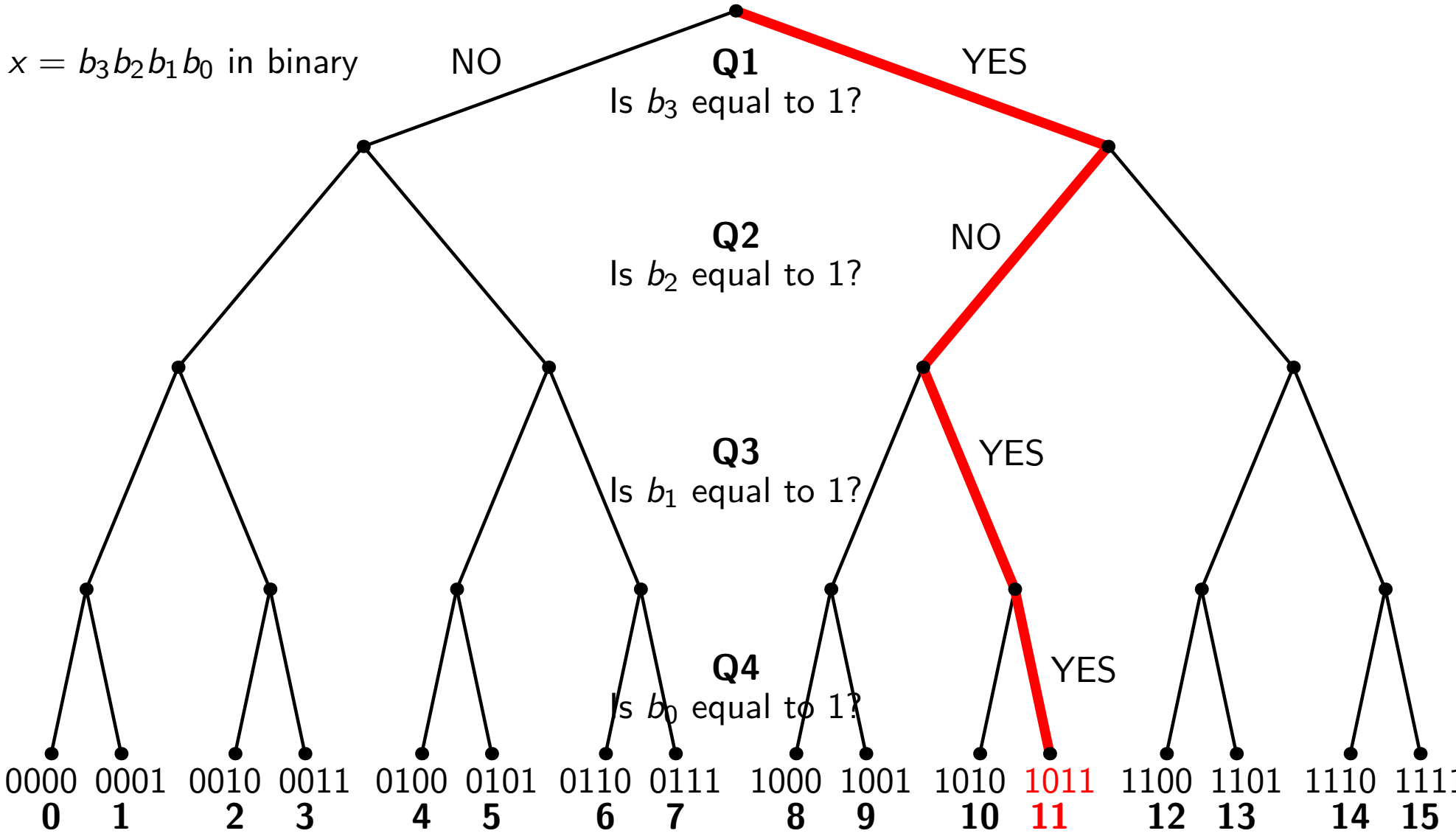
When playing the game as Bob, you could use the answers you had heard so far to help choose the next question.

But to turn a questioning strategy into a code, we need to be able to write down all the questions *in advance*.

(1) Let Alice's number be $m = b_3 2^3 + b_2 2^2 + b_1 2 + b_0$, so $m = b_3 b_2 b_1 b_0$ in binary. Encode m as $b_3 b_2 b_1 b_0$ and ask about b_3, b_2, b_1, b_0. The corresponding code has as its codewords all binary words of length 4 and the decoder is

$$0000 \mapsto 0, \quad 0001 \mapsto 1, \quad 0010 \mapsto 2, \quad \ldots, \quad 1111 \mapsto 15.$$

Example 1.9: A Decision Tree for the Guessing Game



Code from a 9 Question Strategy for the Liar Game

(2) Let Alice's number be $x = b_3 2^3 + b_2 2^2 + b_1 2 + b_0$.

Questioning strategy from Thursday: ask about each bit of Alice's number, twice. Then ask about $e = b_3 + b_2 + b_1 + b_0$. The answer to the final question will determine which (if any) of the first eight answers was a lie.

Code from a 9 Question Strategy for the Liar Game

(2) Let Alice's number be $x = b_3 2^3 + b_2 2^2 + b_1 2 + b_0$.
Questioning strategy from Thursday: ask about each bit of Alice's number, twice. Then ask about $e = b_3 + b_2 + b_1 + b_0$. The answer to the final question will determine which (if any) of the first eight answers was a lie.

In the corresponding code, the number x is encoded as

$$(b_3, b_2, b_1, b_0, b_3, b_2, b_1, b_0, e)$$

where $e = b_3 + b_2 + b_1 + b_0 \bmod 2$. For example,

$$\begin{aligned} 0 &\mapsto 0000\,0000\,0 \\ 1 &\mapsto 0001\,0001\,1 \\ 2 &\mapsto 0010\,0010\,1 \\ &\vdots \\ 15 &\mapsto 1111\,1111\,0. \end{aligned}$$

Code from a 9 Question Strategy for the Liar Game

(2) Let Alice's number be $x = b_3 2^3 + b_2 2^2 + b_1 2 + b_0$.
Questioning strategy from Thursday: ask about each bit of Alice's number, twice. Then ask about $e = b_3 + b_2 + b_1 + b_0$. The answer to the final question will determine which (if any) of the first eight answers was a lie.

In the corresponding code, the number x is encoded as

$$(b_3, b_2, b_1, b_0, b_3, b_2, b_1, b_0, e)$$

where $e = b_3 + b_2 + b_1 + b_0 \bmod 2$. For example,

$$\begin{aligned} 0 &\mapsto 0000\,0000\,0 \\ 1 &\mapsto 0001\,0001\,1 \\ 2 &\mapsto 0010\,0010\,1 \\ &\vdots \\ 15 &\mapsto 1111\,1111\,0. \end{aligned}$$

Exercise: Decode the following received words:

$$011001100, 110111001, 100110011, 001101010$$

Rate of a Code

Definition 1.10

Let C be a code of length n and size M. We define the *rate* of C to be $(\log_2 M)/n$.

Roughly put, the rate of a binary code measures the proportion of the bits of the codewords that give direct information about the encoded message. For example, the binary codes $\{00, 11\}$ and $\{000, 111\}$ used by Alice and Bob in Example 1.2 have rates $1/2$ and $1/3$, respectively.

Rate of a Code

Definition 1.10

Let C be a code of length n and size M. We define the *rate* of C to be $(\log_2 M)/n$.

Roughly put, the rate of a binary code measures the proportion of the bits of the codewords that give direct information about the encoded message. For example, the binary codes $\{00, 11\}$ and $\{000, 111\}$ used by Alice and Bob in Example 1.2 have rates $1/2$ and $1/3$, respectively.

The code of length 9 used for the guessing game has size 16 so has rate $(\log_2 16)/9 = 4/9$.

The Square Code

The Square Code on Question 2 of the preliminary sheet is a binary code of length 8. Its codewords are all binary words of the form

$$(u_1, u_2, u_3, u_4, u_1 + u_2, u_3 + u_4, u_1 + u_3, u_2 + u_4)$$

where $u_1, u_2, u_3, u_4 \in \{0, 1\}$ and the addition is done modulo 2.

(a) Check that 11000011 is a codeword in the square code. Draw it as a square diagram.

(b) Suppose Alice sends 11000011 and Bob receives 01000011. How can Bob work out that Alice sent 11000011?

The Square Code

The Square Code on Question 2 of the preliminary sheet is a binary code of length 8. Its codewords are all binary words of the form

$$(u_1, u_2, u_3, u_4, u_1 + u_2, u_3 + u_4, u_1 + u_3, u_2 + u_4)$$

where $u_1, u_2, u_3, u_4 \in \{0, 1\}$ and the addition is done modulo 2.

(a) Check that 11000011 is a codeword in the square code. Draw it as a square diagram.

(b) Suppose Alice sends 11000011 and Bob receives 01000011. How can Bob work out that Alice sent 11000011?

Exercise: Alice wants to send Bob a number m between 0 and 15. She writes m in binary as $m = 2^3 b_3 + 2^2 b_2 + 2^1 b_1 + 2^0 b_0$ and then sends Bob the codeword in the Square Code starting $b_3 b_2 b_1 b_0 \ldots$

Imagine you are Bob and you receive 10011001. What do you think Alice's number is?

The Square Code

The Square Code on Question 2 of the preliminary sheet is a binary code of length 8. Its codewords are all binary words of the form

$$(u_1, u_2, u_3, u_4, u_1 + u_2, u_3 + u_4, u_1 + u_3, u_2 + u_4)$$

where $u_1, u_2, u_3, u_4 \in \{0, 1\}$ and the addition is done modulo 2.

(a) Check that 11000011 is a codeword in the square code. Draw it as a square diagram.

(b) Suppose Alice sends 11000011 and Bob receives 01000011. How can Bob work out that Alice sent 11000011?

Exercise: Alice wants to send Bob a number m between 0 and 15. She writes m in binary as $m = 2^3 b_3 + 2^2 b_2 + 2^1 b_1 + 2^0 b_0$ and then sends Bob the codeword in the Square Code starting $b_3 b_2 b_1 b_0 \ldots$

Imagine you are Bob and you receive 10011001. What do you think Alice's number is?

(A) 8 (B) 9 (C) 11 (D) 13.

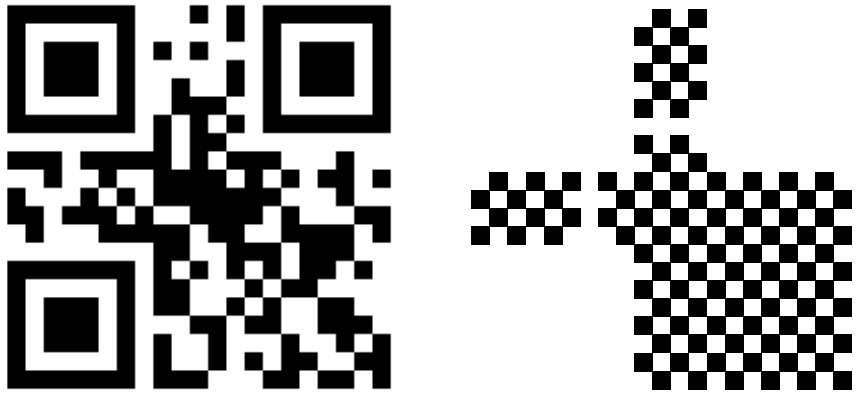
Example 1.11: Quick-Response Codes

A *Quick-Response code*, or *QR-code* is a 21×21 grid of small black and white squares. The squares represent a binary sequence (black is 1 and white is 0) encoding a short message. If your mobile phone has a camera, you might be able to use it to decode the QR-code below.

The MATHEMATICA notebook `QRCodes.nb` can be used to encode short messages.

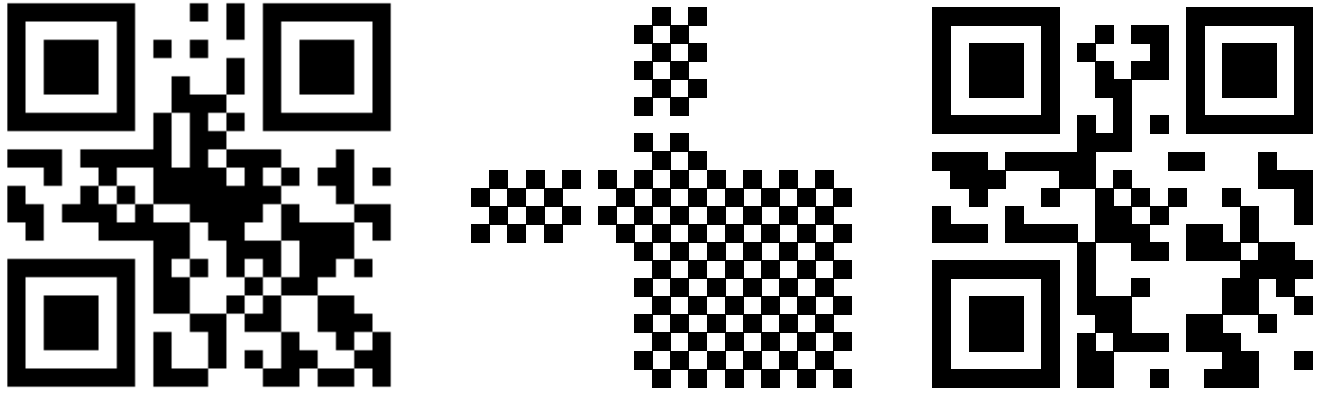
Example 1.11: Quick-Response Codes

The first step in decoding is to 'subtract' a masking pattern, specified by a format string included in two places in the grid.



Example 1.11: Quick-Response Codes

The first step in decoding is to 'subtract' a masking pattern, specified by a format string included in two places in the grid.



Most of the data in this QR-code consists of $26 \times 8 = 208$ bits which (after unmasking) form a single codeword in a Reed–Solomon code of length 26 over an alphabet of size 2^8.

The encoder begins by converting each character in the message into a number between 0 and 44: for example, 'C' is sent to 12, 'D' to 13, and so on. Each pair of numbers is then converted into an 11 bit binary word. These words are then concatenated into a string of $19 \times 8 = 152$ bits which is encoded using the Reed–Solomon code.

QR-Codes

Exercise: Why do you think the format string is included twice? What is the point of the three squares in the corners? (Including the white border, these squares occupy $8^2 \times 3 = 192$ of the $21^2 = 441$ little squares, so they must be important for some reason!)

Exercise: You are invited to see how much damage the copy of the QR-code below in the printed notes can tolerate before decoding fails.

Example 1.12: Reed–Solomon CD Code

A compact-disc contains information in the form of a sequence of microscopic pits on a disc that are read by a laser. Here the compact disc is the channel, and its main purpose is to transmit information reliably through time, rather than through space.

The pits encode a long sequence of the bits 0 and 1. The encoding scheme combines two different Reed–Solomon codes. The first code alone guarantees to correct one error in each block of 128 consecutive bits.

If, however, the errors occur in adjacent bits, as is usual for a scratch then, mainly because of the clever way in which the two codes are combined, many more errors can be corrected. Up to $16 \times 32 \times 8 = 4096$ adjacent bits can be corrupted in a block of $124 \times 28 \times 8 = 27776$ bits and the compact disc will still be decoded successfully.

Example 1.13: Australian Railways Telegraph Code

In this code the encoder encodes a message as a list of codewords, and then each codeword is sent, separately, down the channel.

Ayah	Provide locomotive to work
Aybu	Return locomotive at once
Azaf	Breakdown train left at ...
Azor	Arrange to provide assistance locomotive
Azub	A second locomotive will be attached to ...

In telegraph transmission only upper case were used. So a typical message might be something like 'Breakdown train left at Sydney, provide locomotive to work', encoded as AZAF SYDNEY AYAH.

Exercise: Most codewords are not English words, although a few are: '**Coma**' is an instruction about extra trucks, '**Cosy**' is an instruction about loading trucks. Why do you think English words were usually avoided?

Example 1.14: Mariner 9

The Mariner 9 probe, launched in 1971, took the first pictures of Mars, ultimately transmitting 7239 pictures at a resolution of 700×832 pixels. The images were grey-scale, using $64 = 2^6$ different shades of grey.

- The probe did not have the internal memory to store even one image, so the image had to be transmitted as it was captured.
- The pictures were transmitted back to Earth by sending one pixel at a time, so we can think of a message as a single number between 0 and 63.
- The channel could send the two binary digits 0 and 1. The probability of each bit being flipped in the channel was about 0.05.
- Had each pixel been encoded a 6 bit number, about 26% of the image would have been wrong.

Codes for Mariner 9

It was acceptable for each pixel to be encoded by up to 32 bits, so increasing the amount of data to be stored and transmitted by a factor of 5.

A repetition code where each of the 6 bits needed to specify a grey level was repeated 5 times would reduce the percentage of incorrect pixels to less than 4%.

The Hadamard code of length 32 and size 64 actually used could correct up to 7 errors in each transmitted word. This reduced the percentage of incorrect pixels to about 0.014%.

Codes for Mariner 9

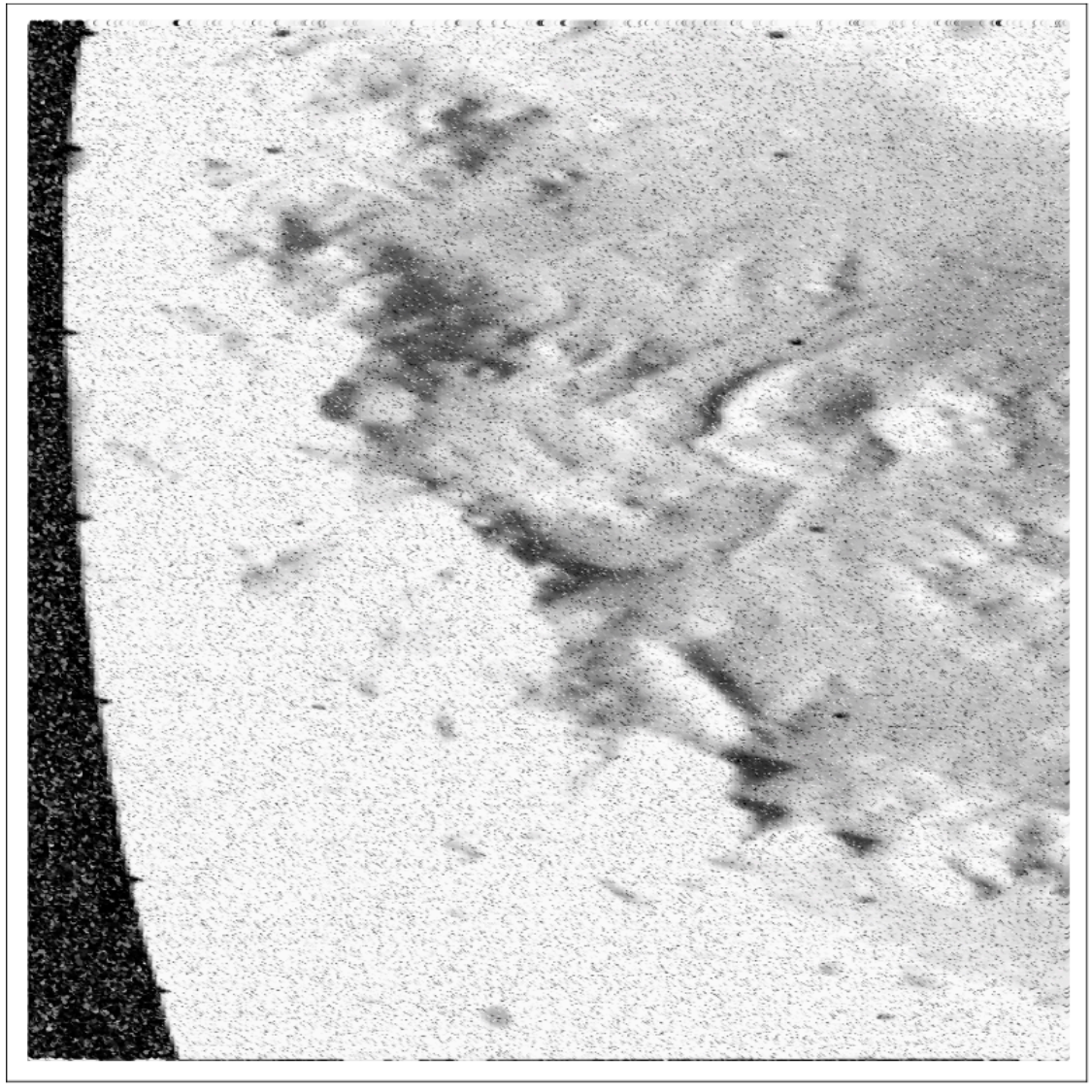
It was acceptable for each pixel to be encoded by up to 32 bits, so increasing the amount of data to be stored and transmitted by a factor of 5.

A repetition code where each of the 6 bits needed to specify a grey level was repeated 5 times would reduce the percentage of incorrect pixels to less than 4%.

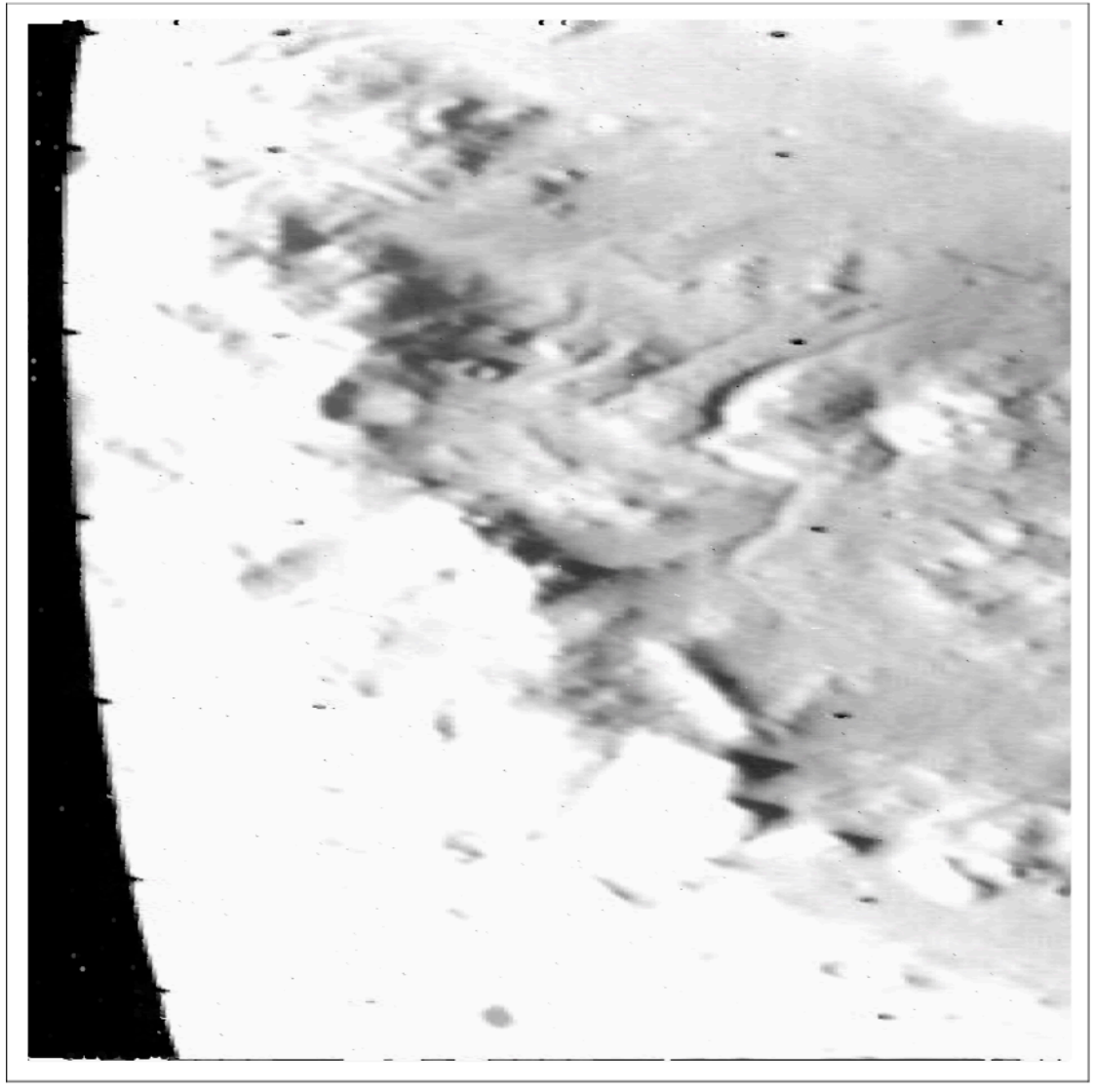
The Hadamard code of length 32 and size 64 actually used could correct up to 7 errors in each transmitted word. This reduced the percentage of incorrect pixels to about 0.014%.

The next two slides show one of the Mariner 9 transmissions, as it might have been received had each shade been encoded as a 6 bit binary word, and one of the actual images sent using the Hadamard code.

Mariner 9 Image: Improvement Due to Error Correction

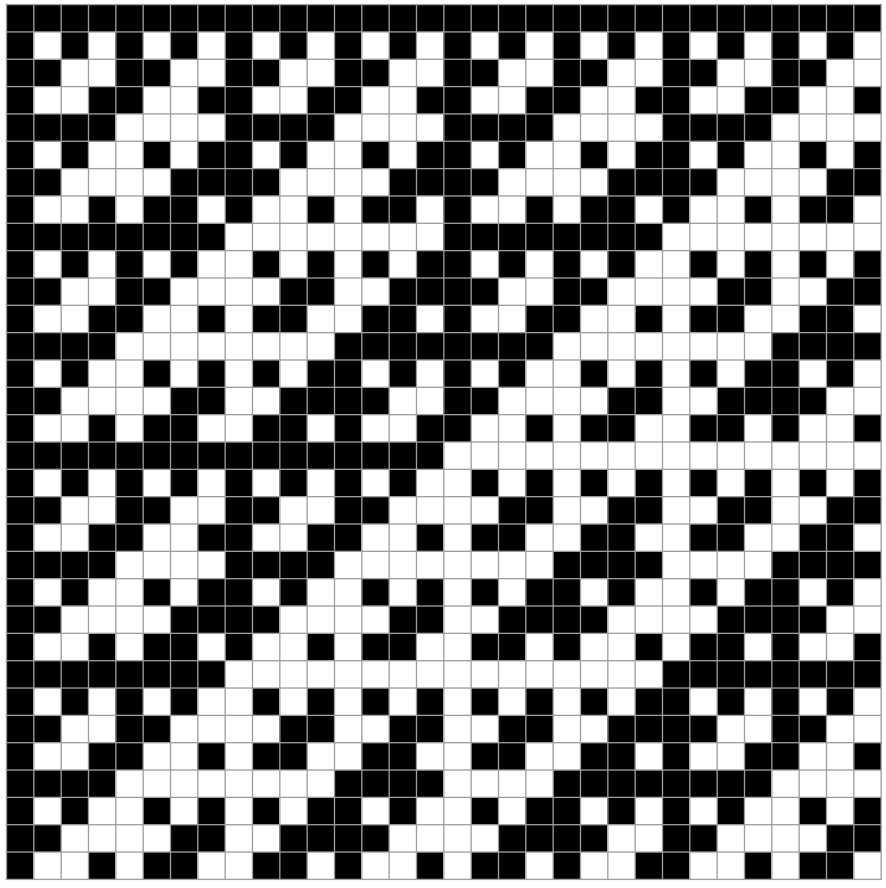


Mariner 9 Image: Improvement Due to Error Correction



The Mariner 9 Code

32 of the 64 Mariner 9 codewords: ■ = 0 and □ = 1. Suppose we receive the word below. How should we decide which codeword was sent? Comparing with all 64 codewords takes some time . . .



MT361/MT461/MT5461
Error Correcting Codes

Mark Wildon, mark.wildon@rhul.ac.uk

MT361/MT461/MT5461
Error Correcting Codes

Mark Wildon, mark.wildon@rhul.ac.uk

Administration

- Reminder: this Thursday the extra lecture for MSc/MSci students is moved to 2pm at Friday in C219.
- All future Tuesday lectures will, like this one, be in QBLT.
- If you left a pair of gloves in my office, they can be collected from the Maths Office (243). [Also your suggested first weighing for the coins problem does lead to a three-question strategy.]
- Answers to the Preliminary Sheet are on Moodle. Please take a copy of Sheet 1. You are welcome to discuss any problem sheets with me in office hours.

§2 Hamming Distance and Nearest Neighbour Decoding

Definition 2.1

Let A be an alphabet. Let $u, v \in A^n$ be words of length n. The *Hamming distance* between u and v, denoted $d(u, v)$, is the number of positions in which u and v are different.

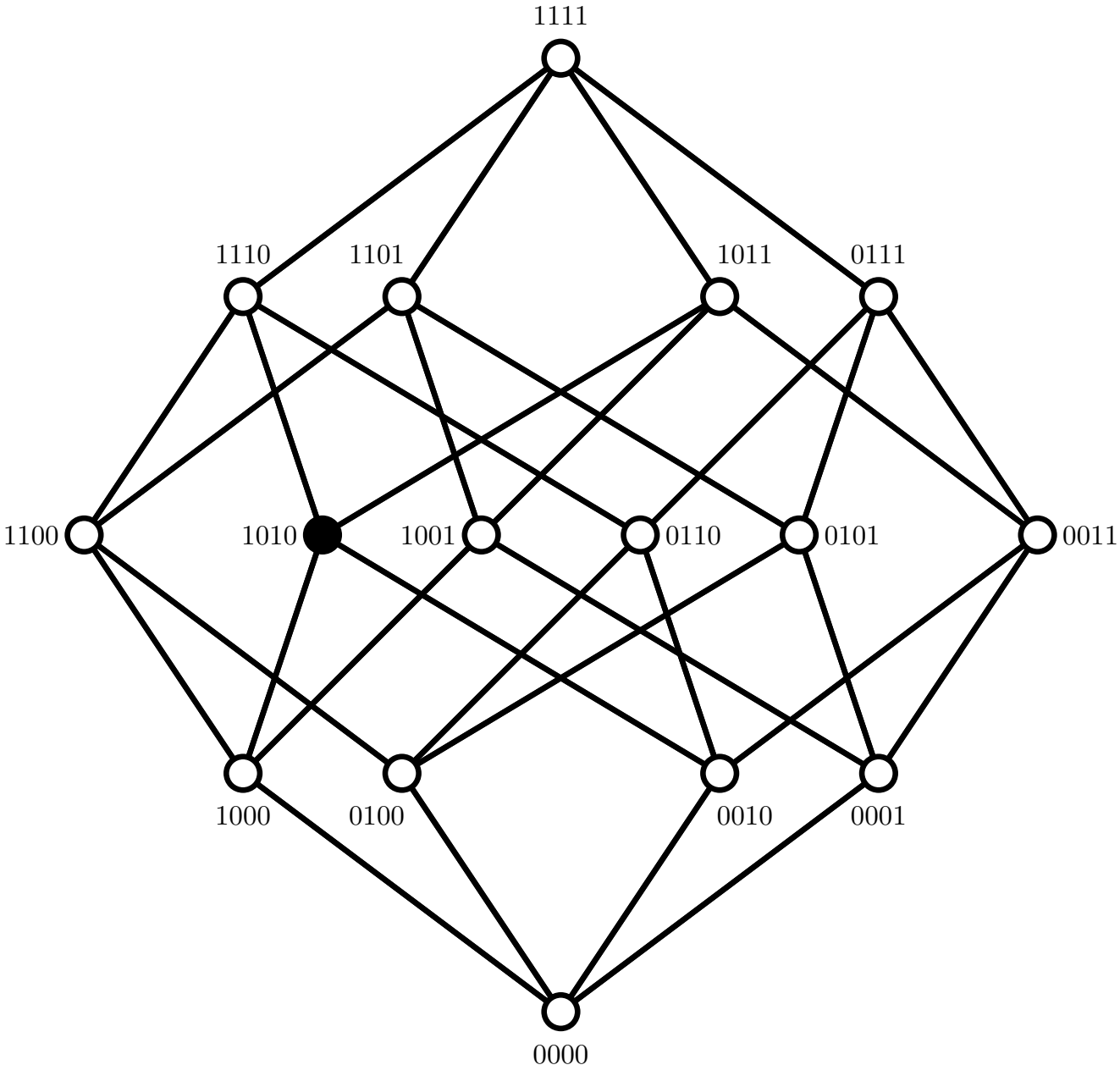
Example 2.2

Working with binary words of length 4, we have

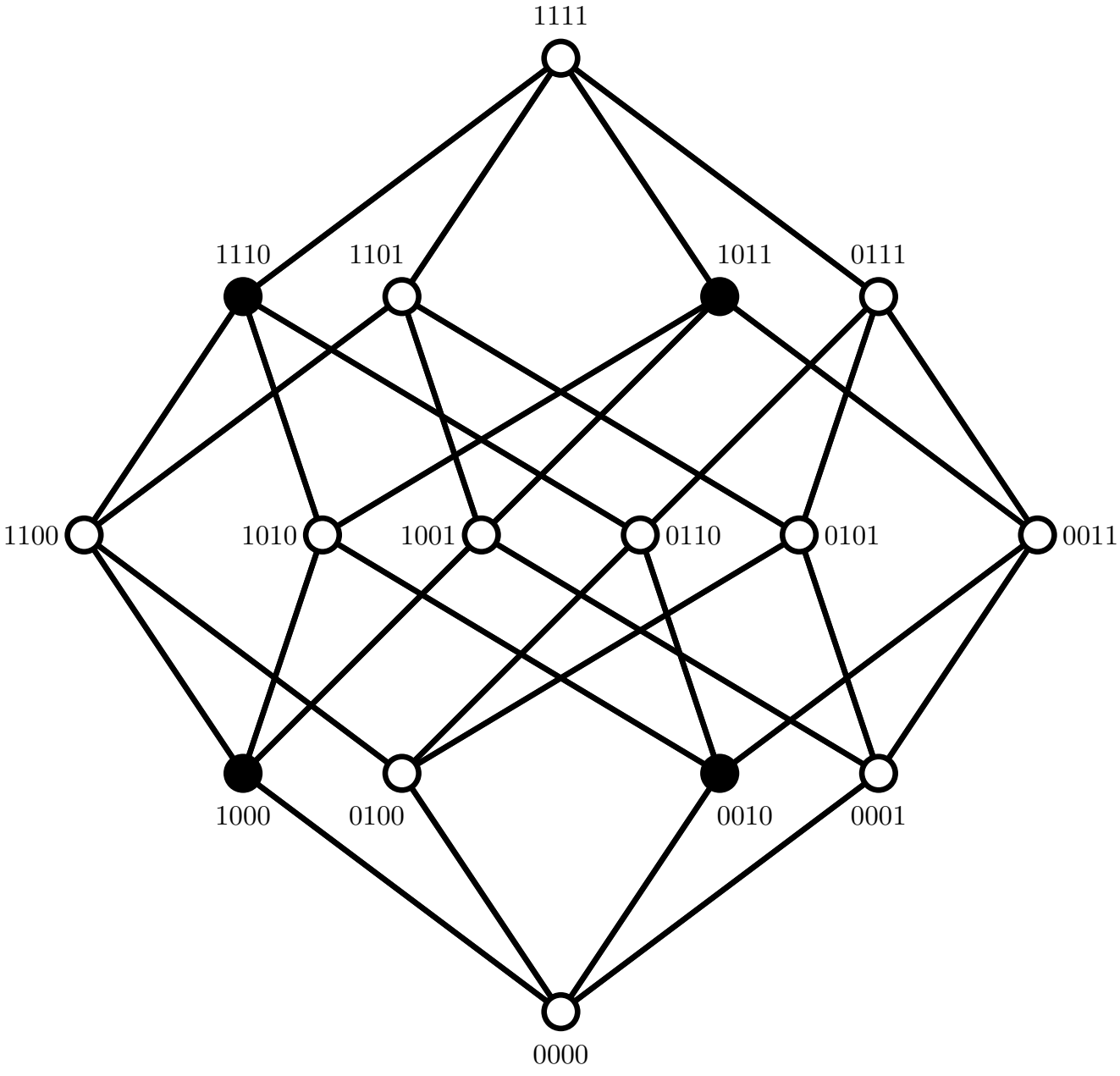
$$d(0011, 1101) = 3$$

because the words 0011 and 1101 differ in their first three positions, and are the same in their final position. Working with words over the alphabet $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \ldots, \mathrm{X}, \mathrm{Y}, \mathrm{Z}\}$ we have $d(\mathrm{TALE}, \mathrm{TAKE}) = 1$ and $d(\mathrm{TALE}, \mathrm{TILT}) = 2$.

