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

Admin

From Monday 23rd, the Monday 3pm lecture will be in **ABLT3**.

From **this Tuesday**, the Tuesday 3pm lecture will be in **ABLT1**.

The Thursday 10am lecture will continue to be in **BLT2**.

Preliminary Sheet: Answers now available on Moodle.

Correction: Q1(c) should read 'Using Scheme 2 **[not 1]**, the probability that Bob decodes Alice's message wrongly was found to be $3p^2 - 2p^3$. How many bits does Alice send to Bob when this scheme is used?'

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.

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.)

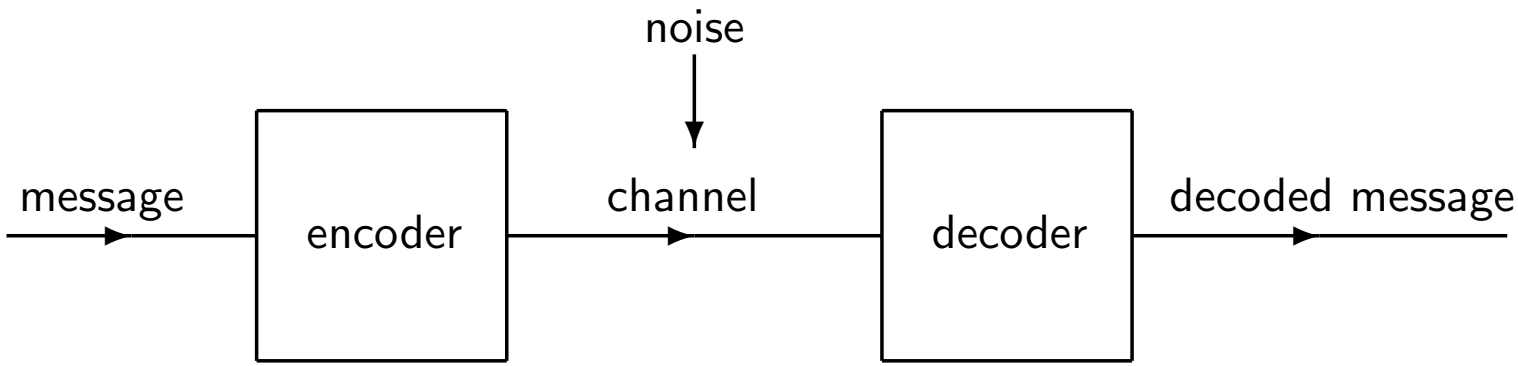
Prerequisites

- 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 linear algebra: vector spaces, subspaces, matrices, image and kernel, rank-nullity theorem, reduced row-echelon form. I will issue a handout later in term to remind you of these ideas.

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.

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

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

(This is **misprinted** as $p^3 + 3p(1-p)^2$ on page 6 of the handout: **please correct**.) See the preliminary problem sheet for an analysis of Scheme 1.

Remarks on the definition of codes

Remarks 1.6

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 1.7.

Using the Same Code in Different Ways

Example 1.7

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

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

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

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 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; this corresponds to noise in the channel. Repeat the game in the previous exercise: try not to use too many questions!

Example 1.8

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

Example 1.8: 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) Binary search strategy:

Example 1.8: 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) Binary search strategy: full binary code of length 4, size 16

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

Example 1.8: 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) Binary search strategy: full binary code of length 4, size 16

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

(2) Ask every question 3 times:

Example 1.8: 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) Binary search strategy: full binary code of length 4, size 16

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

(2) Ask every question 3 times: length 12, size 16

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

A better code from the Liar Game

(3) 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. We will encode m as the length 9 word

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

where $e = b_3 + b_2 + b_1 + b_0 \bmod 2$. This corresponds to asking about each bit in m twice, and then asking one final question which will resolve any ambiguity caused by an earlier lie from Alice.

$$\begin{aligned} 0 &\mapsto 0000\,0000\,0 \\ 1 &\mapsto 0001\,0001\,1 \\ 2 &\mapsto 0010\,0010\,1 \\ 3 &\mapsto 0011\,0011\,0 \\ &\vdots \\ 15 &\mapsto 1111\,1111\,0. \end{aligned}$$

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 $m = 2^3 u_1 + 2^2 u_2 + 2^1 u_3 + 2^0 u_4$ and then sends Bob the codeword in the Square Code starting $u_1 u_2 u_3 u_4 \ldots$

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 $m = 2^3 u_1 + 2^2 u_2 + 2^1 u_3 + 2^0 u_4$ and then sends Bob the codeword in the Square Code starting $u_1 u_2 u_3 u_4 \ldots$

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

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

A 7 Question Strategy from the Hamming Code

It is known that no questioning strategy for the Liar Game with 16 possible numbers can *guarantee* to use fewer than 7 questions. Remarkably there is a questioning strategy *with fixed questions* that meets this bound. It comes from the Hamming $[7, 4, 3]$-code that we will see in Part C of the course.

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 $\{1, 3, 5, 7, 9, 11, 13, 15, 17\}$?

Example 1.9: 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 and decoding scheme used will always correct up to 16 errors in a block of 2048 consecutive bits. If, however, the errors occur in adjacent bits, as is usual for a scratch, then at least 128 consecutive errors may be corrected.

Example 1.10: 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 1.11: 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

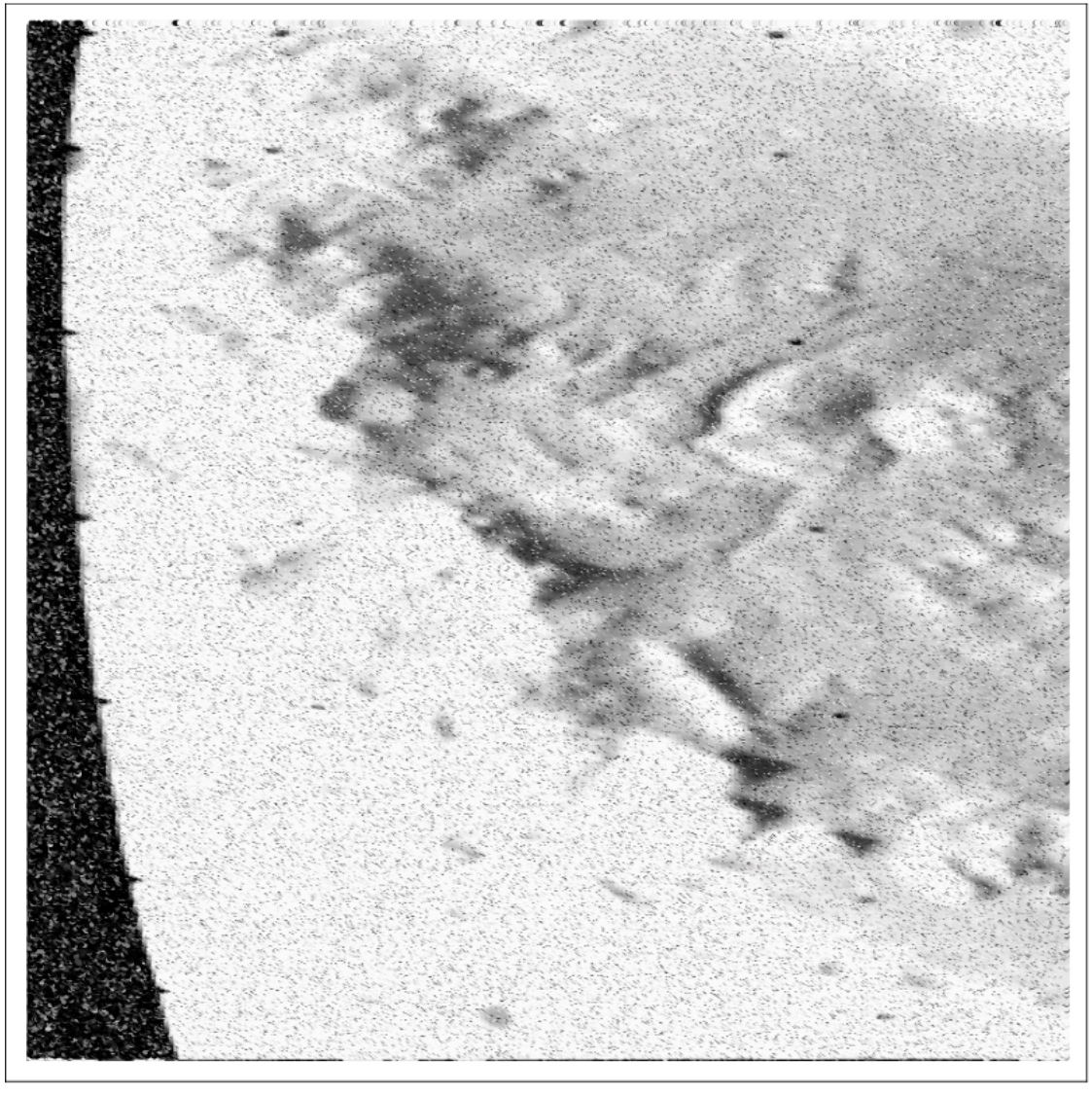
It was acceptable for each pixel to be encoded by up to 32 bits, so increasing the amount of data to be stored and transmitted by a factor of 5.

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%.

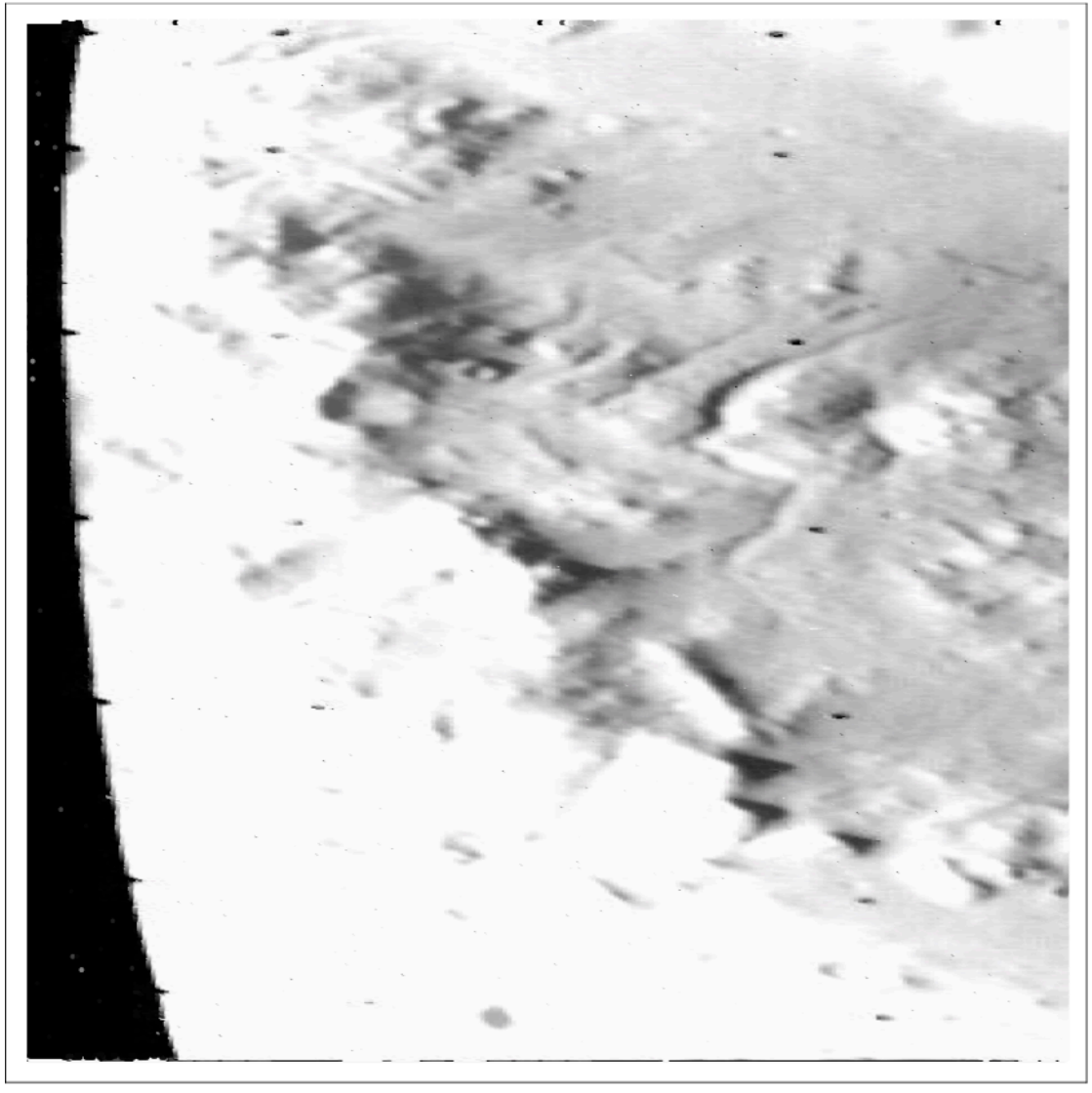
The Hadamard code of length 32 and size 64 actually used could correct up to 7 errors in each transmitted word. This reduced the percentage of incorrect pixels to about 0.014%.

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

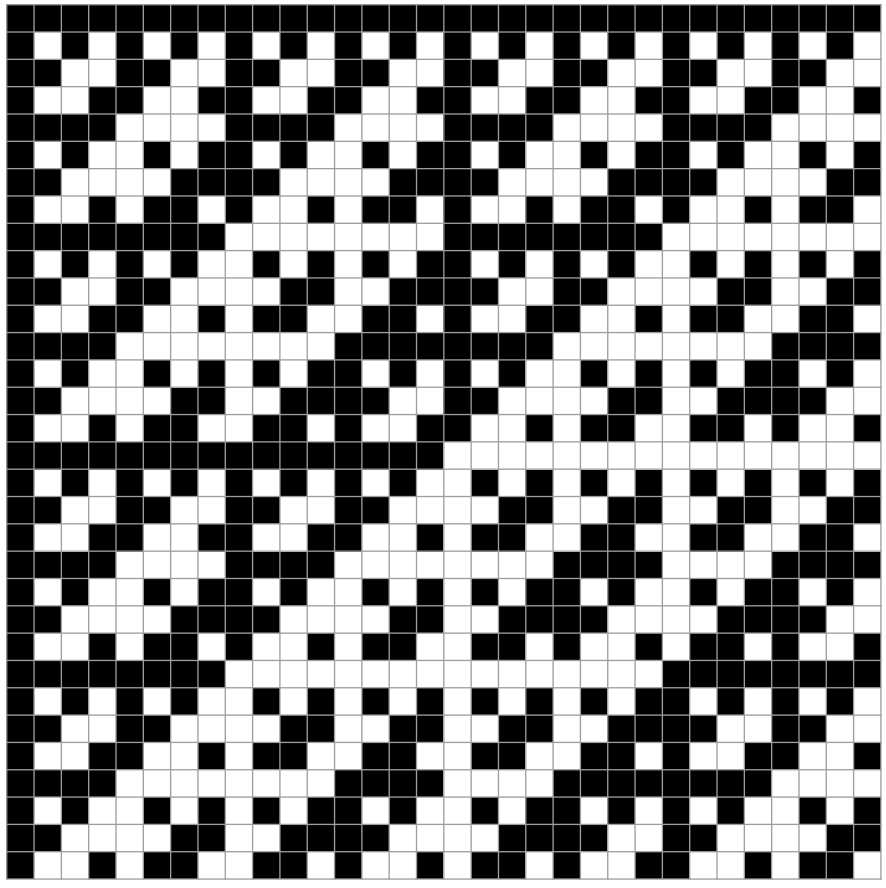


Mariner 9 image: improvement due to error correction



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

Admin

Answers to Sheet 1 on Moodle: answer to Question 2 corrected after Monday's lecture.

If you have the sign-up sheet at end of lecture, please give to the lecturer.

§2 Hamming Distance and Error Detection/Correction

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 $d(\mathrm{TALE}, \mathrm{TILT}) = 2$.