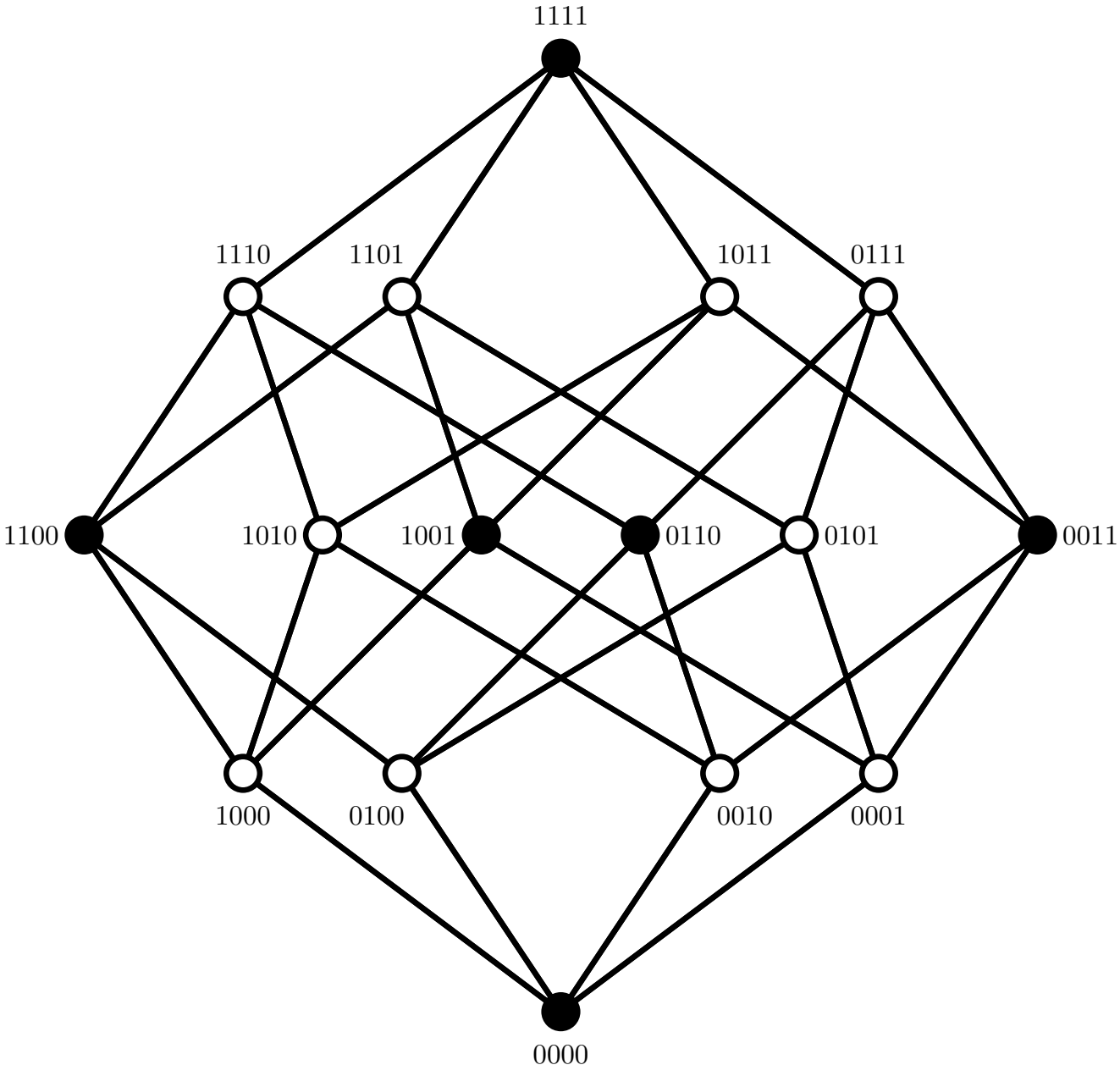
$B_0(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

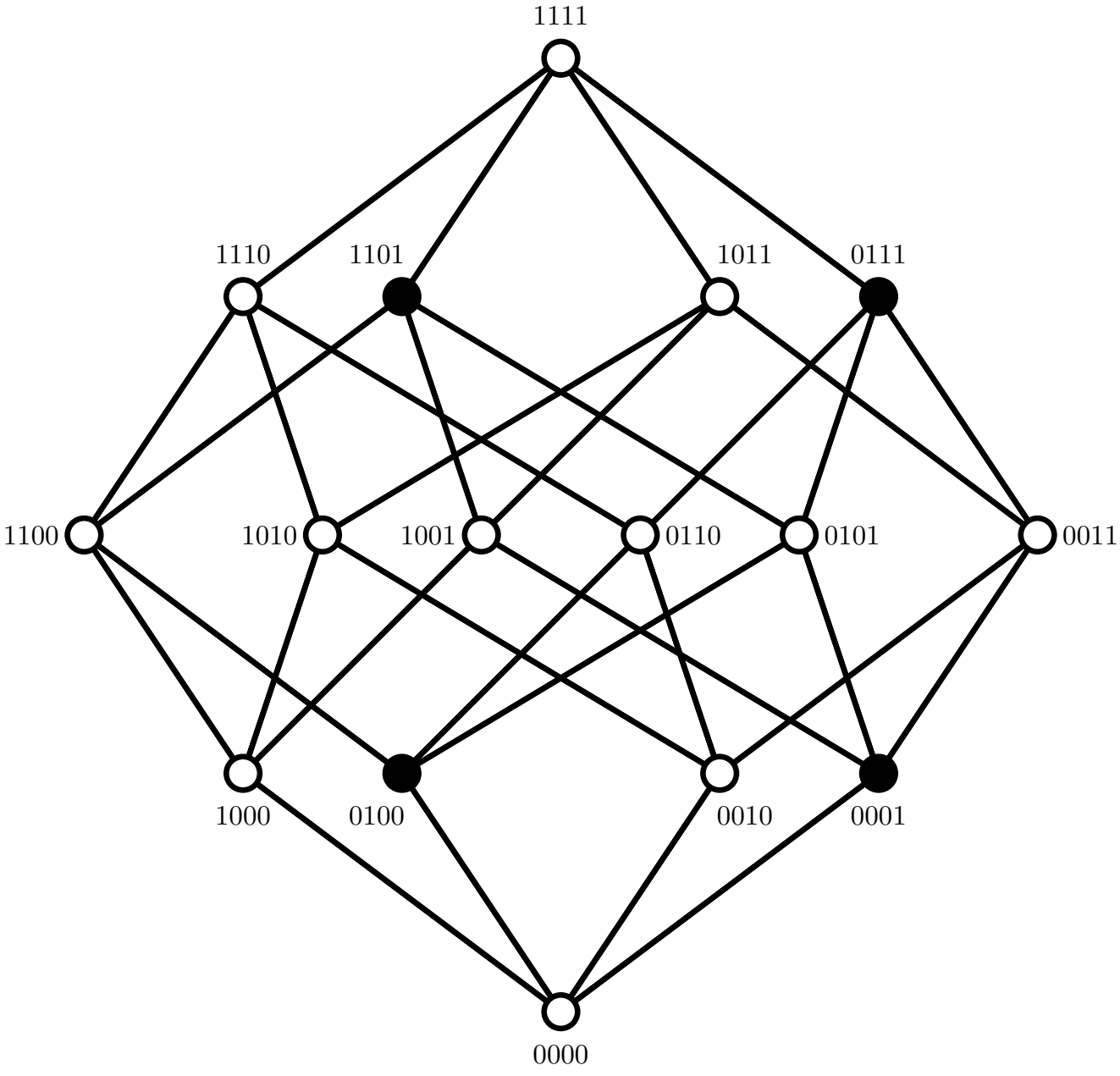
$B_1(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

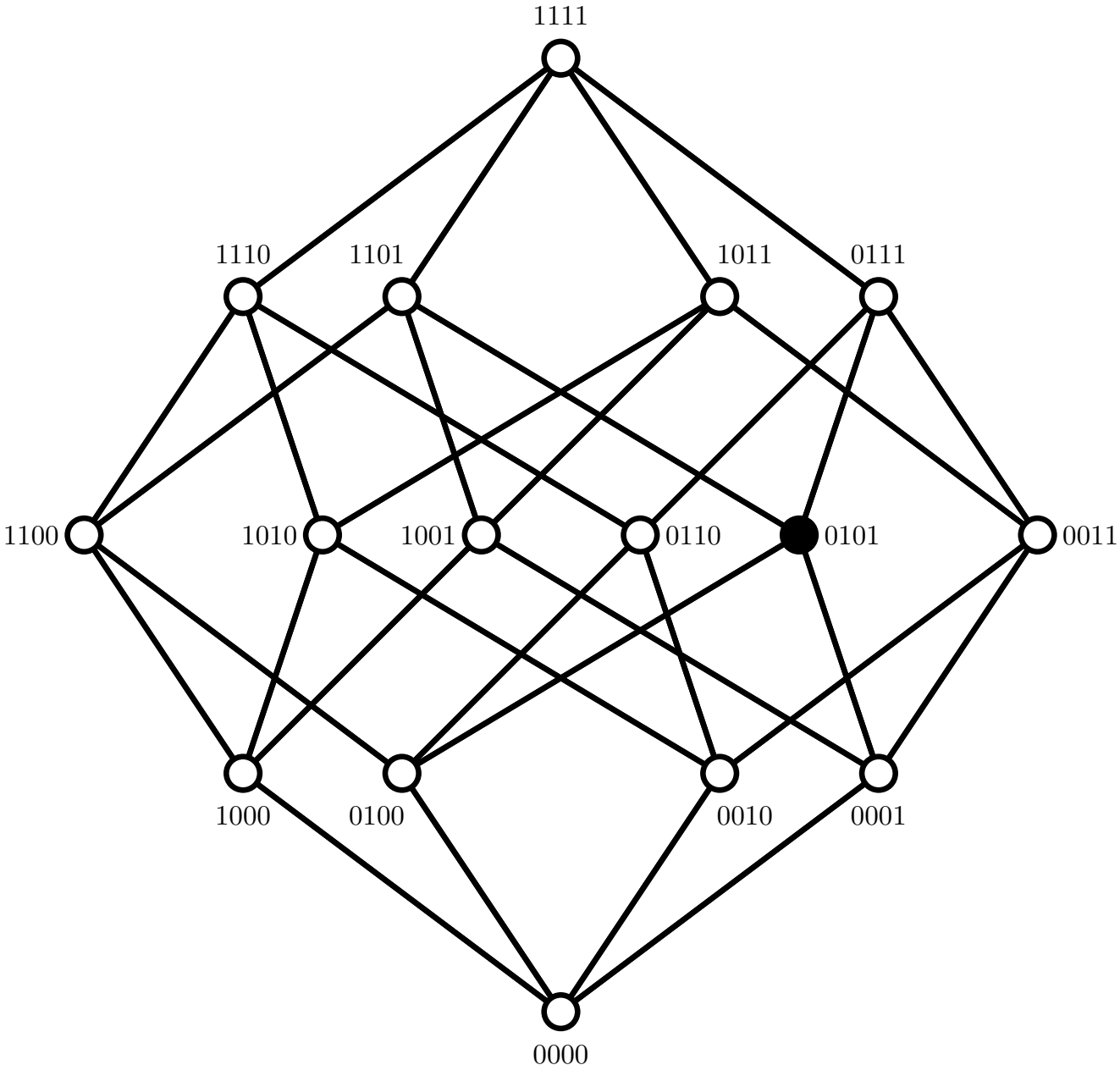
$B_2(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

$B_3(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

$B_4(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

Hamming distance

Theorem 2.3

Let A be a q-ary alphabet and let u, v, w be words over A of length n.

(i) $d(u, v) = 0$ *if and only if* $u = v$;

(ii) $d(u, v) = d(v, u)$;

(iii) $d(u, w) \leq d(u, v) + d(v, w)$.

Exercise: Find all English words v such that

$$d(\mathrm{WARM}, v) = d(\mathrm{COLD}, v) = 2.$$

Check that the triangle inequality holds when u, v, w are WARM, WALL and COLD, respectively.

Repetition Codes

Definition 2.4 (Repetition codes)

Let $n \in \mathbf{N}$ and let A be a q-ary alphabet with $q \geq 2$. The *repetition code* of length n over A has as its codewords all words of length n of the form

$$(s, s, \ldots, s)$$

where $s \in A$. The *binary repetition code* of length n is the repetition code of length n over the binary alphabet $\{0, 1\}$.

Definition 2.5 (Nearest neighbour decoding)

Let C be a code. Suppose that a codeword is sent through the channel and we receive the word v. To decide v using *nearest neighbour decoding* look at all the codewords of C and pick the one that is nearest, in Hamming distance to v. If there is no unique nearest codeword to v, then we say that nearest neighbour decoding *fails*.

Example 2.6

An internal review of the code in Example 1.8 has uncovered several deficiencies. The new proposal uses a ternary repetition code of length 6 over the alphabet $\{0, 1, 2\}$. The new encoder is

$$\text{'Stand-down'} \xmapsto{\text{encoded as}} 000000$$

$$\text{'Stay on your toes'} \xmapsto{\text{encoded as}} 111111$$

$$\text{'Launch nukes'} \xmapsto{\text{encoded as}} 222222.$$

Suppose ALICE sends 'Stand-down'. If BOB receives 001102, then

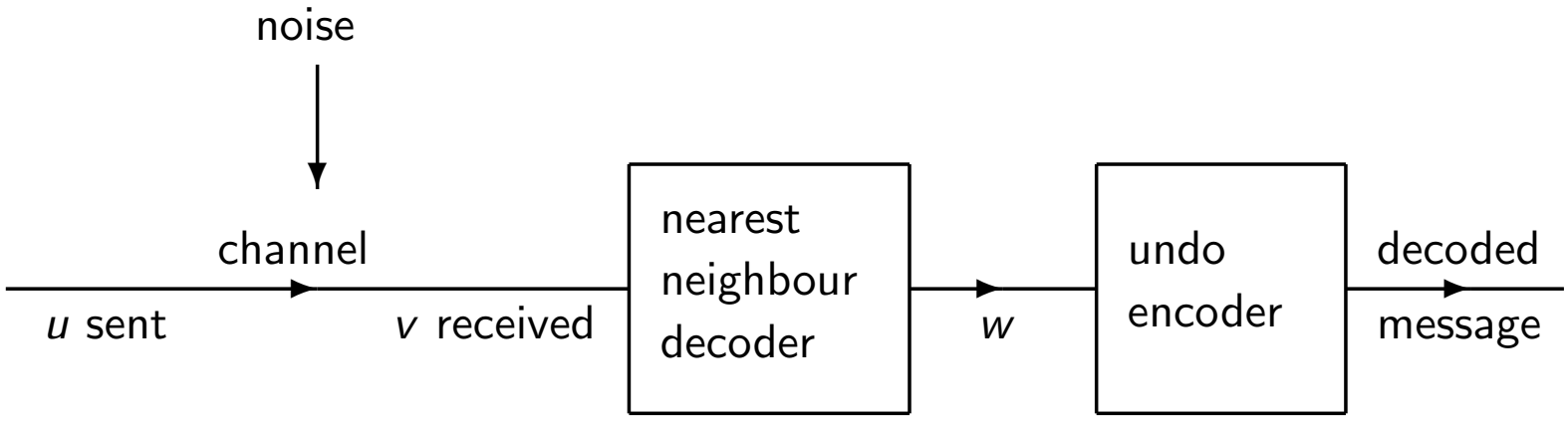
$$d(001102, 000000) = 3, d(001102, 111111) = 4, d(001102, 222222) = 5,$$

so under nearest neighbour decoding, 001102 is decoded as 000000, which in turn BOB will decode as 'Stand-down'.

- What happens if 000111 is received?
- What if 020222 is received?

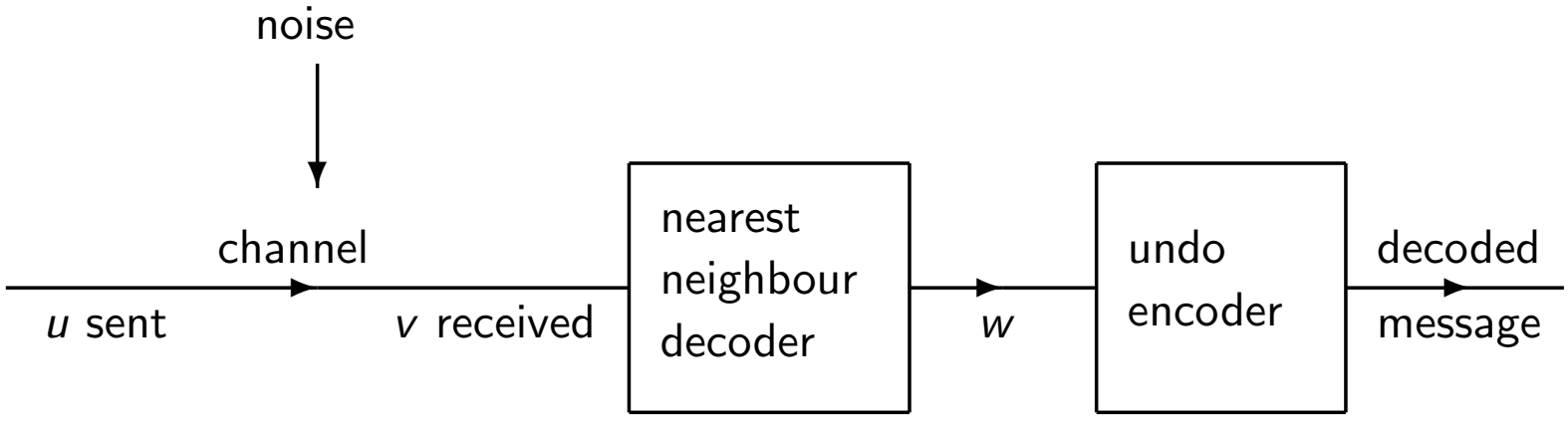
Nearest Neighbour Decoding as Part of a Two-step Process

Nearest neighbour decoding is only one step in the decoding process. To recover the sent word, the decoder must take the codeword given by nearest neighbour decoding and then undo the encoder by translating the codeword back into a message.



Nearest Neighbour Decoding as Part of a Two-step Process

Nearest neighbour decoding is only one step in the decoding process. To recover the sent word, the decoder must take the codeword given by nearest neighbour decoding and then undo the encoder by translating the codeword back into a message.



The decoded message is correct if and only if $w = u$. This is the case if and only if u is the unique nearest codeword to v, i.e.

$$d(u, v) < d(u', v)$$

for all codewords u'.

Nearest Neighbour Decoding: when failure is not an option

In practice it might be essential that the decoder always gives some result, even if nearest neighbour decoding has failed. Then the decoder will have to make an arbitrary choice between two or more codewords that are equally likely to be the sent word. Mathematically it is much better just to report that nearest neighbour decoding has failed.

Quiz on Nearest Neighbour Decoding

Let C be the ternary code of length 4 with codewords

0000 0111 0222 1012 1120 1201 2021 2102 2210

The construction of this code using orthogonal Latin squares will be seen later in the course. It has many interesting properties: in particular, if $u, u' \in C$ are distinct then $d(u, u') = 3$.

Please decode the received words (a) 2222, (b) 1201, (c) 2121 using nearest neighbour decoding. In each case write down one of the next closest codewords, and determine the minimum number of errors that must have occurred in the channel if your answer is not the sent codeword.

Quiz on Nearest Neighbour Decoding

Let C be the ternary code of length 4 with codewords

0000 0111 0222 1012 1120 1201 2021 2102 2210

The construction of this code using orthogonal Latin squares will be seen later in the course. It has many interesting properties: in particular, if $u, u' \in C$ are distinct then $d(u, u') = 3$.

Please decode the received words (a) 2222, (b) 1201, (c) 2121 using nearest neighbour decoding. In each case write down one of the next closest codewords, and determine the minimum number of errors that must have occurred in the channel if your answer is not the sent codeword.

(a) 0222, distance 2 from 2021, 2102, 2210, if wrong 2 errors;

Quiz on Nearest Neighbour Decoding

Let C be the ternary code of length 4 with codewords

0000 0111 0222 1012 1120 1201 2021 2102 2210

The construction of this code using orthogonal Latin squares will be seen later in the course. It has many interesting properties: in particular, if $u, u' \in C$ are distinct then $d(u, u') = 3$.

Please decode the received words (a) 2222, (b) 1201, (c) 2121 using nearest neighbour decoding. In each case write down one of the next closest codewords, and determine the minimum number of errors that must have occurred in the channel if your answer is not the sent codeword.

(a) 0222, distance 2 from 2021, 2102, 2210, if wrong 2 errors;

(b) 1201, distance 3 from any other codeword, if wrong 3 errors;

Quiz on Nearest Neighbour Decoding

Let C be the ternary code of length 4 with codewords

0000 0111 0222 1012 1120 1201 2021 2102 2210

The construction of this code using orthogonal Latin squares will be seen later in the course. It has many interesting properties: in particular, if $u, u' \in C$ are distinct then $d(u, u') = 3$.

Please decode the received words (a) 2222, (b) 1201, (c) 2121 using nearest neighbour decoding. In each case write down one of the next closest codewords, and determine the minimum number of errors that must have occurred in the channel if your answer is not the sent codeword.

(a) 0222, distance 2 from 2021, 2102, 2210, if wrong 2 errors;

(b) 1201, distance 3 from any other codeword, if wrong 3 errors;

(c) 2021, distance 2 from 0111, 1120, 2102, if wrong 2 errors.

Probabilistic Justification for Nearest Neighbour Decoding

Lemma 2.7

Let C be a binary code of length n used to communicate on a binary symmetric channel with cross-over probability p. Suppose that the codeword $u \in C$ is sent. If v is a word of length n, then the probability that v is received $p^{d(u,v)}(1-p)^{n-d(u,v)}$. **[corrected to $n - d(u,v)$ from $1 - d(u,v)$ on 17th January]**

In symbols, we write
$\mathbf{P}[v \text{ received} \mid u \text{ sent}] = p^{d(u,v)}(1-p)^{n-d(u,v)}$. For example, if C is the binary repetition code of length 3, then according to Lemma 2.7,

$$\mathbf{P}[001 \text{ received} \mid 111 \text{ sent}] = p^2(1-p)$$
$$\mathbf{P}[000 \text{ received} \mid 111 \text{ sent}] = p^3.$$

These agree with the calculations in Example 1.2.

Probabilistic Justification for Nearest Neighbour Decoding

Theorem 2.8

Suppose that we use a binary code C of length n to send messages through the binary symmetric channel **with crossover probability** *p, and that each codeword in C is equally likely to be sent. Suppose we receive a binary word v. For each $u \in C$,*

$$\mathbf{P}[u \text{ sent} \mid v \text{ received}] = p^{d(u,v)}(1-p)^{n-d(u,v)}C(v)$$

where $C(v)$ does not depend on u. Hence $\mathbf{P}[u \text{ sent} \mid v \text{ received}]$ is maximized by choosing u to be the nearest codeword to v.

Probabilistic Justification for Nearest Neighbour Decoding

Theorem 2.8

Suppose that we use a binary code C of length n to send messages through the binary symmetric channel **with crossover probability** *p, and that each codeword in C is equally likely to be sent. Suppose we receive a binary word v. For each $u \in C$,*

$$\mathbf{P}[u \text{ sent} \mid v \text{ received}] = p^{d(u,v)}(1-p)^{n-d(u,v)}C(v)$$

where $C(v)$ does not depend on u. Hence $\mathbf{P}[u \text{ sent} \mid v \text{ received}]$ is maximized by choosing u to be the nearest codeword to v.

Thus, provided we accept that maximizing $\mathbf{P}[u \text{ sent} \mid v \text{ received}]$ is a good idea, we are inevitably led to nearest neighbour decoding. The syllabus talks only about 'probability calculations', so while interesting, Theorem 2.8 may be regarded as non-examinable.

Proof of Theorem 2.8 (non-examinable)

By the argument used in Bayes' Law we have

$$\begin{aligned}\mathbf{P}[u \text{ sent} \mid v \text{ received}] &= \mathbf{P}[u \text{ sent and } v \text{ received}]\mathbf{P}[v \text{ received}] \\ &= \frac{\mathbf{P}[v \text{ sent and } u \text{ received}]}{\mathbf{P}[u \text{ sent}]}\mathbf{P}[v \text{ received}].\end{aligned}$$

Now

- $\mathbf{P}[u \text{ sent}] = 1/|C|$
- $\mathbf{P}[v \text{ sent and } u \text{ received}] = p^{d(u,v)}(1-p)^{n-d(u,v)}$ by Lemma 2.7.

So if we define $C(v) = \dfrac{1}{\mathbf{P}[v \text{ received}]}$ then we have

$$\mathbf{P}[u \text{ sent} \mid v \text{ received}] = p^{d(u,v)}(1-p)^{n-d(u,v)}C(v)$$

as required.

Example: Different Transmission Probabilities

Suppose that Alice sends 111 with probability 0.999 and 000 with probability 0.001, over a binary symmetric channel with crossover probability 0.2. If you receive 000, what do you think Alice sent?

$$\begin{aligned}
&\mathbf{P}[111 \text{ sent} \mid 000 \text{ received}] \\
&\qquad = \mathbf{P}[111 \text{ sent and } 000 \text{ received}]\mathbf{P}[000 \text{ received}] \\
&\qquad = \frac{\mathbf{P}[000 \text{ received} \mid 111 \text{ sent}]}{\mathbf{P}[111 \text{ sent}]}\mathbf{P}[000 \text{ received}] \\
&\qquad = \frac{(0.2)^3 \times 0.999}{\mathbf{P}[000 \text{ received}]} \\
&\qquad = \frac{0.007992}{\mathbf{P}[000 \text{ received}]}
\end{aligned}$$

and similarly

$$\mathbf{P}[000 \text{ sent} \mid 000 \text{ received}] = \frac{(0.8)^3 \times 0.001}{\mathbf{P}[000 \text{ received}]} = \frac{0.000512}{\mathbf{P}[000 \text{ received}]}.$$

So even if you receive 000, it is most likely that Alice sent 111. In fact, it is over 15 times more likely that Alice sent 111 than 000.

Binary Parity Check Codes

Example 2.9 (Parity check codes)

Let $n \in \mathbf{N}$ and let C be the binary code consisting of all binary words of length n. Let C_{ext} be the code of length $n+1$ whose codewords are obtained by appending an extra bit to each codeword in C, so that the total number of 1s in each codeword is even.

Binary Parity Check Codes

Example 2.9 (Parity check codes)

Let $n \in \mathbf{N}$ and let C be the binary code consisting of all binary words of length n. Let C_{ext} be the code of length $n+1$ whose codewords are obtained by appending an extra bit to each codeword in C, so that the total number of 1s in each codeword is even.

Suppose that a codeword $u \in C_{\text{ext}}$ is sent through the channel and the word v is received. If a single bit is corrupted, so $d(u, v) = 1$, then v must have an odd number of 1s. Hence $v \notin C_{\text{ext}}$ and we detect that an error has occurred.

This shows that if u, $u' \in C_{\text{ext}}$ are distinct codewords then $d(u, u') \geq 2$.

Example 2.10 (ISBN-10 code)

All recent books have an International Standard Book Number (ISBN) assigned by the publisher. In this scheme, each book is assigned a codeword of length 10 over the 11-ary alphabet $\{0, 1, 2, \ldots, 9, \mathrm{X}\}$.

For example, [5] in the list of recommended reading has ISBN

0-444-85193-3.

- 0 identifies the country of publication;
- 444 identifies the publisher;
- 85193 is the item number assigned by the publisher;
- 3 is the *check digit*.

The check digit is chosen so that if $u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10}$ is an ISBN then

$$\sum_{j=1}^{10} (11 - j) u_j = 10u_1 + 9u_2 + \cdots + 2u_9 + u_{10} \equiv 0 \bmod 11.$$

ISBNs continued

We will say that $u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10}$ is an *ISBN* if it satisfies the check condition above, ignoring the question of whether it was ever assigned to a book.

Lemma 2.11

If a single error is made when writing down an ISBN, the result is not an ISBN.

ISBNs continued

We will say that $u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10}$ is an *ISBN* if it satisfies the check condition above, ignoring the question of whether it was ever assigned to a book.

Lemma 2.11

If a single error is made when writing down an ISBN, the result is not an ISBN.

The ISBN code also detects when two unequal adjacent symbols are swapped. (See Question 5 on Sheet 2.) This is a sort of error likely to be made by a busy person. However it does not detect all errors involving two symbols. For example, starting from 0000000000 we can change two symbols to get 1000000001, which is also an ISBN.

Exercise: In the next section we will see the accepted definitions of what it means for a code C to be t-error detecting or t-error correcting. From the examples you have seen so far, how would you define these terms?

Administration

- Please take the next installment in the handout (pages 21 to 26).
- Please take Problem Sheet 2.
- Reminder that Problem Sheet 1 is due in by 11am Thursday.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error detecting if ...

- it can detect when there are precisely t errors.
- it can detect up to t errors.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error detecting if ...

- it can detect when there are precisely t errors.
- it can detect up to t errors.

What do you mean by 'detect'?
A definition should relate to things already defined!

- t errors occuring in the channel will result in a word $v \notin C$ being received.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error detecting if ...

- it can detect when there are precisely t errors.
- it can detect up to t errors.

What do you mean by 'detect'?
A definition should relate to things already defined!

- t errors occuring in the channel will result in a word $v \notin C$ being received.

Perfect, if change to 'up to t errors'

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error detecting if ...

- it can detect when there are precisely t errors.
- it can detect up to t errors.

What do you mean by 'detect'?
A definition should relate to things already defined!

- t errors occuring in the channel will result in a word $v \notin C$ being received.

Perfect, if change to 'up to t errors'

- for any codewords $u, v \in C$, $d(u, v) > t$.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error detecting if ...

- it can detect when there are precisely t errors.
- it can detect up to t errors.

What do you mean by 'detect'?
A definition should relate to things already defined!

- t errors occuring in the channel will result in a word $v \notin C$ being received.

Perfect, if change to 'up to t errors'

- for any codewords $u, v \in C$, $d(u, v) > t$.

Equivalent to accepted definition

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error correcting if ...

- it can correct precisely t errors.
- it can detect and correct up to t errors.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error correcting if ...

- it can correct precisely t errors.
- it can detect and correct up to t errors.

What do you mean by 'correct'?

- We can deduce the input from the output containing at most t errors.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error correcting if ...

- it can correct precisely t errors.
- it can detect and correct up to t errors.

What do you mean by 'correct'?

- We can deduce the input from the output containing at most t errors.

What counts as an allowable way to deduce the input?

- Can detect p-errors and provide enough information for the errors to be resolved without resending of message.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error correcting if ...

- it can correct precisely t errors.
- it can detect and correct up to t errors.

What do you mean by 'correct'?

- We can deduce the input from the output containing at most t errors.

What counts as an allowable way to deduce the input?

- Can detect p-errors and provide enough information for the errors to be resolved without resending of message.

'Enough information' seems a bit vague

- When u is sent and [up to] t errors occur and v is received such that $v \notin C$ then we know which t bits have flipped.

Definitions of t-Error Detecting and t-Error Correctng

Let C be a code of length n over an alphabet A. Let $t \in \mathbf{N}$.

Definition. The code C is t-error correcting if ...

- it can correct precisely t errors.
- it can detect and correct up to t errors.

What do you mean by 'correct'?

- We can deduce the input from the output containing at most t errors.

What counts as an allowable way to deduce the input?

- Can detect p-errors and provide enough information for the errors to be resolved without resending of message.

'Enough information' seems a bit vague

- When u is sent and [up to] t errors occur and v is received such that $v \notin C$ then we know which t bits have flipped.

How do we know? (Also this assumes a binary code.)

t-Error Detecting and t-Error Correcting Codes

Definition 3.1

Let C be a code of length n over an alphabet A and let $t \in \mathbf{N}$. We say that C is

- *t-error detecting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $v \neq u$ and $d(u, v) \leq t$, then $v \notin C$.
- *t-error correcting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $d(u, v) \leq t$, then v is decoded to u using nearest neighbour decoding.

t-Error Detecting and t-Error Correcting Codes

Definition 3.1

Let C be a code of length n over an alphabet A and let $t \in \mathbf{N}$. We say that C is

- *t-error detecting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $v \neq u$ and $d(u, v) \leq t$, then $v \notin C$.
- *t-error correcting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $d(u, v) \leq t$, then v is decoded to u using nearest neighbour decoding.

Equivalently, a code C is t-error detecting if if whenever a codeword is sent, and **between** 1 **and** t **errors occur**, the received word is not a codeword, and so the receiver will know that something has gone wrong in the channel.

Remarks 3.2

(1) Recall that when there is no unique nearest codeword to u then nearest neighbour decoding fails. So a code C is t-error correcting if and only if whenever v is a word within distance t of a codeword $u \in C$ then

$$d(u, v) < d(u', v) \quad \text{for all } u' \in C \text{ with } u' \neq u.$$

This gives a more abstract way to state the definition of t-error correcting that does not mention nearest neighbour decoding.

(2) It may seem a bit odd to say that a code is 't-error correcting' when it is the decoder (be it a human or a computer) that has to do all the work of decoding. Moreover, we have seen in Example 1.7 that the same code can reasonably be used with different decoders. A code that is t-error correcting promises to be able to correct up to t-errors *provided nearest neighbour decoding is used.*

Remarks 3.2 [continued]

(3) In both definitions we have $d(u, v) \leq t$, so v is obtained from u by changing **up to** t positions. Hence if $s < t$ then a t-error detecting code is also s-error detecting, and a t-error correcting code is also s-error correcting.

If instead we required **exactly** t changes then, by Question 11 on Sheet 1, there would be codes that are 2-error detecting but not 1-error detecting. This could be confusing in practical applications. Mathematically, it would lead to theorems with long-winded hypotheses of the form 'Suppose C is a code that is t-error detecting for all $t \leq c$, ...'. Both are undesirable.

Earlier Examples

We will now show that Definition 3.1 agrees with our findings in Examples 2.6 and 2.9.

Lemma 3.3

Let $n \in \mathbf{N}$. Let C be the repetition code of length n over a q-ary alphabet A, where $q \geq 2$.

(i) *C is $(n-1)$-error detecting but not n-error detecting.*

(ii) *If $n = 2m+1$ then C is m-error correcting but not $(m+1)$-error correcting.*

(iii) *If $n = 2m$ then C is $(m-1)$-error correcting, but not m-error correcting.*

The proof of part (iii) is left to you in Question 2 of Sheet 2.

Lemma 3.4

Let $n \in \mathbf{N}$ and let C_{ext} be the binary parity check code of length $n+1$ defined in Example 2.9. Then C_{ext} is 1-error detecting but not 2-error detecting. It is not 1-error correcting.

ISBNs

We will say that $u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10}$ is an *ISBN* if it satisfies the check condition

$$\sum_{j=1}^{10} (11 - j) u_j = 10u_1 + 9u_2 + \cdots + 2u_9 + u_{10} \equiv 0 \bmod 11,$$

ignoring the question of whether it was ever assigned to a book. (If we need $u_{10} = 10$ then put X instead of 10 in the last position.)

Lemma 2.11

If a single error is made when writing down an ISBN, the result is not an ISBN.

ISBNs

We will say that $u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}$ is an *ISBN* if it satisfies the check condition

$$\sum_{j=1}^{10}(11-j)u_j = 10u_1 + 9u_2 + \cdots + 2u_9 + u_{10} \equiv 0 \bmod 11,$$

ignoring the question of whether it was ever assigned to a book. (If we need $u_{10} = 10$ then put X instead of 10 in the last position.)

Lemma 2.11

If a single error is made when writing down an ISBN, the result is not an ISBN.

The ISBN code also detects when two unequal adjacent symbols are swapped. (See Question 5 on Sheet 2.) This is a sort of error likely to be made by a busy person. However it does not detect all errors involving two symbols.

Lemma 3.5

The ISBN code is 1*-error detecting but not* 2*-error detecting. It is not even* 1*-error correcting.*

Administration

- Please take your work for Sheet 1: surnames A–J in red folder, K–Z in blue folder.
- Error in Question 4 of Sheet 2: please change $\{0,1,2\}^7$ to $\{0,1,2\}^6$, so reads

  In the ternary repetition code of length 6 *(considered in Example 2.6), what is the maximum distance of a word* $v \in \{0,1,2\}^{\mathbf{6}}$ *from a codeword?*

- Error in Question 6 of Sheet 2: please change w to v, so reads

  Let C *be a* t*-error correcting code over the alphabet* A*. Suppose that you receive the word* $v \in A^n$ *and, after some hunting through the code, you find a codeword* $u \in C$ *such that* $d(u, \mathbf{v}) \leq t$, ...

§4 Minimum Distance and Hamming Balls

Question 1 on Sheet 2 shows that if C is a code such that $d(u, u') \geq 3$ for all distinct u, $u' \in C$, then C is 2-error detecting and 1-error correcting. This is a special case of a very useful general result. We need the following definition.

Definition 4.1

Let C be a code. The *minimum distance* of C, denoted $d(C)$, is defined by $d(C) = \min\{d(u, w) : u, w \in C, u \neq w\}$.

By Definition 1.4, any code has at least two codewords, so the minimum distance of a code is always well-defined.

Example on Minimum Distance

Example 4.2

Here is an example from Hamming's original paper (see reference on page 3 of the printed notes).

Let C be the binary code of length 3 with codewords

$$001, \quad 010, \quad 100, \quad 111$$

as seen on Sheet 1, Question 2.

Then $d(u, w) = 2$ for all distinct u and w in C, so $d(C) = 2$. If we adjoin 000 as another codeword then the minimum distance goes down to 1 since $d(000, 001) = 1$.

More Examples of Minimum Distance

It is not hard to find the minimum distance of the codes seen in the examples so far.

Lemma 4.3

Let $n \in \mathbf{N}$.

(i) *The minimum distance of any length n repetition code is n.*

(ii) *The minimum distance of the length $n+1$ binary parity check code C_{ext} in Example 2.9 is 2.*

(iii) *The minimum distance of the square code is 3.*

More Examples of Minimum Distance

It is not hard to find the minimum distance of the codes seen in the examples so far.

Lemma 4.3

Let $n \in \mathbf{N}$.

(i) *The minimum distance of any length n repetition code is n.*

(ii) *The minimum distance of the length $n+1$ binary parity check code C_{ext} in Example 2.9 is 2.*

(iii) *The minimum distance of the square code is 3.*

Exercise: Let C be a code. What do you think is the relationship between the maximum t for which C is t-error detecting and the minimum distance of C? Now think about the same question with 'error detecting' replaced with 'error correcting'.

Reminder of Definition 3.1

Definition 3.1

Let C be a code of length n over an alphabet A and let $t \in \mathbf{N}$. We say that C is

- *t-error detecting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $v \neq u$ and $d(u, v) \leq t$, then $v \notin C$.
- *t-error correcting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $d(u, v) \leq t$, then v is decoded to u using nearest neighbour decoding.

Reminder of Definition 3.1

Definition 3.1

Let C be a code of length n over an alphabet A and let $t \in \mathbf{N}$. We say that C is

- *t-error detecting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $v \neq u$ and $d(u, v) \leq t$, then $v \notin C$.
- *t-error correcting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $d(u, v) \leq t$, then v is decoded to u using nearest neighbour decoding.

We observed in Remark (3.2)(1) that nearest neighbour decoding successfully decodes v to u if and only if u is the unique closest codeword to v, i.e.

$$d(u, v) < d(u', v) \quad \text{for all } u' \in C \text{ with } u' \neq u.$$

Code	Minimum distance	Maximum t s.t. t-error detecting	Maximum t s.t. t-error correcting
Binary parity check code of length 5	2	1	none
Ternary repetition code of length 6	6	5	2
Ternary repetition code of length 7	7	6	3
Square code	3	2	1
$\{u_1u_2u_3\ u_1u_2u_3\ u_1u_2u_3 : u_1, u_2, u_3 \in \{0,1\}\}$	3	2	1

Code	Minimum distance	Maximum t s.t. t-error detecting	Maximum t s.t. t-error correcting
Binary parity check code of length 5	2	1	none
Ternary repetition code of length 6	6	5	2
Ternary repetition code of length 7	7	6	3
Square code	3	2	1
$\{u_1u_2u_3\ u_1u_2u_3\ u_1u_2u_3 : u_1, u_2, u_3 \in \{0,1\}\}$	3	2	1
Code of length 9 in Q3 Sheet 2	3	2	1
Any code with minimum distance ≥ 3 (by Q1 Sheet 2)	≥ 3	≥ 2	≥ 1

Minimum Distance Controls Error Detection and Correction

Theorem 4.5

Let C be a code with minimum distance d. Let $t \in \mathbf{N}$.

(i) *C is t-error detecting* $\iff$ $t \leq d - 1$;

(ii) *C is t-error correcting* $\iff$ $2t \leq d - 1$.

It is usually easier to compute the minimum distance of a code than it is to determine (without using any other results) the maximum t for which it is t-error detecting or correcting. This makes the '$\Longleftarrow$' directions in Theorem 4.5 very useful.

Minimum Distance Controls Error Detection and Correction

Theorem 4.5

Let C be a code with minimum distance d. Let $t \in \mathbf{N}$.

(i) *C is t-error detecting* $\iff$ $t \leq d-1$;

(ii) *C is t-error correcting* $\iff$ $2t \leq d-1$.

It is usually easier to compute the minimum distance of a code than it is to determine (without using any other results) the maximum t for which it is t-error detecting or correcting. This makes the '$\Longleftarrow$' directions in Theorem 4.5 very useful.

Recall that if $x \in \mathbf{R}$ then $\lfloor x \rfloor$ is the greatest integer n such that $n \leq x$. For instance $\lfloor 2 \rfloor = \lfloor 2\frac{1}{2} \rfloor = 2$.

Corollary 4.6

A code of minimum distance d is $(d-1)$-error detecting and $\lfloor \frac{d-1}{2} \rfloor$-error correcting.

Question 4 on Sheet 2

> *In the ternary repetition code of length* 6 *(considered in Example 2.6), what is the maximum distance of a word* $v \in \{0,1,2\}^6$ *from a codeword?*
>
> **Clarification:** v **could be any word, and the 'distance of** v **from a codeword' means the minimum of** $d(v, 000000), d(v, 111111), d(v, 222222)$**.**

Maybe a better way to state the question:

> *Suppose you receive* $v \in \{0,1,2\}^6$*. What is the maximum number of changes you might need to make to* v *to turn* v *into one of the codewords in the ternary repetition code?*

For example, if $v = 001112$ then the closest codeword is 111111; 3 changes are needed since $d(111111, v) = 3$. But there are words that are even further away from *all* the codewords.

Table from Hamming's Original Paper

Corollary 4.6

A code of minimum distance d is $(d-1)$-error detecting and $\lfloor \frac{d-1}{2} \rfloor$-error correcting.

The table below (taken from Hamming's original paper) shows the maximum number of errors a code of small minimum distance can detect and correct.

$d(C)$	error detection / correction
1	no detection possible
2	1-error detecting
3	2-error detecting / 1-error correcting
4	3-error detecting / 1-error correcting
5	4-error detecting / 2-error correcting

(n, M, d)-notation

There is a special notation for recording the most important parameters of a code.

Notation 4.4

If C is a code of length n, size M and minimum distance d, then C is said to be a (n, M, d)-code.

For example, a repetition code of length n over a q-ary alphabet is a (n, q, n)-code, and the binary parity check code of length $n + 1$ in Example 2.6 is a $(n, 2^n, 2)$-code. The square code is a $(8, 16, 3)$-code.

Hamming Balls

Definition 4.7

Let A be a q-ary alphabet and let u be a word of length n. The *Hamming ball of radius t about u* is

$$B_t(u) = \{v \in A^n : d(u, v) \leq t\}.$$

An equivalent definition is that $B_t(u)$ consists of all words that can be obtained from u by changing up to t of its positions. For example, $B_1(000) = \{000, 100, 010, 001\}$.

Example 4.8

The Hamming balls about the binary word 0000 are

$$B_0(0000) = \{0000\}$$
$$B_1(0000) = \{0000, 1000, 0100, 0010, 0001\},$$
$$B_2(0000) = B_1(0000) \cup \{1100, 1010, 1001, 0110, 0101, 0011\}$$
$$B_3(0000) = B_2(0000) \cup \{1110, 1101, 1011, 0111\}$$

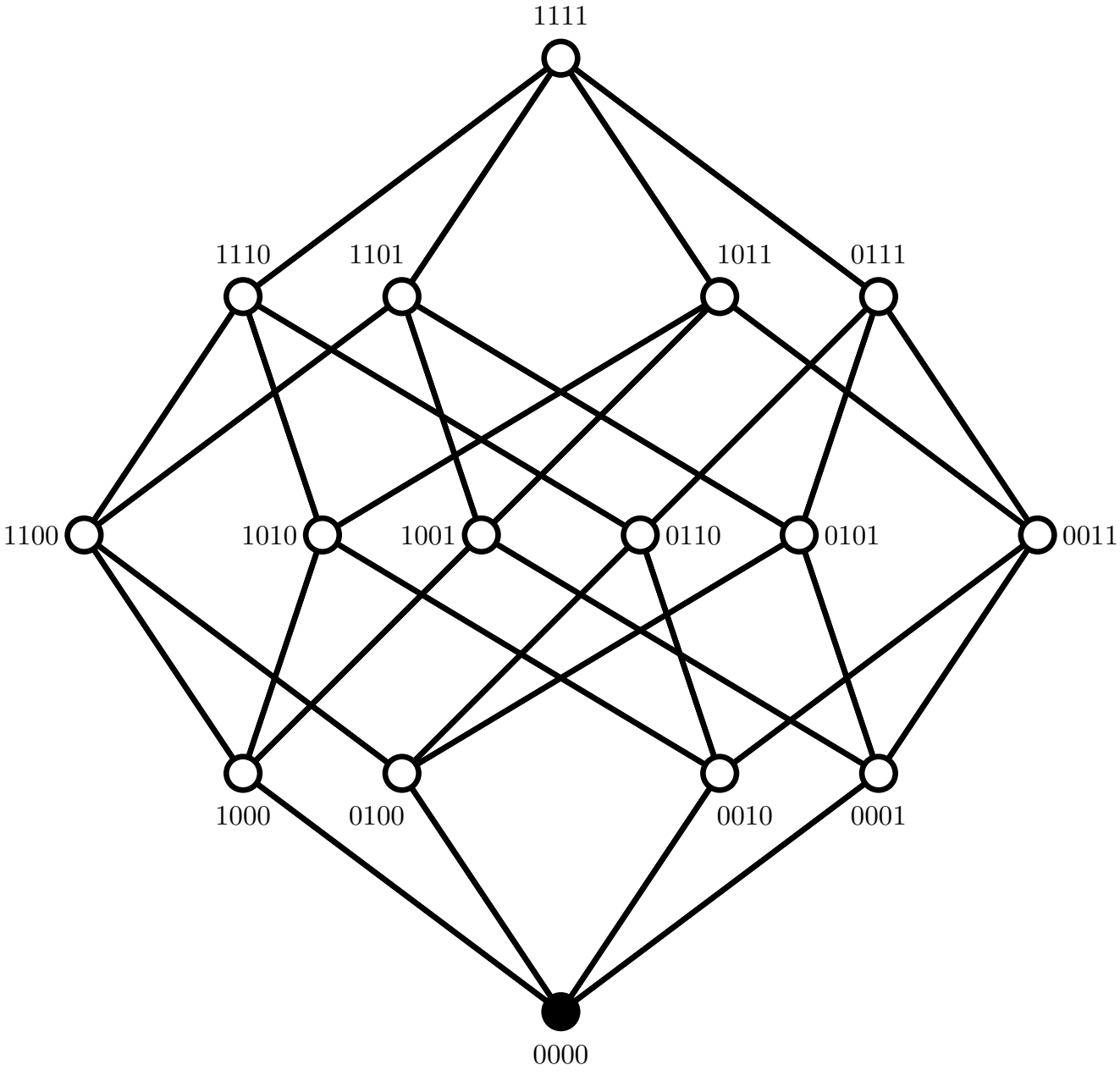
and $B_4(0000)$ is the set of all binary words of length 4.

(ii) If $u = 1010$ then

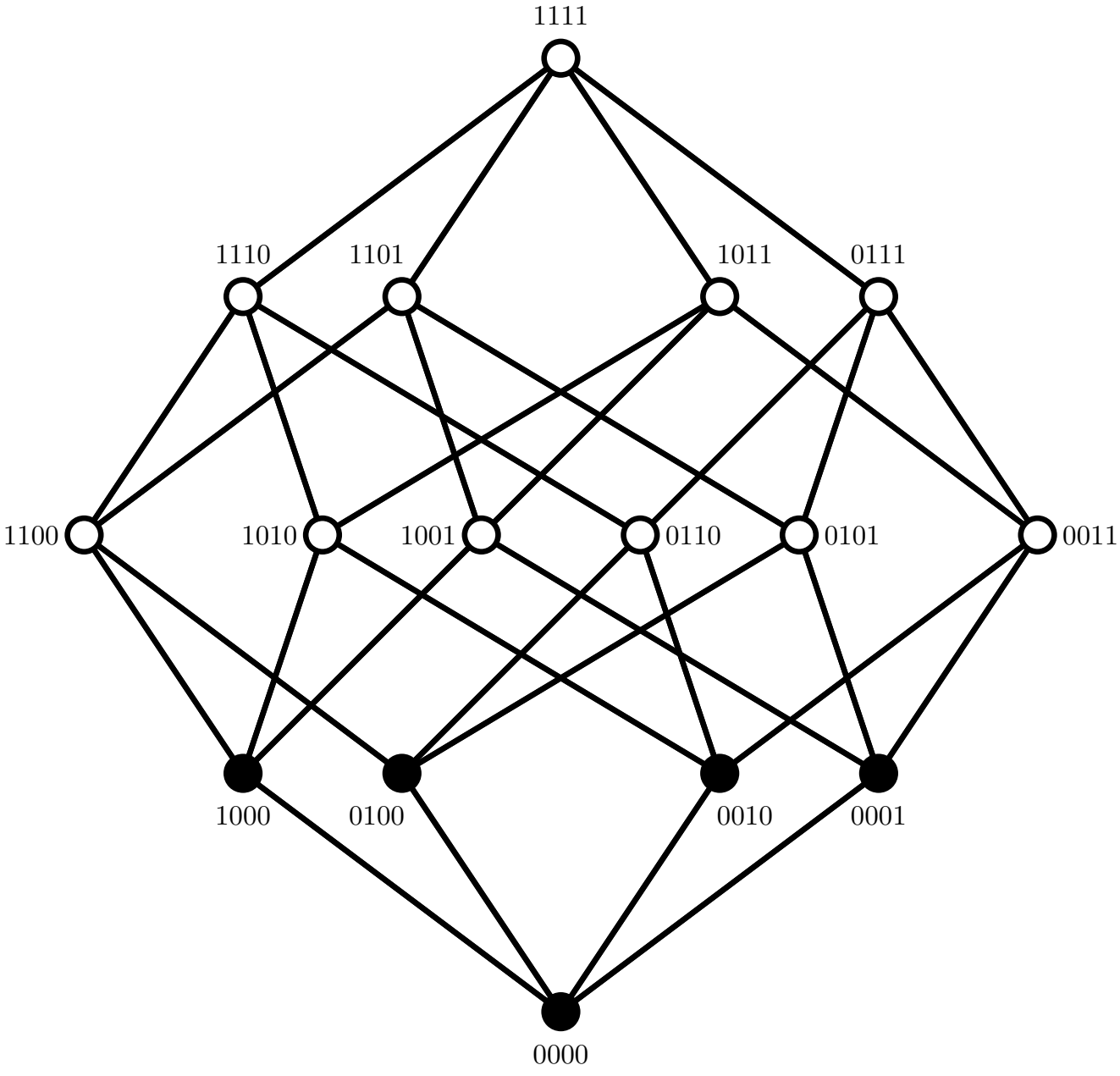
$$B_0(1010) = \{1010\},$$
$$B_1(1010) = \{1010, 0010, 1110, 1000, 1011\},$$
$$B_2(1010) = B_1(1010) \cup \{0110, 0000, 0001, 1100, 1111, 1001\}.$$

Also $B_3(1010)$ consists of all binary words of length 4 *except* 0101, and $B_4(1010)$ is the set of all binary words of length 4.

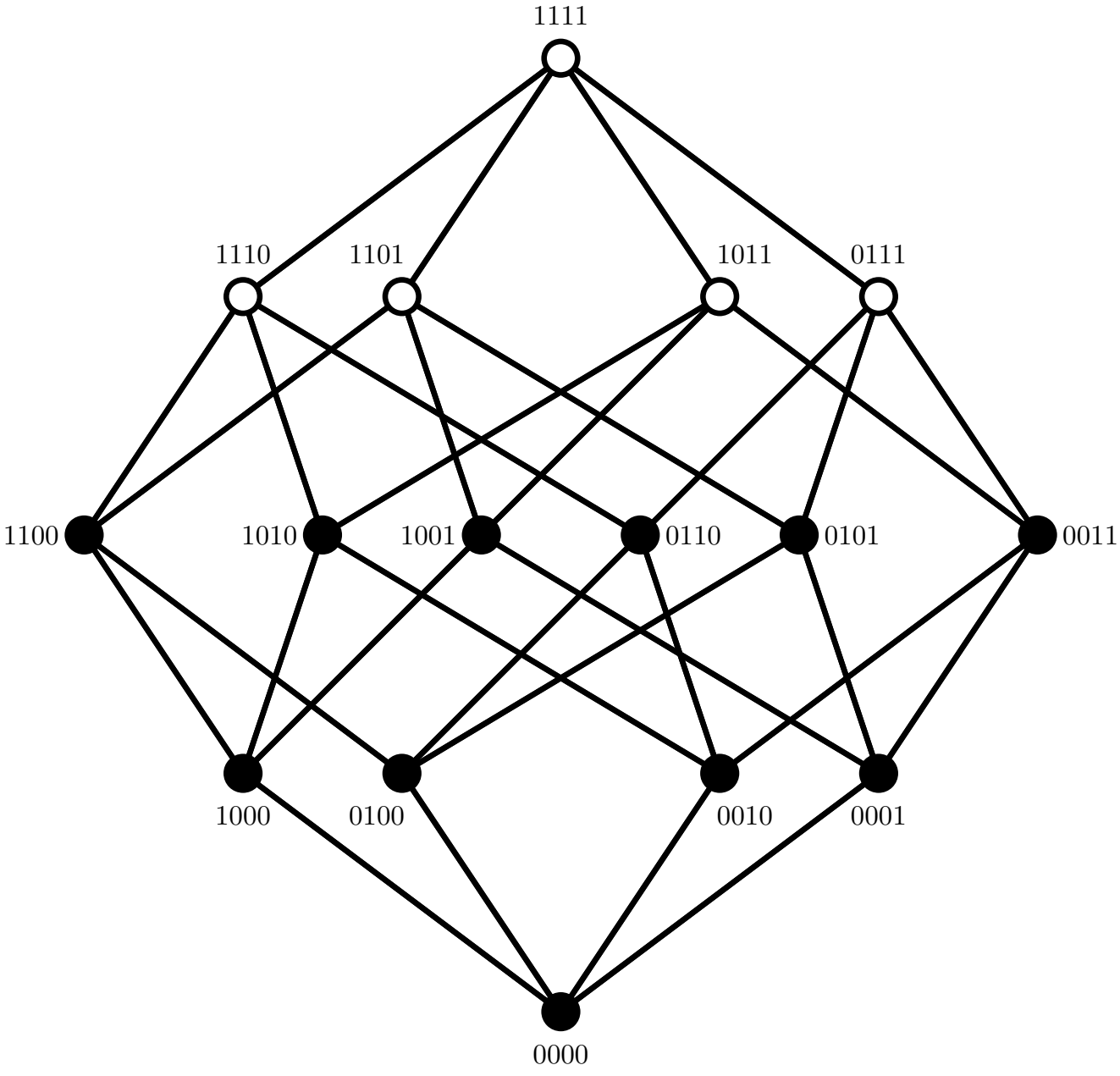
$B_0(0000)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

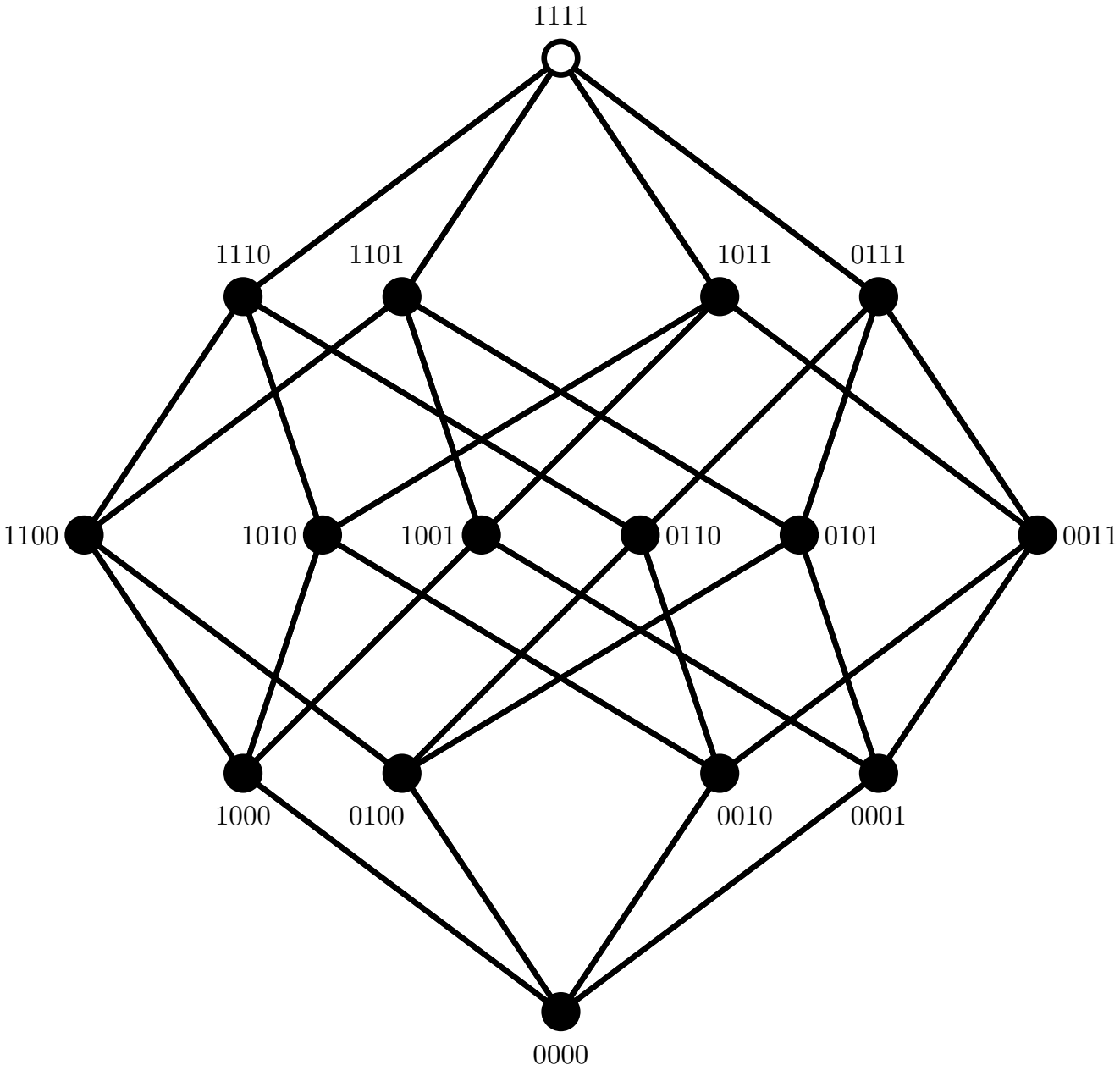
$B_1(0000)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

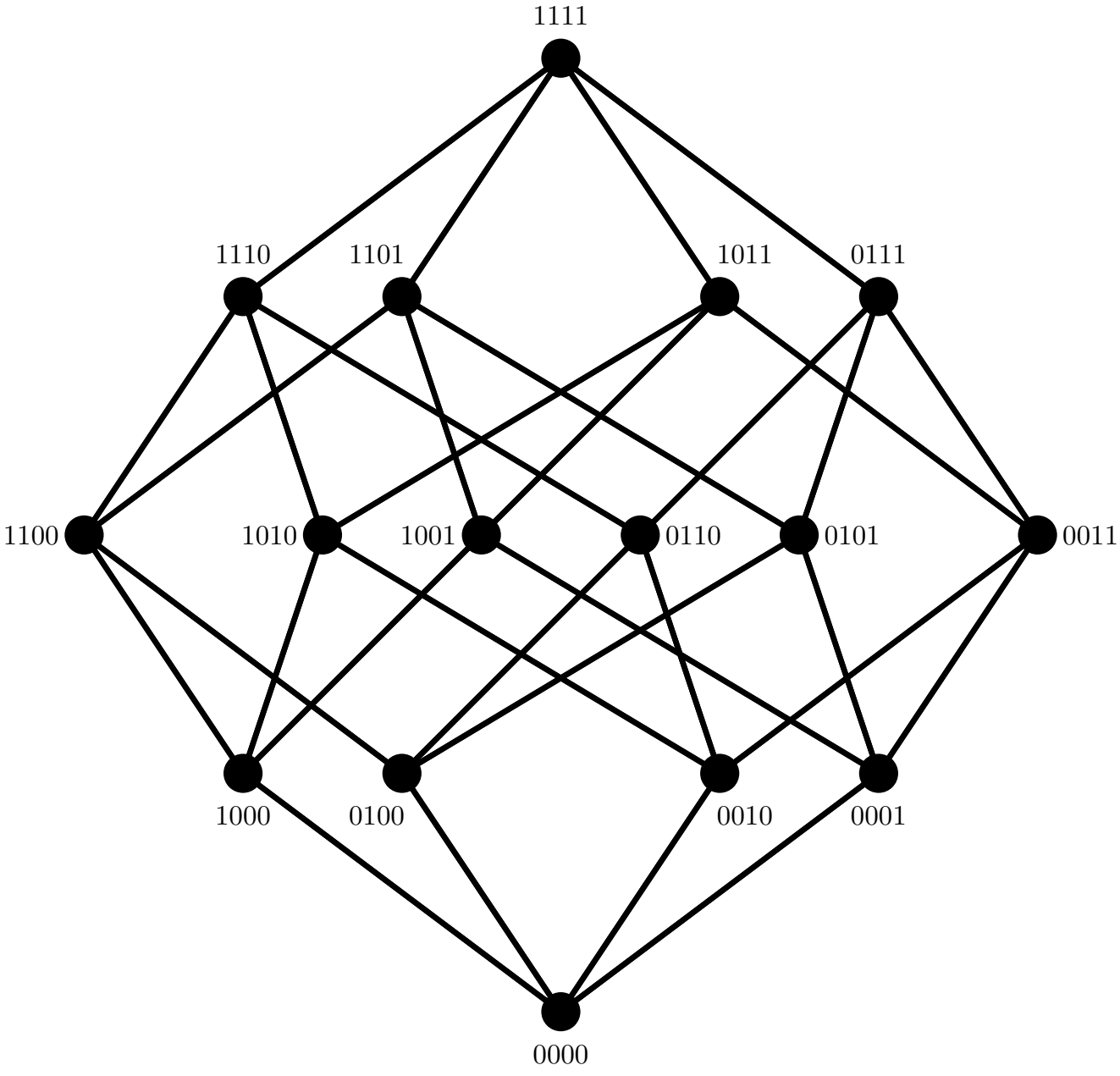
$B_2(0000)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

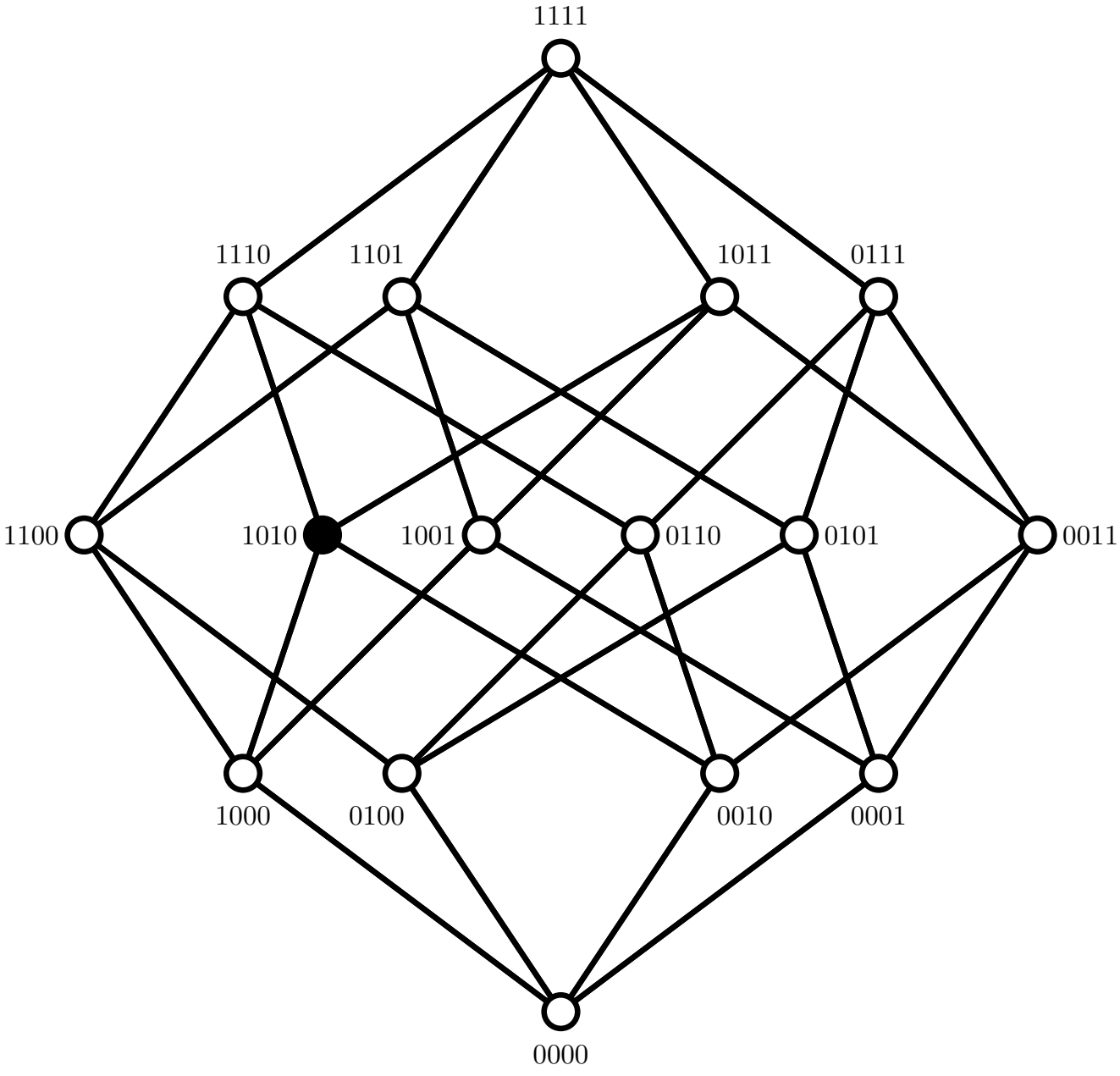
$B_3(0000)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

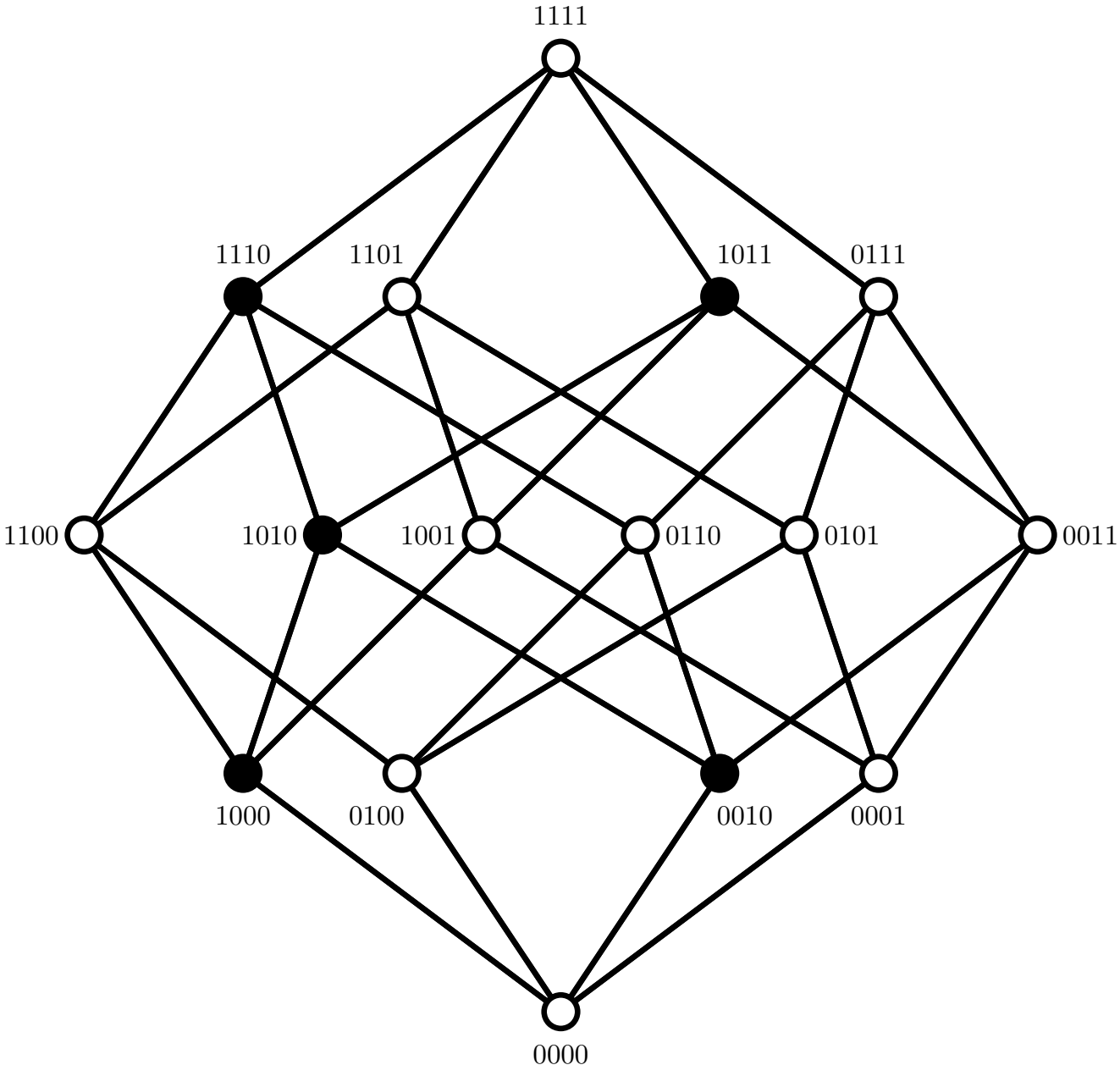
$B_4(0000)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

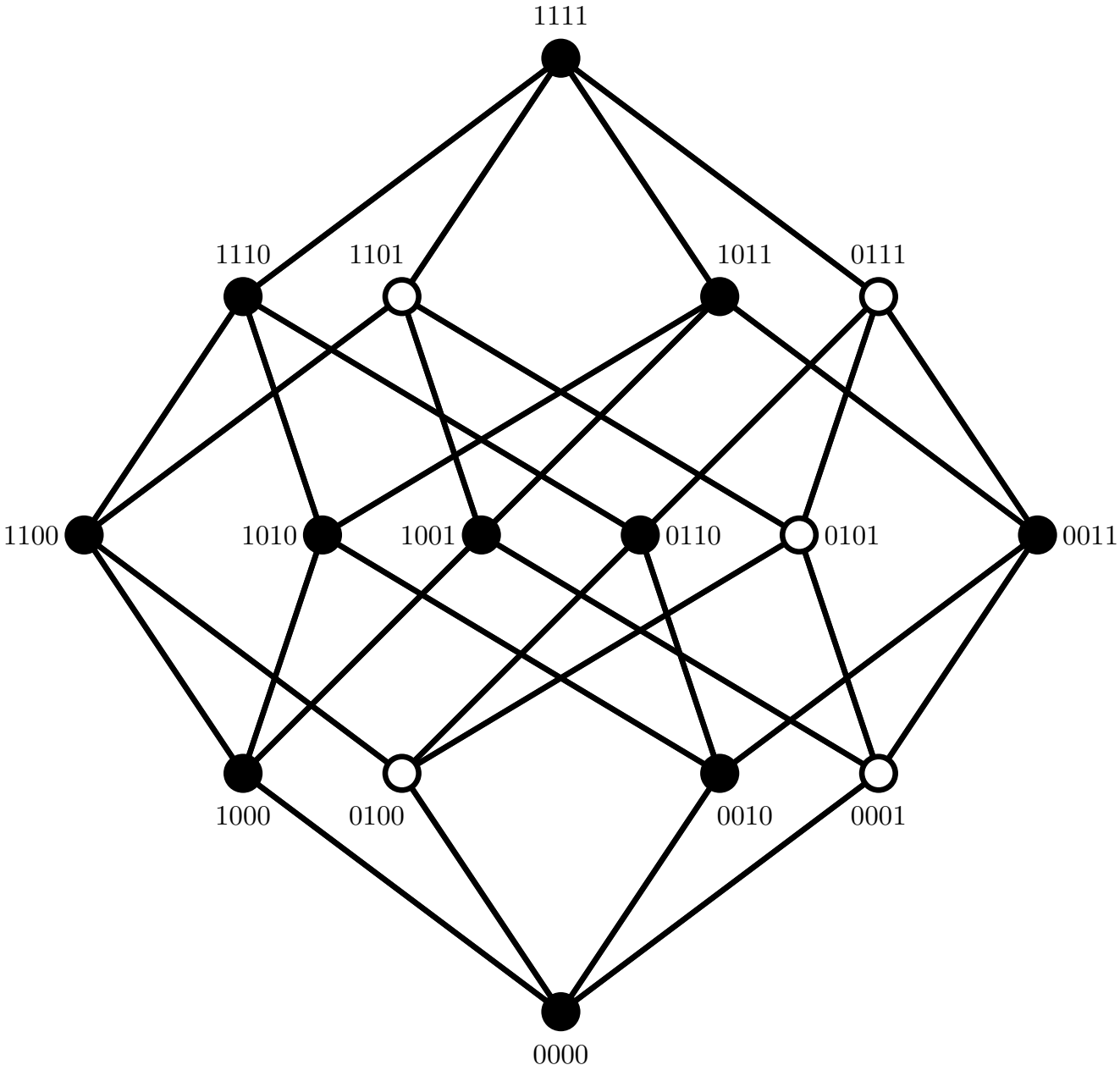
$B_0(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