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. For the connection with Lewis Carroll's 'Doublets Game', see Question 9 on Sheet 1.

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 $\{0, 1\}$.

Example 2.5

Consider the ternary repetition code of length 7 over the alphabet $\{0, 1, 2\}$ with codewords

$$0000000, \quad 1111111 \quad 2222222.$$

Any two distinct codewords are at distance 7.

Example 2.5 (continued)

Suppose the codeword 0000000 is sent and e errors occur in the channel, so that we receive v where $d(0000000, v) = e$.

If $1 \leq e \leq 6$ then the received word is not a codeword, since $d(0000000, 1111111) = d(0000000, 2222222) = 7$. So we can detect errors when ≤ 6 errors occur.

Example 2.5 (continued)

Suppose the codeword 0000000 is sent and e errors occur in the channel, so that we receive v where $d(0000000, v) = e$.

If $1 \leq e \leq 6$ then the received word is not a codeword, since $d(0000000, 1111111) = d(0000000, 2222222) = 7$. So we can detect errors when ≤ 6 errors occur.

If $1 \leq e \leq 3$ then $d(0000000, v) \leq 3$, and $d(1111111, v) \geq 4$ and $d(2222222, v) \geq 4$. Hence the nearest codeword to v is 0000000.

Example 2.5 (continued)

Suppose the codeword 0000000 is sent and e errors occur in the channel, so that we receive v where $d(0000000, v) = e$.

If $1 \leq e \leq 6$ then the received word is not a codeword, since $d(0000000, 1111111) = d(0000000, 2222222) = 7$. So we can detect errors when ≤ 6 errors occur.

If $1 \leq e \leq 3$ then $d(0000000, v) \leq 3$, and $d(1111111, v) \geq 4$ and $d(2222222, v) \geq 4$. Hence the nearest codeword to v is 0000000.

If $e \geq 4$ then it is possible that the nearest codeword to v will be either 1111111 or 2222222. For example, if $e = 4$ we may have

$$v = 1010110$$

and then $d(1111111, v) = 3$, $d(0000000, v) = 4$. So we can only promise to decode reliably when ≤ 3 errors occur.

Quiz on Example 2.5

Let C be the ternary repetition code in Example 2.5 with codewords 0000000, 1111111, 2222222. Using C as an error correcting code, decode the following received words by choosing a nearest codeword in C.

(a) 1100000 (b) 2211211 (c) 0122021 (d) 1112220

In each case write down how many errors you think occurred in the channel, and a lower bound for the number of errors that must have occurred if your answer is **not** the sent codeword.

Quiz on Example 2.5

Let C be the ternary repetition code in Example 2.5 with codewords 0000000, 1111111, 2222222. Using C as an error correcting code, decode the following received words by choosing a nearest codeword in C.

(a) 1100000 (b) 2211211 (c) 0122021 (d) 1112220

In each case write down how many errors you think occurred in the channel, and a lower bound for the number of errors that must have occurred if your answer is **not** the sent codeword.

(a) 0000000
2/5 errors

(b) 1111111
3/4 errors

(c) 2222222
4/5 errors

(d) $xxxxxxx$
4/4 errors

where $x = 1$ or $x = 2$.

Binary Parity Check Codes

Example 2.6 (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 $d(u, u') \geq 2$.

Quiz on Example 2.6.

Exercise: How would you use the length 5 code C_{ext} of size 16 to encode a number between 0 and 15?

Quiz on Example 2.6.

Exercise: How would you use the length 5 code C_{ext} of size 16 to encode a number between 0 and 15?

Using C_{ext} as an error detecting code (i.e. don't try to correct any errors) decide for each of the received words below whether you would request retransmission. Decode the words you accept.

(a) 11000 (b) 11100 (c) 11011

How many errors must have occurred in the channel if any of your decoded messages are wrong?

Key Definition

Definition 2.7

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

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

for all codewords $u' \in C$ such that $u' \neq u$.

Remarks 2.8 on Definition 2.7

(1) Equivalently, a code C is t-error detecting if making up to t changes in a codeword does not give another codeword. It is t-error correcting if whenever v is a word within distance t of a codeword $u \in C$, then v is strictly nearer to u than to any other codeword of C. (So $d(u, v) < d(u', v)$ for all $u' \in C$ such that $u \neq u'$.)

(2) In both definitions we have $d(u, v) \leq t$, so v is obtained from u by changing **up to** t positions. Thus 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.

(3) It may seem a bit odd to say that a code is 't-error correcting' when it is the decoder (be it human or computer) that has to do all the work of decoding! A code that is t-error correcting promises to be able to correct up to t-errors, *provided a suitable decoder is used*.

Exercise from Thursday

Let C be the ternary repetition code in Example 2.5 with codewords 0000000, 1111111, 2222222.

Exercise: show that C is 3-error correcting. *Hint: show that if* $u \in C$ *and* $v \in \{0,1,2\}^7$ *is such that* $d(u,v) \leq 3$, *then* $d(w,v) \geq 4$ *for all codewords* $w \in C$ *with* $w \neq u$.

Ternary Repetition Code and Definition 2.7

Lemma 2.9

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.*

Ternary Repetition Code and Definition 2.7

Lemma 2.9

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.*

In Example 2.5 we agreed, informally, that the ternary repetition code of length 7 can detect when up to 6 errors occur, and correct reliably when up to 3 errors occur. And, by Lemma 2.8, this code is 6-error detecting and 3-error correcting.

Binary Parity Check Code and Definition 2.7

Lemma 2.10

Let $n \in \mathbf{N}$ and let C_{ext} be the binary parity check code of length $n+1$ defined in Example 2.6. Then C_{ext} is 1-error detecting but not 2-error detecting. It is not 1-error correcting.

End Monday: let $u = 000\ldots0 \in C_{\text{ext}}$ and let $v = 110\ldots0$. Then $d(u,v) = 2$ and $v \in C_{\text{ext}}$, so C_{ext} is not 2-error detecting.

(I twice wrote C rather than C_{ext}: please correct.)

Exercise: show that C_{ext} is not 1-error correcting.

Binary Parity Check Code and Definition 2.7

Lemma 2.10

Let $n \in \mathbf{N}$ and let C_{ext} be the binary parity check code of length $n+1$ defined in Example 2.6. Then C_{ext} is 1-error detecting but not 2-error detecting. It is not 1-error correcting.

End Monday: let $u = 000\ldots0 \in C_{\text{ext}}$ and let $v = 110\ldots0$. Then $d(u, v) = 2$ and $v \in C_{\text{ext}}$, so C_{ext} is not 2-error detecting.

(I twice wrote C rather than C_{ext}: please correct.)

Exercise: show that C_{ext} is not 1-error correcting.

In Example 2.6 we agreed, informally, that the length 5 binary parity check code can detect when a single error occurs. We also found that it cannot detect when 2 errors occur, and cannot reliably be used as an error-correcting code. This agrees with the lemma.

Final Example of Definition 2.7

Lemma 2.11

Let C be the binary code of length 12 seen in Example 1.8 whose codewords are all words of the form

$$u_1\,u_2\,u_3\,u_4\;u_1\,u_2\,u_3\,u_4\;u_1\,u_2\,u_3\,u_4$$

where $u_1, u_2, u_3, u_4 \in \{0, 1\}$. Then C is 2-error detecting and one 1-error correcting. It is not 3-error detecting or 2-error correcting.

Final Example of Definition 2.7

Lemma 2.11

Let C be the binary code of length 12 seen in Example 1.8 whose codewords are all words of the form

$$u_1\,u_2\,u_3\,u_4\;u_1\,u_2\,u_3\,u_4\;u_1\,u_2\,u_3\,u_4$$

where $u_1, u_2, u_3, u_4 \in \{0, 1\}$. Then C is 2-error detecting and one 1-error correcting. It is not 3-error detecting or 2-error correcting.

Suppose that C is used to encode binary numbers between 0 and 15. Decode the following received words, first to codewords, then to numbers between 0 and 15. In each case write down how many errors you think occurred in the channel, and a lower bound for the number of errors that must have occurred if your answer is **not** the sent codeword.

(a) 0110 0110 0110 (b) 0110 0111 0110 (c) 0100 0111 0010

Answers to Quiz on Lemma 2.11

	(a)	(b)	(c)
Received word v	0110 0110 0110	0110 0111 0110	0100 0111 0010
Nearest codeword u	0110 0110 0110	0110 0110 0110	0110 0110 0110
Decode as	6	6	6
Errors expected	0	1	3
Min. errors if wrong	3	2	4

Answers to Quiz on Lemma 2.11

	(a)	(b)	(c)
Received word v	0110 0110 0110	0110 0111 0110	0100 0111 0010
Nearest codeword u	0110 0110 0110	0110 0110 0110	0110 0110 0110
Decode as	6	6	6
Errors expected	0	1	3
Min. errors if wrong	3	2	4

The values in the final row are the distances from the received word to the second nearest codeword; in all cases $w = 0111\ 0111\ 0111$ is one such codeword, and w has distances 3, 2 and 4, from the received word.

Answers to Quiz on Lemma 2.11

	(a)	(b)	(c)
Received word v	0110 0110 0110	0110 0111 0110	0100 0111 0010
Nearest codeword u	0110 0110 0110	0110 0110 0110	0110 0110 0110
Decode as	6	6	6
Errors expected	0	1	3
Min. errors if wrong	3	2	4

The values in the final row are the distances from the received word to the second nearest codeword; in all cases $w = 0111\ 0111\ 0111$ is one such codeword, and w has distances 3, 2 and 4, from the received word.

In (a) the received word is the codeword $u = 0110\ 0110\ 0110$ so $d(u, w) = 3$; this shows that C is not 3-error detecting.

In (b) the received word is $v = 0110\ 0111\ 0110$ which is distance 1 from u and distance 2 from w. This shows that C is not 2-error correcting.

Hamming's Adversarial Approach

Note that, while the code C in Lemma 2.11 can be used to detect some cases where 3 or more errors occur, it **does not guarantee to do so**. The definitions of t-error detecting and t-error correcting codes follow Hamming's 'adversarial' approach, in which we aim to decode reliably even if the adversary, who wants to cause us the maximum inconvenience, is allowed to choose the positions in which the (up to t) errors occur.

In practice the situation when using a t-error correcting code is

- **better** because random errors will not inconvenience the decoder as much as errors carefully chosen by the adversary;

Hamming's Adversarial Approach

Note that, while the code C in Lemma 2.11 can be used to detect some cases where 3 or more errors occur, it **does not guarantee to do so**. The definitions of t-error detecting and t-error correcting codes follow Hamming's 'adversarial' approach, in which we aim to decode reliably even if the adversary, who wants to cause us the maximum inconvenience, is allowed to choose the positions in which the (up to t) errors occur.

In practice the situation when using a t-error correcting code is

- **better** because random errors will not inconvenience the decoder as much as errors carefully chosen by the adversary;
- **worse** because there will always be a chance that more than t errors occur. (But we can choose a code with t large, so that this is unlikely.)

Example 2.12 (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 $\{0, 1, 2, \ldots, 9, \mathrm{X}\}$.

All we need to know is 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 and only if

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

where any symbol X in the ISBN is interpreted as 10. (In fact X can only appear as the final 'check digit' u_{10}.)

Lemma 2.13

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

§3 Minimum Distance

Question 4 on Sheet 1 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 3.1

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

Example 3.2

Here is an example from Hamming's original paper. 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 $d(000, 001) = 1$.

Examples of Minimum Distance

Lemma 3.3

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 2.6 is 2.*

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

Question 2 on Sheet 2 gives a step-by-step proof that the 1-error correcting code of length 9 seen in Example 1.8 has minimum distance 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 / t-error correcting and the minimum distance of C?

Code	Maximum t s.t. t-error detecting	Maximum t s.t. t-error correcting	Minimum distance
Binary parity check code of length 5	1	0	2
Ternary repetition code of length 7	6	3	7
Any code with min. distance ≥ 3	≥ 2	≥ 1	≥ 3

Code	Maximum t s.t. t-error detecting	Maximum t s.t. t-error correcting	Minimum distance
Binary parity check code of length 5	1	0	2
Ternary repetition code of length 7	6	3	7
Any code with min. distance ≥ 3	≥ 2	≥ 1	≥ 3
Square code	2	1	3
Lemma 2.11 code	2	1	3

Main Theorem on Minimum Distance

Recall that, by Definition 3.1, the minimum distance of a code C is

$$d(C) = \min\{d(u, w) : u, w \in C, u \neq w\}.$$

Theorem 3.4

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* $\iff$ $2t \leq d - 1$.

On Monday we proved everything except the '$\Leftarrow$' ('if') direction of (ii). In Question 4 on Sheet 1 we saw the special case that if $d \geq 3$ then C is 1-error correcting

Main Theorem on Minimum Distance

Recall that, by Definition 3.1, the minimum distance of a code C is

$$d(C) = \min\{d(u, w) : u, w \in C, u \neq w\}.$$

Theorem 3.4

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* $\iff$ $2t \leq d - 1$.

On Monday we proved everything except the '$\Leftarrow$' ('if') direction of (ii). In Question 4 on Sheet 1 we saw the special case that if $d \geq 3$ then C is 1-error correcting

For example, let C be a repetition code of length $2m + 1$. The minimum distance of C is $2m + 1$. By Theorem 3.4

$$C \text{ is } t\text{-error detecting} \iff t \leq 2m$$

$$C \text{ is } t\text{-error correcting} \iff t \leq m$$

Corollary of Theorem 3.4

The table below (taken from Hamming's original paper, see reference on page 3) shows the 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

Corollary of Theorem 3.4

The table below (taken from Hamming's original paper, see reference on page 3) shows the 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

Corollary 3.5

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

Notation

Notation 3.6

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.

Exercise: What are the parameters of the binary code in Example 1.8 and Lemma 2.11 with codewords

$$u_1 u_2 u_3 u_4 \;\; u_1 u_2 u_3 u_4 \;\; u_1 u_2 u_3 u_4$$

for $u_1, u_2, u_3, u_4 \in \{0, 1\}$?

Sheet 1 Question 3

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

Most people gave a good proof that C was $(m-1)$-error correcting. (Some 'model' answers are on Moodle.)

Sheet 1 Question 3

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

Most people gave a good proof that C was $(m-1)$-error correcting. (Some 'model' answers are on Moodle.)

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.

Sheet 1 Question 3

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

Most people gave a good proof that C was $(m-1)$-error correcting. (Some 'model' answers are on Moodle.)

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 we change m positions in a codeword then can't correct.". This is very vague. Improve to:

Sheet 1 Question 3

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

Most people gave a good proof that C was $(m-1)$-error correcting. (Some 'model' answers are on Moodle.)

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 we 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 $d(u, v) = d(u', v) = m$." This is nearly right, but it depends how we change positions. So we need to be more specific.

Sheet 1 Question 3

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

Most people gave a good proof that C was $(m-1)$-error correcting. (Some 'model' answers are on Moodle.)

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 we 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 $d(u, v) = d(u', v) = m$." This is nearly right, but it depends how we change positions. So we need to be more specific.

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

§4 Nearest Neighbour Decoding

Example 4.1

Let $C = \{00000, 11100, 00111, 11011\}$.

(i) Suppose $v = 11111$ is received. Then since $d(11011, 11111) = 1$ and $d(u', 11111) \geq 2$ for all codewords $u' \neq 11011$, we would decode v as 11011 using nearest neighbour decoding.

(ii) Suppose $v = 01110$ is received. Then nearest neighbour decoding fails because the nearest codewords to v are 11100 and 00111, both at distance 2.

Quiz on Nearest Neighbour Decoding

Let C be the ternary code of length 4 on Question 1 of Sheet 3 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 on Question 1 of Sheet 3 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 on Question 1 of Sheet 3 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.