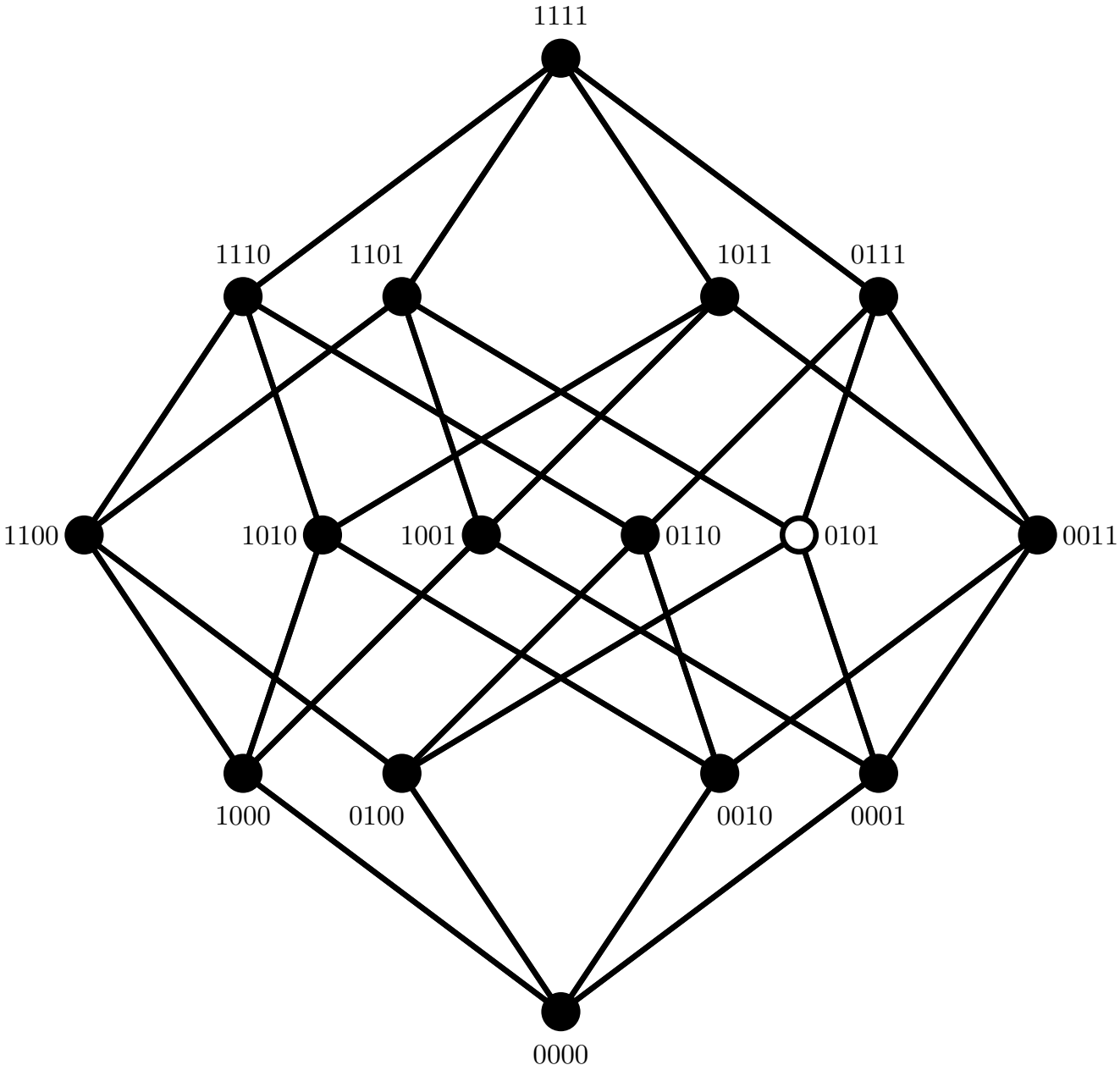
$B_1(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

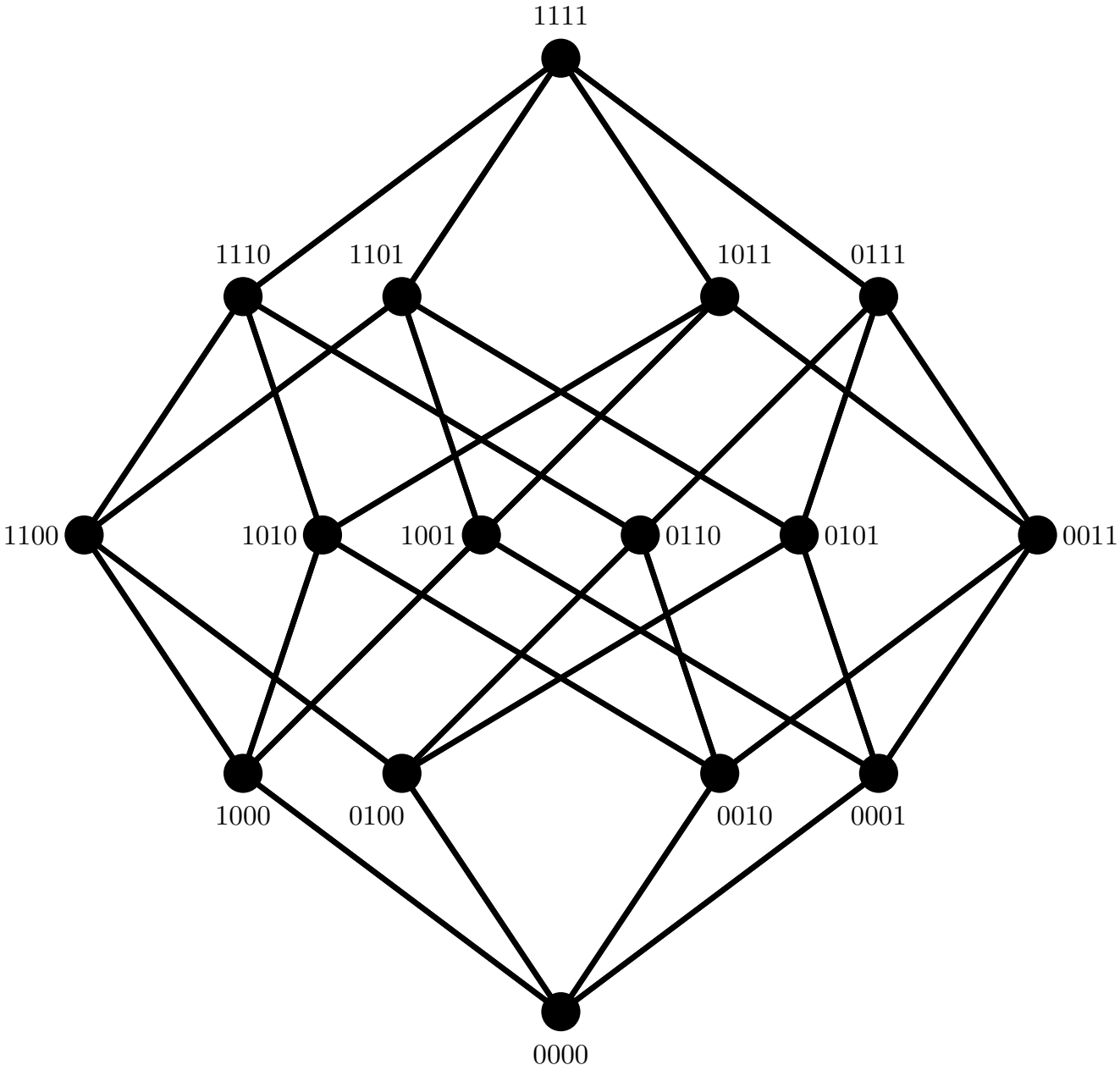
$B_2(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

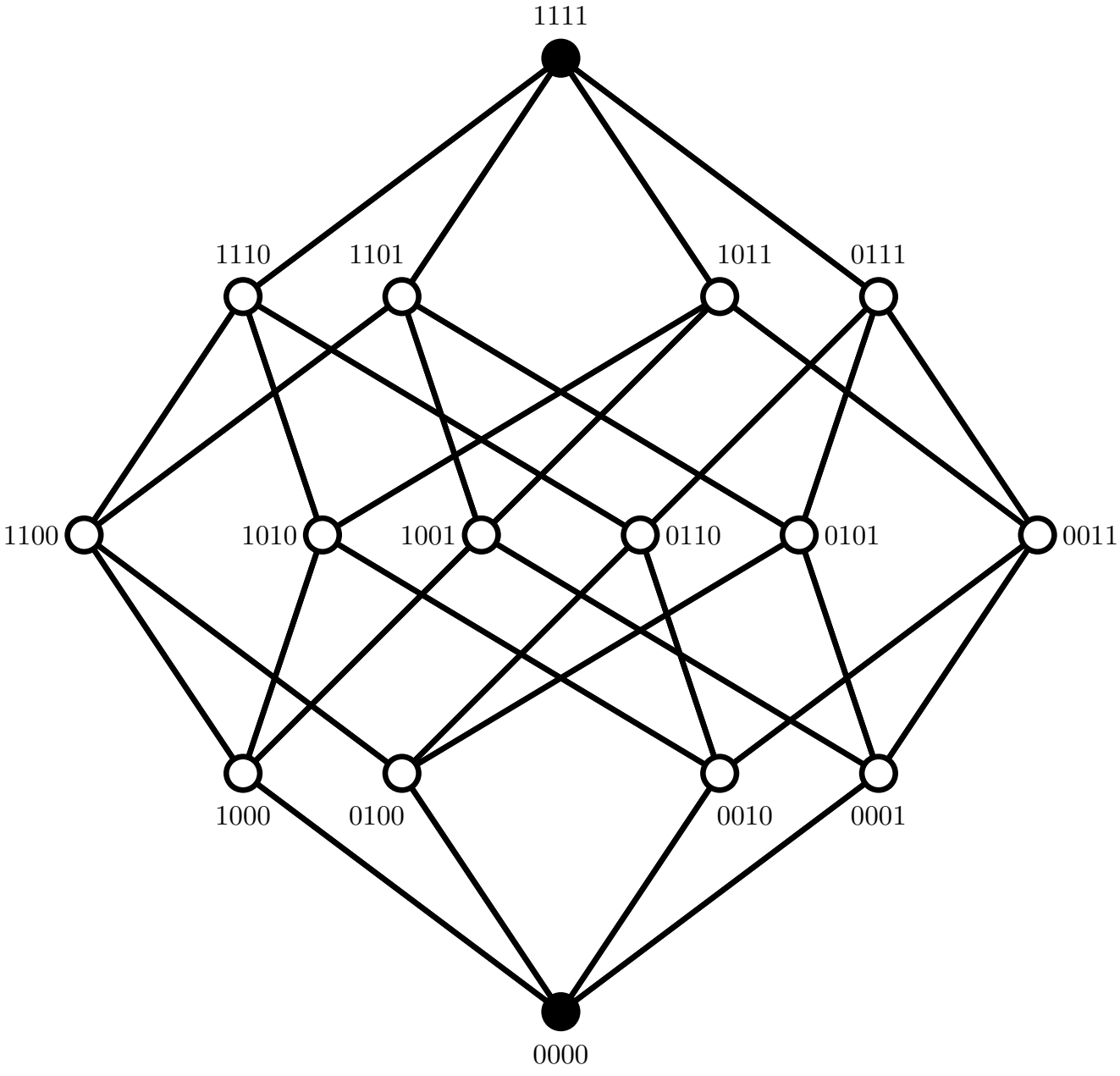
$B_3(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

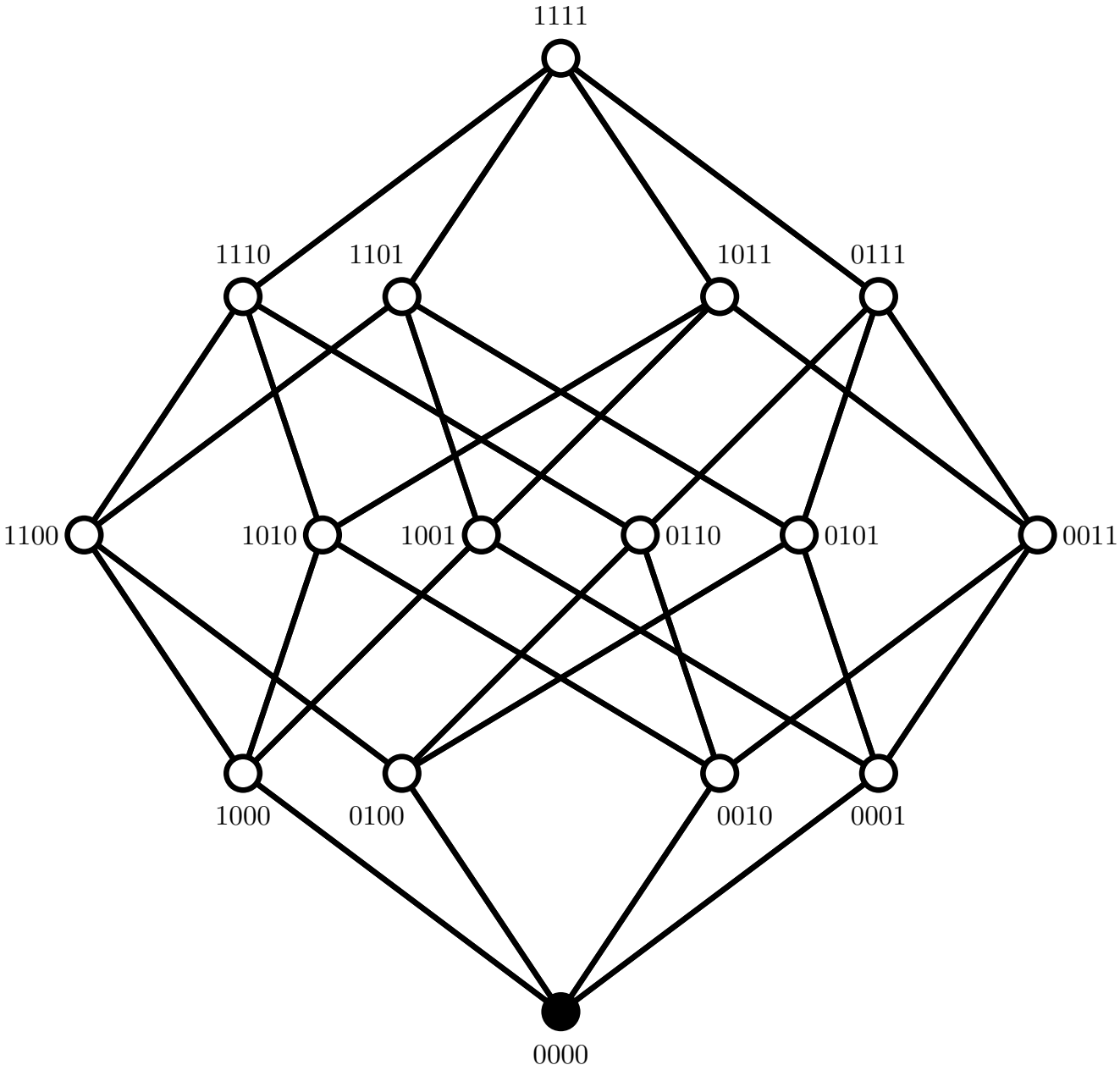
$B_4(1010)$


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

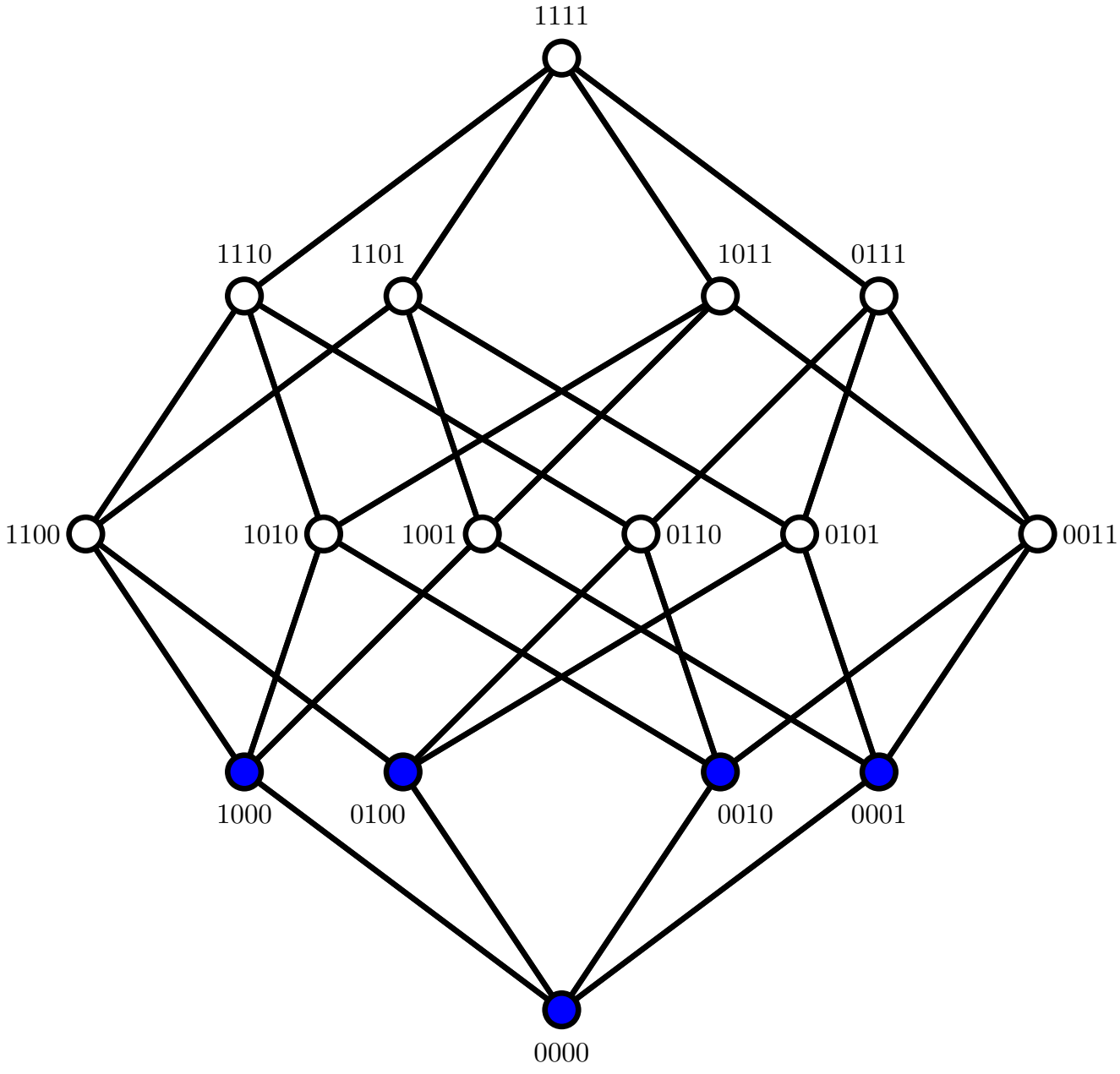
Nearest Neighbour Decoding for $\{0000, 1111\}$



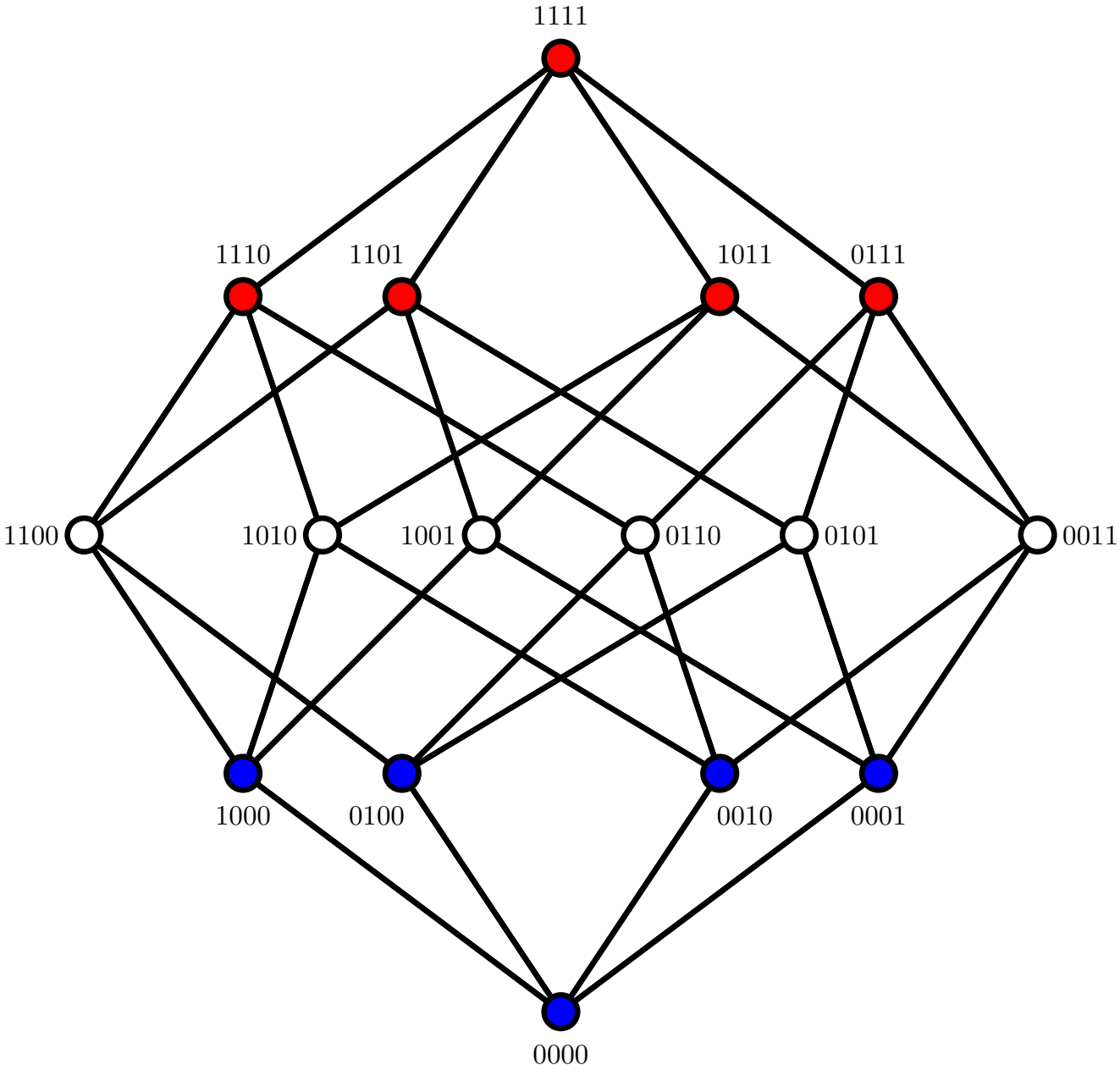
0000 Sent


1111
1110
1101
1011
0111
1100
1010
1001
0110
0101
0011
1000
0100
0010
0001
0000

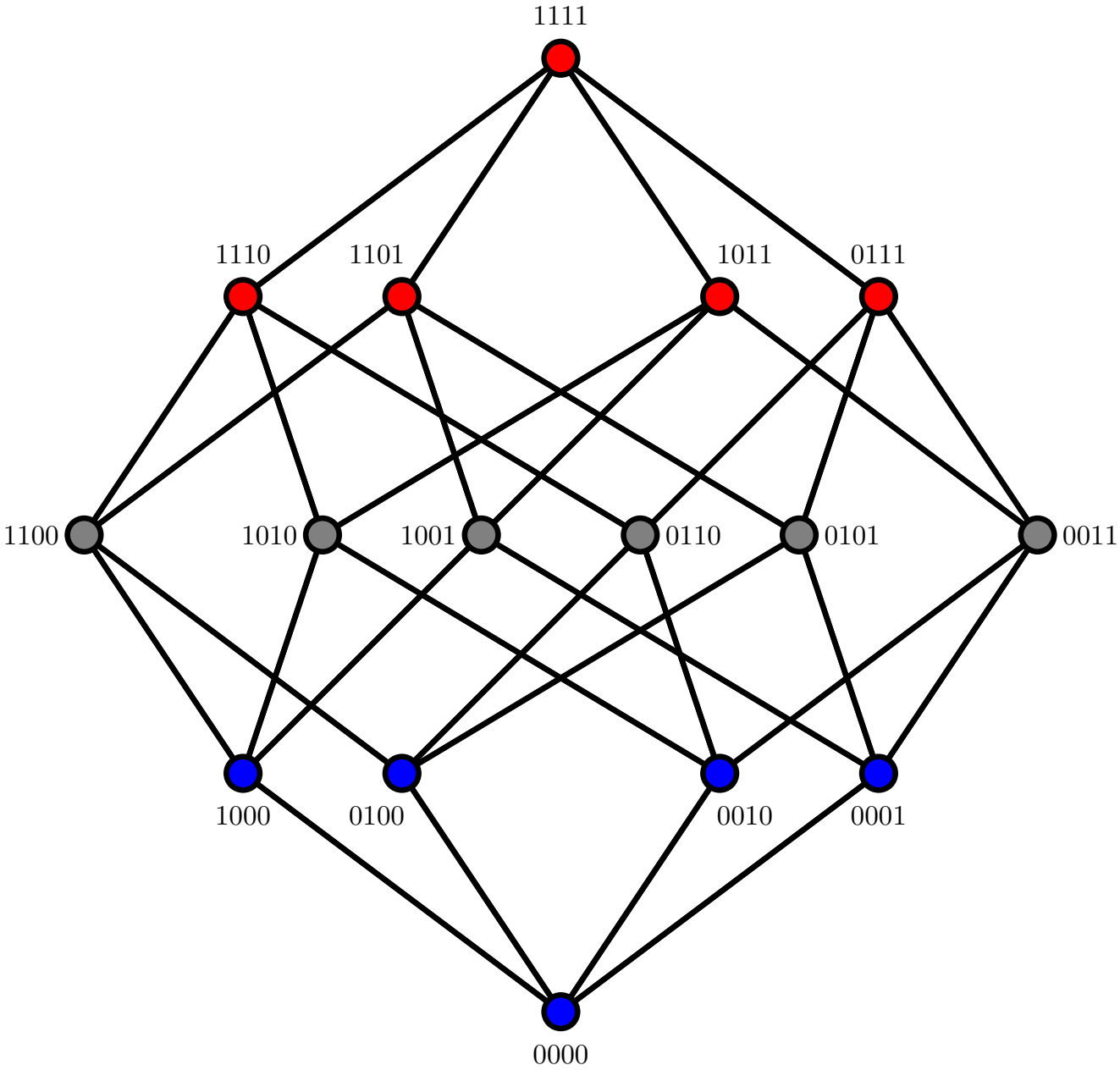
Blue: Received Words Decoded as 0000



Red: Received Words Decoded as 1111



Grey: Nearest Neighbour Decoding Fails



Lemma on Disjoint Hamming Balls

Lemma 4.9

Let C be a code. If C is t-error correcting then for all distinct codewords $u, u' \in C$, the Hamming balls $B_t(u)$ and $B_t(u')$ are disjoint.

Proof: codewords in $B_t(u)$ are decoded to u under nearest neighbour decoding. Codewords in $B_t(u')$ are decoded to u' under nearest neighbour decoding. So no word can be in both. $\square$

We will use Lemma 4.9 to complete the proof of Theorem 4.5(ii). We have already proved '$\Longleftarrow$', so only '$\Longrightarrow$' remains.

Theorem 4.5

Let C be a code with minimum distance d. Let $t \in \mathbf{N}$.

(i) *C is t-error detecting* $\iff$ $t \leq d-1$;

(ii) *C is t-error correcting* $\iff$ $2t \leq d-1$.

Summary of Part A

In Part A we have seen the formal definition (Definition 3.1) of t-error detecting and correcting codes. The examples in §2 and the results in Lemmas 3.3 and 3.4 show that this definition is a reasonable one. We then saw other ways of thinking about t-error correcting codes, using minimum distance (Definition 4.1) and Hamming balls (Lemma 4.9).

Theorem 4.10

Let C be a code. The following are equivalent

(a) *C is t-error correcting;*

(b) *Nearest neighbour decoding always gives the sent codeword (without failing), provided at most t errors occur;*

(c) *If $u \in C$ and $d(u, v) \leq t$ then $d(u, v) < d(u', v)$ for all $u' \in C$ such that $u' \neq u$;*

(d) *If u, $u' \in C$ are distinct codewords then the Hamming balls $B_t(u)$ and $B_t(u')$ are disjoint;*

(e) *The minimum distance of C is at least $2t + 1$.*

Problem Sheet 2 Question 2

Work will be returned tomorrow in the Tuesday lecture.

2. Let A be a q-ary alphabet where $q \geq 2$. Let $m \in \mathbf{N}$ and let C be the repetition code of length $2m$ over A. Prove Lemma 3.3(iii) that C is $(m-1)$-error correcting but not m-error correcting.

Most people gave a good proof that C is $(m-1)$-error correcting. (See Moodle for model answers.)

To show that C is not m-error correcting

Problem Sheet 2 Question 2

Work will be returned tomorrow in the Tuesday lecture.

2. Let A be a q-ary alphabet where $q \geq 2$. Let $m \in \mathbf{N}$ and let C be the repetition code of length $2m$ over A. Prove Lemma 3.3(iii) that C is $(m-1)$-error correcting but not m-error correcting.

Most people gave a good proof that C is $(m-1)$-error correcting. (See Moodle for model answers.)

To show that C is not m-error correcting

- "C is $(m-1)$-error correcting so can't be m-error correcting". This is simply wrong.

Problem Sheet 2 Question 2

Work will be returned tomorrow in the Tuesday lecture.

2. Let A be a q-ary alphabet where $q \geq 2$. Let $m \in \mathbf{N}$ and let C be the repetition code of length $2m$ over A. Prove Lemma 3.3(iii) that C is $(m-1)$-error correcting but not m-error correcting.

Most people gave a good proof that C is $(m-1)$-error correcting. (See Moodle for model answers.)

To show that C is not m-error correcting

- "C is $(m-1)$-error correcting so can't be m-error correcting". This is simply wrong.
- "If you change m positions in a codeword then can't correct". This is very vague. Improve to:

Problem Sheet 2 Question 2

Work will be returned tomorrow in the Tuesday lecture.

2. Let A be a q-ary alphabet where $q \geq 2$. Let $m \in \mathbf{N}$ and let C be the repetition code of length $2m$ over A. Prove Lemma 3.3(iii) that C is $(m-1)$-error correcting but not m-error correcting.

Most people gave a good proof that C is $(m-1)$-error correcting. (See Moodle for model answers.)

To show that C is not m-error correcting

- "C is $(m-1)$-error correcting so can't be m-error correcting". This is simply wrong.
- "If you change m positions in a codeword then can't correct". This is very vague. Improve to:
- "If we change m positions in a codeword u to get v then $d(u,v) = d(u',v) = m$ so nearest neighbour decoding fails." Almost right. But it depends how we change u to get v.

Problem Sheet 2 Question 2

Work will be returned tomorrow in the Tuesday lecture.

2. Let A be a q-ary alphabet where $q \geq 2$. Let $m \in \mathbf{N}$ and let C be the repetition code of length $2m$ over A. Prove Lemma 3.3(iii) that C is $(m-1)$-error correcting but not m-error correcting.

Most people gave a good proof that C is $(m-1)$-error correcting. (See Moodle for model answers.)

To show that C is not m-error correcting

- "C is $(m-1)$-error correcting so can't be m-error correcting". This is simply wrong.
- "If you change m positions in a codeword then can't correct". This is very vague. Improve to:
- "If we change m positions in a codeword u to get v then $d(u,v) = d(u',v) = m$ so nearest neighbour decoding fails." Almost right. But it depends how we change u to get v.

Summary: to show that some object does not have a certain property, one example suffices. Be as specific as possible.

Problem Sheet 2 Question 6

6. Let C be a t-error correcting code over the alphabet A. Suppose that you receive the word $v \in A^n$ and, after some hunting through the code, you find a codeword $u \in C$ such that $d(u, v) \leq t$. Show that if u is not the sent codeword then at least $t + 1$ errors have occurred in the channel.

Part B: Main Coding Theorem Problem

§5 Main Coding Theorem Problem and Hamming's Bound

Problem 5.1

The *Main Coding Theory Problem* is to find codes over a specified alphabet with

(1) small length;

(2) high minimum distance;

(3) large size.

Remark 5.2

Another desirable property is that there should be an efficient way to perform nearest neighbour decoding on received words. For example, the Reed–Solomon code used on compact discs has the largest possible size for its length and minimum distance. There are now efficient decoding algorithms, but when it was first invented, it was impractical because there was no good way to decode received words.