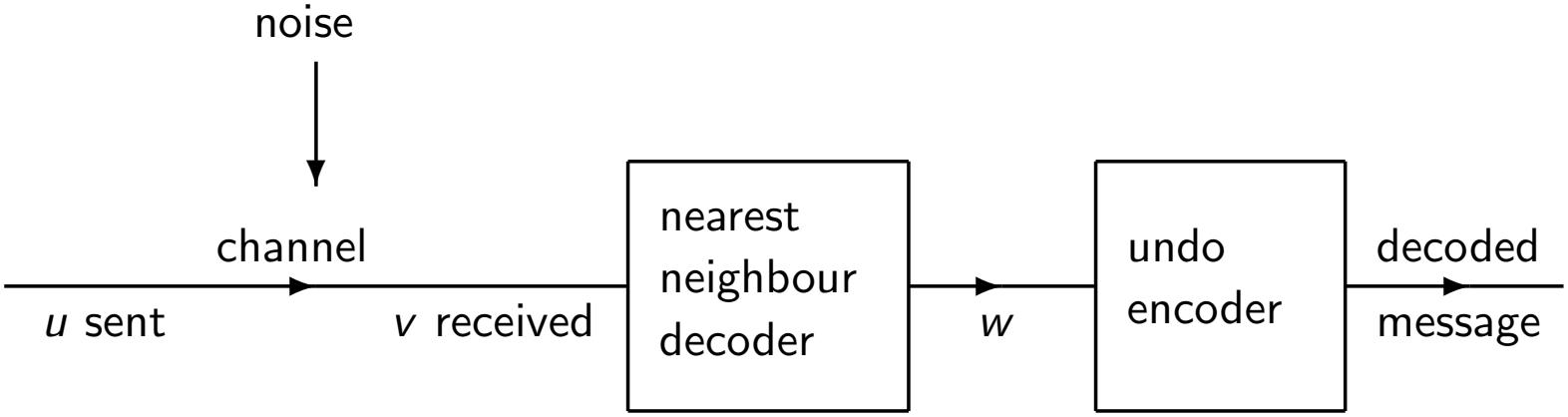
(a) 0222, distance 2 from 2021, 2102, 2210, if wrong 2 errors;

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

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

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.



For earlier examples of this two-stage process, see quiz on Lemma 2.11 or the square code example (Lecture 4).

Connection with t-Error Correcting Codes

In Remark 2.8(3) we remarked that a t-error correcting code promises to correct up to t errors, *provided a suitable decoder is used*. The nearest neighbour decoder is this suitable decoder.

Exercise: Let C be a code. Show that C is t-error correcting $\iff$ whenever at most t errors occur in the channel, decoding a received word using nearest neighbour decoding gives the sent codeword.

Hamming balls

Definition 4.2

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

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

Example 4.3

The Hamming balls about the binary word 0000 are

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

and $B_4(0000)$ consists of all binary words of length 4.

Clarification of Question 3 on Sheet 3

(3) Let u and w be binary words of length n.

(a) Let $1 \leq i \leq n$ and let u' and w' be the binary words obtained by flipping the bit in position i of u and w, respectively. Show that $d(u, w) = d(u', w')$.

(b) Let $1 \leq i < j \leq n$ and let u' and w' be the binary words obtained by swapping the bits in positions i and j of u and w, respectively. Show that $d(u, w) = d(u', w')$.

In (b) u' is obtained by swapping the bits in position i and j of u; v' is obtained by swapping the bits in position i and j of v.

For example if $i = 1$, $j = 2$ and $u = 10101$, $v = 00101$ then

$$u' = 01101, \quad v' = 00101$$

(Also in Question 4, 'Theorem 4.1' should be replaced with 'Theorem 4.6'.)

The Final Interpretation of 't-Error Correcting'

Lemma 4.4

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

Exercise: Let C be a t-error correcting code. Suppose that you receive a word v, and that after searching through some of the codewords in C for its nearest neighbour, you find a codeword u such that $d(u, v) \leq t$. Explain why u must be the unique nearest codeword to v.

Reminder of Key Definition

Definition 2.7

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

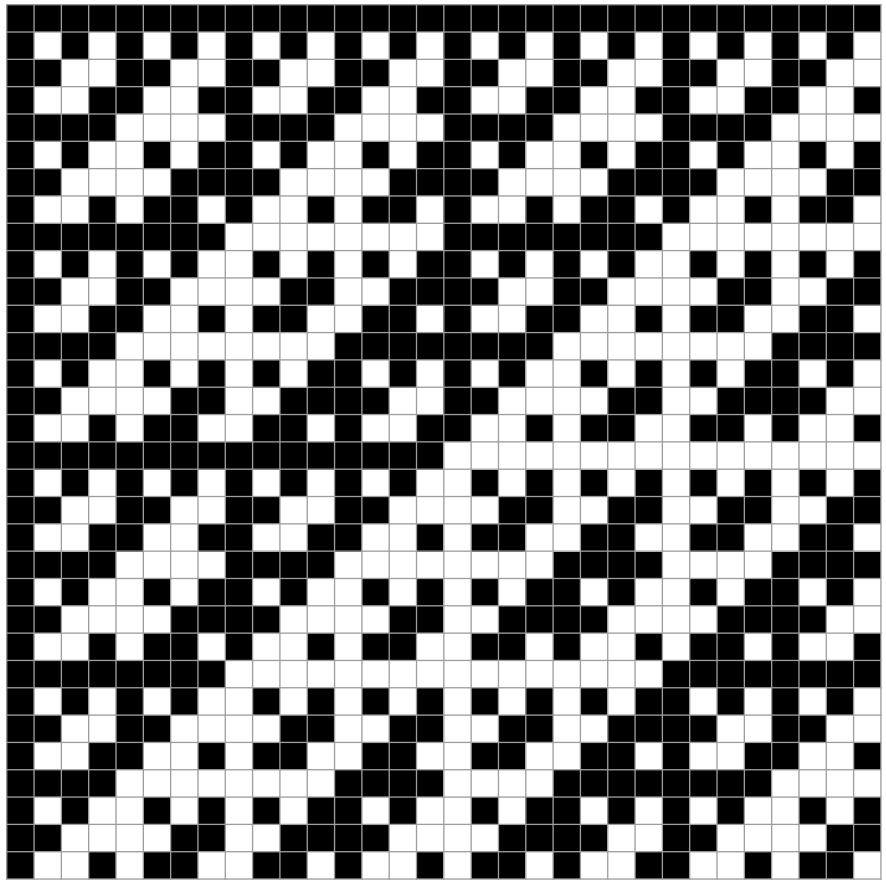
- *t-error detecting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $v \neq u$ and $d(u, v) \leq t$, then $v \notin C$.
- *t-error correcting* if whenever $u \in C$ is a codeword, and v is a word of length n over A such that $d(u, v) \leq t$, then

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

for all codewords $u' \in C$ such that $u' \neq u$.

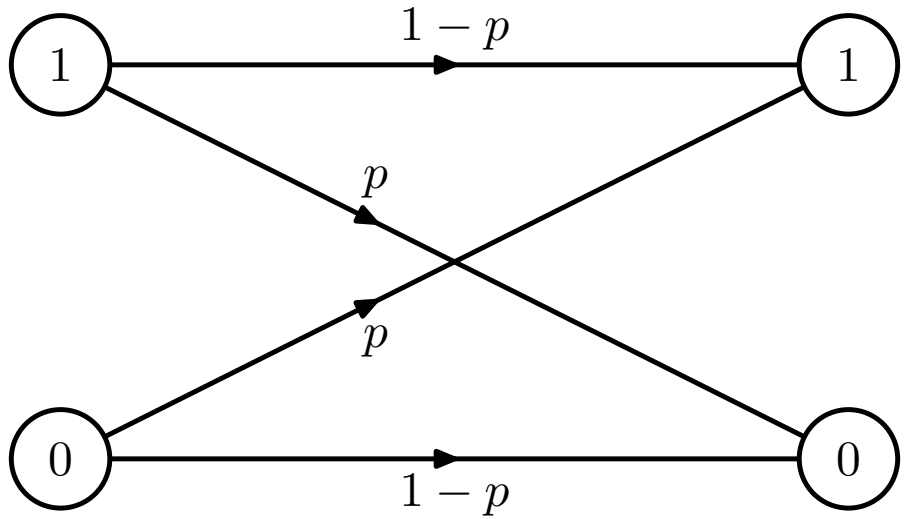
The Mariner 9 Code

32 of the 64 Mariner 9 codewords: ■ = 0 and □ = 1. Suppose we receive the word below. Even with the speed-up in the exercise, perfoming nearest neighbour decoding will take a while . . .



Binary Symmetric Channel

Consider the binary channel in which each transmitted bit flips, so a 0 becomes a 1 and a 1 becomes a 0, independently with probability $p > 0$. This channel is known as the *binary symmetric channel*.



The Probabilistic Justification for Nearest Neighbour Decoding

Theorem 4.5

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

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

where $A(v)$ 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.

See Question 7 on Sheet 3 for an example of the problems that arise if one codeword is much more likely to be sent than the others.

Summary of Part A

In Part A we have seen the formal definition (Definition 2.7) of t-error detecting/correcting codes. The examples in §2 and the results in Lemmas 2.9 and 2.10 show that this definition is a reasonable one.

We then saw other ways of thinking about t-error correcting codes, using minimum distance (Definition 3.1) and nearest neighbour decoding. These are summarised in the following theorem.

Theorem 4.6

Let C be a code. The following are equivalent

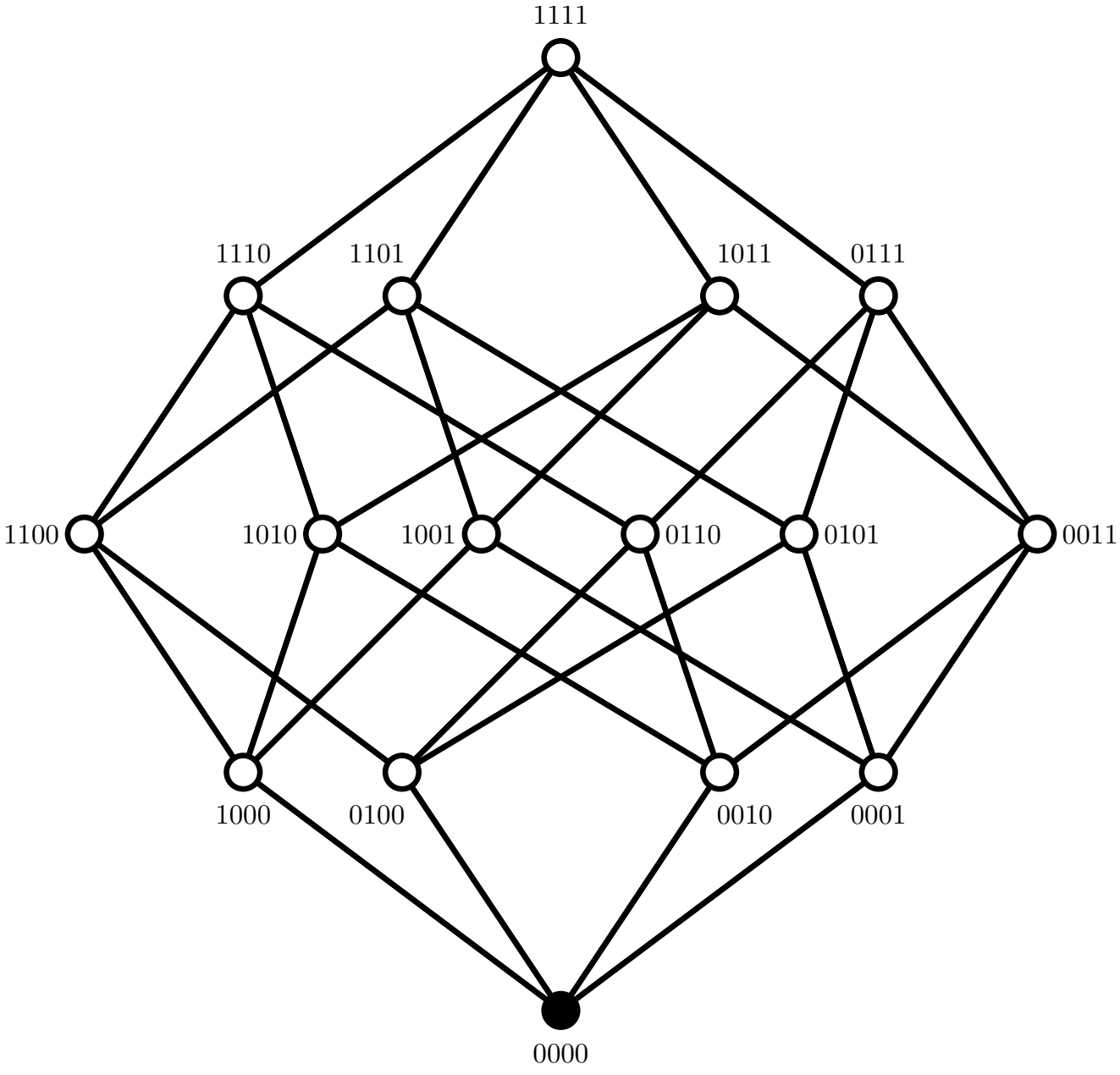
(a) *C is t-error correcting;*

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

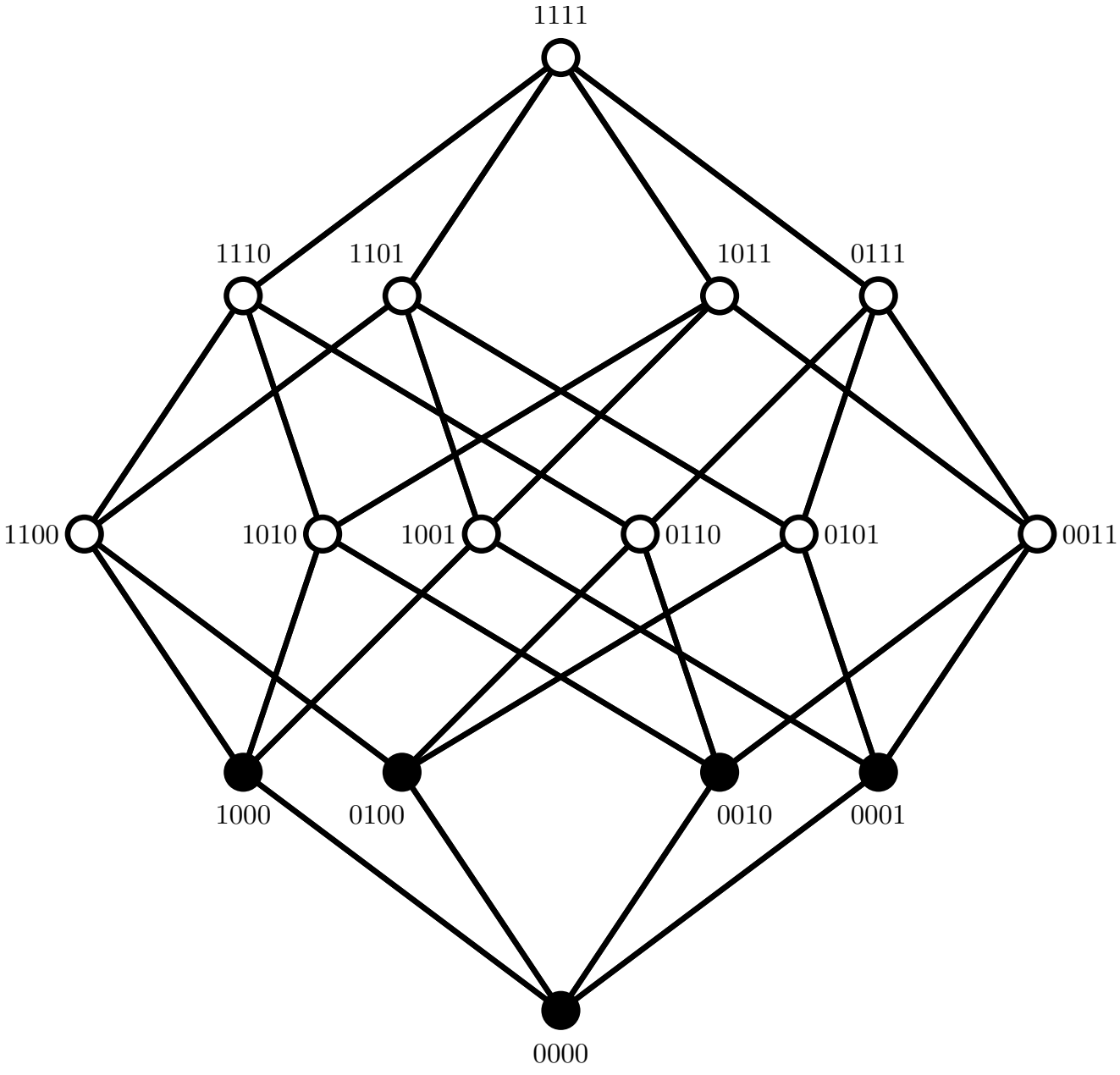
(c) *Nearest neighbour decoding always gives the sent codeword, provided at most t errors occur;*

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

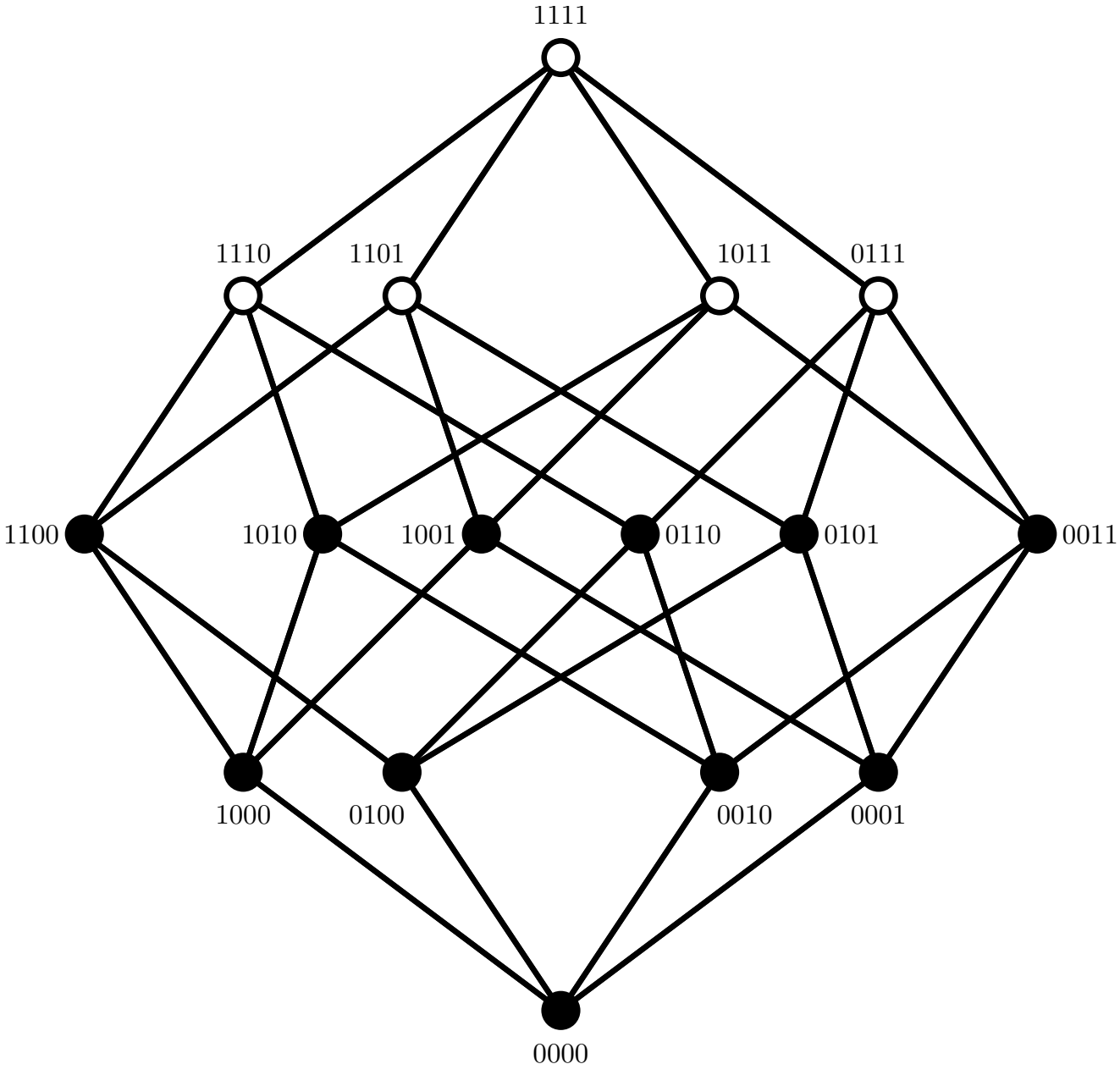
$B_0(0000)$


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

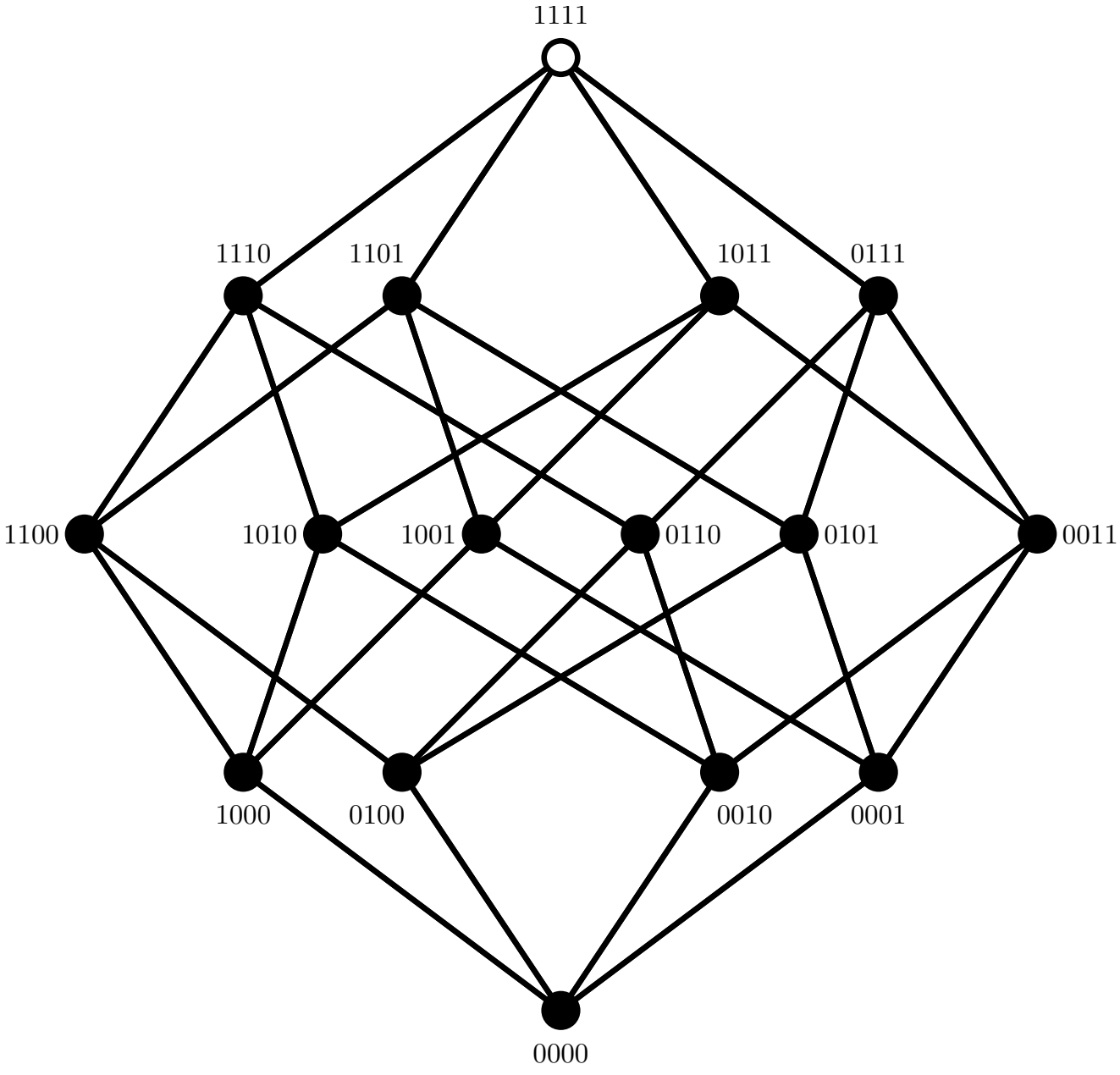
$B_1(0000)$


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

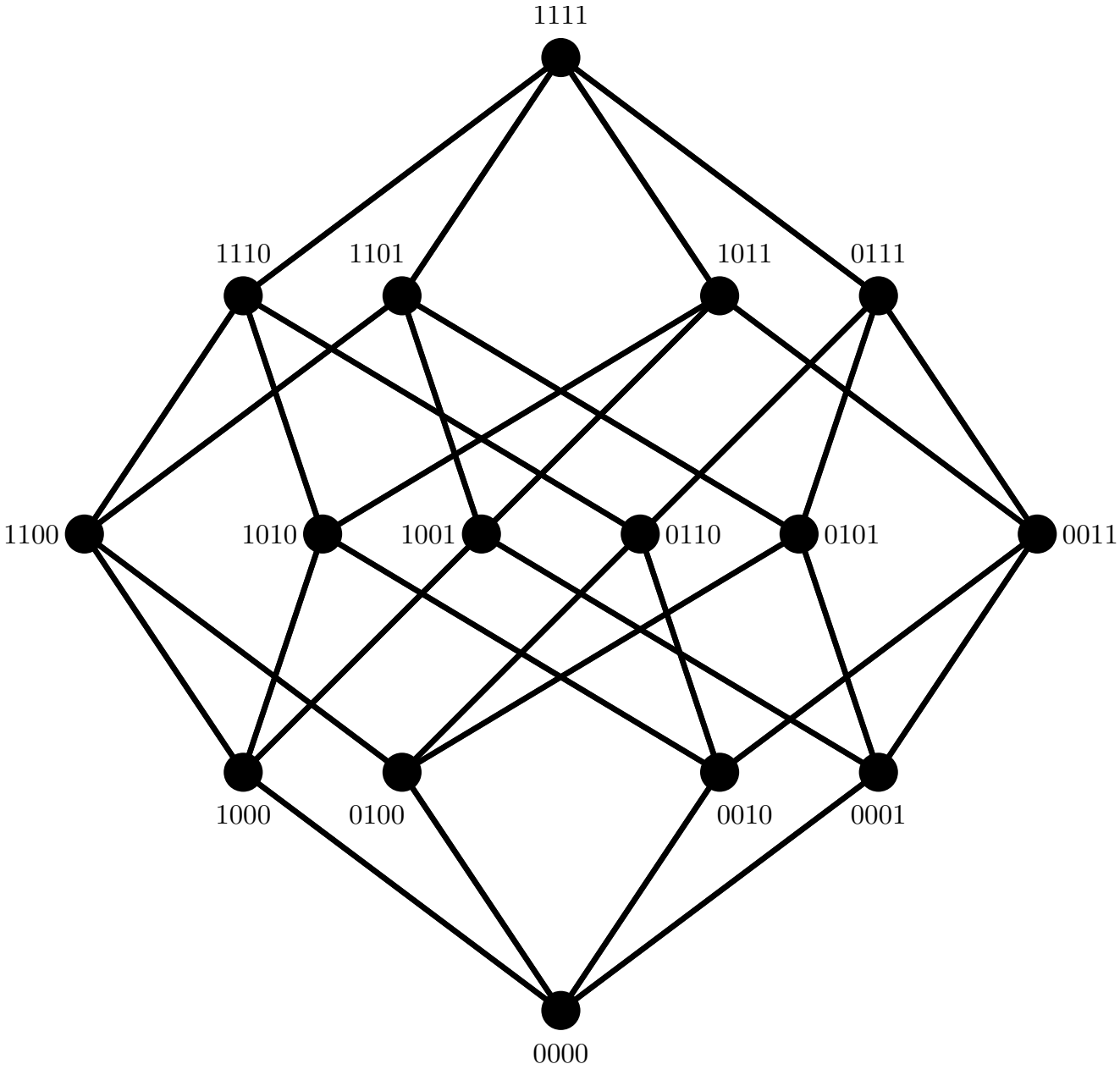
$B_2(0000)$


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

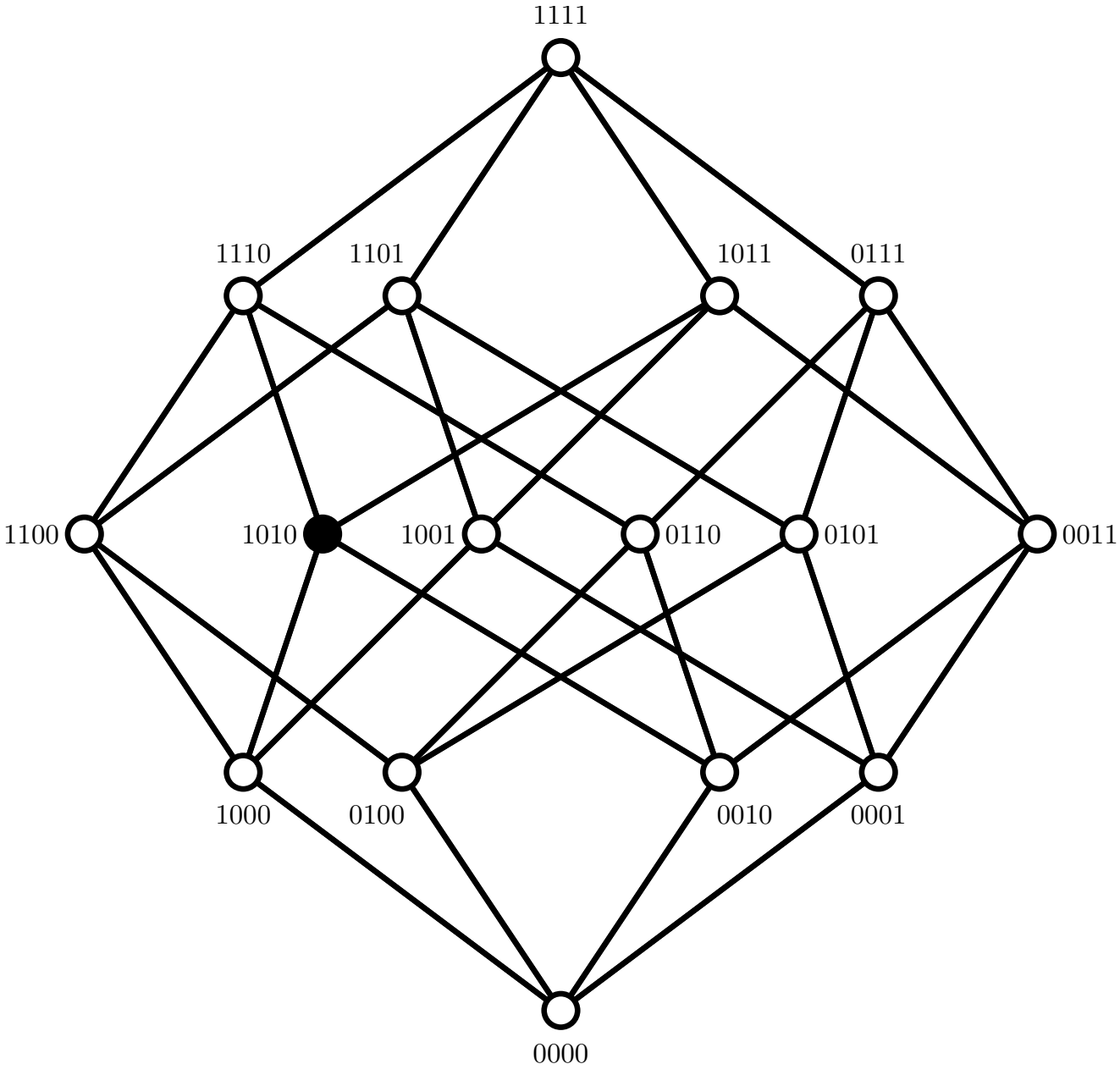
$B_3(0000)$


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

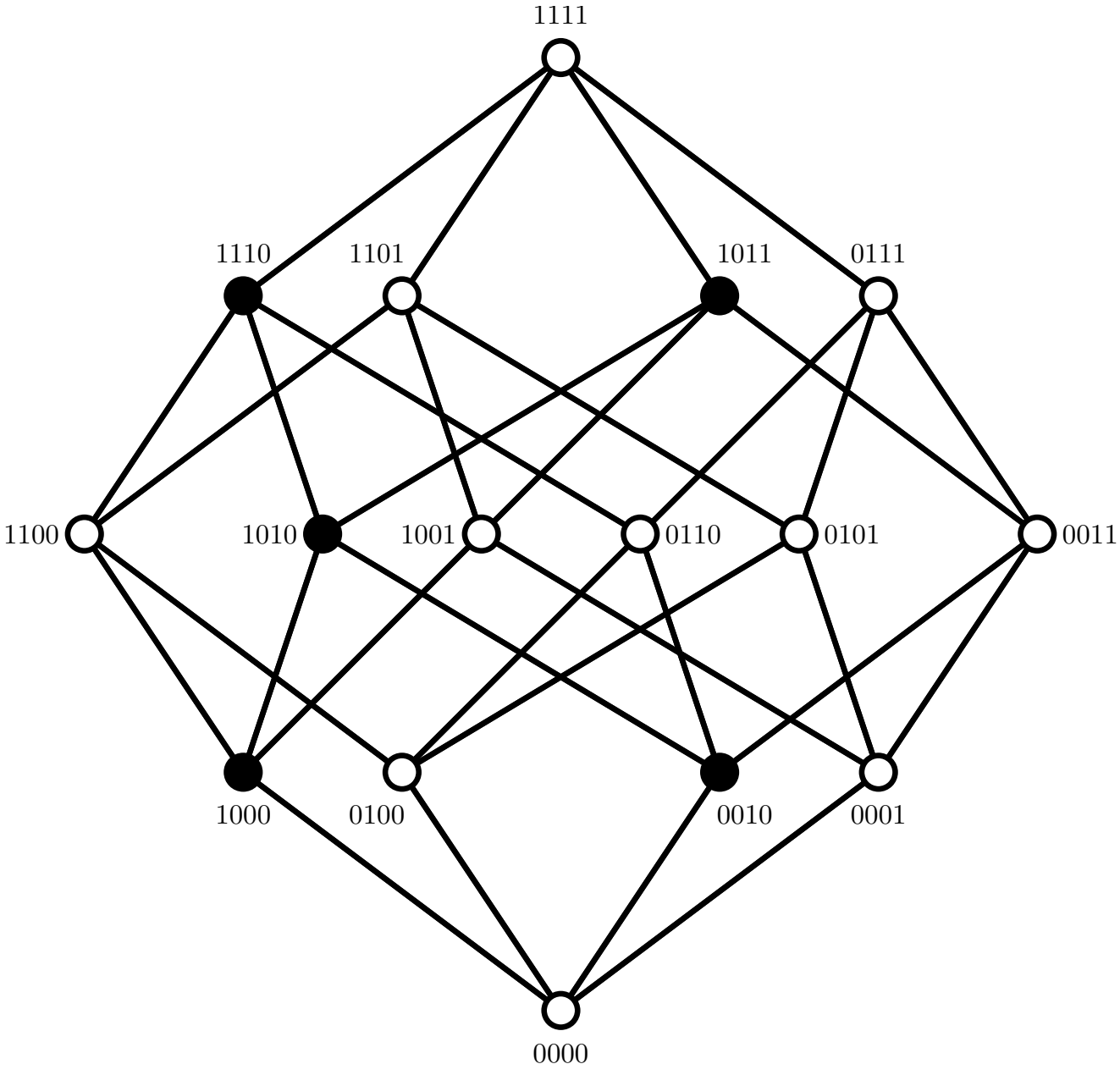
$B_4(0000)$


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