Hamming's Packing Bound: Preliminaries

In the next lemma we need binomial coefficients. Recall that the binomial coefficient $\binom{n}{k}$ is the number of ways to choose k objects from a set of size n. It is given by the formula

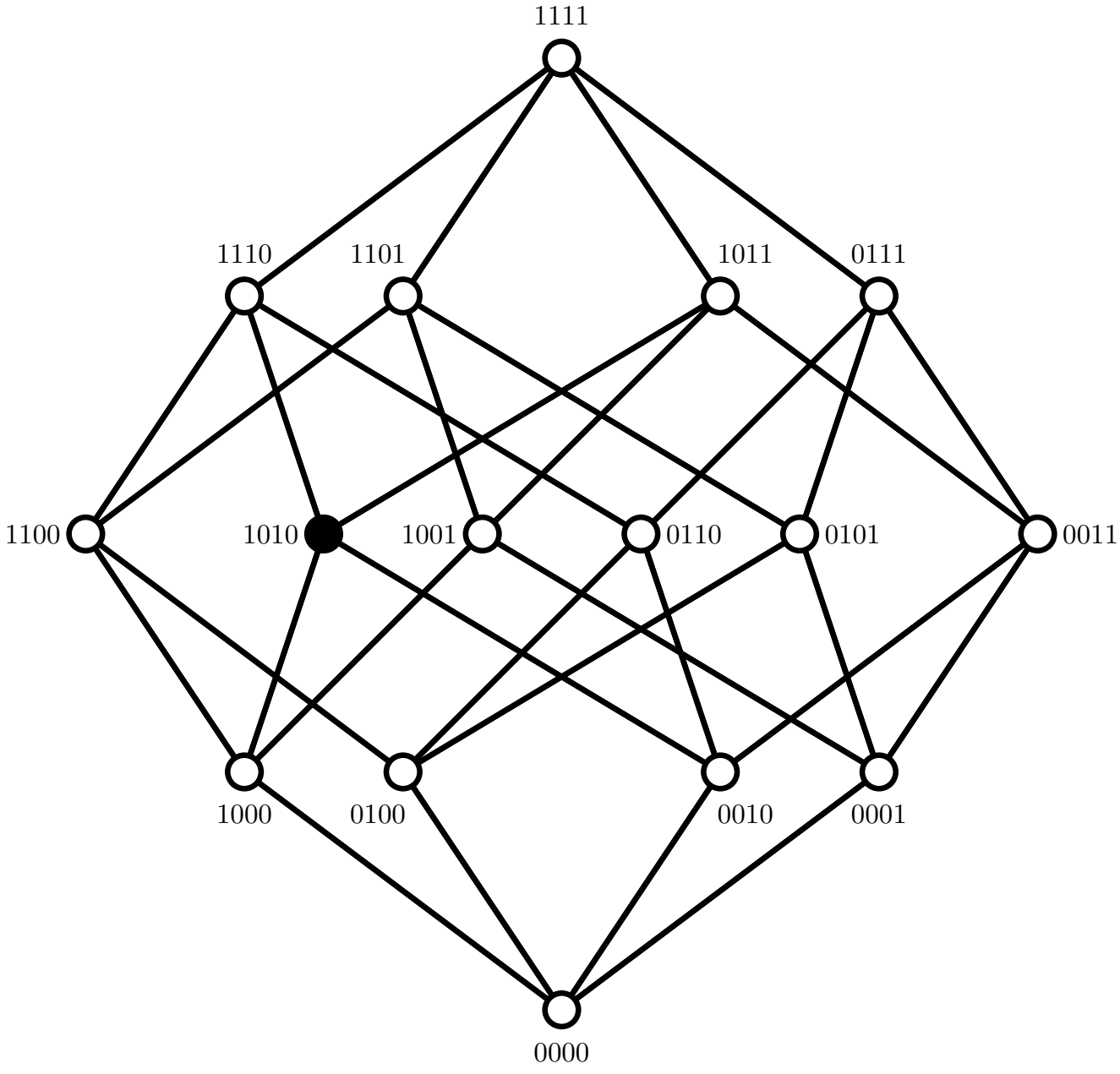
$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{for } 0 \leq k \leq n.$$

Lemma 5.3

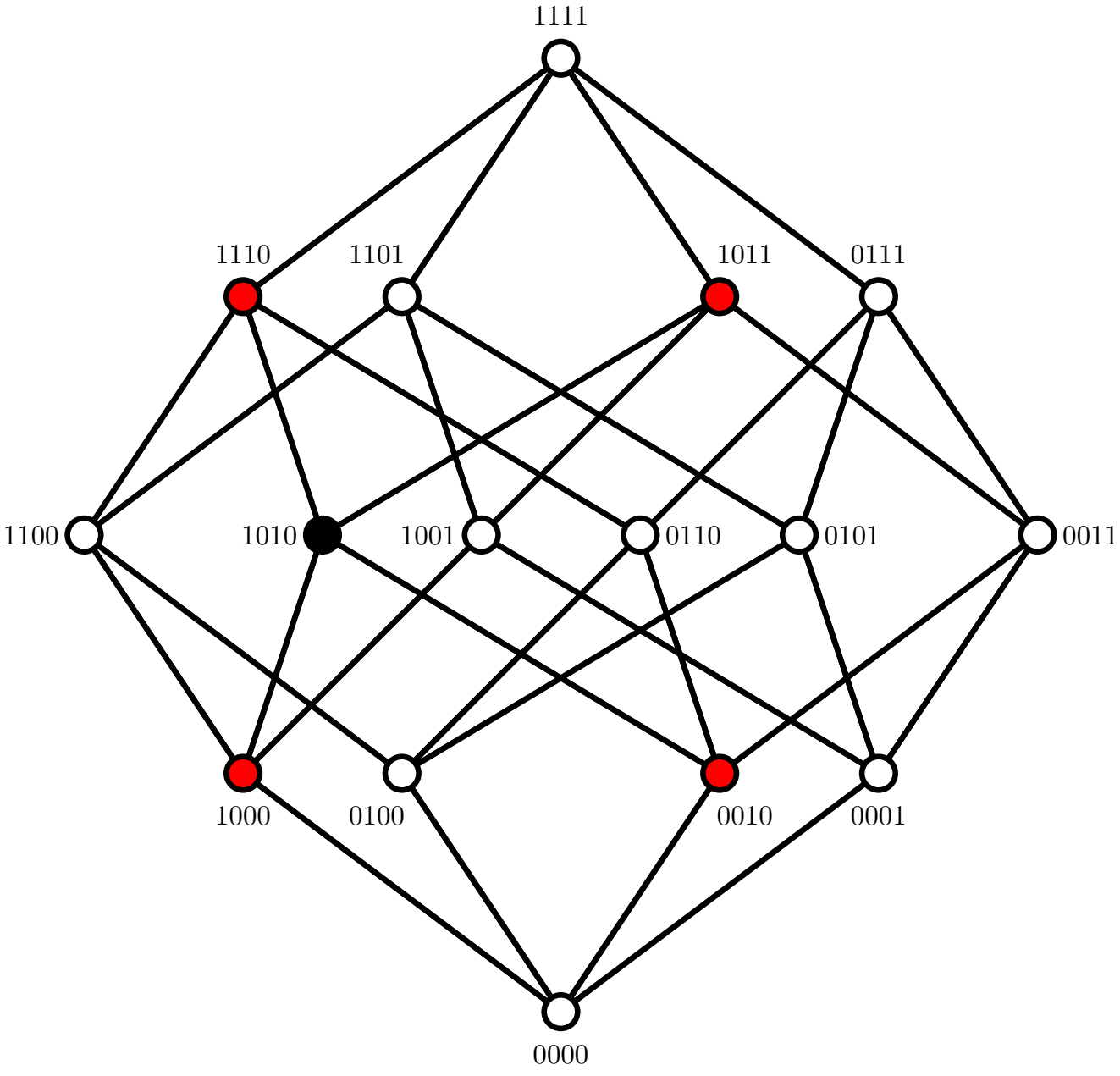
Let u be a binary word of length n. The number of words in the Hamming ball $B_t(u)$ of radius t about u is

$$\sum_{k=0}^{t} \binom{n}{k}.$$

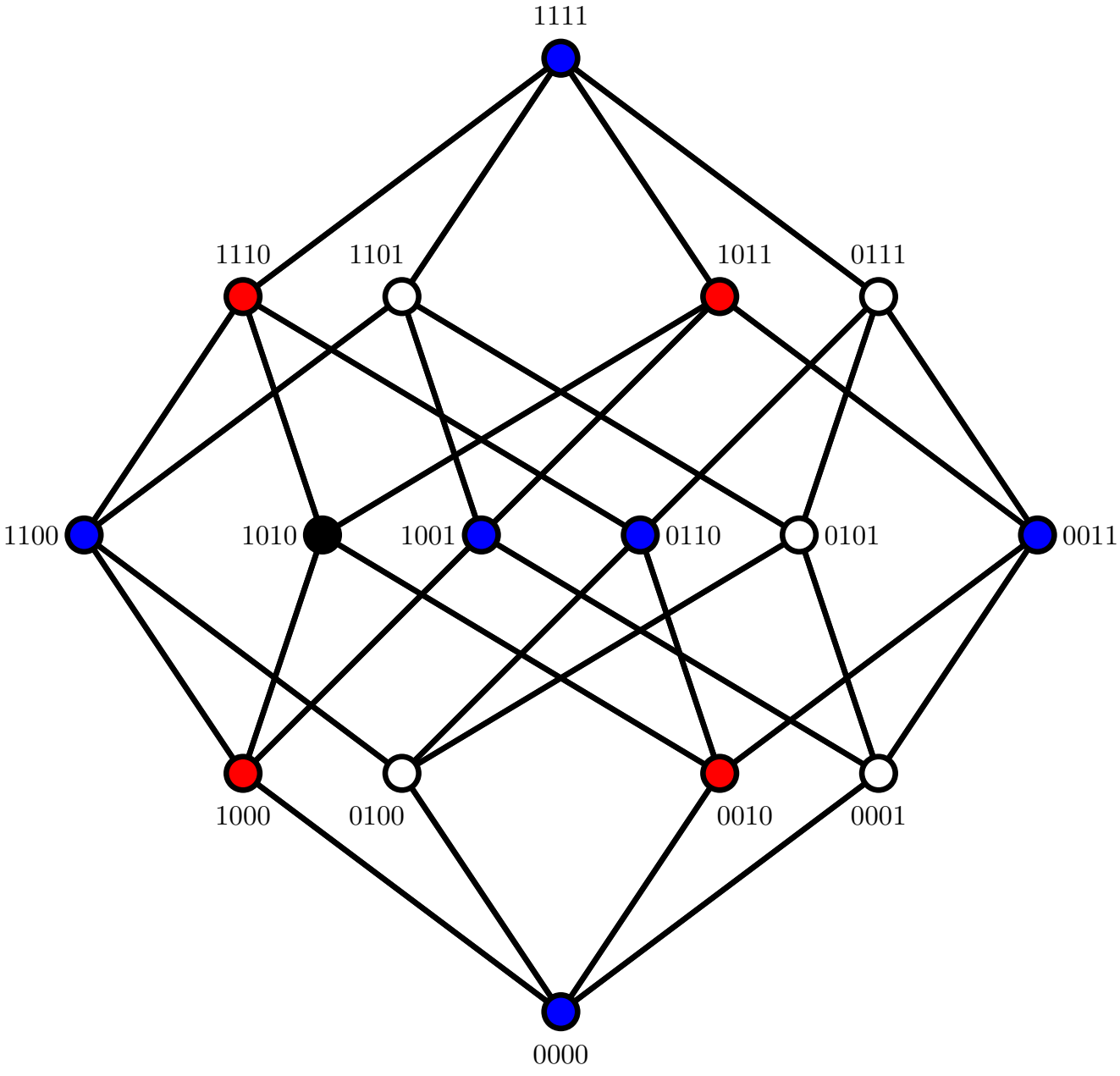
Decomposition of Hamming Ball about 0101 into Spheres



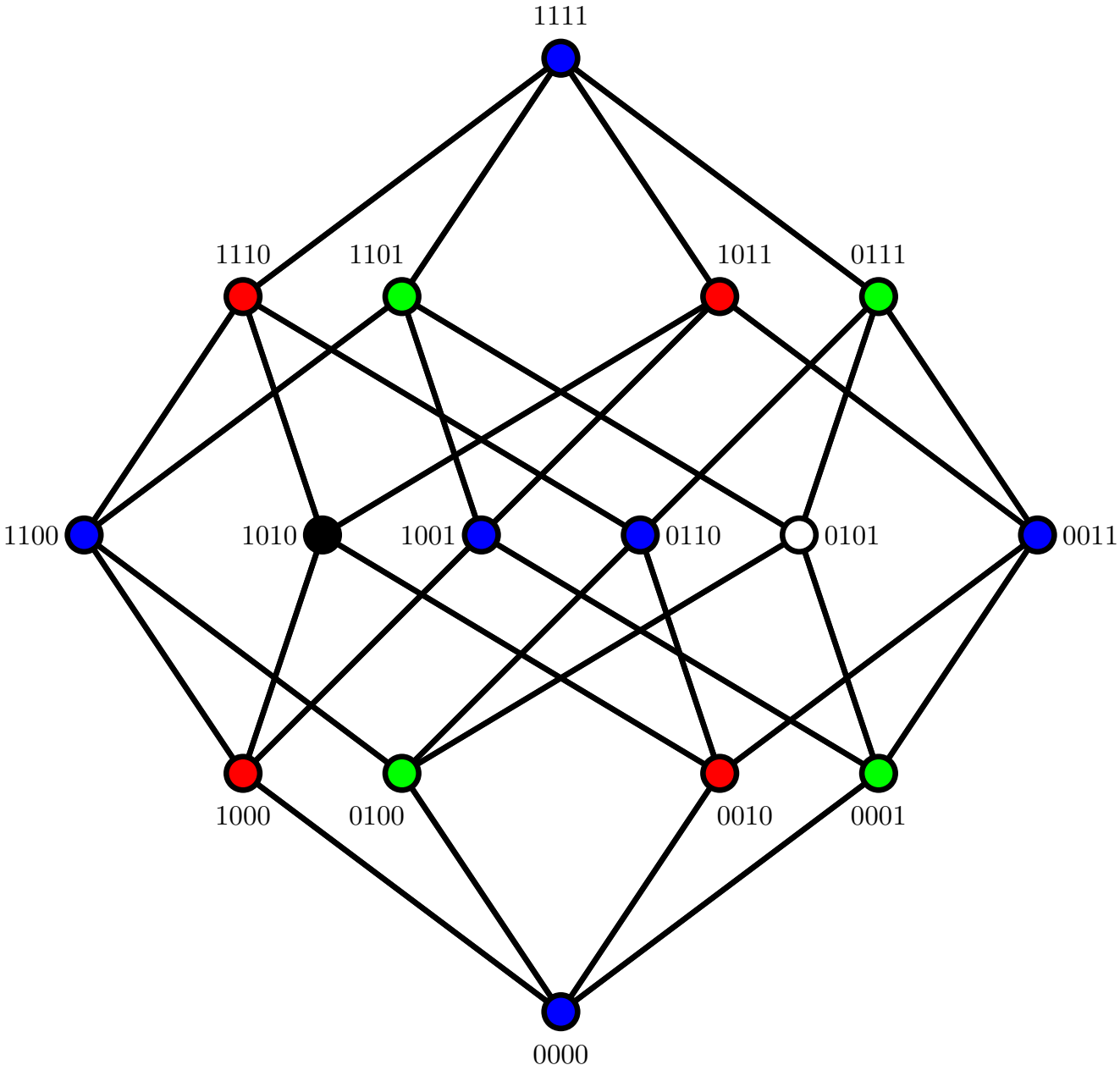
Decomposition of Hamming Ball about 0101 into Spheres



Decomposition of Hamming Ball about 0101 into Spheres



Decomposition of Hamming Ball about 0101 into Spheres



Hamming's Packing Bound

Theorem 5.4 (Hamming's Packing Bound)

Let C be a binary (n, M, d)-code. If $e = \lfloor \frac{d-1}{2} \rfloor$ then

$$M \leq \frac{2^n}{\sum_{k=0}^{e} \binom{n}{k}}.$$

Hamming's Packing Bound

Theorem 5.4 (Hamming's Packing Bound)

Let C be a binary (n, M, d)-code. If $e = \lfloor \frac{d-1}{2} \rfloor$ then

$$M \leq \frac{2^n}{\sum_{k=0}^{e} \binom{n}{k}}.$$

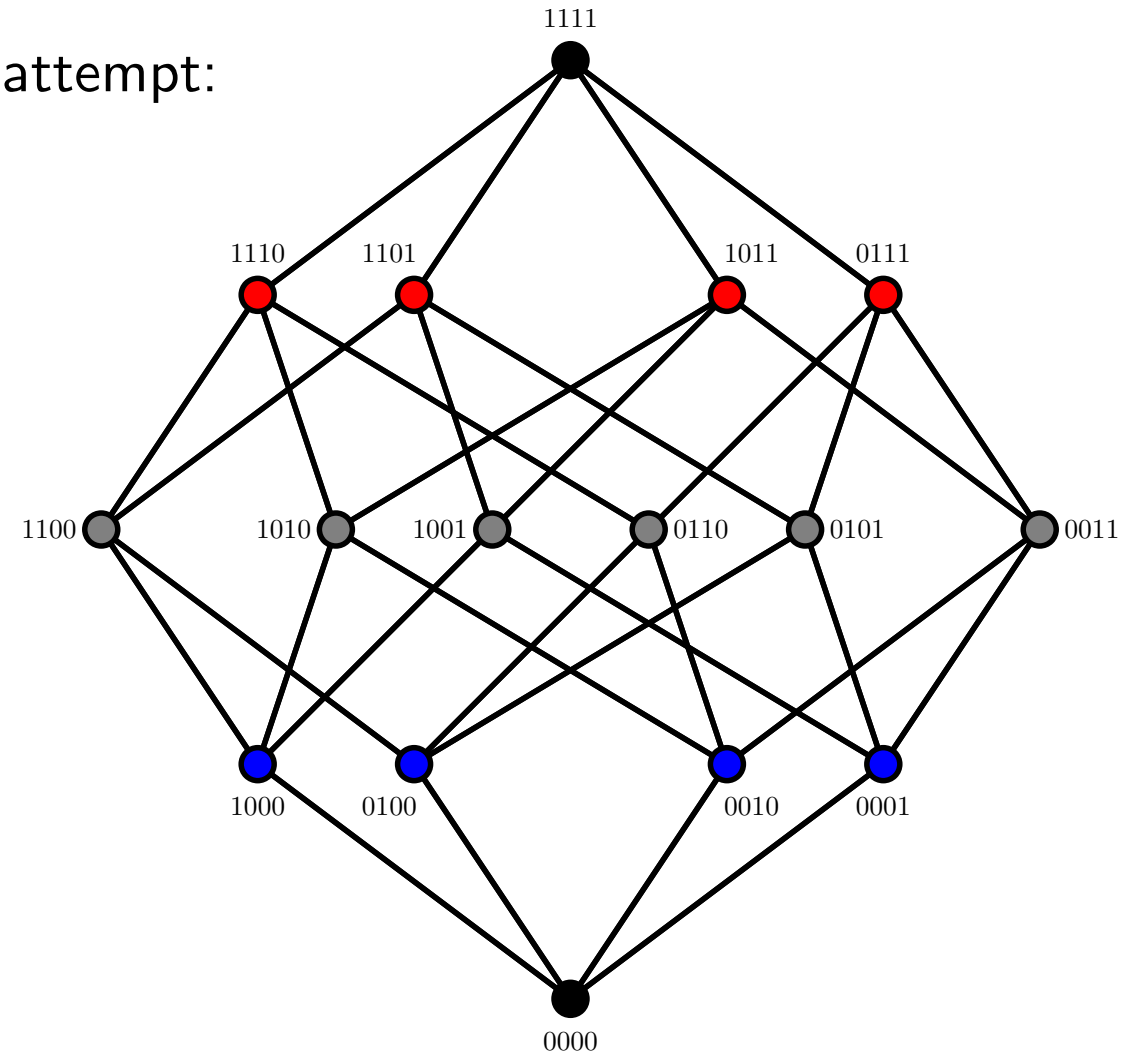
By Theorem 4.5(ii) that a 1-error correcting code has minimum distance at least 3. Hamming's bound therefore implies that a 1-error correcting binary code of length n has size at most $2^n/(1+n)$. Some values are shown below.

length n	3	4	5	6	7	8	9	10
bound on size M	2	3	5	9	16	28	51	93

Hamming's Packing Bound is Necessary NOT Sufficient

Hamming's Packing Bound says that if C is a code of length 4 and minimum distance 3 then $|C| \leq 2^4/(1+4) = 3\frac{1}{5}$. So $|C| \leq 3$. But there is no way to fit 3 disjoint Hamming balls of radius 1 in $\{0,1\}^4$.

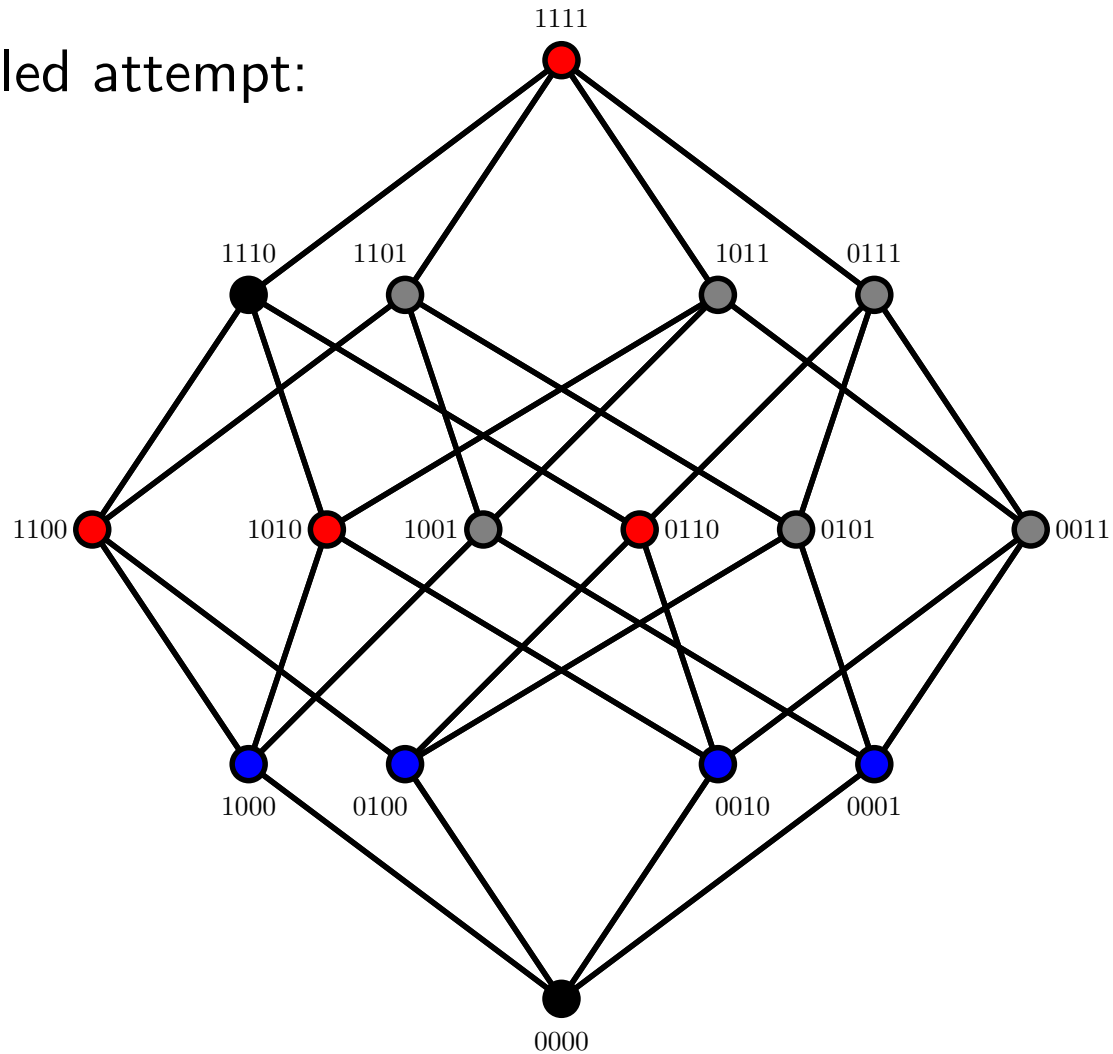
One failed attempt:



Hamming's Packing Bound is Necessary NOT Sufficient

Hamming's Packing Bound says that if C is a code of length 4 and minimum distance 3 then $|C| \leq 2^4/(1+4) = 3\frac{1}{5}$. So $|C| \leq 3$. But there is no way to fit 3 disjoint Hamming balls of radius 1 in $\{0,1\}^4$.

Another failed attempt:



Hamming's Packing Bound is Necessary NOT Sufficient

Hamming's Packing Bound says that if C is a code of length 4 and minimum distance 3 then $|C| \leq 2^4/(1+4) = 3\frac{1}{5}$. So $|C| \leq 3$. But there is no way to fit 3 disjoint Hamming balls of radius 1 in $\{0,1\}^4$.

It is very important to realise that while Hamming's bound is a necessary condition for a binary code of specified length, size and minimum distance to exist, it is **not in general sufficient**.

$A_q(n, d)$ Notation

Definition 5.5

Let $q \geq 2$ and let $n \in \mathbf{N}$, $d \in \mathbf{N}$ be such that $n \geq d$. We denote by $A_q(n, d)$ the largest size of a code of length n and minimum distance d over a q-ary alphabet.

Exercise: Convince yourself that $A_q(n, d)$ does not depend on which q-ary alphabet is used. Working over the q-ary alphabet $\{0, 1, \ldots, q-1\}$, show that there is at least one code of length n and minimum distance d.

The previous exercise implies that $A_q(n, d)$ is well-defined. We can now restate Hamming's Packing Bound as

$$A_2(n, d) \leq 2^n \Big/ \sum_{k=0}^{e} \binom{n}{k},$$

where $e = \lfloor (d-1)/2 \rfloor$.

Optimal Binary Codes: Values of $A_2(n, d)$

The table below shows some values of $A_2(n, d)$; some of these will be proved in the following sections. (You are *not* expected to memorize any part of this table!) One result visible in the table is that $A_2(n, d) = A_2(n + 1, d + 1)$ whenever d is odd: see the optional questions on Sheet 4.

d	$A_2(2, d)$	$A_2(3, d)$	$A_2(4, d)$	$A_2(5, d)$	$A_2(6, d)$	$A_2(7, d)$	$A_2(8, d)$
1	4	8	16	32	64	128	256
2	2	4	8	16	32	64	128
3		2	2	4	8	16	20
4			2	2	4	8	16
5				2	2	2	4
6					2	2	2
7						2	2
8							2

§6 Bounds from Equivalences of Codes

Looking at the first row and the main diagonal in the table you might conjecture that the following lemma holds.

Lemma 6.1

Let $q \geq 2$ and let $n \in \mathbf{N}$. Then

(i) $A_q(n, 1) = q^n$;

(ii) $A_q(n, n) = q$.

Equivalences of Codes

Definition 6.2

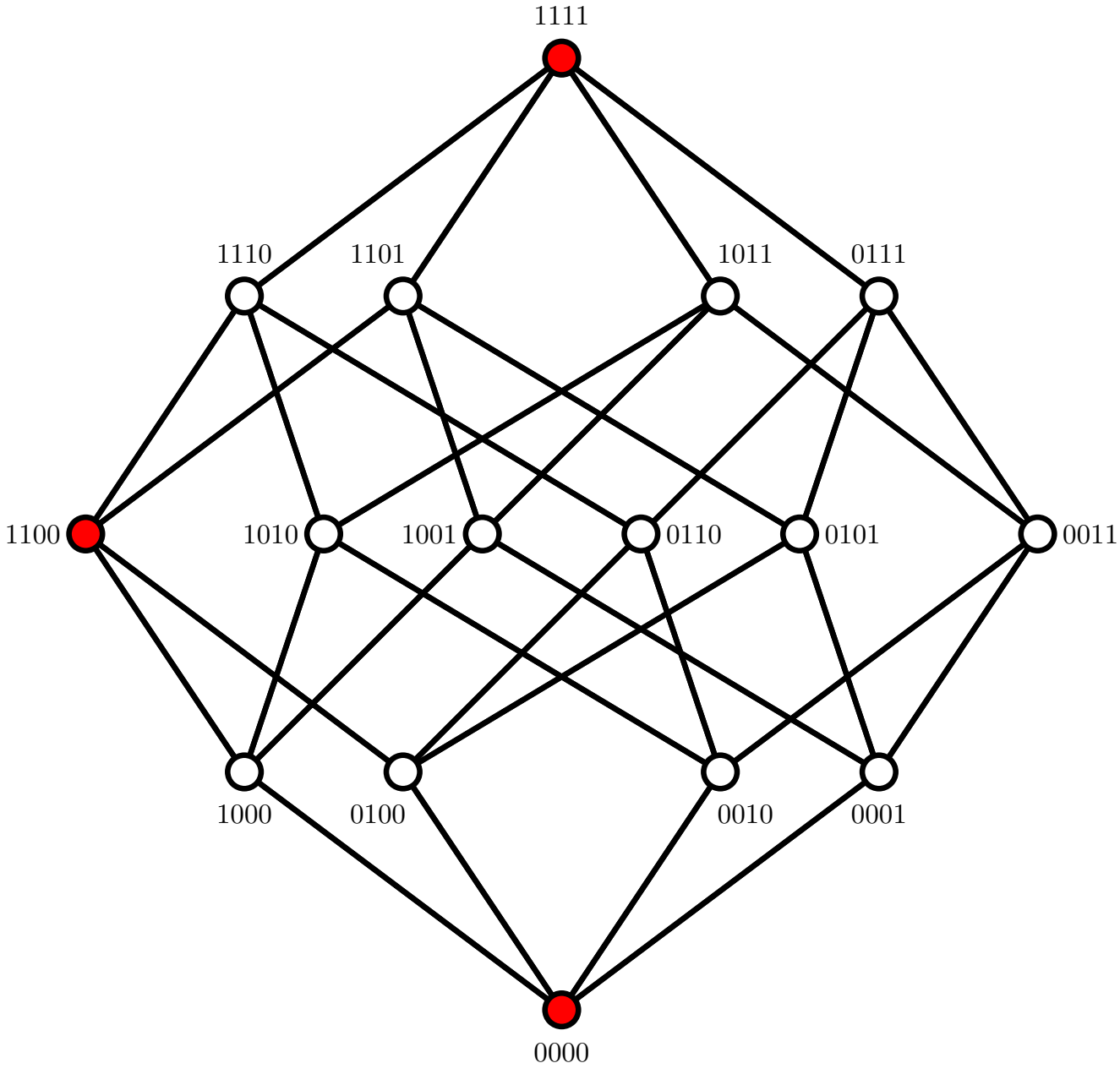
Let C and C' be codes over a q-ary alphabet A. We say that C and C' are *equivalent* if one can be obtained from the other by repeatedly applying the following two operations to all the codewords:

(a) relabelling the symbols appearing in a fixed position;

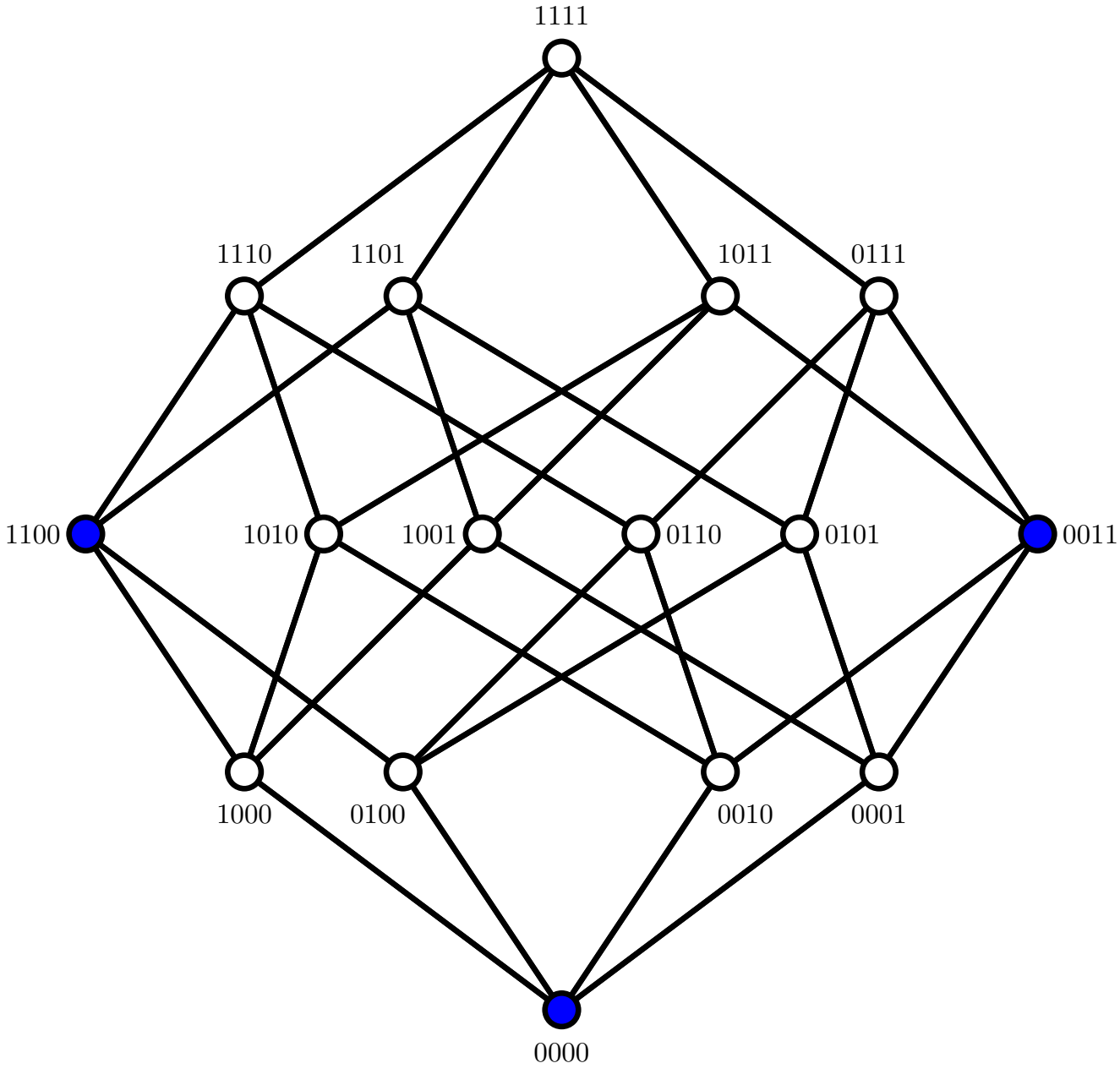
(b) shuffling the positions within each codeword.

Exercise: Are $\{0000, 1100, 1111\}$ and $\{0000, 1100, 0011\}$ equivalent?

Exercise: Are $\{0000, 1100, 1111\}$ and $\{0000, 1100, 0011\}$ equivalent?



Exercise: Are $\{0000, 1100, 1111\}$ and $\{0000, 1100, 0011\}$ equivalent?



Why Equivalences are Useful

We are interested in equivalences because, by the following lemma, equivalent codes have the same distances between codewords.

Lemma 6.3

If u and w are codewords in a code C, and u' and w' are the corresponding codewords in an equivalent code C' obtained by relabelling and/or shuffling positions then $d(u, w) = d(u', w')$.

Proof.

For binary words this was proved in Question 4 on Sheet 3. The only shuffles allowed in this question were swaps on two positions, but any shuffle can be obtained by repeated swaps, so this suffices. The extension to a general alphabet is routine. □

In particular, if C and C' are equivalent then C and C' have the same minimum distance. However the converse does not hold.

Example of Equivalences

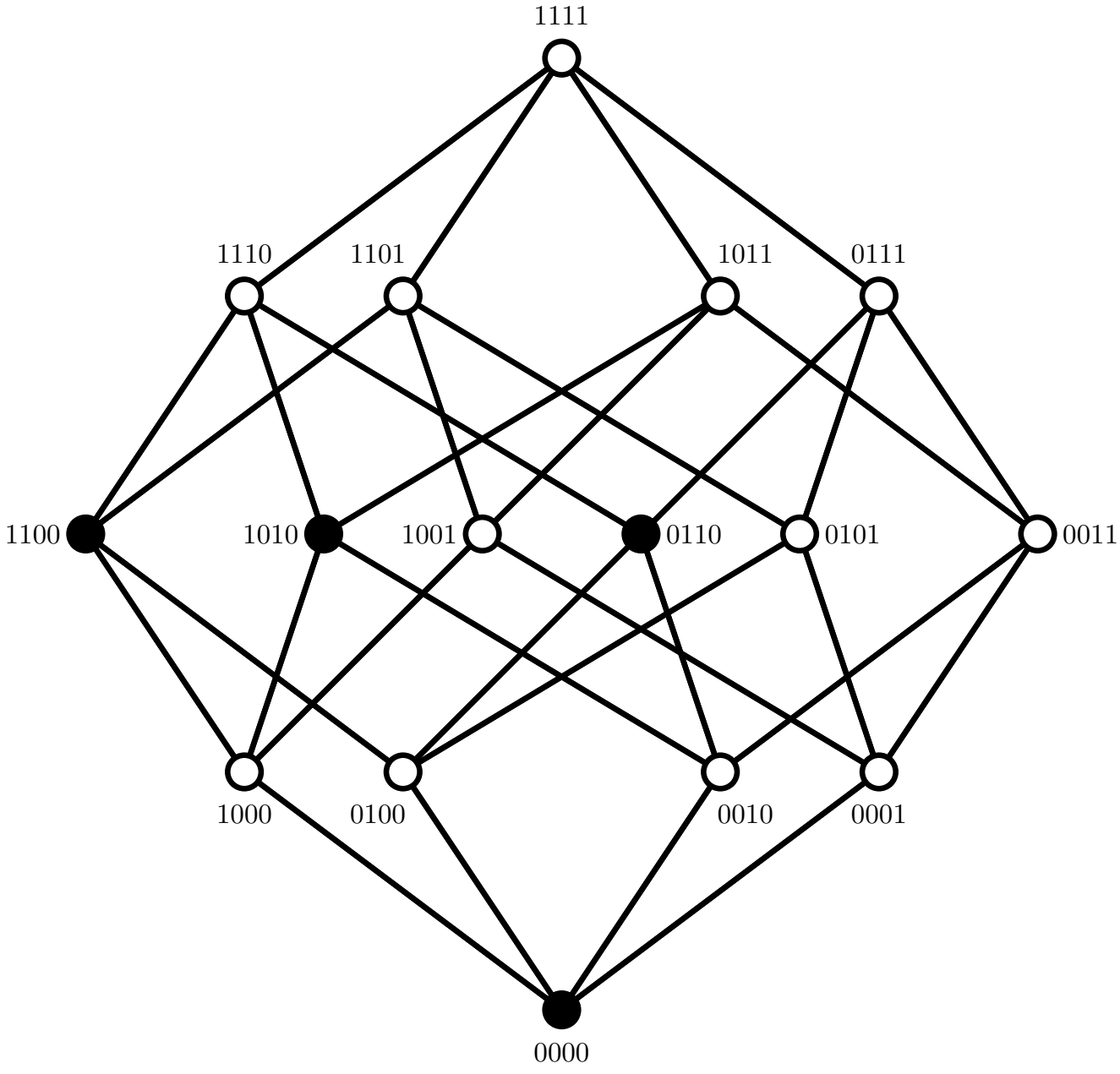
Example 6.4

Consider the four binary codes

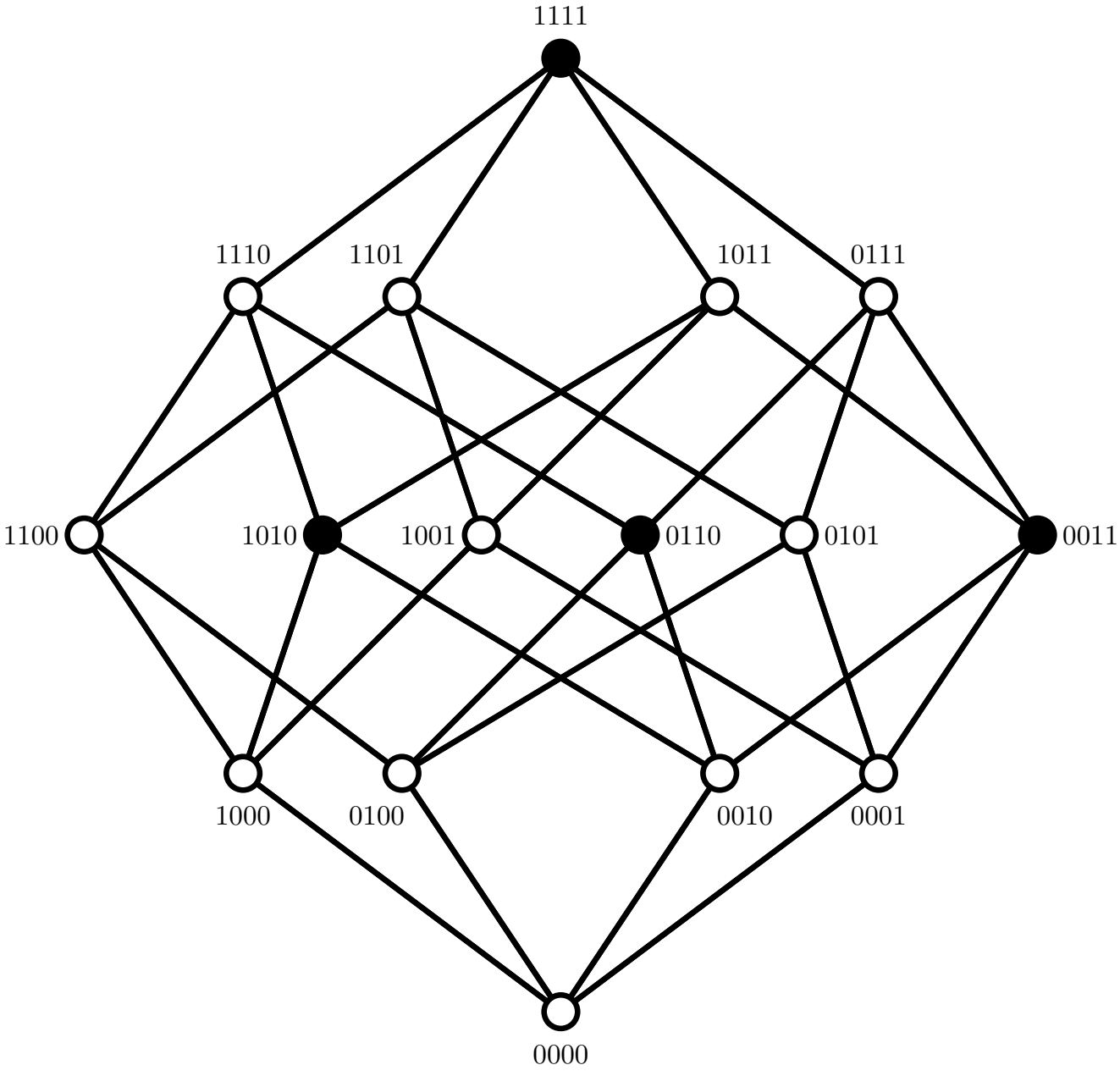
$$C = \{0000, 1100, 1010, 0110\}$$
$$C' = \{1010, 0110, 0011, 1111\}$$
$$D = \{0000, 1100, 0011, 1111\}$$
$$E = \{1000, 0100, 0010, 0001\}.$$

All four codes have minimum distance 2. By applying operations (a) and (b) we will show that C and C' are equivalent. No other two of these codes are equivalent.