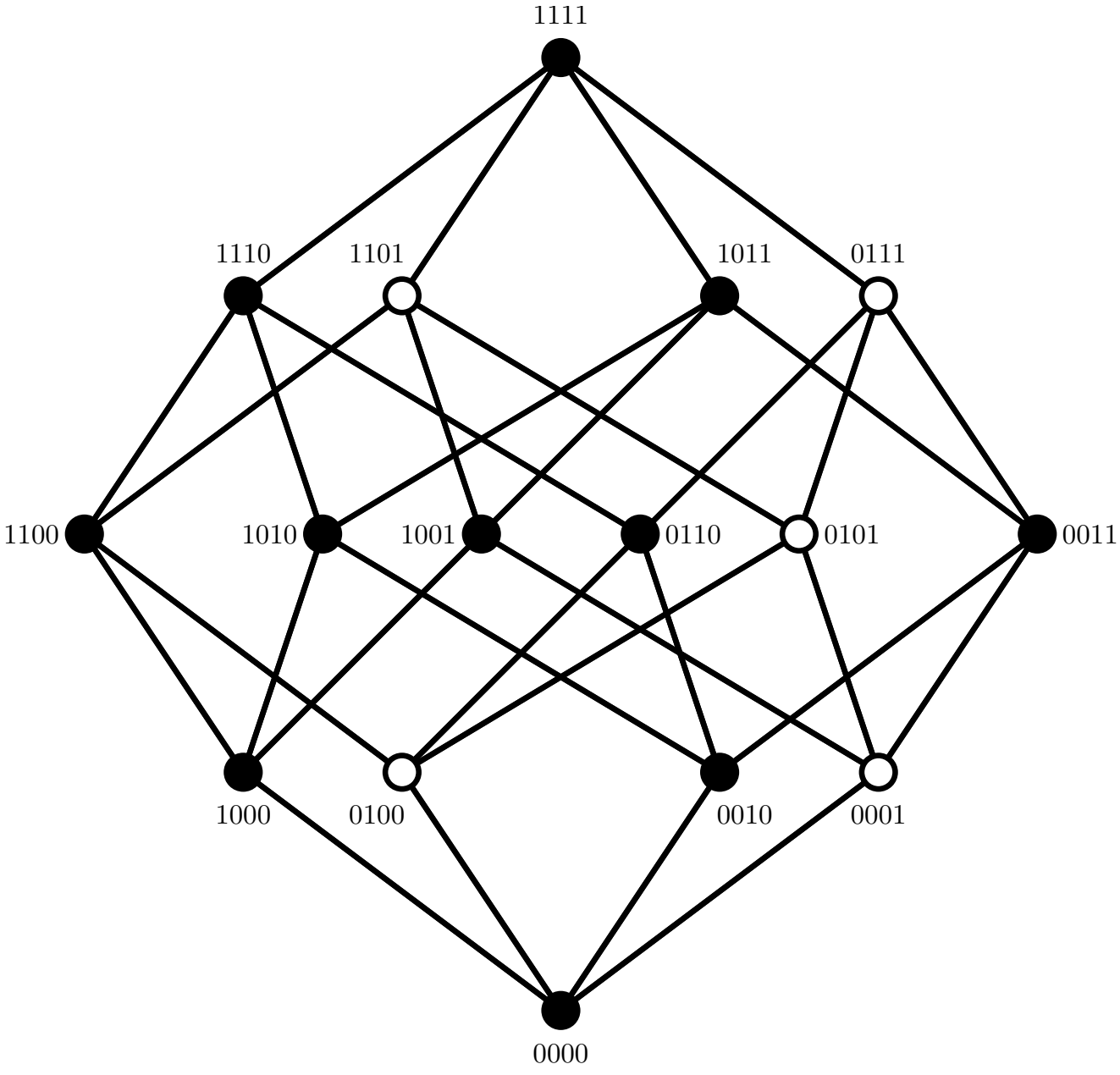
$B_0(1010)$


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

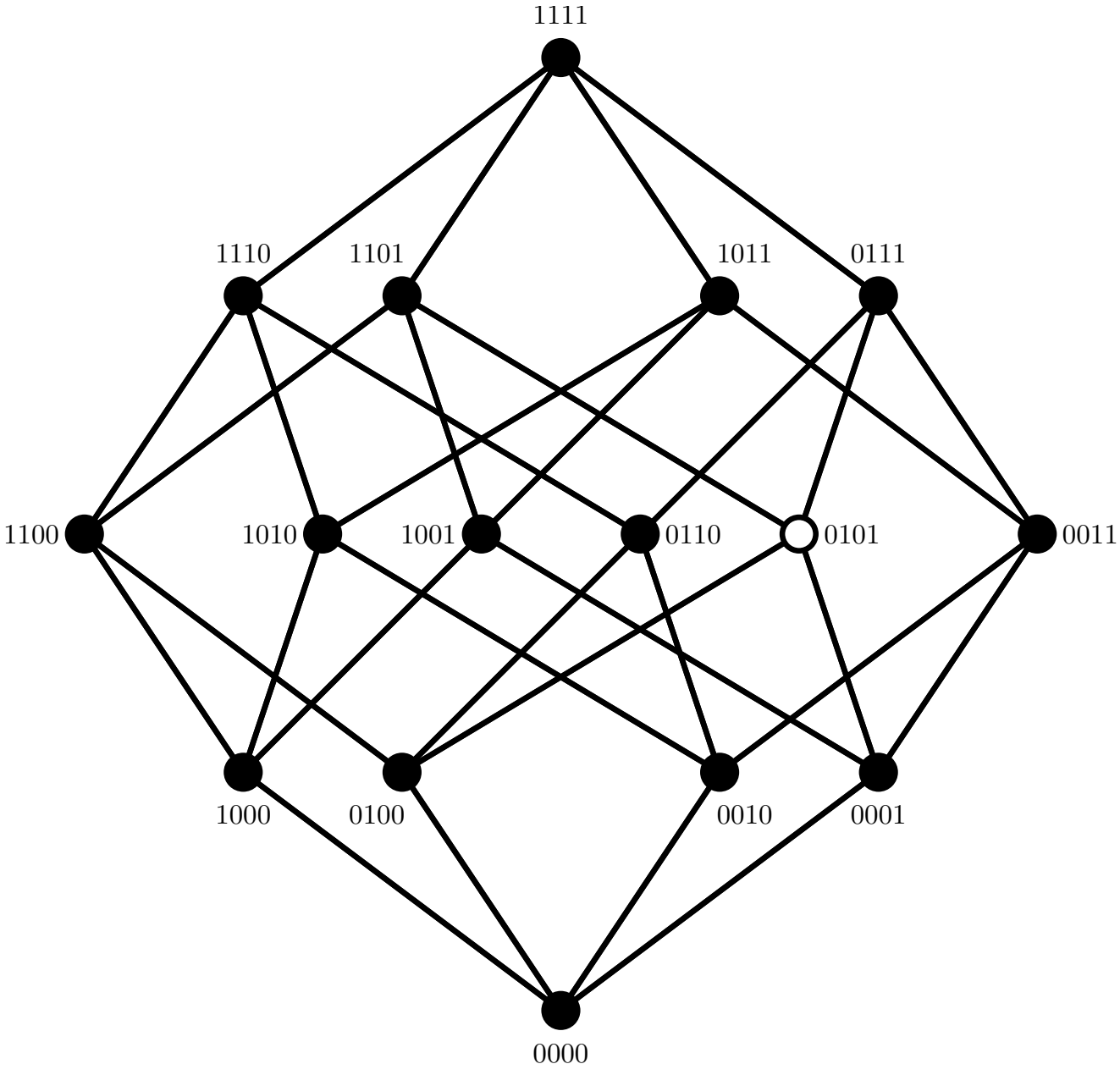
$B_1(1010)$


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

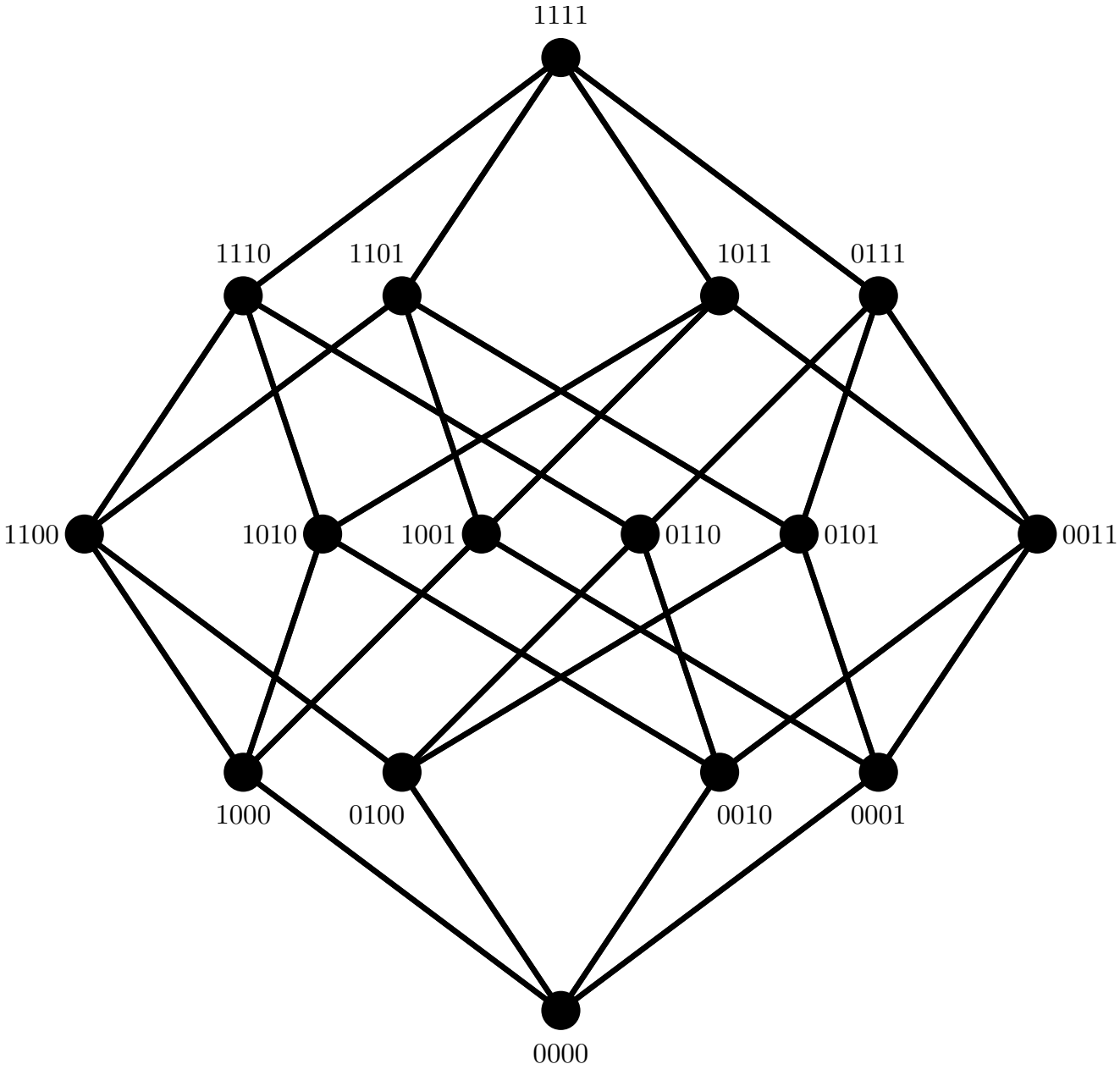
$B_2(1010)$


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

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

Admin

Correction to Question 3 on Sheet 5: minimum distance should be $n - 1$, not 3. A corrected version of the sheet is on Moodle.

If you have the sign-up sheet at end of lecture, please give to the lecturer.

§5 Main Coding Theory Problem and Hamming's Bound

Problem 5.1

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

(1) small length;

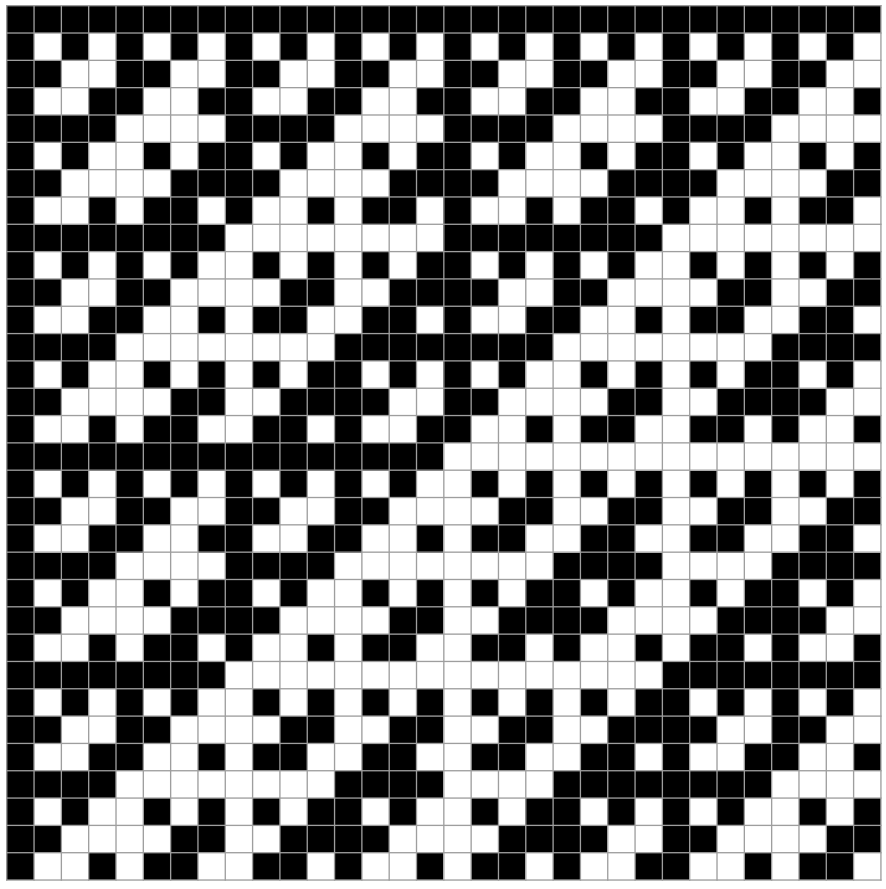
(2) large size;

(3) high minimum distance.

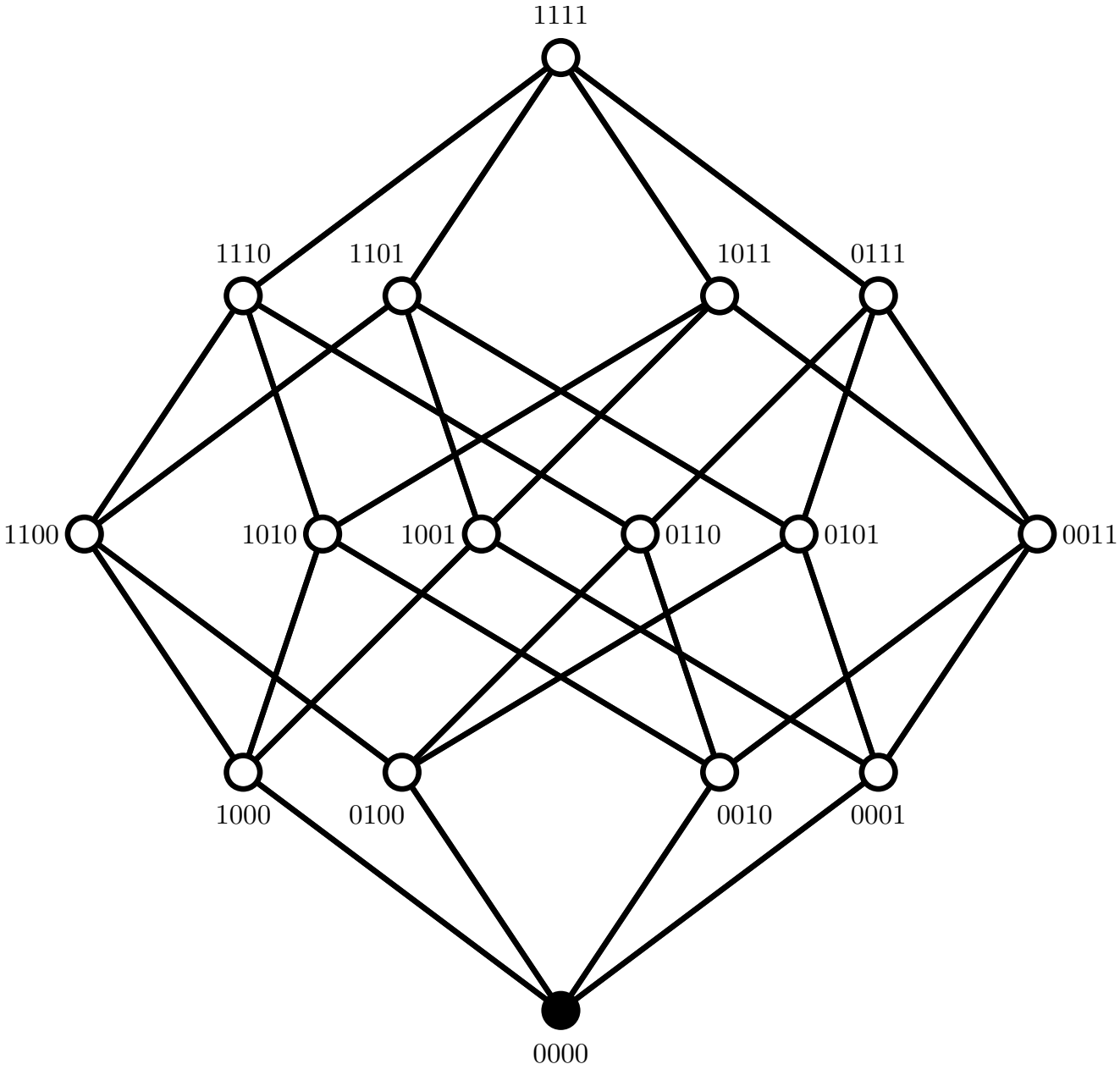
Equivalently, we want to find (n, M, d)-codes over the given alphabet with small n, high M and high d.

The Mariner 9 Code

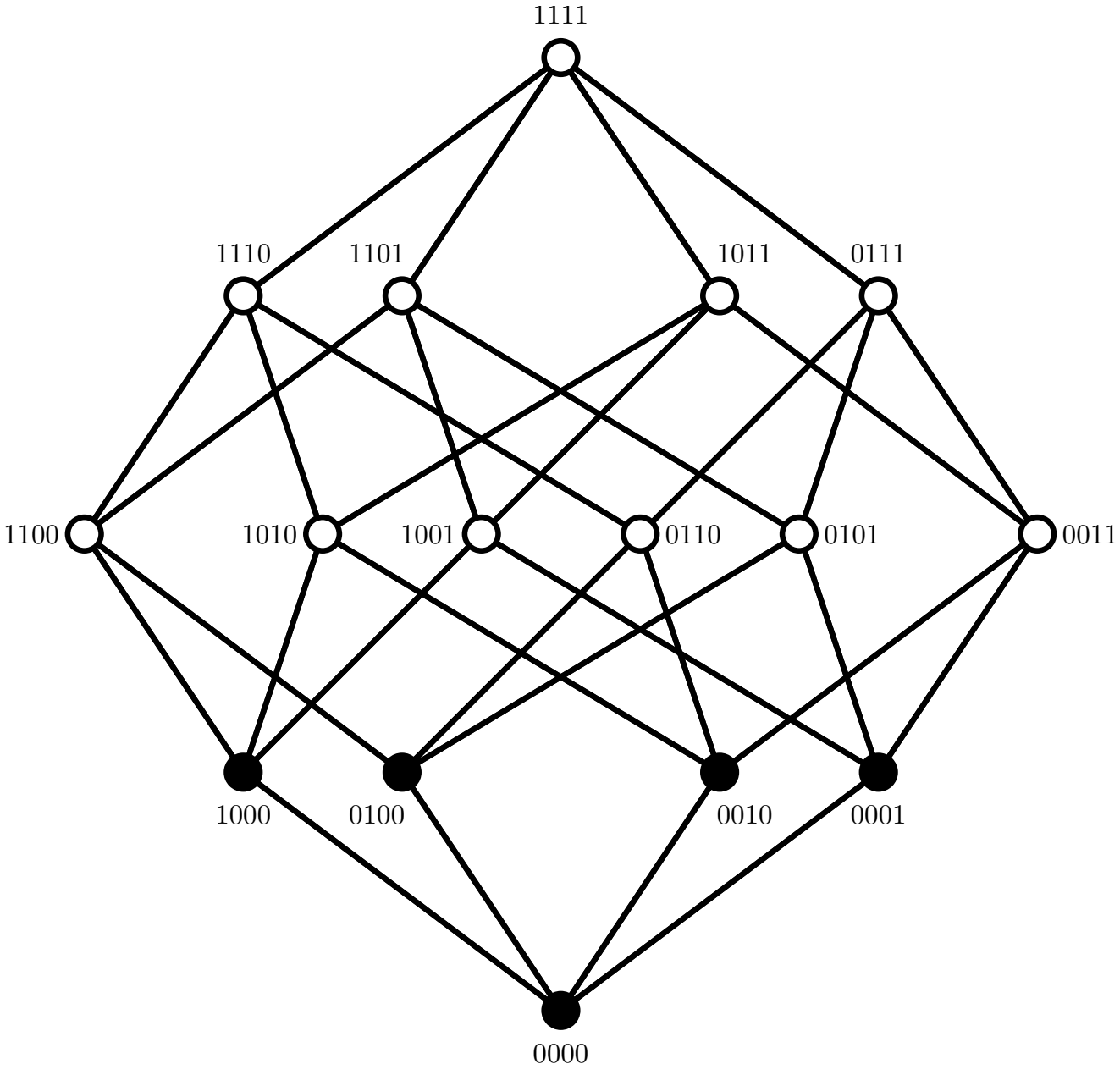
32 of the 64 Mariner 9 codewords: ■ = 0 and □ = 1. Suppose we receive the word below. Looking through all 64 codewords to find the closest one will take some time.