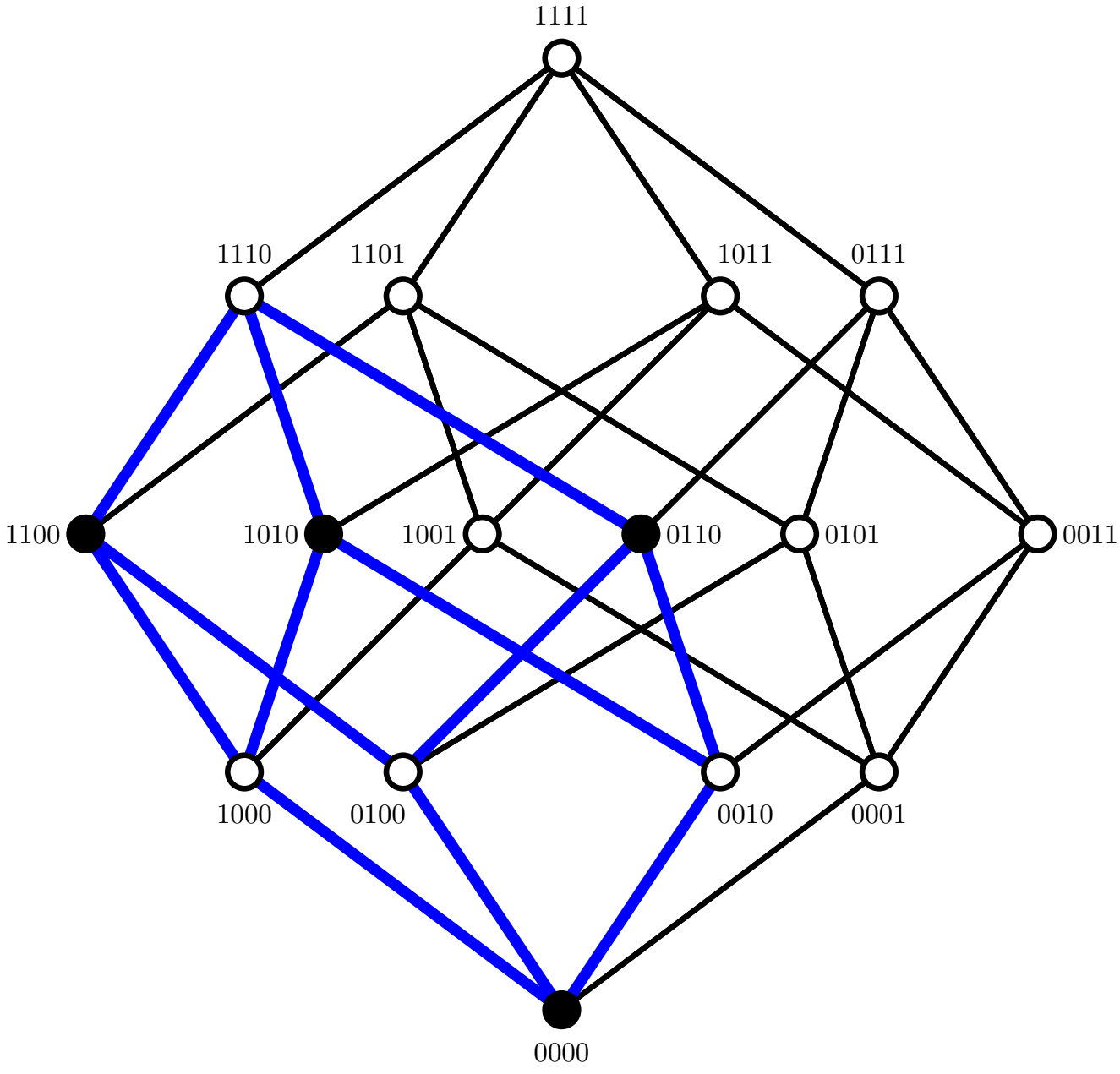
$C = \{0000, 1100, 1010, 0110\}$



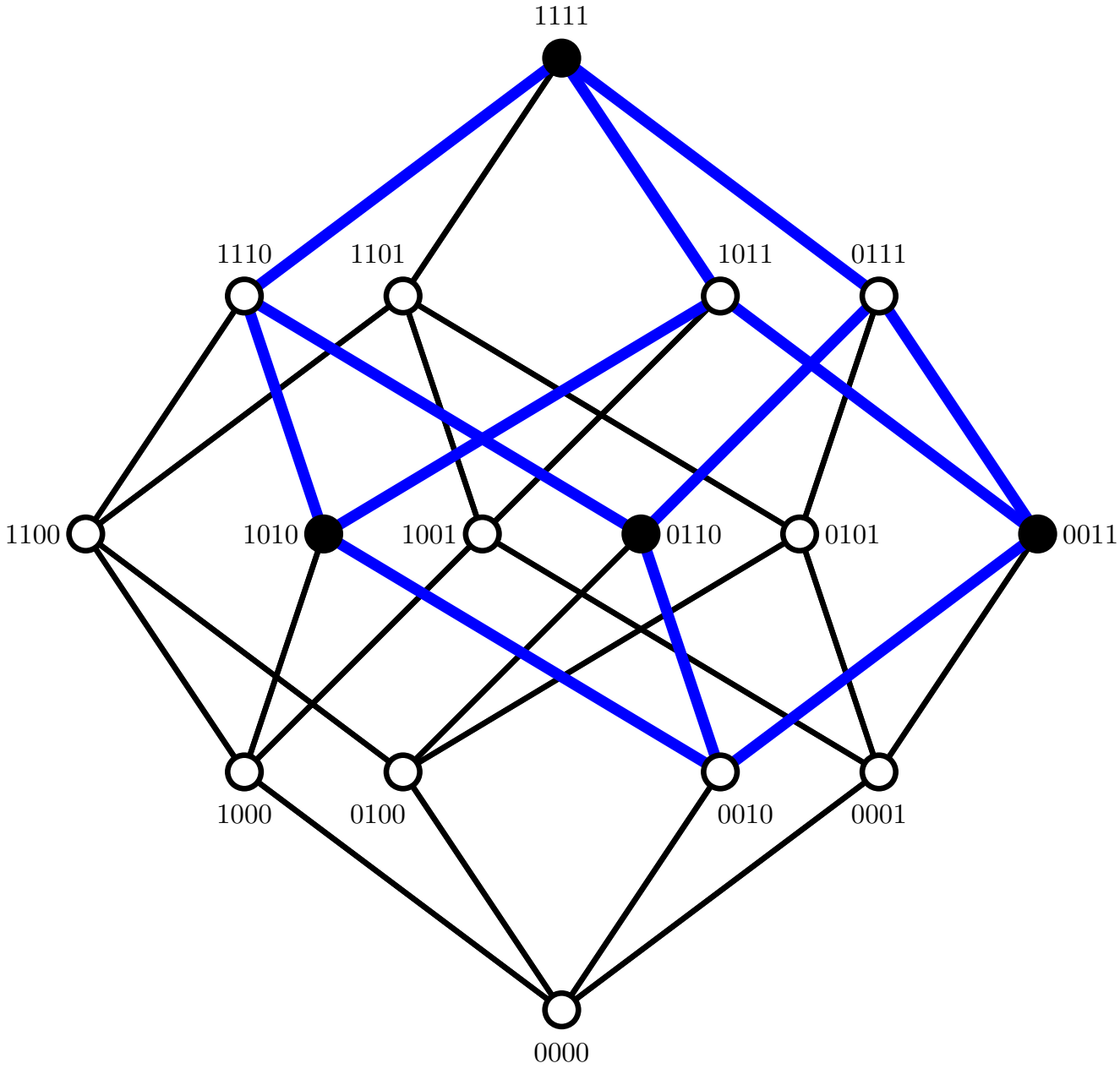
$C' = \{1010, 0110, 0011, 1111\}$



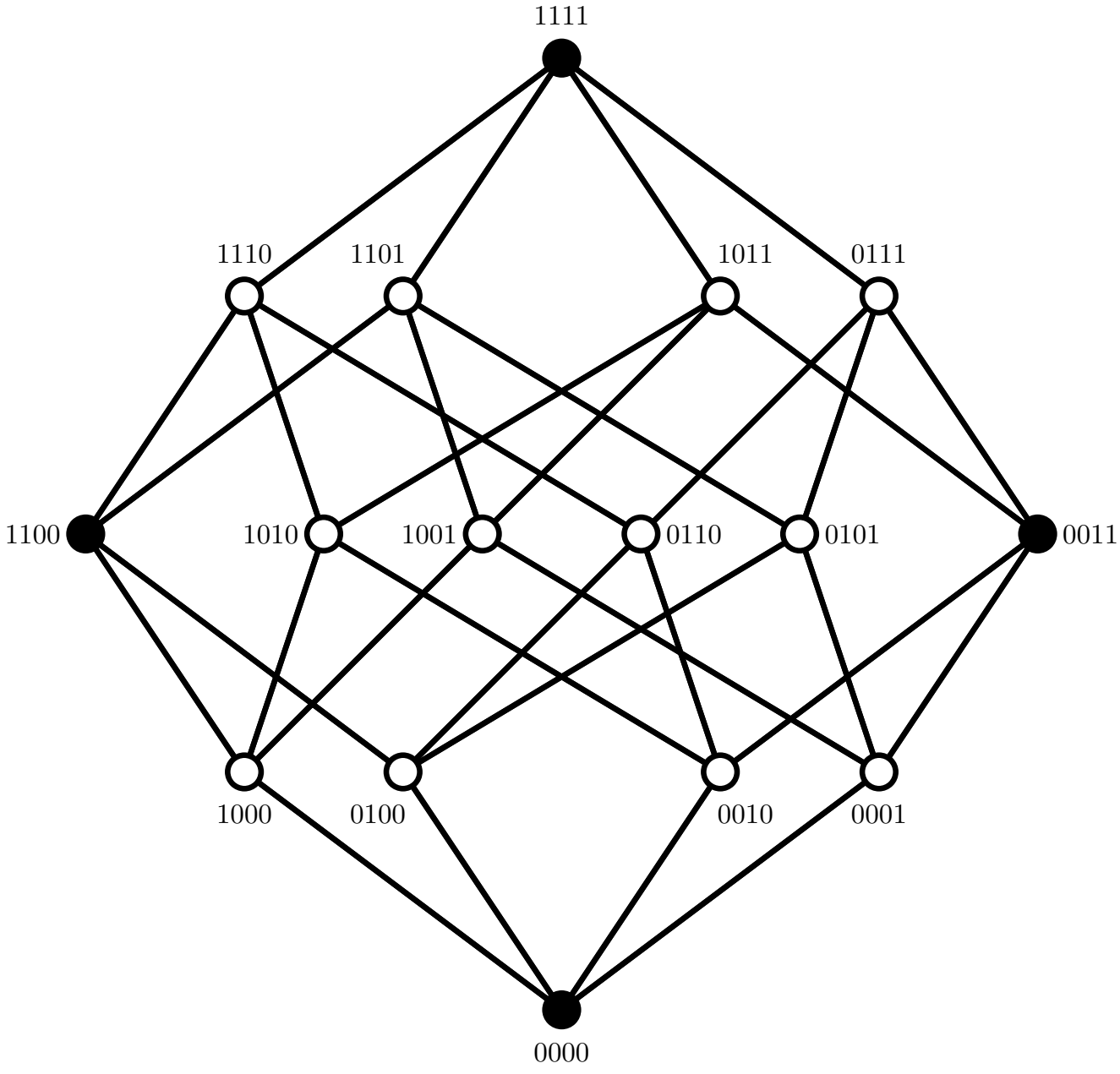
$C = \{0000, 1100, 1010, 0110\}$



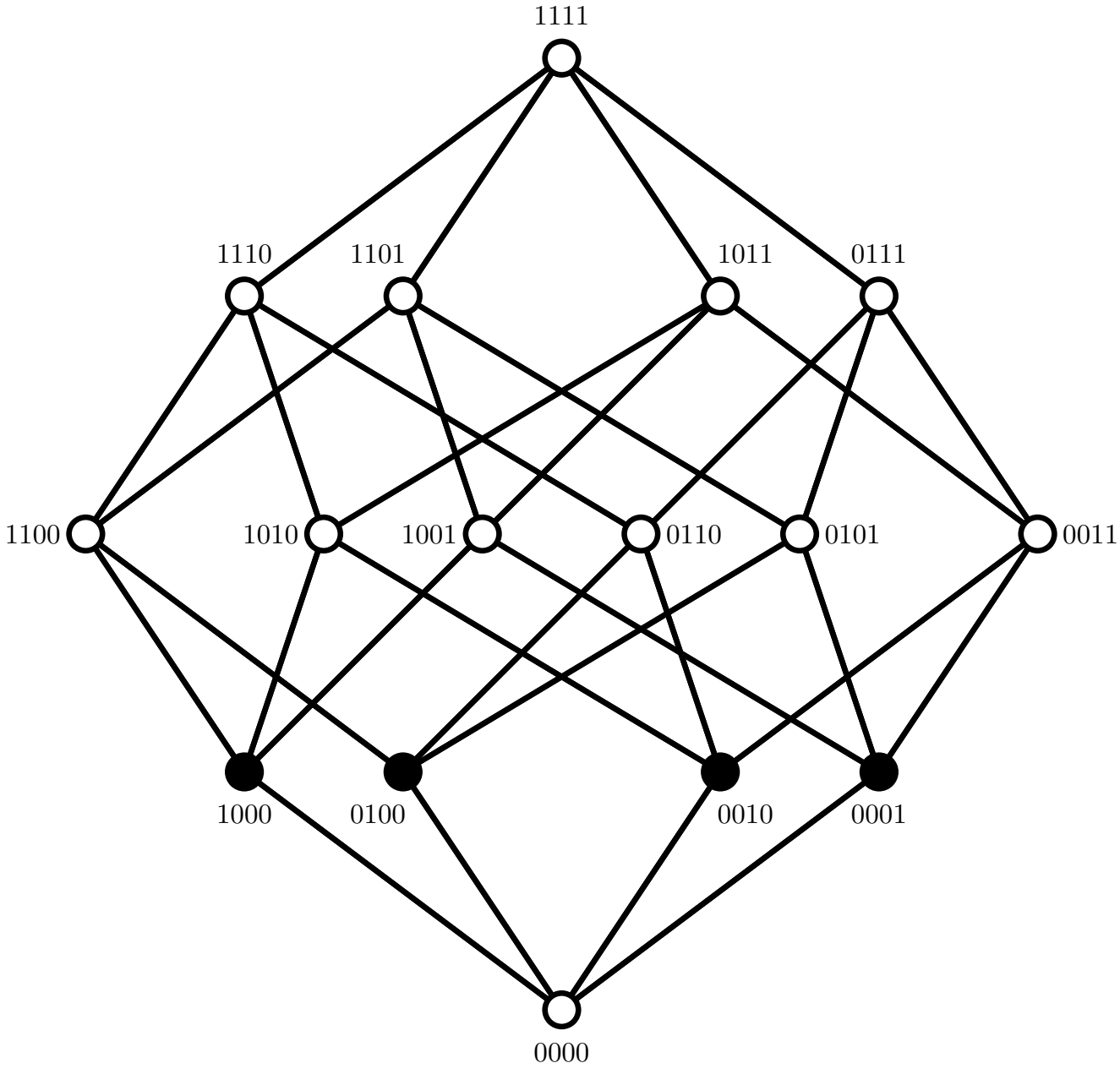
$C' = \{1010, 0110, 0011, 1111\}$



$D = \{0000, 1100, 0011, 1111\}$



$E = \{1000, 0100, 0010, 0001\}$



Example of Equivalences

Example 6.4

Consider the four binary codes

$$C = \{0000, 1100, 1010, 0110\}$$
$$C' = \{1010, 0110, 0011, 1111\}$$
$$D = \{0000, 1100, 0011, 1111\}$$
$$E = \{1000, 0100, 0010, 0001\}.$$

All four codes have minimum distance 2. By applying operations (a) and (b) we will show that C and C' are equivalent. No other two of these codes are equivalent.

Lemma 6.5

$A_2(5, 3) = 4$.

The proof of Lemma 6.5 actually shows something stronger: up to equivalence there is a unique binary $(5, 4, 3)$-code, namely the code $\{00000, 11100, 00111, 11011\}$ first seen in Question 3 on Sheet 1.

Weights and $A_2(8,5)$

The next lemma isolates a result that is useful for Lemma 6.5 and Theorem 8.7. In it, we say that a binary word u has *weight* r, and write $\mathrm{wt}(u) = r$, if exactly r positions of u are equal to 1, and the rest are equal to 0.

Lemma 6.6

Let u and w be binary codewords of length n. Suppose that $\mathrm{wt}(u) = r$ and $\mathrm{wt}(w) = s$. If $r + s \geq n$ then

$$d(u, w) \leq 2n - (r + s).$$

In Question 2 of Sheet 4 you are asked to fill in the details of the proof of the following theorem.

Theorem 6.7

$A_2(8,5) = 4$.

Quiz: True or False?

Let C be a 1-error correcting binary code of length n and size M.

(0) If $R = \left\lfloor \frac{2^n}{1+n} \right\rfloor$ then $M \leq R$.

(1) If $n = 5$ then $M \leq 6$.

(2) If $n = 5$ then $M \leq 4$.

(3) If $n = 5$ then $M \geq 4$.

Let $v \in \{0,1\}^n$.

(4) There exists $u \in C$ such that $d(u,v) \leq 1$.

(5) There exists a unique $u \in C$ such that $d(u,v) \leq 1$.

(6) If $u, u' \in C$ are codewords such that $d(u,v) = d(u',v) = 1$ then $u = u'$.

(7) Suppose that $v \in C$. Then $d(v,u') \geq 3$ for all $u' \in C$.

Quiz: True or False?

Let C be a 1-error correcting binary code of length n and size M.

(0) If $R = \left\lfloor \frac{2^n}{1+n} \right\rfloor$ then $M \leq R$.

True: Hamming's Packing Bound for a 1-error correcting code states that $M \leq 2^n/(1+n)$. We can replace $2^n/(1+n)$ with its floor because C has an integer number of codewords.

Quiz: True or False?

Let C be a 1-error correcting binary code of length n and size M.

(0) If $R = \left\lfloor \frac{2^n}{1+n} \right\rfloor$ then $M \leq R$.

True: Hamming's Packing Bound for a 1-error correcting code states that $M \leq 2^n/(1+n)$. We can replace $2^n/(1+n)$ with its floor because C has an integer number of codewords.

(1) If $n = 5$ then $M \leq 6$.

True: C has minimum distance at least 3 and so by Hamming's Packing Bound

$$|C| \leq \frac{2^5}{1+5} = \frac{32}{6} = 5\tfrac{1}{3}.$$

Hence $|C| \leq 5$, so $|C| \leq 6$. (Or use next question.)

Quiz: True or False?

Let C be a 1-error correcting binary code of length n and size M.

(0) If $R = \left\lfloor \frac{2^n}{1+n} \right\rfloor$ then $M \leq R$.

True: Hamming's Packing Bound for a 1-error correcting code states that $M \leq 2^n/(1+n)$. We can replace $2^n/(1+n)$ with its floor because C has an integer number of codewords.

(1) If $n = 5$ then $M \leq 6$.

True: C has minimum distance at least 3 and so by Hamming's Packing Bound

$$|C| \leq \frac{2^5}{1+5} = \frac{32}{6} = 5\tfrac{1}{3}.$$

Hence $|C| \leq 5$, so $|C| \leq 6$. (Or use next question.)

(2) If $n = 5$ then $M \leq 4$.

True: C has minimum distance at least 3 and we saw on Tuesday that $A_2(5,3) \leq 4$. So $|C| \leq 4$. (This also does (1).)

Quiz: True or False?

Let C be a 1-error correcting binary code of length n and size M.

(0) If $R = \left\lfloor \frac{2^n}{1+n} \right\rfloor$ then $M \leq R$.

True: Hamming's Packing Bound for a 1-error correcting code states that $M \leq 2^n/(1+n)$. We can replace $2^n/(1+n)$ with its floor because C has an integer number of codewords.

(1) If $n = 5$ then $M \leq 6$.

True: C has minimum distance at least 3 and so by Hamming's Packing Bound

$$|C| \leq \frac{2^5}{1+5} = \frac{32}{6} = 5\tfrac{1}{3}.$$

Hence $|C| \leq 5$, so $|C| \leq 6$. (Or use next question.)

(2) If $n = 5$ then $M \leq 4$.

True: C has minimum distance at least 3 and we saw on Tuesday that $A_2(5,3) \leq 4$. So $|C| \leq 4$. (This also does (1).)

(3) If $n = 5$ then $M \geq 4$.

False: $\{00000, 11111\}$ is 1-error correcting and has size 2.

Quiz: True or False? [Continued]

Let C be a 1-error correcting binary code of length n and size M.
Let $v \in \{0,1\}^n$.

(4) There exists $u \in C$ such that $d(u,v) \leq 1$.
False: let $C = \{0000, 1111\}$. Let $v = 1100$. Then there is no codeword in C within distance 1 of v.

Quiz: True or False? [Continued]

Let C be a 1-error correcting binary code of length n and size M.
Let $v \in \{0,1\}^n$.

(4) There exists $u \in C$ such that $d(u,v) \leq 1$.
False: let $C = \{0000, 1111\}$. Let $v = 1100$. Then there is no codeword in C within distance 1 of v.

(5) There exists a unique $u \in C$ such that $d(u,v) \leq 1$.
False: the same example as (4) does it.

Quiz: True or False? [Continued]

Let C be a 1-error correcting binary code of length n and size M.
Let $v \in \{0,1\}^n$.

(4) There exists $u \in C$ such that $d(u,v) \leq 1$.
False: let $C = \{0000, 1111\}$. Let $v = 1100$. Then there is no codeword in C within distance 1 of v.

(5) There exists a unique $u \in C$ such that $d(u,v) \leq 1$.
False: the same example as (4) does it.

(6) If $u, u' \in C$ are codewords such that $d(u,v) = d(u',v) = 1$ then $u = u'$.
True: the Hamming balls of radius 1 about distinct codewords are disjoint by Lemma 4.9. Hence $u = u'$. (Or directly: the words in $B_1(u)$ are decoded to u using nearest neighbour decoding, and the words in $B_1(u')$ are decoded to u' using nearest heighbour decoding, so if $v \in B_1(u) \cap B_1(u')$ then $u = u'$.)

Quiz: True or False? [Continued]

Let C be a 1-error correcting binary code of length n and size M. Let $v \in \{0,1\}^n$.

(4) There exists $u \in C$ such that $d(u,v) \leq 1$.
False: let $C = \{0000, 1111\}$. Let $v = 1100$. Then there is no codeword in C within distance 1 of v.

(5) There exists a unique $u \in C$ such that $d(u,v) \leq 1$.
False: the same example as (4) does it.

(6) If $u, u' \in C$ are codewords such that $d(u,v) = d(u',v) = 1$ then $u = u'$.
True: the Hamming balls of radius 1 about distinct codewords are disjoint by Lemma 4.9. Hence $u = u'$. (Or directly: the words in $B_1(u)$ are decoded to u using nearest neighbour decoding, and the words in $B_1(u')$ are decoded to u' using nearest heighbour decoding, so if $v \in B_1(u) \cap B_1(u')$ then $u = u'$.)

(7) Suppose that $v \in C$. Then $d(v,u') \geq 3$ for all $u' \in C$.
False: take $u' = v$, then $d(v,u') = 0$. (Sorry: if you got everything right except this one, have half a prize.)

§7 Codes from Mutually Orthogonal Latin Squares

Definition 7.1

Let $q \in \mathbf{N}$ and let A be a q-ary alphabet. A *Latin square with entries from* A is a $q \times q$ array in which every row and column contains each symbol in A exactly once. We say that q is the *order* of the square.

Note that since there are q symbols and each row and column has length q, it is equivalent to require *either*

(i) each row and column contains every symbol in A; *or*

(ii) no symbol appears twice in any row or column of A.

Examples of Latin Squares

Example 7.2

A Latin square of order 4 over the alphabet $\{0, 1, 2, 3\}$, constructed using the addition table for the integers modulo 4, is shown below.

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 3 & 0 & 1 & 2 \end{array}$$

Convention: It will be convenient to number the rows and columns of a Latin square over the alphabet $A = \{0, 1, \ldots, q-1\}$ by the numbers in A. So if X is the Latin square in Example 7.2 then $X_{00} = 0$, $X_{12} = 3$ and $X_{33} = 2$.

Mutually Orthogonal Latin Squares

Definition 7.3

Let X and Y be Latin squares over an alphabet A. We say that X and Y are *orthogonal* if for each each a, $b \in A$ there exist unique i, $j \in A$ such that $X_{ij} = a$ and $Y_{ij} = b$.

Equivalently, X and Y are orthogonal if for all a, $b \in A$ there is a unique position in which X contains a and Y contains b. We shall abbreviate 'X and Y are a pair of mutually orthogonal Latin squares', as 'X and Y are MOLs'.

Example 7.4

Two MOLs over the alphabet $\{0, 1, 2, 3\}$ are shown below.

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{array} \qquad \begin{array}{cccc} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \\ 1 & 0 & 3 & 2 \end{array}$$

Exercise: Show that there is no pair of MOLs of order 2.

Hunting for MOLs

Remark 7.5

In 1782 Euler posed the following problem: 36 officers belong to six regiments and hold six different ranks, so that each combination of rank and regiment corresponds to a unique officer. Can be the officers be paraded on a 6×6 parade ground so that in any line each regiment and rank occurs precisely once? Equivalently, does there exist a pair of MOLs of order 6? Euler conjectured that the answer was no, but this was not proved until 1900.

In fact there are pairs of MOLs of all orders other than 2 and 6. Here we will only prove existence for odd prime orders.

Lemma 7.6

Let $q \geq 3$ be prime and let $A = \{0, 1, \ldots, q-1\}$. For $i, j \in A$ let

$$X_{ij} = i + j \bmod q$$
$$Y_{ij} = 2i + j \bmod q$$

Then X and Y are mutually orthogonal Latin squares.

From MOLs to Codes

We now show how to use MOLs to construct a family of 1-error correcting codes. These codes all have length 4 and minimum distance 3.

Theorem 7.7

Let A be the alphabet $\{0, 1, \ldots, q-1\}$. There is a pair of MOLs over A of order q $\iff$ there is a $(4, q^2, 3)$-code over A.

In lectures we will prove the '$\Longrightarrow$' direction. See Question 1 on Sheet 5 for the '$\Longleftarrow$' direction.

Example 7.8

Let X and Y be the MOLs in Example 7.4. The corresponding code has a codeword (i, j, X_{ij}, Y_{ij}) for every i, j such that $0 \leq i, j \leq q-1$. So the codewords are

$$\begin{array}{cccccccc} 0000 & 0111 & 0222 & 0333 & 1012 & 1103 & 1230 & 1321 \\ 2023 & 2132 & 2201 & 2310 & 3031 & 3120 & 3213 & 3302. \end{array}$$

Overview of Theorem 7.7

Let X and Y be MOLs and let C be the corresponding code, so

$$C = \{(i, j, X_{ij}, Y_{ij}) : 1 \leq i, j \leq q - 1\}.$$

In words: C has a codeword (i, j, x, y) if and only if the MOLS X and Y have symbols x and y in position (i, j).

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

Overview of Theorem 7.7

Let X and Y be MOLs and let C be the corresponding code, so

$$C = \{(i, j, X_{ij}, Y_{ij}) : 1 \leq i, j \leq q - 1\}.$$

In words: C has a codeword (i, j, x, y) if and only if the MOLS X and Y have symbols x and y in position (i, j).

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

Overview of Theorem 7.7

Let X and Y be MOLs and let C be the corresponding code, so

$$C = \{(i, j, X_{ij}, Y_{ij}) : 1 \leq i, j \leq q - 1\}.$$

In words: C has a codeword (i, j, x, y) if and only if the MOLS X and Y have symbols x and y in position (i, j).

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

Overview of Theorem 7.7

Let X and Y be MOLs and let C be the corresponding code, so

$$C = \{(i, j, X_{ij}, Y_{ij}) : 1 \leq i, j \leq q - 1\}.$$

In words: C has a codeword (i, j, x, y) if and only if the MOLS X and Y have symbols x and y in position (i, j).

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

Overview of Theorem 7.7

Let X and Y be MOLs and let C be the corresponding code, so

$$C = \{(i, j, X_{ij}, Y_{ij}) : 1 \leq i, j \leq q - 1\}.$$

In words: C has a codeword (i, j, x, y) if and only if the MOLS X and Y have symbols x and y in position (i, j).

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

Administration

- Sheet 5, Question 3(a) should read 'Show that if $u = u_1 u_2 \dots u_n$ and $v = v_1 v_2 \dots v_n$ are **distinct** codewords in C then $\dots$'.
- Example 7.8: replace $1 \leq i, j \leq q - 1$ with $0 \leq i, j \leq q - 1$.

In Praise of Abstraction

Lemma 7.6

Let $q \geq 3$ be prime and let $A = \{0, 1, \ldots, q-1\}$. For $i, j \in A$ let $X_{ij} = i + j$ mod q and let $Y_{ij} = 2i + j$ mod q. Then X and Y are mutually orthogonal Latin squares.

- To show that the columns of Y have no repeats we needed:

$$2i + j = 2i' + j \text{ mod } q \Longrightarrow i = i' \text{ mod } q.$$

The obvious first step is to subtract off j, so we have $2i = 2i'$ mod q. To finish we can either:

- multiply through by $(q+1)/2$ (as in lecture).
- use that 2 is invertible mod q with inverse $(q+1)/2$.
- note that since q is prime, $\mathbf{Z}/q\mathbf{Z}$ is a field, so 2 is invertible.

In Praise of Abstraction

Lemma 7.6

Let $q \geq 3$ be prime and let $A = \{0, 1, \ldots, q-1\}$. For $i, j \in A$ let $X_{ij} = i + j \bmod q$ and let $Y_{ij} = 2i + j \bmod q$. Then X and Y are mutually orthogonal Latin squares.

- To show that X and Y are MOLS:
  - In lecture: if $i + j = x$, $2i + j = y$ then, solving the equations, we get $i = y - x$, $j = 2x - y$. So the pair (x, y) appears only in the position $(y - x, 2x - y)$, taken mod q, and there are no repeated pairs of symbols.
  - If $i + j = x$, $2i + j = y$ then

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}.$$

    This equation has a unique solution for i and j because the matrix is invertible.

Question 1 on Sheet 3: Show that C_4 and C_5 are not equivalent

Recall that

$$C_4 = \{0000, 1100, 0110, 0011\}$$
$$C_5 = \{0110, 1100, 1001, 0011\}.$$

It is tempting to make an argument using that three of the codewords in C_4 appear in C_5. But this is hard to justify: **it is not true that an equivalence of codes must fix the codewords common to both codes**.

For example, we have seen that

$$\{0000, 1100, 1111\},$$
$$\{0000, 1100, 0011\}$$

are equivalent, but only by an equivalence that sends 0000 in the first code to either 1100 or 0011 in the second.

Question 1 on Sheet 3: Show that C_4 and C_5 are not equivalent

Recall that

$$C_4 = \{0000, 1100, 0110, 0011\}$$
$$C_5 = \{0110, 1100, 1001, 0011\}.$$

It is tempting to make an argument using that three of the codewords in C_4 appear in C_5. But this is hard to justify: **it is not true that an equivalence of codes must fix the codewords common to both codes**.