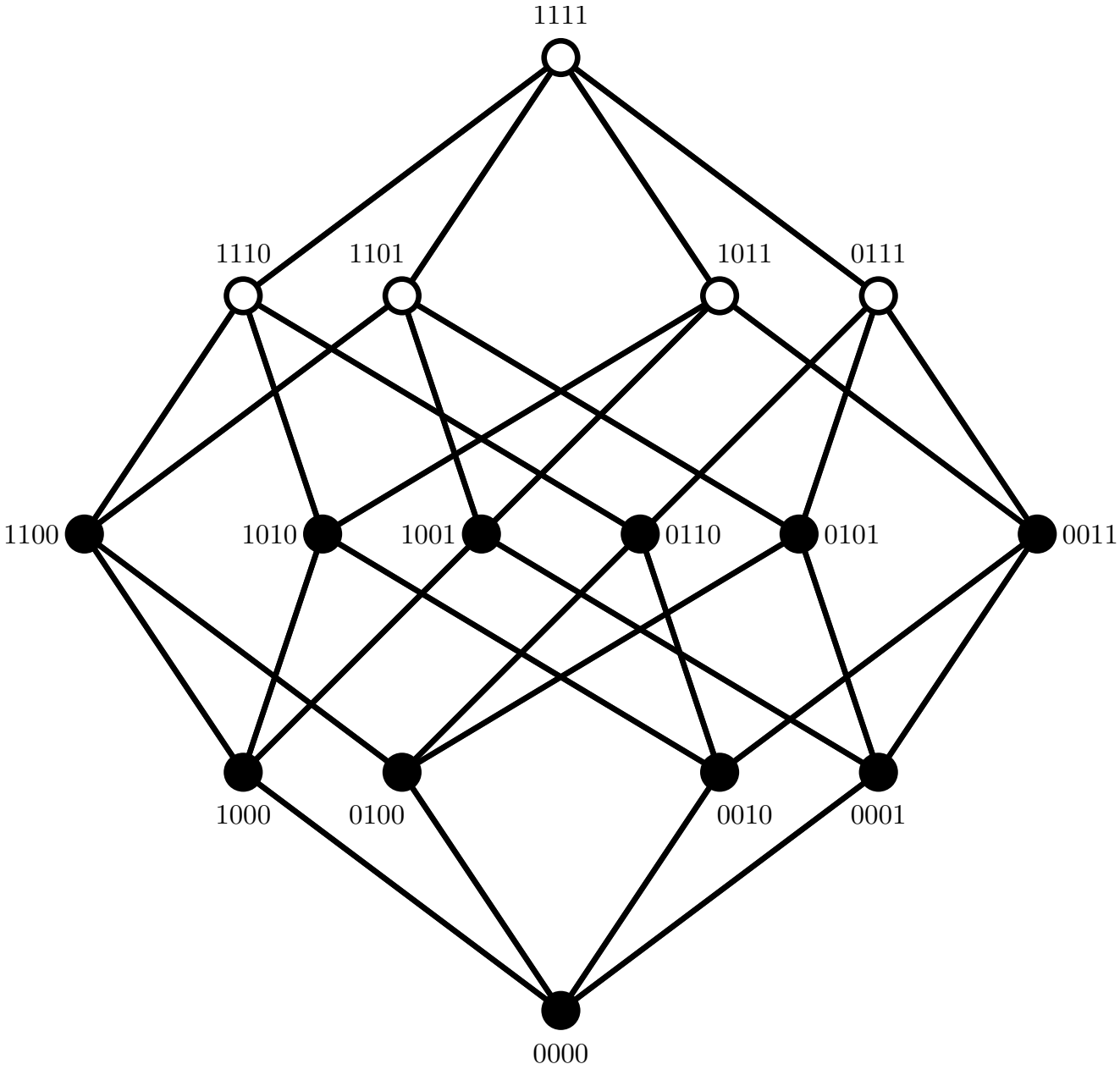
$B_0(0000)$


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

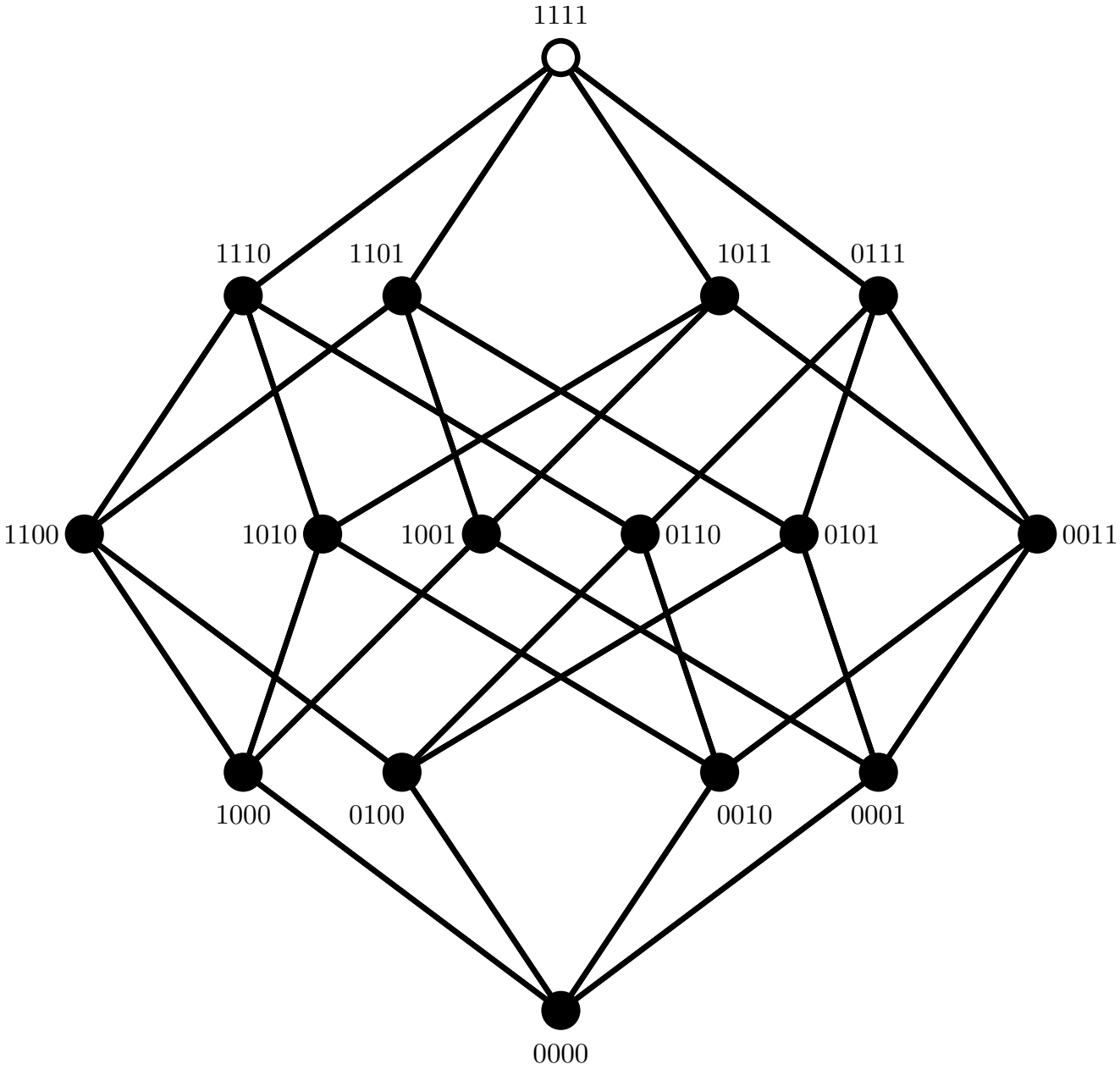
$B_1(0000)$


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

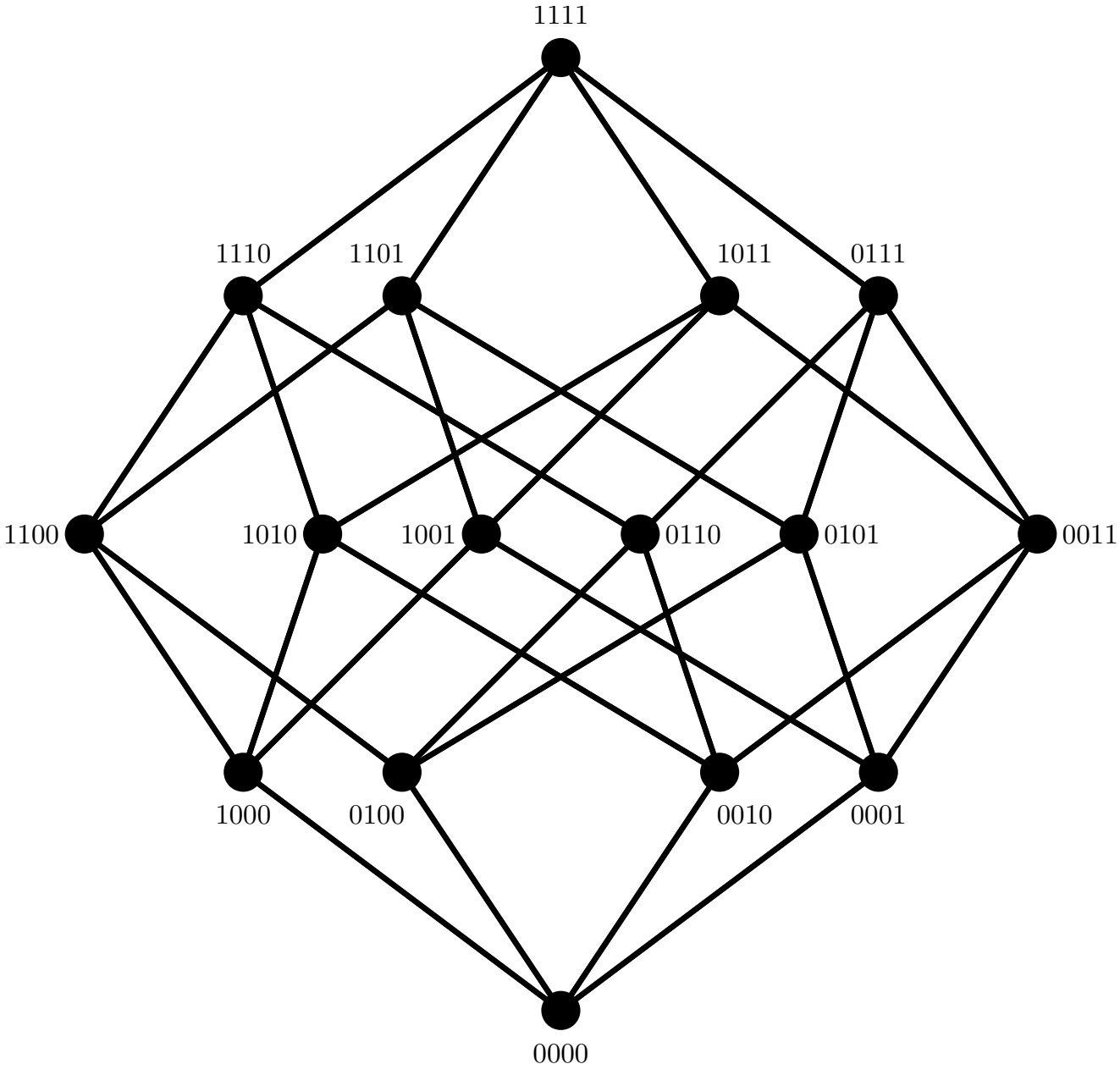
$B_2(0000)$


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

Hamming's Packing Bounds

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

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

We also need the result of Question 5 of Sheet 3, that if C has minimum distance at least $2t+1$ then the Hamming balls of radius t about distinct codewords are disjoint.

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

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

Correction to Proof of Theorem 5.4

Plase change $e = \lfloor (n-1)/2 \rfloor$ to $e = \lfloor (d-1)/2 \rfloor$. Then the calculation that $2e + 1 \leq d$, and so the Hamming balls of radius e about codewords are disjoint, should read

$$2\lfloor \frac{d-1}{2} \rfloor + 1 \leq 2\left(\frac{d-1}{2}\right) + 1 = (d-1) + 1 \leq d.$$

My apologies for this error.

Examples of Hamming's Packing Bound

Hamming's Packing Bound for the maximum size of a binary $(n, M, 3)$-code.

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

Hamming's bound is a **necessary condition** for a code to exist. It is not **sufficient**.

Examples of Hamming's Packing Bound

Hamming's Packing Bound for the maximum size of a binary $(n, M, 3)$-code.

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

Hamming's bound is a **necessary condition** for a code to exist. It is not **sufficient**. For example, the table shows that a binary code of length 5 and minimum distance 3 has size at most 5. But by Question 5 on Sheet 3, such a code has size at most 4; one example is

$$00000, \quad 11100, \quad 00111, \quad 11011.$$

§6 Equivalences of Codes and $A_q(n, d)$

Definition 6.1

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.

§6 Equivalences of Codes and $A_q(n, d)$

Definition 6.1

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 the choice of which particular q-ary alphabet to use makes no difference to $A_q(n, d)$.

§6 Equivalences of Codes and $A_q(n, d)$

Definition 6.1

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 the choice of which particular q-ary alphabet to use makes no difference to $A_q(n, d)$.

Exercise: 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.

§6 Equivalences of Codes and $A_q(n, d)$

Definition 6.1

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 the choice of which particular q-ary alphabet to use makes no difference to $A_q(n, d)$.

Exercise: 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.

Lemma 6.2

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

Definition 6.3

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.

Lemma 6.4

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 (a) and/or shuffling positions (b) then $d(u, w) = d(u', w')$.

Example of Equivalences

Example 6.5

Consider the four binary codes

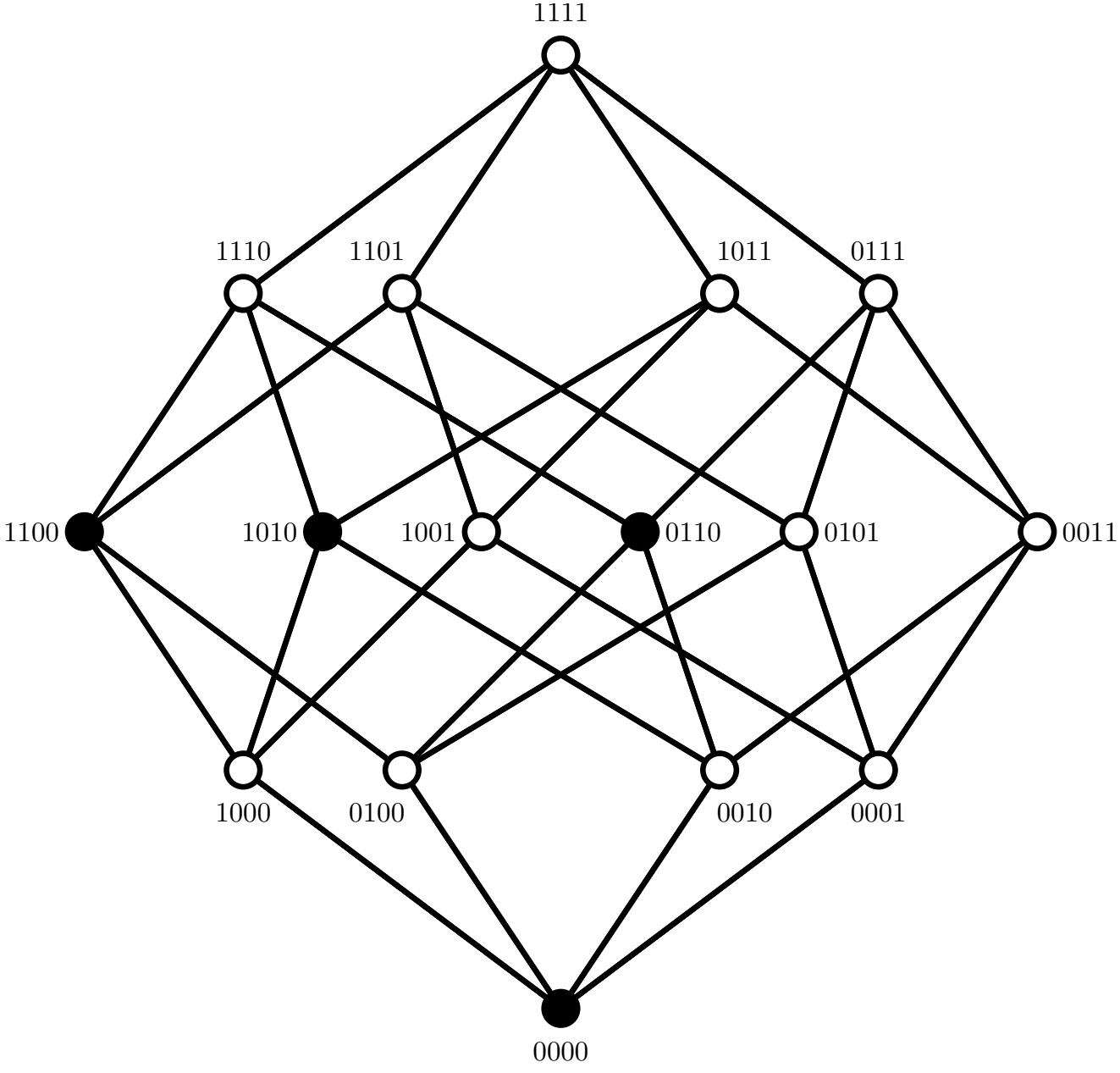
$$C = \{0000, 1100, 1010, 0110\}$$
$$C' = \{1010, 0110, 0011, 1111\}$$
$$D = \{0000, 1100, 0011, 1111\}$$
$$E = \{1000, 0100, 0010, 0001\}$$

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.

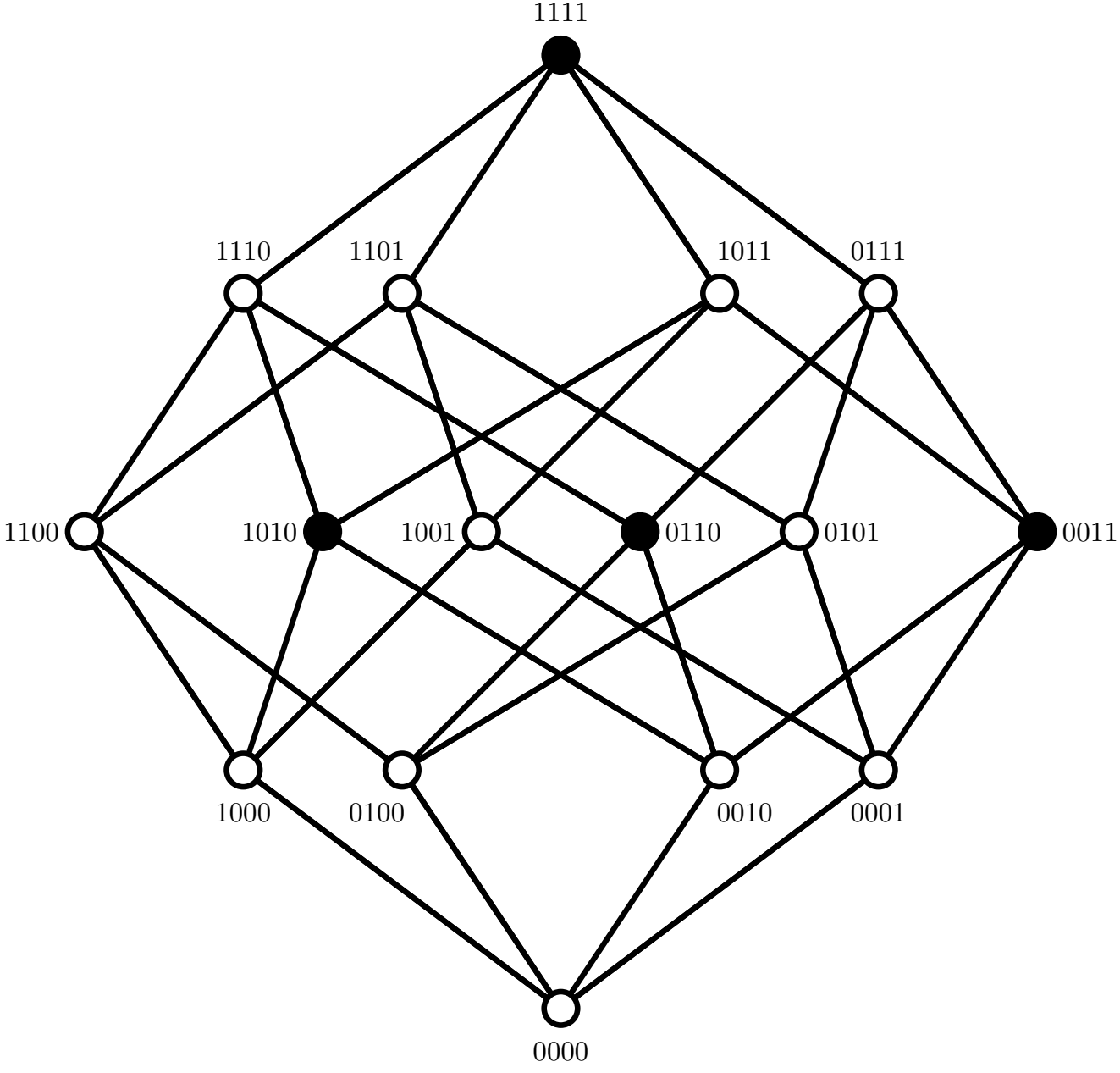
For C and D this can be shown quite easily: there are codewords in D that are distance 4 apart, for example $d(0000, 1111) = 4$, but all codewords in C are distance 2 apart.