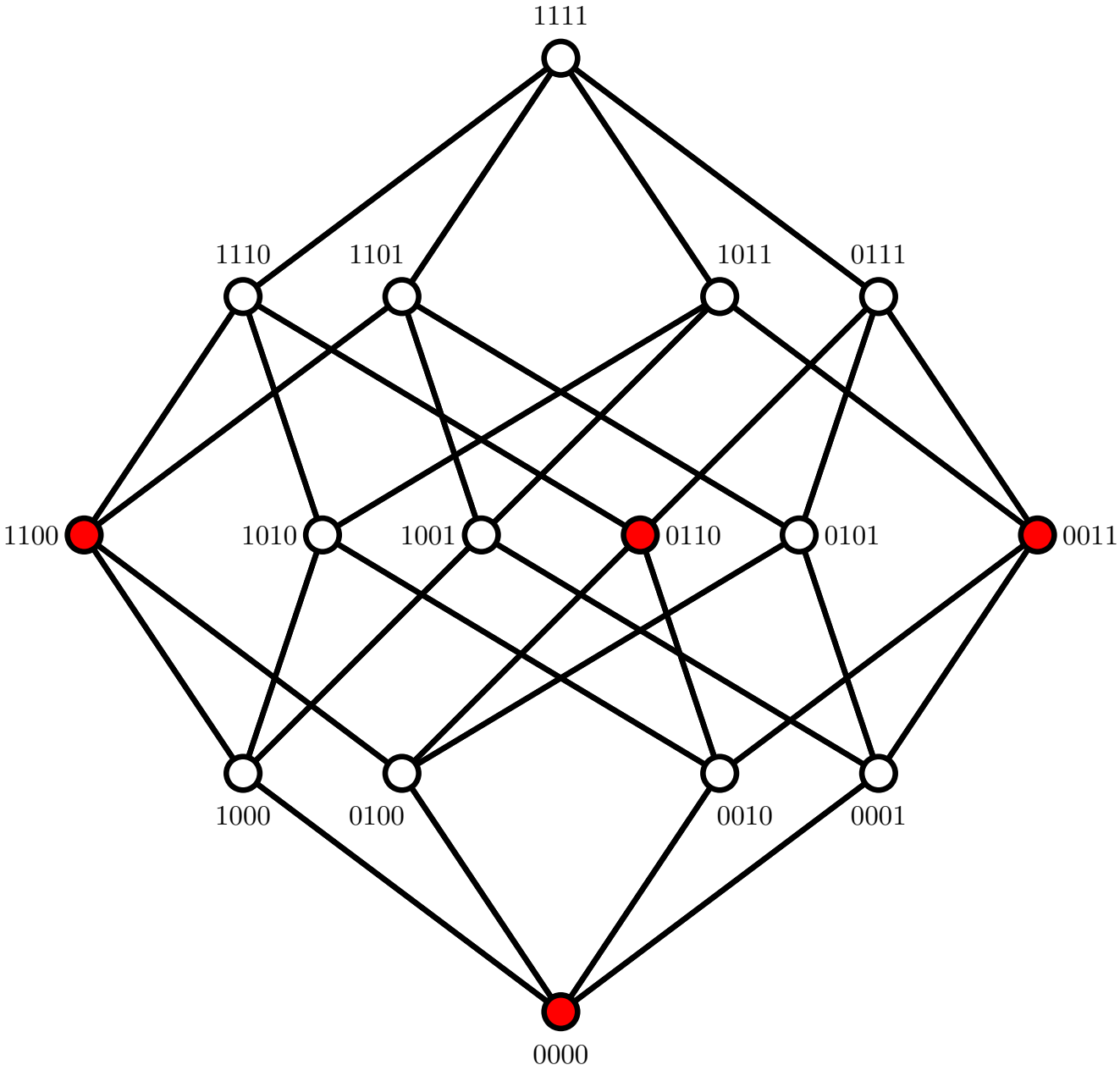
For example, we have seen that

$$\{0000, 1100, 1111\},$$
$$\{0000, 1100, 0011\}$$

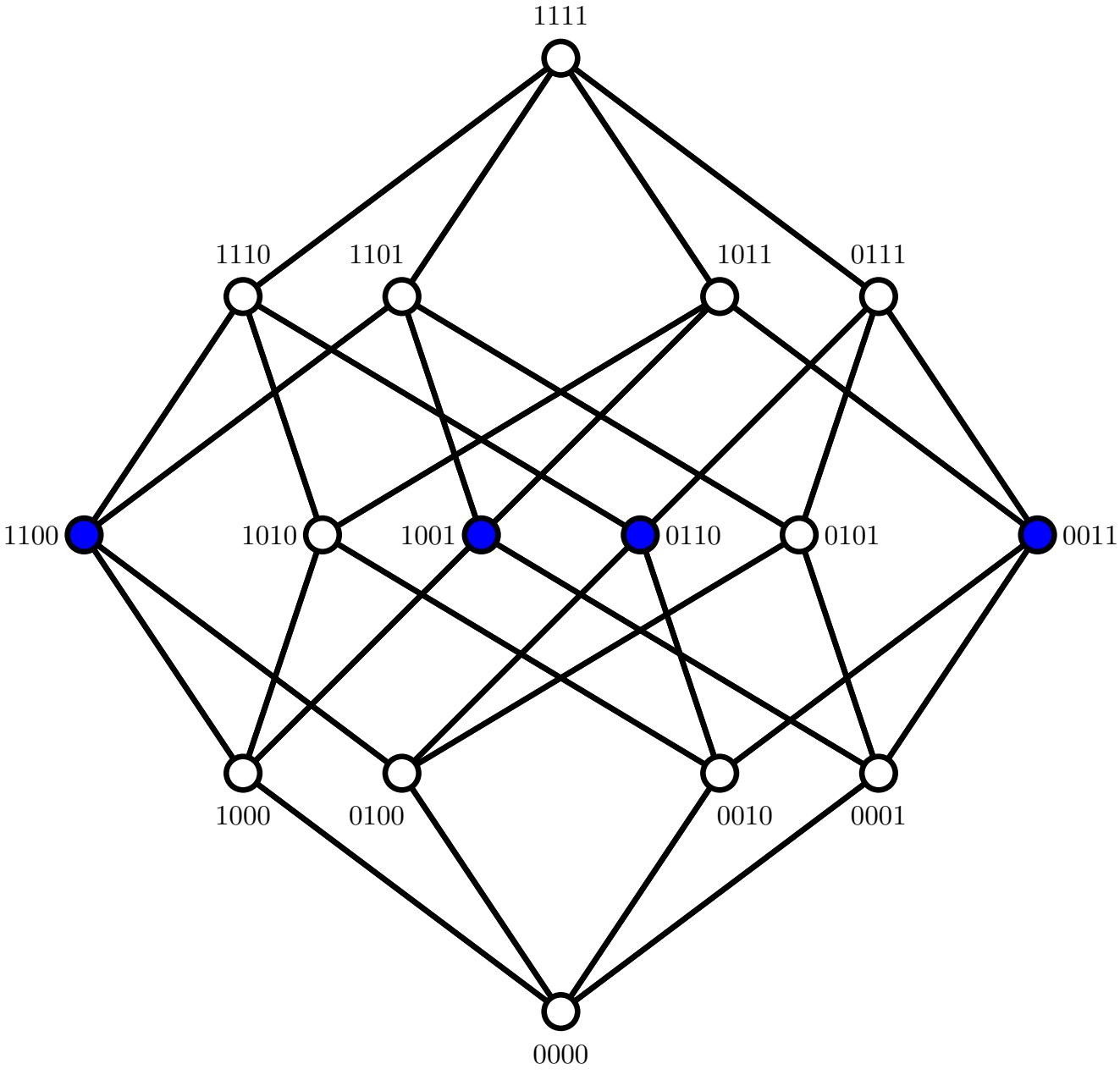
are equivalent, but only by an equivalence that sends 0000 in the first code to either 1100 or 0011 in the second.

Finally, please do not write 'code' if you mean 'codeword'.

Question 1, Sheet 3: C_4



Question 1, Sheet 4: C_5



§8 The Singleton Bound and Puncturing a Code

Definition 8.1

Let C be a code of length $n \geq 2$ and minimum distance ≥ 2. Let $C^\star$ be the code whose codewords are obtained by removing the final position from each codeword in C. We say that $C^\star$ is obtained by *puncturing C in its final position*.

Note that since C has minimum distance ≥ 2, it is impossible for two codewords in C to become equal when their final position is removed. So $C^\star$ has the same size as C.

Example 8.2

Let C be the binary code whose codewords are all binary words of length 4 with an even number of 1s. Let $C^\star$ be the code obtained by puncturing C in its final position. Then

$$C = \{0000, 1100, 1010, 0110, 1001, 0101, 0011, 1111\}$$
$$C^\star = \{000, 110, 101, 011, 100, 010, 001, 111\}$$

Thus C has minimum distance 2 and $C^\star$ has minimum distance 1.

Extra Example using the MOLS code from Example 7.4

Recall that starting from the MOLS

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

we obtained a 4-ary $(4, 16, 3)$-code

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

Let $u, w \in C$ be distinct. If we puncture u and w twice we get distinct words, since $d(u, w) \geq 3$.

So puncturing C twice, in any two positions, gives the code containing all q-ary codewords of length 2. For example puncture in second and final position:

00 01 02 03 11 10 13 12
22 23 20 21 33 32 31 30.

Extra Example using the MOLS code from Example 7.4

Recall that starting from the MOLS

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	1	2	3
2	3	0	1
3	2	1	0
1	0	3	2

00	11	22	33
12	03	30	21
23	32	01	10
31	10	13	02

we obtained a 4-ary $(4, 16, 3)$-code

0000 0111 0222 0333 1012 1103 1230 1321
2023 2132 2201 2310 3031 3120 3213 3302.

Let $u, w \in C$ be distinct. If we puncture u and w twice we get distinct words, since $d(u, w) \geq 3$.

So puncturing C twice, in any two positions, gives the code containing all q-ary codewords of length 2. For example puncture in third and final position:

00 11 22 33 12 03 30 21
23 32 01 10 31 20 13 02.

Singleton Bound

Lemma 8.3

Let C be a code of length n and minimum distance d. The punctured code $C^\star$ has length $n-1$ and minimum distance $\geq d-1$.

Theorem 8.4 (Singleton Bound)

If C is a q-ary code of length n and minimum distance d then $|C| \leq q^{n-d+1}$. Hence $A_q(n,d) \leq q^{n-d+1}$.

Remarks 8.5

(1) If $n = 4$ and $d = 3$ then the Singleton bound gives $A_q(4, 3) \leq q^{4-3+1} = q^2$. The codes constructed by MOLs achieve the bound. So there is a pair of MOLs of order q if and only if $A_q(4, 3) = q^2$.

(2) The Reed–Solomon codes constructed in the MSc/MSci course achieve the Singleton bound. They show that $A_q(n, d) = q^{n-d+1}$ whenever q is a prime power and $q \geq n$.

(3) The special case of the Singleton bound when $d = n$ is

$$A_q(n, n) \leq q.$$

This was proved in Lemma 6.1(ii) by putting codewords into pigeonholes according to their first position. A similar argument can be used to prove the general Singleton bound: see Questions 3 and **7 [not 6]** on Sheet 5.

§9 Hadamard Codes and the Plotkin Bound

Hadamard codes are a family of binary codes that have high minimum distance and so can detect and correct many errors. We shall see that, like the codes constructed from MOLs, Hadamard codes have the largest possible size for their length and minimum distance.

Hadamard codes are constructed using certain matrices with entries $+1$ and -1.

Definition 9.1

Let $n \in \mathbf{N}$. A *Hadamard matrix of order n* is an $n \times n$ matrix H such that each entry of H is either $+1$ or -1 and $HH^{tr} = nI$. Here I is the $n \times n$ identity matrix and H^{tr} is the transpose matrix of H.

Example 9.2

If $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ then H is a Hadamard matrix of order 2. Two Hadamard matrices of order 4 are shown below; in these matrices we write $+$ for 1 and $-$ for -1.

$$\begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}, \qquad \begin{pmatrix} + & + & + & - \\ + & + & - & + \\ + & - & + & + \\ - & + & + & + \end{pmatrix}.$$

Except for the 1×1 matrices $(+1)$ and (-1), all Hadamard matrices have even order.

Lemma 9.3

Suppose H is a Hadamard matrix of order n where $n \geq 2$. If i, $k \in \{1, 2, \ldots, n\}$ and $i \neq k$ then row i and row k of H are equal in exactly $n/2$ positions.

Sheet 5

- MSc/MSci: Red folder
- 361: Green folder
- Model answers available from Moodle. Please see lecturer if you want to go through any of it.

From Hadamard Matrices to Codes

The connection with coding theory is as follows.

Theorem 9.4

Suppose that H is a Hadamard matrix of order $n \geq 2$. Let B be the $2n \times n$ matrix defined by

$$B = \begin{pmatrix} H \\ -H \end{pmatrix}.$$

The rows of B are the codewords in a $(n, 2n, n/2)$-code over the alphabet $\{+, -\}$.

We say that any code given by the construction in Theorem 9.4 is a *Hadamard code*. These codes can be converted into binary codes over the usual alphabet of bits $\{0, 1\}$ by replacing each $+$ with 0 and each $-$ with 1.

Example of a Hadamard Code

Example 9.5

Let

$$H = \begin{pmatrix} + & + & + & - \\ + & + & - & + \\ + & - & + & + \\ - & + & + & + \end{pmatrix}.$$

The construction in Theorem 9.4 gives the binary code with codewords

$$\begin{array}{cccc} 0001 & 0010 & 0100 & 1000 \\ 1110 & 1101 & 1011 & 0111. \end{array}$$

Example 1.14: Mariner 9

The Mariner 9 probe, launched in 1971, took the first pictures of Mars, ultimately transmitting 7239 pictures at a resolution of 700×832 pixels. The images were grey-scale, using $64 = 2^6$ different shades of grey.

- The probe did not have the internal memory to store even one image, so the image had to be transmitted as it was captured.
- The pictures were transmitted back to Earth by sending one pixel at a time, so we can think of a message as a single number between 0 and 63.
- The channel could send the two binary digits 0 and 1. The probability of each bit being flipped in the channel was about 0.05.
- Had each pixel been encoded using 6 bits, about 26% of the image would have been wrong.

Codes for Mariner 9

It was acceptable for each pixel to be encoded by up to 32 bits, so increasing the amount of data to be stored and transmitted by a factor of 5.

A repetition code where each of the 6 bits needed to specify a grey level was repeated 5 times would reduce the percentage of incorrect pixels to less than 4%.

Instead a $(32, 64, 16)$-Hadamard code was used. Since the minimum distance is 16, by Theorem 4.5(ii), the code is 7-error correcting. This reduced the percentage of incorrect pixels to about 0.014%.

Importantly, there was a way to do nearest neighbour decoding on received words that was fast even on the primitive computers available in the 70s.

Codes for Mariner 9

It was acceptable for each pixel to be encoded by up to 32 bits, so increasing the amount of data to be stored and transmitted by a factor of 5.

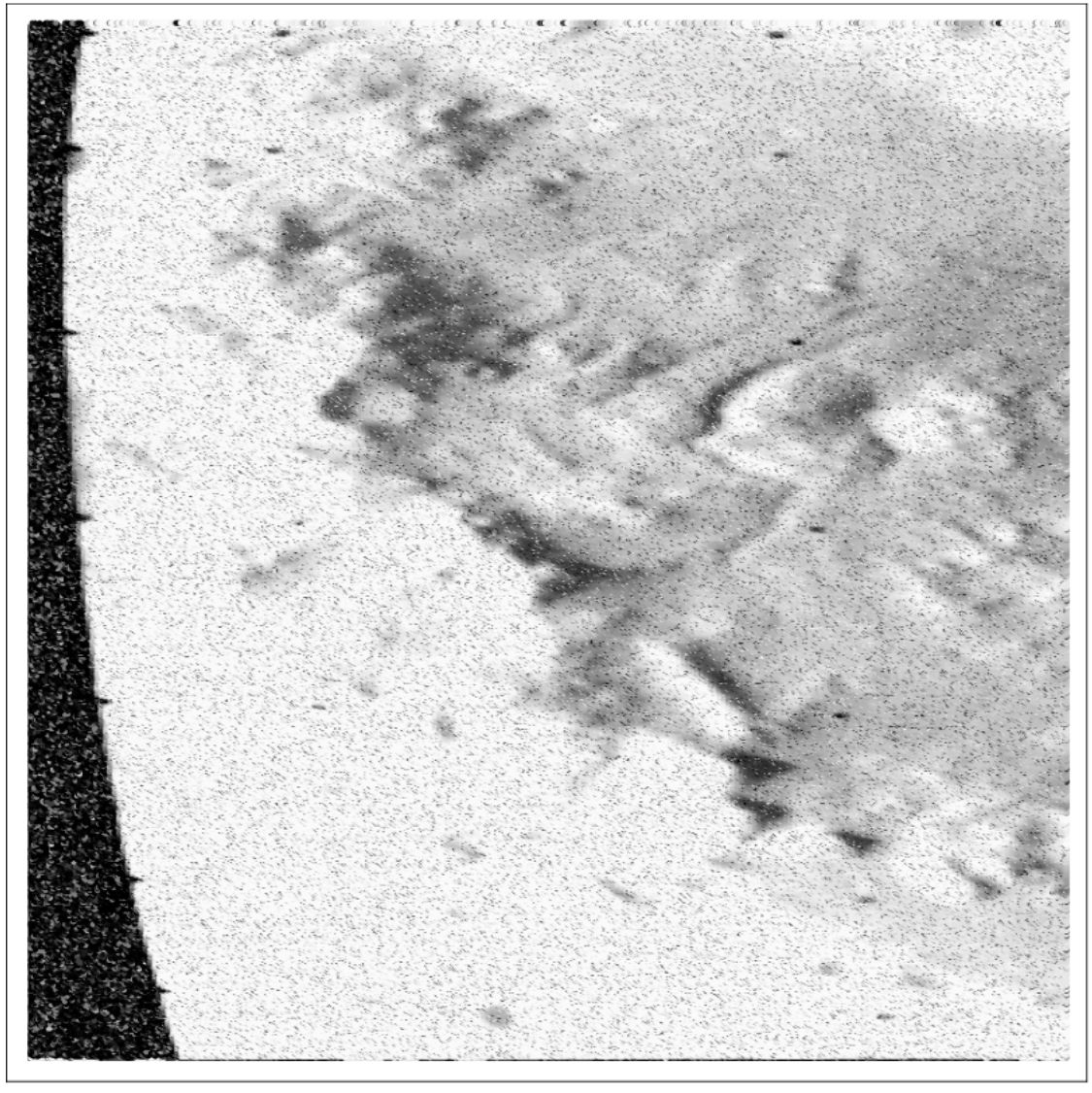
A repetition code where each of the 6 bits needed to specify a grey level was repeated 5 times would reduce the percentage of incorrect pixels to less than 4%.

Instead a $(32, 64, 16)$-Hadamard code was used. Since the minimum distance is 16, by Theorem 4.5(ii), the code is 7-error correcting. This reduced the percentage of incorrect pixels to about 0.014%.

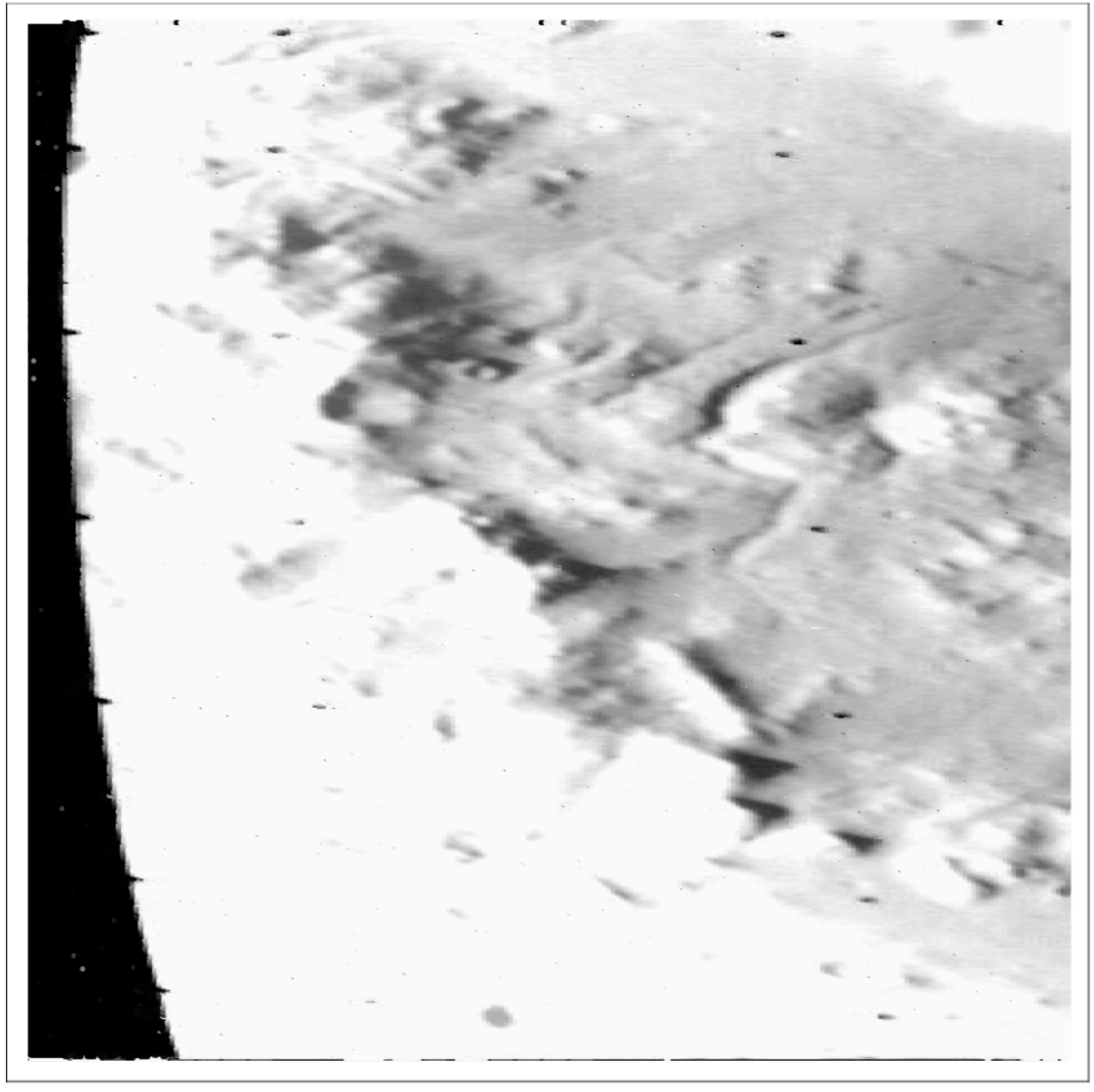
Importantly, there was a way to do nearest neighbour decoding on received words that was fast even on the primitive computers available in the 70s.

The next slide shows a Mariner 9 as it might have been received had each pixel been encoded as a 6 bit binary word, without any attempt at error correcting.

Mariner 9 Image: Using One Word of Length 6 Per Pixel

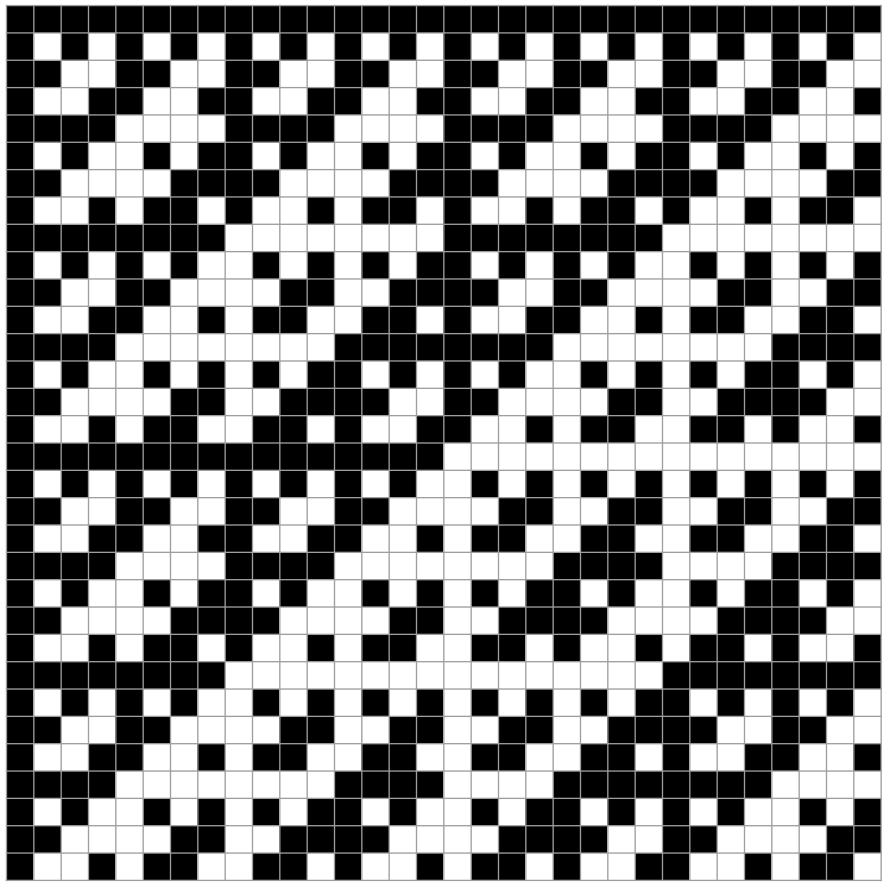


Mariner 9 Image: Using Hadamard Code of Length 32



The Mariner 9 Code

32 of the 64 Mariner 9 codewords: ■ = 0 and □ = 1. Suppose we receive the word below. How should we do nearest neighbour decoding? Comparing with all 64 codewords takes some time ...



Plotkin Bound

The Singleton bound is often the strongest bound for codes over a large alphabet, but for a binary $(2d, M, d)$-code it only gives the bound $M \leq 2^{d+1}$. The following result leads to a stronger bound on $A_2(2d, d)$.

Theorem 9.6 (Plotkin bound)

Let n, $d \in \mathbf{N}$ be such that $2d > n$. Then

$$A_2(n, d) \leq \frac{2d}{2d - n}.$$

Exercise: Use the Plotkin bound to prove that $A_2(9, 6) = 4$. Can the Plotkin bound be used to show that $A_2(8, 5) = 4$?

Hadamard Codes are as Large as Possible

A related bound is attained by Hadamard codes.

Corollary 9.7 (Another Plotkin bound)

If $d \in \mathbf{N}$ *then*

$$A_2(2d, d) \leq 4d.$$

If there is a Hadamard matrix of order $2d$ *then*

$$A_2(2d, d) = 4d.$$

Hadamard Codes are as Large as Possible

A related bound is attained by Hadamard codes.

Corollary 9.7 (Another Plotkin bound)

If $d \in \mathbf{N}$ *then*

$$A_2(2d, d) \leq 4d.$$

If there is a Hadamard matrix of order $2d$ *then*

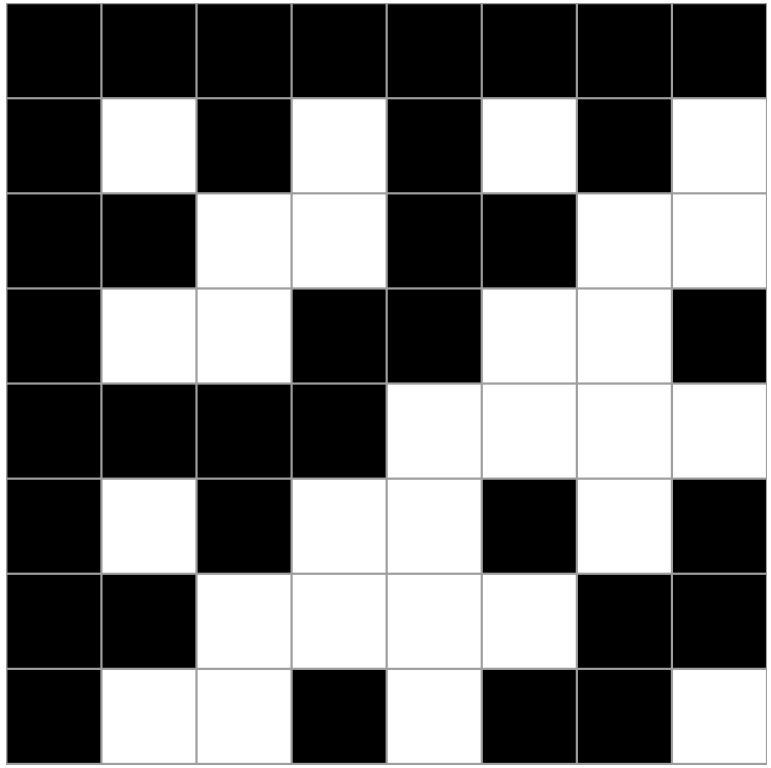
$$A_2(2d, d) = 4d.$$

It is quite easy to show that if there is a Hadamard matrix of order n then either $n = 1$, or $n = 2$ or n is divisible by 4. It is a major open problem to show that there are Hadamard matrices of all orders divisible by 4.

There is also a related 'asymptotic' Plotkin bound, which states that $A_2(n, d) \leq 2^{n-2d+1}n$ for all n and d.

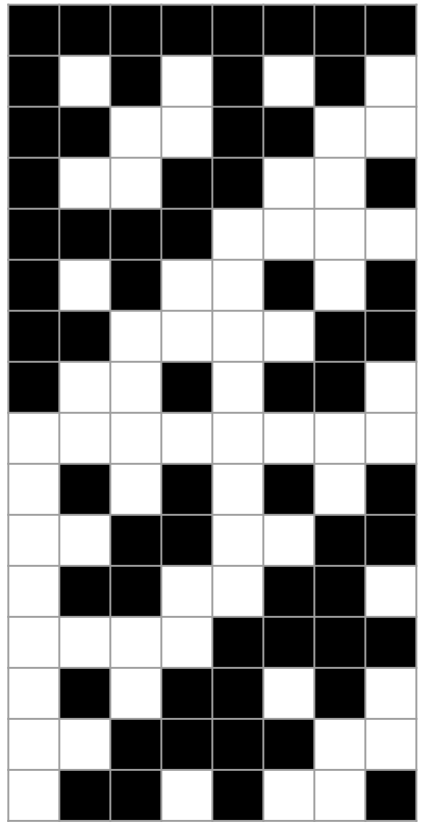
Example of Corollary 9.7

To see how the arguments works we take the Hadamard matrix of order 8 shown below. (This matrix comes from the 'doubling' construction in Question 2 of Sheet 6 by doubling the Hadamard matrix of order 4 seen on Monday.) Black squares show $+1$ and white squares show -1.



Example of Corollary 9.7

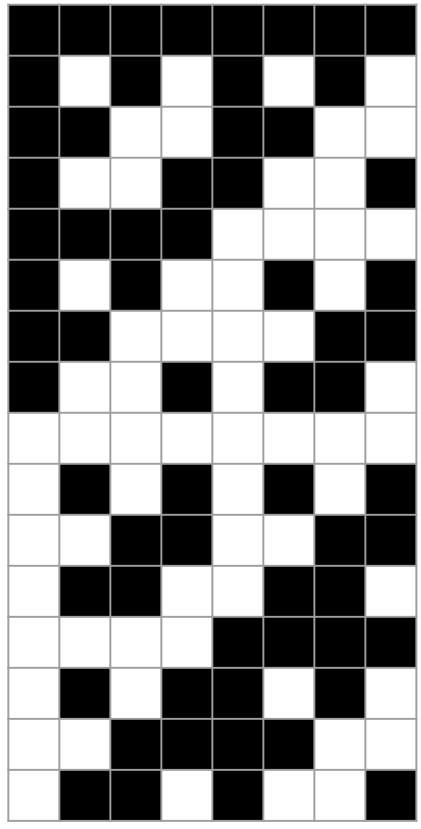
The corresponding code has 16 codewords and minimum distance 4.



There are 8 codewords ending $+1$ (black) and 8 codewords ending -1 white, so we can take either set.

Example of Corollary 9.7

The corresponding code has 16 codewords and minimum distance 4.

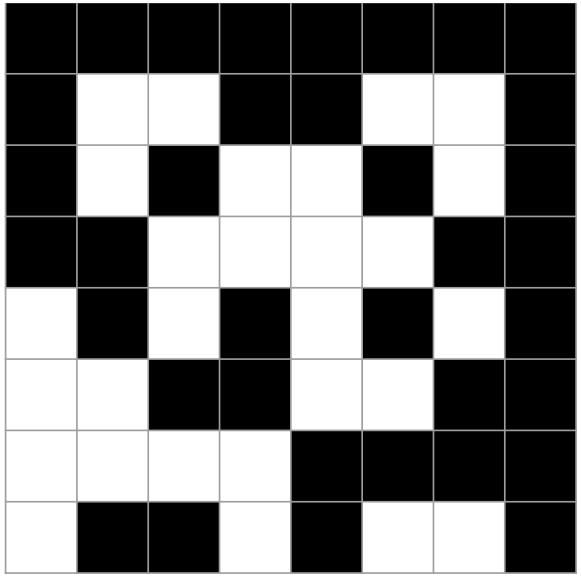


+	+	+	+	+	+	+	+
+	−	+	−	+	−	+	−
+	+	−	−	+	+	−	−
+	−	−	+	+	−	−	+
+	+	+	+	−	−	−	−
+	−	+	−	−	+	−	+
+	+	−	−	−	−	+	+
+	−	−	+	−	+	+	−
−	−	−	−	−	−	−	−
−	+	−	+	−	+	−	+
−	−	+	+	−	−	+	+
−	+	+	−	−	+	+	−
−	−	−	−	+	+	+	+
−	+	−	+	+	−	+	−
−	−	+	+	+	+	−	−
−	+	+	−	+	−	−	+

There are 8 codewords ending $+1$ (black) and 8 codewords ending -1 white, so we can take either set.

Example of Corollary 9.7

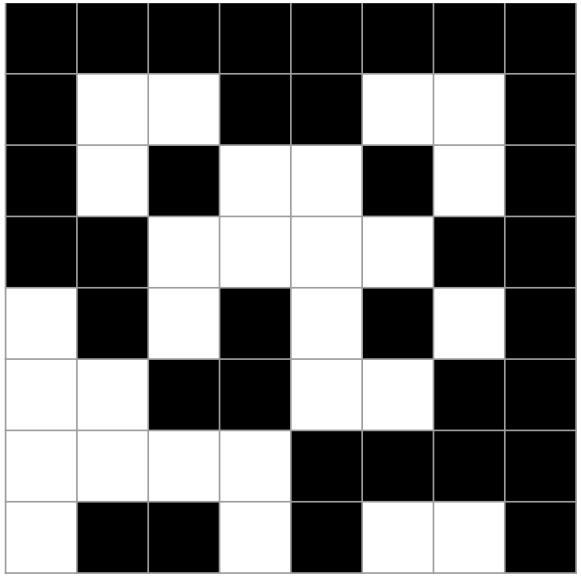
Taking the 8 codewords ending $+1$ (black) gives this code.

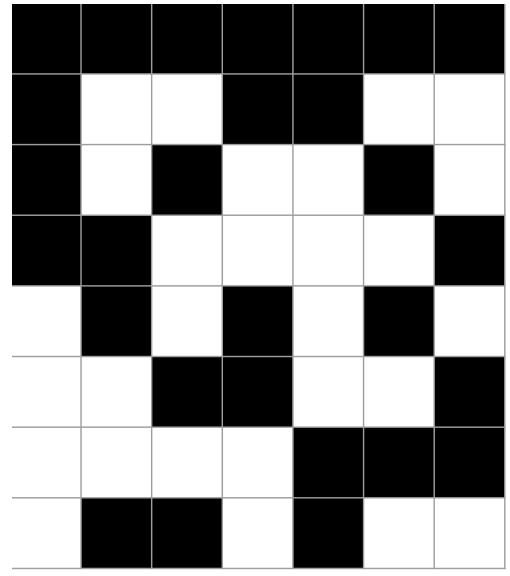


We then puncture to remove the final constant position. In the last step of the proof we apply the Plotkin Bound to the resulting code of length 7 and minimum distance 4. (In this example, this code is as large as possible.)

Example of Corollary 9.7

Taking the 8 codewords ending $+1$ (black) gives this code.





We then puncture to remove the final constant position. In the last step of the proof we apply the Plotkin Bound to the resulting code of length 7 and minimum distance 4. (In this example, this code is as large as possible.)

§10 Gilbert–Varshamov Bound

Recall from Definition 4.7 that the Hamming ball of radius t about a binary word $u \in \{0,1\}^n$ was defined by

$$B_t(u) = \{v \in \{0,1\}^n : d(u,v) \leq t\}.$$

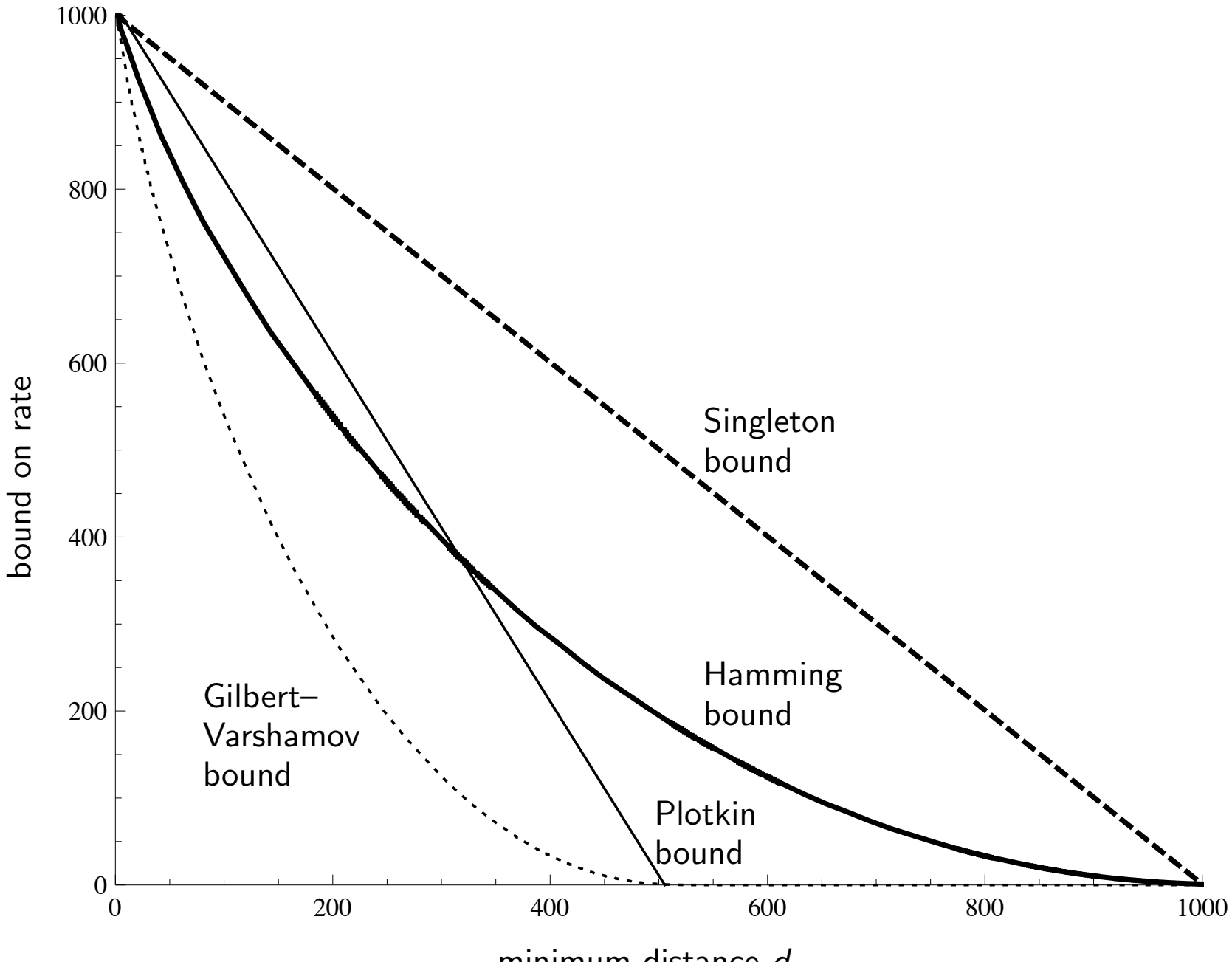
The idea in the next theorem is to construct a code of minimum distance d in the most naïve way possible: we put in new codewords until the Hamming balls of radius $(d-1)$ about codewords cover $\{0,1\}^n$, and so every word is distance $\leq (d-1)$ from some codeword.

Theorem 10.1 (Gilbert–Varshamov bound)

If $n, d \in \mathbf{N}$ *then*

$$A_2(n,d) \geq \frac{2^n}{\sum_{k=0}^{d-1} \binom{n}{k}}.$$

Comparison of Bounds



Question 3 Sheet 4

A binary code C of length n is said to be *perfect* if there exists $e \in \mathbf{N}$ such that

$$\{0,1\}^n = \bigcup_{u \in C} B_e(u)$$

where the union is disjoint. (In words: the Hamming balls of radius e about codewords are disjoint, and every binary word of length n is in one of these balls.)

(a) Show that if n is odd then the binary repetition code of length n is perfect.

(b) Show that if C is a perfect binary code of length n with $e = 1$ then C is 1-error correcting and $n = 2^m - 1$ for some $m \in \mathbf{N}$. Express $|C|$ in terms of m. [*You may use any general results proved earlier in the course.*]

(c) (**Optional**) Show that any perfect binary code has odd minimum distance.

Part C: Linear codes

§11 Linear Codes and Weights

From now on the alphabet of of bits $\{0, 1\}$ should be thought of as $\mathbf{Z}_2$, that is, the integers modulo 2. So we have

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0.$$

Binary words of length n are elements of $\mathbf{Z}_2^n$. Given $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n) \in \mathbf{Z}_2^n$, we define

$$(u_1, u_2, \ldots, u_n) + (v_1, v_2, \ldots, v_n) = (u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n).$$

Definition 11.1

Let C be a binary code of length n. We say that C is *linear* if for all $u, w \in C$ we have $u + w \in C$.

Example 11.2

(1) The length 5 code $\{00000, 11100, 00111, 11011\}$ is linear.

(2) For any $n \in \mathbf{N}$, the binary repetition code of length n is a linear $(n, 2, n)$-code.

(3) For any $n \in \mathbf{N}$, the code of size 2^n consisting of all binary words of length n is a linear $(n, 2^n, 1)$-code.

(4) Let C be all binary words of length 4. As in Example 2.9, let C_{ext} be the code obtained by adding an extra bit at the end of each codeword to make the total number of 1s in each codeword even. Then, C_{ext} is a $(5, 16, 2)$-code and

$$C_{\text{ext}} = \{(u_1, u_2, u_3, u_4, u_5) \in \mathbf{Z}_2^5 : u_1 + u_2 + u_3 + u_4 + u_5 = 0\}.$$

We will show that C_{ext} is linear.

Exercise: Show that the Square Code (see Question 2 on the Preliminary Problem Sheet) is linear.

Sheet 6. 361: Blue folder, MSci/Msc: Red folder.

Spare copies of Part C handout and Sheet 7

(b) Starting from the 2×2 Hadamard matrix

$$\begin{pmatrix} + & + \\ + & - \end{pmatrix}$$

use (a) to construct Hadamard matrices with orders 4 and 8. Hence write down the codewords in (i) a binary $(4, 8, 2)$ code and (ii) a binary $(8, 16, 4)$-code.

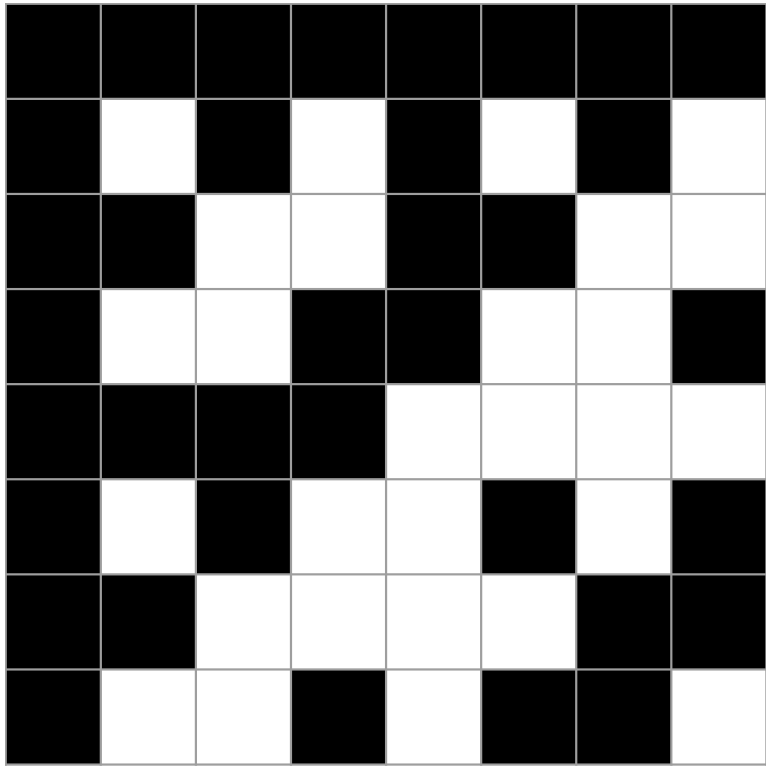
(c) Use nearest neighbour decoding to decode (where possible) the received words 01010111, 10011110 and 11000000 sent using the code in (b)(ii).

(d) Use (b)(ii) to show that there is a binary $(7, 16, 3)$-code. Deduce from the Hamming Packing Bound that $A_2(7, 3) = 16$.

(e) Using (b)(ii) and the Plotkin bound, prove that $A_2(7, 4) = 8$.

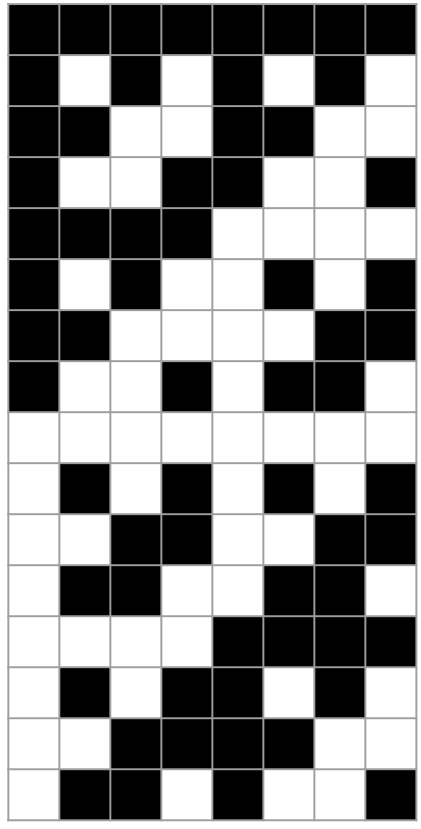
Sheet 6, Question 2

From Hadamard matrix of order 8 to $(8, 16, 4)$-code



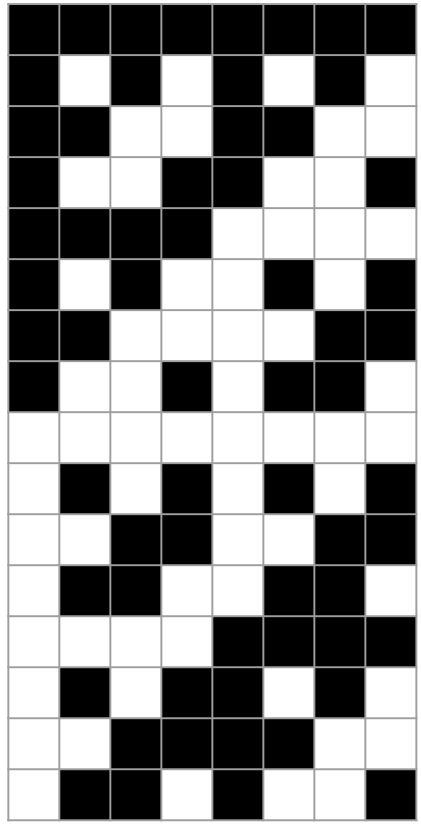
Sheet 6, Question 2

From Hadamard matrix of order 8 to $(8, 16, 4)$-code



Sheet 6, Question 2

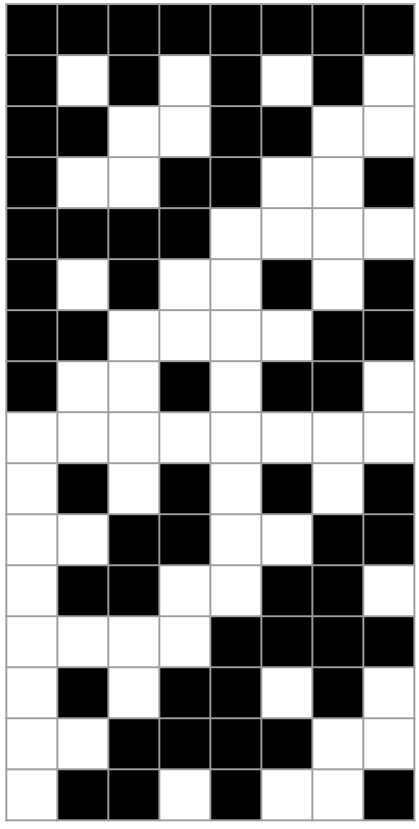
From Hadamard matrix of order 8 to $(8, 16, 4)$-code



+	+	+	+	+	+	+	+
+	−	+	−	+	−	+	−
+	+	−	−	+	+	−	−
+	−	−	+	+	−	−	+
+	+	+	+	−	−	−	−
+	−	+	−	−	+	−	+
+	+	−	−	−	−	+	+
+	−	−	+	−	+	+	−
−	−	−	−	−	−	−	−
−	+	−	+	−	+	−	+
−	−	+	+	−	−	+	+
−	+	+	−	−	+	+	−
−	−	−	−	+	+	+	+
−	+	−	+	+	−	+	−
−	−	+	+	+	+	−	−
−	+	+	−	+	−	−	+

Sheet 6, Question 2

From Hadamard matrix of order 8 to $(8, 16, 4)$-code



0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
0	0	0	0	1	1	1	1
0	1	0	1	1	0	1	0
0	0	1	1	1	1	0	0
0	1	1	0	1	0	0	1
1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	1	0	0
1	0	0	1	1	0	0	1
1	1	1	1	0	0	0	0
1	0	1	0	0	1	0	1
1	1	0	0	0	0	1	1
1	0	0	1	0	1	1	0

So we have a $(8, 16, 4)$-code. By taking all the codewords starting 0 and puncturing the first position we get a $(7, 8, 4)$-code. (Or take all codewords ending 1 and puncture in final position, or . . .)

What does this say about $A_2(7, 4)$?

Hamming Distance under Addition

The next lemma shows that Hamming distance behaves well under addition.

Lemma 11.3

Let u, w be binary words of length $n \in \mathbf{N}$. For any binary word $v \in \mathbf{Z}_2^n$ we have

$$d(u, w) = d(u + v, w + v).$$

Recall that the *weight* of a binary word u was defined just before Lemma 6.6 to be the number of positions of u equal to 1.

Lemma 11.4

Let C be a linear binary code. The minimum distance of C is equal to the minimum weight of a non-zero codeword of C.

Quiz on Lemma 11.4

To find the minimum distance of a code C we have to think about $d(u, w)$ for all distinct $u, w \in C$. If C is linear it is much easier to find

$$\min\{\mathrm{wt}(u) : u \in C, u \neq 0\}$$

and then apply Lemma 11.4.

Recall that the square code has codewords

$$(u_1, u_2, u_3, u_4, u_1 + u_2, u_3 + u_4, u_1 + u_3, u_2 + u_4)$$

for u_1, u_2, u_3, $u_4 \in \mathbf{Z}_2$, represented by

u_1	u_2	$u_1 + u_2$
u_3	u_4	$u_3 + u_4$
$u_1 + u_3$	$u_2 + u_4$	

What is the minimum weight of a non-zero codeword?

(a) 1 (b) 2 (c) 3 (d) 4

Parity Check Extensions

The last result in this section generalises the parity check extension codes seen in Example 2.9 and Example 11.2(2). For an optional related result see Questions 6 and 7 on Sheet 4.

Definition 11.5

Let C be a binary code of length n. The *parity check extension* of C is the code C_{ext} of length $n+1$ defined by

$$C_{\text{ext}} = \{(u_1, \ldots, u_n, u_{n+1}) : (u_1, \ldots, u_n) \in C, u_1 + \cdots + u_n + u_{n+1} = 0.\}$$

Theorem 11.6

Let C be a linear binary (n, M, d)-code. Then C_{ext} is a linear binary code of length $n+1$ and size M. The minimum distance of C_{ext} is d if d is even and $d+1$ if d is odd.

Example of Theorem 11.6

Suppose we start with the binary square code S, which is a $(8, 16, 3)$-code. To form the parity check extension we take each codeword

u_1	u_2	$u_1 + u_2$
u_3	u_4	$u_3 + u_4$
$u_1 + u_3$	$u_2 + u_4$	

and append a final bit to make the weight even.

Example of Theorem 11.6

Suppose we start with the binary square code S, which is a $(8, 16, 3)$-code. To form the parity check extension we take each codeword

u_1	u_2	$u_1 + u_2$
u_3	u_4	$u_3 + u_4$
$u_1 + u_3$	$u_2 + u_4$	

and append a final bit to make the weight even. The codewords in the parity check extension of S can be represented as

u_1	u_2	$u_1 + u_2$
u_3	u_4	$u_3 + u_4$
$u_1 + u_3$	$u_2 + u_4$	$u_1 + u_2 + u_3 + u_4$

and the minimum distance of the extended code is 4.

§12 Bases, Generator Matrices and Encoding

In this section we will see an efficient way to encode using a linear binary code. The next exercise shows the basic idea.

Exercise: Suppose that $u(1), \ldots, u(k)$ are codewords in a linear binary code C. Show that if $c_1, c_2, \ldots, c_k \in \mathbf{Z}_2$, then $c_1 u(1) + \cdots + c_k u(k)$ is a codeword in C and

$$c_1 u(1) + \cdots + c_k u(k) = \big(c_1, \ldots, c_k\big) \begin{pmatrix} u(1) \\ \vdots \\ u(k) \end{pmatrix}.$$

This suggests an encoding strategy where we first convert messages to binary words of length k, and then encode the binary word $(c_1, \ldots, c_k)$ as the codeword $c_1 u(1) + \cdots + c_k u(k) \in C$.

Mariner 9

Example 12.1

(For interest only.) The binary $(32, 64, 16)$-Hadamard code used by Mariner 9 is linear. The encoder used the idea just outlined, with the matrix G shown below.

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}$$

To encode a pixel whose blackness is represented by the number $m \in \{0, 1, \ldots, 63\}$, write m in binary as $b_5 b_4 b_3 b_2 b_1 b_0$ and then encode it as

$$(b_5, \ldots, b_1, b_0)G.$$

For example, since 34 is 100010 in binary, 34 is encoded as the codeword

$$(1, 0, 0, 0, 1, 0)G = 01010101010101011010101010101010.$$

Bases

To make this encoding scheme work, **it is essential** to make a careful choice of $u(1), \ldots, u(k)$.

Definition 12.2

Let C be a linear binary code of length n. We say that words $u(1), \ldots, u(k) \in \mathbf{Z}_2^n$ are

(a) *linearly independent* if the only solution to the equation

$$c_1 u(1) + \cdots + c_k u(k) = 0$$

with $c_1, \ldots, c_k \in \mathbf{Z}_2$ is $c_1 = c_2 = \ldots = c_k = 0$.

(b) *span* C if for every $w \in C$ there exist $c_1, \ldots, c_k \in \mathbf{Z}_2$ such that

$$w = c_1 u(1) + \cdots + c_k u(k).$$

(c) *a basis of* C if they are linearly independent and span C.

Example 12.3

(1) Let $C = \{00000, 11100, 00111, 11011\}$, as in Example 11.2(1) Then a basis for C is $11100, 00111$. If we take

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix},$$

the codewords in C are $(0,0)G$, $(0,1)G$, $(1,0)G$ and $(1,1)G$.

(2) Let C_{ext} be the parity check extension of all binary words of length 4, considered in Example 11.2(4). Then

$$10001, 01001, 00101, 00011$$

is a basis for C_{ext}.

It is **very important** to note that a linear binary code usually does not have a unique basis.

Exercise: Find a different basis for C_{ext}.

Adminstration

Please take Problem Sheet 8. Note deadline is Tuesday 19 March.

On Tuesday I intended to write

$$c_1(1,1,1,0,0) + c_2(0,0,1,1,1) + c_3(1,1,0,1,1) = (0,0,0,0,0)$$

and state that this holds when $c_1 = c_2 = c_3 = 1$. Some of c_1, c_2, c_3 were written as u_1, u_2, u_3.

What Good is a Basis?

Definition 12.4

Suppose that C is a linear binary code of length n and minimum distance d. If $u(1), \ldots, u(k)$ is a basis of C then we say that C has *dimension* k and that C is a $[n, k, d]$-code.

Thus a linear binary $(n, 2^k, d)$-code is a $[n, k, d]$-code. The codes in Example 12.3 have parameters $[5, 2, 3]$ and $[5, 4, 2]$, respectively.

The next result connects dimension with the rate of a binary code, as defined in Definition 1.10.

Theorem 12.5

Let C be a linear binary code having $u(1), \ldots, u(k)$ as a basis. For each $w \in C$ there exist unique $c_1, \ldots, c_k \in \mathbf{Z}_2$ such that

$$w = c_1 u(1) + \cdots + c_k u(k).$$

Hence $|C| = 2^k$ and the rate of C is k/n.

Quiz on Linear Codes

Let

$$C = \{000, 011, 110, 101\}$$
$$D = \{100, 010, 001, 111\}$$
$$E = \{000, 011\}$$
$$F = \{000, 011, 110\}$$

Which are the linear binary codes?

(1) C only (2) E and F only (3) C and E only (4) E only

Generator Matrices

The matrices used in Example 12.3 are generator matrices, as defined in the following definition.

Definition 12.6

Suppose that C is a linear binary code of length n having $u(1), \ldots, u(k) \in \mathbf{Z}_2^n$ as a basis. The $k \times n$ matrix

$$\begin{pmatrix} u(1) \\ \vdots \\ u(k) \end{pmatrix}$$

is said to be a *generator matrix* for C.

Using Row-Operations to Find a Basis

Example 12.7

Let C be the linear code of length 7 spanned by the codewords 1100110, 1011010, 0110011, 0001111. These codewords are not linearly independent. We can demonstrate this, and find a basis and generator matrix for C, by applying row operations to the matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Encoding Using Generator Matrices

Example 12.8

Let C be the linear code in Example 12.7. We saw that C has generator matrix

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

so C has dimension 3 and size 2^3. To encode the number 7 we write 7 in binary as 111 and take the codeword

$$(1,1,1)\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = (1,1,0,1,0,0,1)$$

In general, the number $4b_2 + 2b_1 + b_0$, written in binary as $b_2b_1b_0$, is encoded as

$$(b_2, b_1, b_2 + b_1, b_0, b_2 + b_0, b_1 + b_0, b_0 + b_1 + b_2).$$

Sheet 7: all work in green folder. Answers on Moodle.

Sophie Christiansen will unveil a new plaque on the golden postbox at 11am tomorrow (Tuesday).

Quiz on Generator Matrices

Let C be the linear binary code with generator matrix

$$G = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

What is the size of C?

(a) 4 (b) 8 (c) 16 (d) 37

What is the minimum distance of C?

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5

How would 7 be encoded using the generator matrix G?

(a) 01111 (b) 11100 (c) 10001 (d) 10101

Standard Form for Generator Matrices

Definition 12.9

A generator matrix

$$\begin{pmatrix} I_k & A \end{pmatrix}$$

where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix is said to be in *standard form*.

Theorem 12.10

Let C be a linear binary code of length n and dimension k. Then C is equivalent, by a permutation of the positions in the codewords, to a code with a generator matrix in standard form

$$\begin{pmatrix} I_k & A \end{pmatrix}$$

where A is an $k \times (n-k)$-matrix.

Standard Form Generator Matrix For Square Code

Question 1(c) on Sheet 7 asked for a basis of the square code. The codewords encoding 1, 2, 4 and 8 give a basis.

$$8 \mapsto (1,0,0,0,1,0,1,0)$$
$$4 \mapsto (0,1,0,0,1,0,0,1)$$
$$2 \mapsto (0,0,1,0,0,1,1,0)$$
$$1 \mapsto (0,0,0,1,0,1,0,1)$$

The corresponding generator matrix is

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

which is in standard form. Note that the square code has $2^4 = 16$ codewords. The codewords in the chosen basis are NOT the whole code!

§13 Decoding by Standard Arrays

In this section we shall see a way to implement nearest neighbour decoding for linear codes that exploits their special structure.

Definition 13.1

Let C be a linear binary code of length n. A *coset* of C is a set of the form

$$C + v = \{u + v : u \in C\}$$

where $v \in \mathbf{Z}_2^n$.

Note that if $v \in \mathbf{Z}_2^n$ then, since $\mathbf{0} \in C$, we have $v = \mathbf{0} + v$ and so $v \in C + v$. Hence the coset containing v is $C + v$.

Example 13.2

Let C be the linear binary code

$$C = \{0000, 1110, 0011, 1101\}$$

obtained by puncturing (see Definition 8.1) the code in Example 12.3(1) in its final position. If we send the codewords through a channel that corrupts position 1 every time, then the received words are

$$C + 1000 = \{1000, 0110, 1011, 0101\}.$$

The other possible one bit errors give cosets

$$\begin{aligned} C + 0100 &= \{0100, 1010, 0111, 1001\}, \\ C + 0010 &= \{0010, 1100, 0001, 1111\}, \\ C + 0001 &= \{0001, 1111, 0010, 1100\}. \end{aligned}$$

We also have the coset $C + 0000 = C$.

Example 13.2

Cosets of C were

$$\begin{aligned} C &= \{0000, 1110, 0011, 1101\} \\ C + 1000 &= \{1000, 0110, 1011, 0101\}, \\ C + 0100 &= \{0100, 1010, 0111, 1001\}, \\ C + 0010 &= \{0010, 1100, 0001, 1111\}. \end{aligned}$$

Suppose that $u \in C$ is sent and $v \in \mathbf{Z}_2^4$ is received such that $v \in C + 0010$. For example, $v = 1100$.

Example 13.2

Cosets of C were

$$\begin{aligned} C &= \{0000, 1110, 0011, 1101\} \\ C + 1000 &= \{1000, 0110, 1011, 0101\}, \\ C + 0100 &= \{0100, 1010, 0111, 1001\}, \\ C + 0010 &= \{0010, 1100, 0001, 1111\}. \end{aligned}$$

Suppose that $u \in C$ is sent and $v \in \mathbf{Z}_2^4$ is received such that $v \in C + 0010$. For example, $v = 1100$.

- We might be tempted to think that an error had occurred in position 3.

Example 13.2

Cosets of C were

$$\begin{aligned} C &= \{0000, 1110, 0011, 1101\} \\ C + 1000 &= \{1000, 0110, 1011, 0101\}, \\ C + 0100 &= \{0100, 1010, 0111, 1001\}, \\ C + 0010 &= \{0010, 1100, 0001, 1111\}. \end{aligned}$$

Suppose that $u \in C$ is sent and $v \in \mathbf{Z}_2^4$ is received such that $v \in C + 0010$. For example, $v = 1100$.

- We might be tempted to think that an error had occurred in position 3.
- But since $C + 0001 = C + 0010$, we also have $v \in C + 0001$. So the same argument would suggest that the error is in position 4.

Example 13.2

Cosets of C were

$$\begin{aligned} C &= \{0000, 1110, 0011, 1101\} \\ C + 1000 &= \{1000, 0110, 1011, 0101\}, \\ C + 0100 &= \{0100, 1010, 0111, 1001\}, \\ C + 0010 &= \{0010, 1100, 0001, 1111\}. \end{aligned}$$

Suppose that $u \in C$ is sent and $v \in \mathbf{Z}_2^4$ is received such that $v \in C + 0010$. For example, $v = 1100$.

- We might be tempted to think that an error had occurred in position 3.
- But since $C + 0001 = C + 0010$, we also have $v \in C + 0001$. So the same argument would suggest that the error is in position 4.
- And it could be that more errors occurred. For instance $C + 0001 = C + 0010 = C + 1100$. So it could be that $u = 0000$ and two errors occurred.

Cosets are Either Equal or Disjoint

Exercise: Let C be a linear binary code of length n. Show that if $v \in \mathbf{Z}_2^n$ then $C + v = C + (u + v)$ for all $u \in C$.

Lemma 13.3

Let C be a linear binary code of length n. If $C + v$ and $C + v'$ are cosets of C then either $C + v = C + v'$ or the cosets $C + v$ and $C + v'$ are disjoint.

Exercise: Check that each binary word of length 4 is in a unique coset of the code in Example 13.2.

Standard Arrays

Definition 13.4

Let C be a linear binary code of length n. A *standard array* for C is a table in which each row consists of the codewords in a coset of C, arranged so that

(i) the first row is C;

(ii) if the word x appears in the first column then $\mathrm{wt}(x) \leq \mathrm{wt}(v)$ for all v in the row of x.

The first word in each row is said to be a *coset leader*. To decode a received word $v \in \mathbf{Z}_2^n$ by *standard array decoding*, decode v as $v + x$ where x is the coset leader for the row containing v.

Standard Arrays

Example 13.5

A standard array for the code C in Example 13.2 is

$$\begin{array}{cccc} 0000 & 1110 & 0011 & 1101 \\ 1000 & 0110 & 1011 & 0101 \\ 0100 & 1010 & 0111 & 1001 \\ 0010 & 1100 & 0001 & 1111 \end{array}$$

Note that we could also taken the fourth row to be

$$0001 \quad 1111 \quad 0010 \quad 1100$$

with 0001 as the coset leader, since both 0010 and 0001 have weight 1. The other coset leaders 0000, 1000 and 0100 are uniquely determined by their cosets.

Justification for Nearest Neighbour Decoding

Theorem 13.6

Let C be a linear binary code of length n. Let $v \in \mathbf{Z}_2^n$. Suppose that the row containing v has coset leader x. Then $v + x \in C$ and

$$d(v + x, v) \leq d(u, v)$$

for all $u \in C$.

Justification for Nearest Neighbour Decoding

Theorem 13.6

Let C be a linear binary code of length n. Let $v \in \mathbf{Z}_2^n$. Suppose that the row containing v has coset leader x. Then $v + x \in C$ and

$$d(v + x, v) \leq d(u, v)$$

for all $u \in C$.

In the proof we used that

$$d(v + x, v) = \mathrm{wt}(x)$$
$$d(u, v) = \mathrm{wt}(u + v).$$

So $v + x$ is the *unique* nearest codeword to v if and only if x is the unique word of minimum weight in the coset $C + v$.

Justification for Nearest Neighbour Decoding

Theorem 13.6

Let C be a linear binary code of length n. Let $v \in \mathbf{Z}_2^n$. Suppose that the row containing v has coset leader x. Then $v + x \in C$ and

$$d(v + x, v) \leq d(u, v)$$

for all $u \in C$.

In the proof we used that

$$d(v + x, v) = \mathrm{wt}(x)$$
$$d(u, v) = \mathrm{wt}(u + v).$$

So $v + x$ is the *unique* nearest codeword to v if and only if x is the unique word of minimum weight in the coset $C + v$.

This shows that standard array decoding implements nearest neighbour decoding when there is a unique possible coset leader. If there are several coset leaders then nearest neighbour decoding fails.

Incomplete Decoding for $\{0000, 1110, 0011, 1101\}$

Decoding Strategy: use standard array decoding if the received word v is in one of the three coset with a unique choice of leader. If $v \in C + 0010 = C + 0001$ then request retransmission.

0000	1110	0011	1101
1000	0110	1011	0101
0100	1010	0111	1001
0010	1100	0001	1111

Incomplete Decoding for $\{0000, 1110, 0011, 1101\}$

Decoding Strategy: use standard array decoding if the received word v is in one of the three coset with a unique choice of leader. If $v \in C + 0010 = C + 0001$ then request retransmission.

0000	1110	0011	1101
1000	0110	1011	0101
0100	1010	0111	1001
0010	1100	0001	1111

Since $A_2(4, 3) = 2$, the largest 1-error correcting code of size 4 has size 2. This decoding strategy shows that if we are willing to request transmission (or make a guess) for two of the four possible errors in one position, then we can use a code of size 4 instead.

§14 Parity Check Matrices and Syndrome Decoding

All the linear codes we have seen so far can be defined by linear equations. For instance, the code $\mathrm{C}_{\mathrm{ext}}$ consisting of all binary words of length 5 with evenly many 1s can be defined by

$$\mathrm{C}_{\mathrm{ext}} = \{(u_1, u_2, u_3, u_4, u_5) \in \mathbf{Z}_2^5 : u_1 + u_2 + u_3 + u_4 + u_5 = 0\}$$

and the square code can be defined by

$$S = \left\{ (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8) \in \mathbf{Z}_2^8 : \begin{array}{l} u_1 + u_2 = u_5, \;\; u_3 + u_4 = u_6 \\ u_1 + u_3 = u_7, \;\; u_2 + u_4 = u_8 \end{array} \right\}$$

Definition 14.1

Let C be a linear binary code of length n and dimension k. A *parity check matrix* for C is an $(n-k) \times n$ matrix H with linearly independent rows such that for each $u \in \mathbf{Z}_2^n$ we have

$$u \in C \iff uH^{tr} = \mathbf{0}.$$

Here $\mathbf{0}$ is the all-zeros word of length $n-k$.

Example 14.2

(1) The code C_{ext} has parity check matrix

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

(2) Let S be the square code. Then S has as a parity check matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Decoding for Square Code Using Parity Check Matrix

To perform the decoding algorithm for the square code seen earlier in the course, we record which linear equations are not satisfied, and then try to flip a single bit (or two if necessary) to make all of them hold.

Exercise: For each of the following received words, decide which of the four defining equations for the square code fail to hold. Decode each word by nearest neighbour decoding.

(Flips bits to make all equations hold, but watch out for failure when two codewords are equally close.)

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10011001

u_1	u_2	u_5
u_3	u_4	u_6
u_7	u_8	

Every Linear Code can be Defined by Linear Equations

Theorem 14.3

Let C be a linear binary code of length n and dimension k. Then C has a parity check matrix. Moreover, if C has a generator matrix G in standard form $G = \begin{pmatrix} I_k & A \end{pmatrix}$ then

$$\begin{pmatrix} A^{tr} & I_{n-k} \end{pmatrix}$$

is a parity check matrix for C.

For example, if $G = \begin{pmatrix} I_k & A \end{pmatrix}$ is the standard form generator matrix for the square code, then applying Theorem 14.3 to G, we get the parity check matrix already found in Example 14.2(2).

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Dual Codes

Definition 14.4

Let C be a linear binary code of length n and dimension k and let H be a parity check matrix for C. The *dual code* $C^{\perp}$ is the linear binary code of length n and dimension $n-k$ with generator matrix H.

Example 14.5

Let C_{ext} be as in Example 14.2(1). Then

$$C_{\mathrm{ext}}^{\perp} = \{00000, 11111\}$$

is the binary repetition code of length 5, and

$$\{00000, 11111\}^{\perp} = C_{\mathrm{ext}}.$$

Hamming Code Defined by a Parity Check Matrix

Example 14.6

Let

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

and let $C = \{u \in \mathbf{Z}_2^7 : uH^{tr} = 0\}$. Then C is a linear binary code with parity check matrix H and generator matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

By Lemma 11.4, the minimum distance of C is equal to the minimum weight of a non-zero codeword. Clearly there are codewords of weight 3 in C, so to show C has minimum distance 3, it suffices to show there are no codewords of weight 1 or 2. This can be done using H.

Minimum Distance from a Parity Check Matrix

Theorem 14.7

Let C be a linear binary code of length n and dimension k. Let H be a parity check matrix for C. The minimum distance of C is equal to the minimum $r \in \mathbf{N}$ such that there exist r linearly dependent columns of H.

Syndromes

Theorem 14.8

Let C be a linear binary code of length n and dimension k with parity check matrix H and let $v, v' \in \mathbf{Z}_2^n$. Then v and v' are in the same coset of $C \iff vH^{tr} = v'H^{tr}$.

Syndromes

Theorem 14.8

Let C be a linear binary code of length n and dimension k with parity check matrix H and let $v, v' \in \mathbf{Z}_2^n$. Then v and v' are in the same coset of $C \iff vH^{tr} = v'H^{tr}$.

This theorem motivates the following definition.

Definition 14.9

Let C be a linear binary code of length n and dimension k with parity check matrix H. The *syndrome* of a word $v \in \mathbf{Z}_2^n$ is defined to be $vH^{tr} \in \mathbf{Z}_2^{n-k}$.

By Theorem 14.8 we can identify the coset of C containing a word $v \in \mathbf{Z}_2^n$ from its syndrome vH^{tr}. So to decode a received word v, calculate its syndrome vH^{tr}, and then decode v as $v + x$ where x is the chosen coset leader for the coset $C + v$ containing v.

Decoding Square Code using Syndromes

Here is the table from Monday with an extra column showing the syndromes vH^{tr} computed using the parity check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Received word v	Equations failing	Syndrome vH^{tr}	Error	Closest codewords
10001100	2 and 3	(0,1,1,0)	00100000	10101100
11001011	1	(1,0,0,0)	00001000	11000011
11000000	3 and 4	(0,0,1,1)	11000000 or 00110000 or 00000011	00000000 and 11110000 and 11000011
10011001	2 and 3	(0,1,1,0)	00100000	10111001

Example 14.10

Let $C = \{0000, 1110, 0011, 1101\}$ be the code used in Examples 13.2 and 13.5. Then C has parity check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

By Theorem 14.8, any two words in the same coset of C have the same syndrome. The map from cosets of C to syndromes is

$$\begin{aligned} C &\mapsto (0,0,0,0)H^{tr} = (0,0) \\ C + (1,0,0,0) &\mapsto (1,0,0,0)H^{tr} = (1,0) \\ C + (0,1,0,0) &\mapsto (0,1,0,0)H^{tr} = (1,1) \\ C + (0,0,1,0) &\mapsto (0,0,1,0)H^{tr} = (0,1). \end{aligned}$$

Thus all words in $C + 1000 = \{1000, 0110, 1011, 0101\}$ have syndrome $(1,0)$, and if any of the words 1000, 0110, 1011, 0101 is received, it will be decoded by adding 1000, since this is the unique coset leader in the coset $C + 1000$.

Example 14.10 [continued]

Using syndrome decoding we can replace the standard array in Example 13.5 with the more concise table below.

syndrome	chosen coset leader
00	0000
10	1000
01	0010
11	0100

We saw in Section 13 that the main defect of the code C is that $C + 0010 = C + 0001$. Correspondingly, the single bit errors 0010 and 0001 have the same syndrome

$$(0, 0, 1, 0)H^{tr} = (0, 0, 0, 1)H^{tr} = (0, 1).$$

Syndrome Decoding for Hamming [7,4,3]-code

Example 14.11

Let C, G and H be as in Example 14.6. Let $e(i)$ be the word with a 1 in position i and 0 in all other positions. The syndrome of $e(i)$ is $e(i)H^{tr}$, which is the ith row of H^{tr}.

The columns of H are distinct and non-zero, so by Lemma 13.3 (cosets are disjoint) and Theorem 14.8 (syndromes correspond to cosets) we have

$$\mathbf{Z}_2^7 = C \cup \big(C + e(1)\big) \cup \cdots \cup \big(C + e(7)\big)$$

where the union is disjoint.

To decode a received word v, we calculate its syndrome vH^{tr}. If vH^{tr} is the ith row of H^{tr} then $vH = e(i)H^{tr}$ and, by Theorem 14.8, $v \in C + e(i)$. So we decode v as $v + e(i)$.

Example 14.11 [continued]

For example, to use C to send the number 13, we would write 13 as 1101 in binary, and encode it as

$$(1,1,0,1)G = (1,0,1,0,1,0,1).$$

Suppose that when we transmit 1010101, an error occurs in position 6, so 1010111 is received. Then the syndrome of the received word is

$$(1,0,1,0,1,1,1)H^{tr} = (0,1,1)$$

which is row 6 of H^{tr}. So we decode by flipping the bit in position 6 to get 1010101. We then read off 1101 from positions 3, 5, 6 and 7.

A 7 Question Strategy from the Hamming Code

Here are the questions for the liar game corresponding to the Hamming $[7, 4, 3]$-code.

1. Is your number in $\{1, 3, 4, 6, 8, 10, 13, 15\}$?
2. Is it in $\{1, 2, 5, 6, 8, 11, 12, 15\}$?
3. Is it in $\{8, 9, 10, 11, 12, 13, 14, 15\}$?
4. Is it in $\{1, 2, 4, 7, 9, 10, 12, 15\}$?
5. Is it in $\{4, 5, 6, 7, 12, 13, 14, 15\}$?
6. Is it in $\{2, 3, 6, 7, 10, 11, 14, 15\}$?
7. Is it in $\{1, 3, 5, 7, 9, 11, 13, 15, 17\}$?

Questionnaires

Please take the questionnaires seriously. The batch number is 965029. Your course code is either MT361 or MT461 or MT5461.

17. For this course, Library study space met my needs.
18. The course books in the Library met my needs for this course.
19. The online Library resources met my needs for this course.
20. I was satisfied with the Moodle elements of this course.
21. I received feedback on my work within the 4 week norm specified by College.

Please write any further comments on the back of the form. (In particular, please answer the old version of Q17: whether you found the speed too fast, too slow, or about right.)