For C and E we need another argument, because all codewords in both codes are distance 2 apart. (See Sheet 4: $C = C_1$, $E = C_3$.)

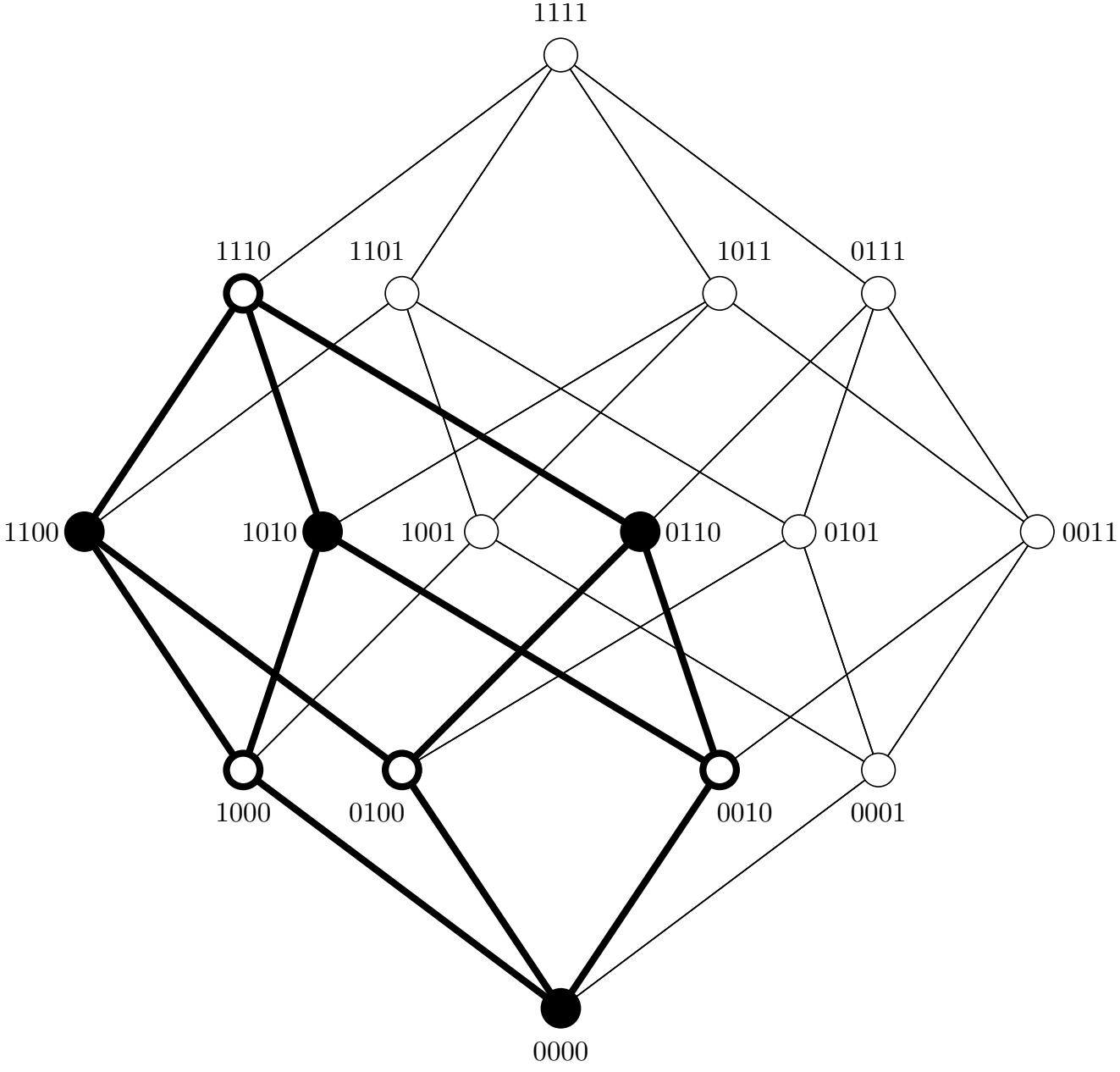
Example 6.5: $C = \{0000, 1100, 1010, 0110\}$



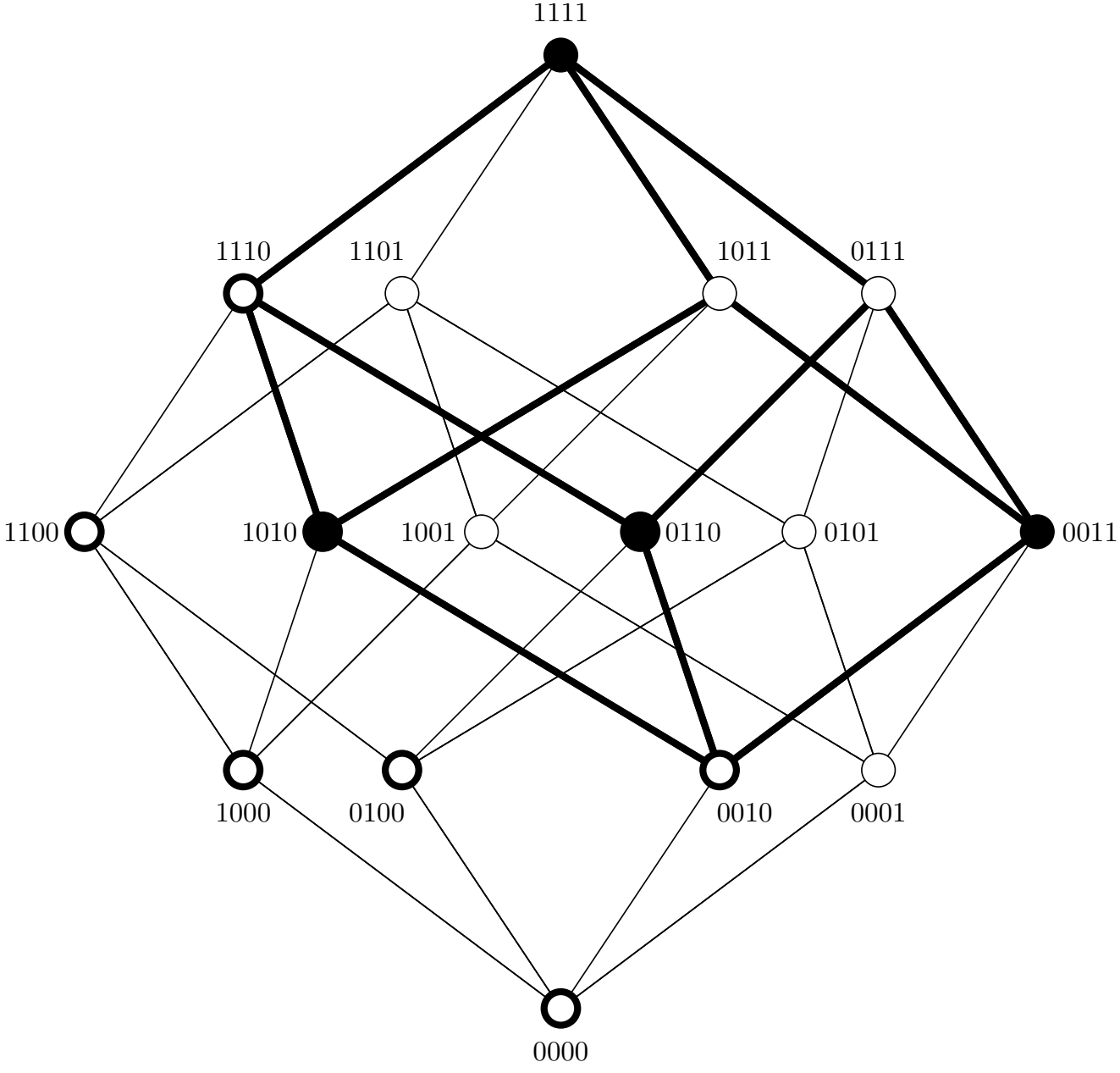
Example 6.5: $C' = \{1010, 0110, 0011, 1111\}$.



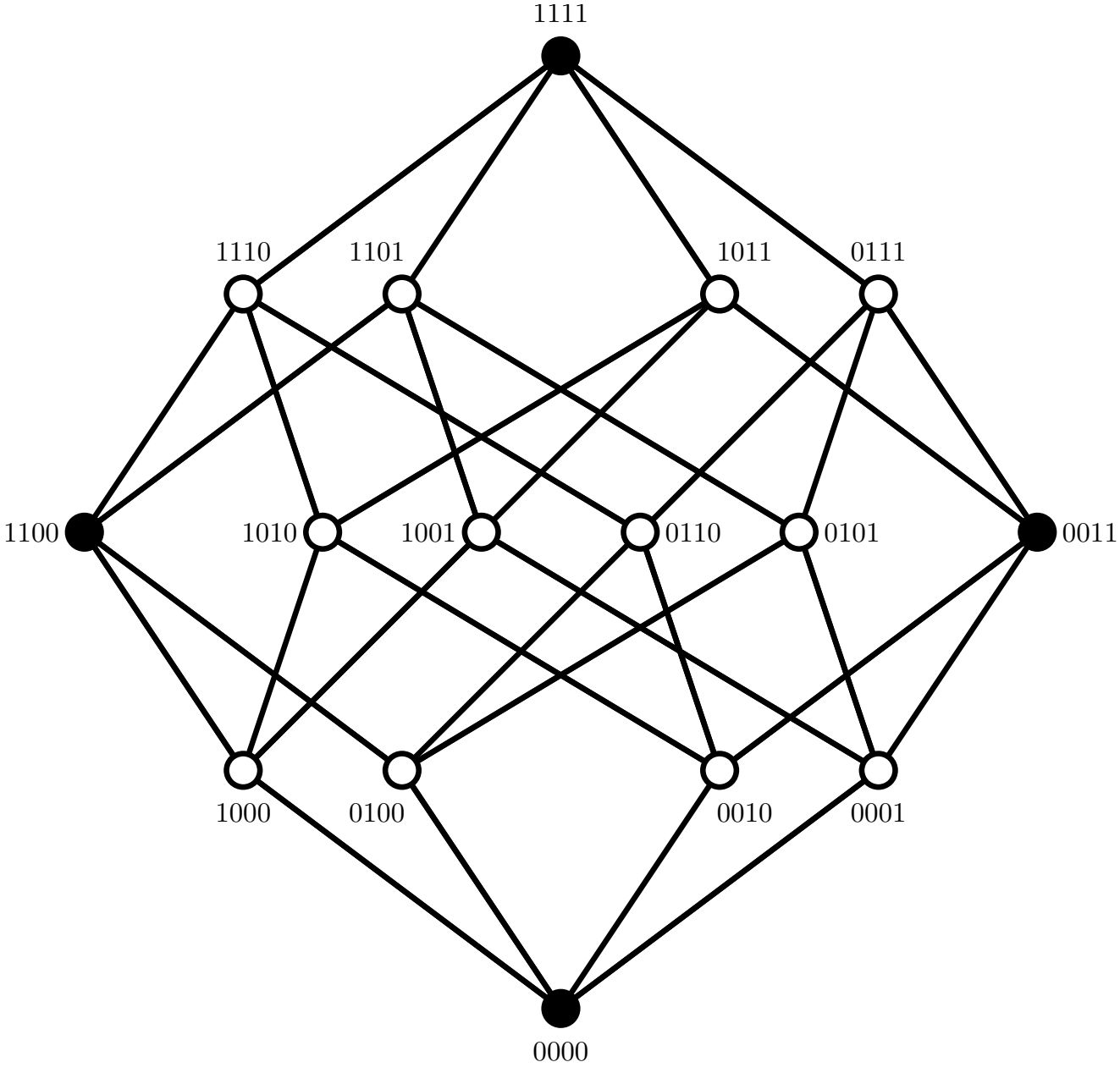
Example 6.5: $C = \{0000, 1100, 1010, 0110\}$



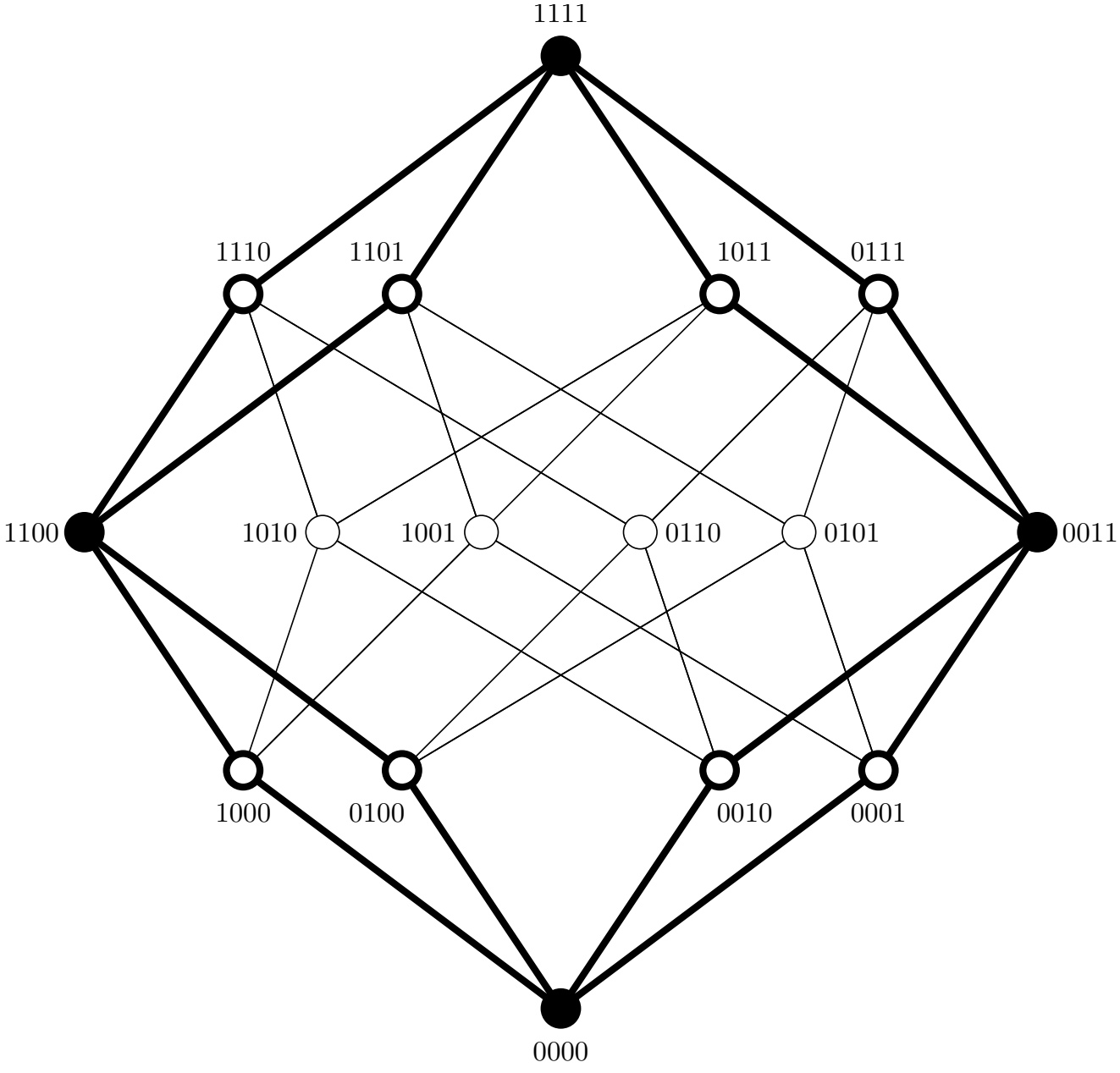
Example 6.5: $C' = \{1010, 0110, 0011, 1111\}$.



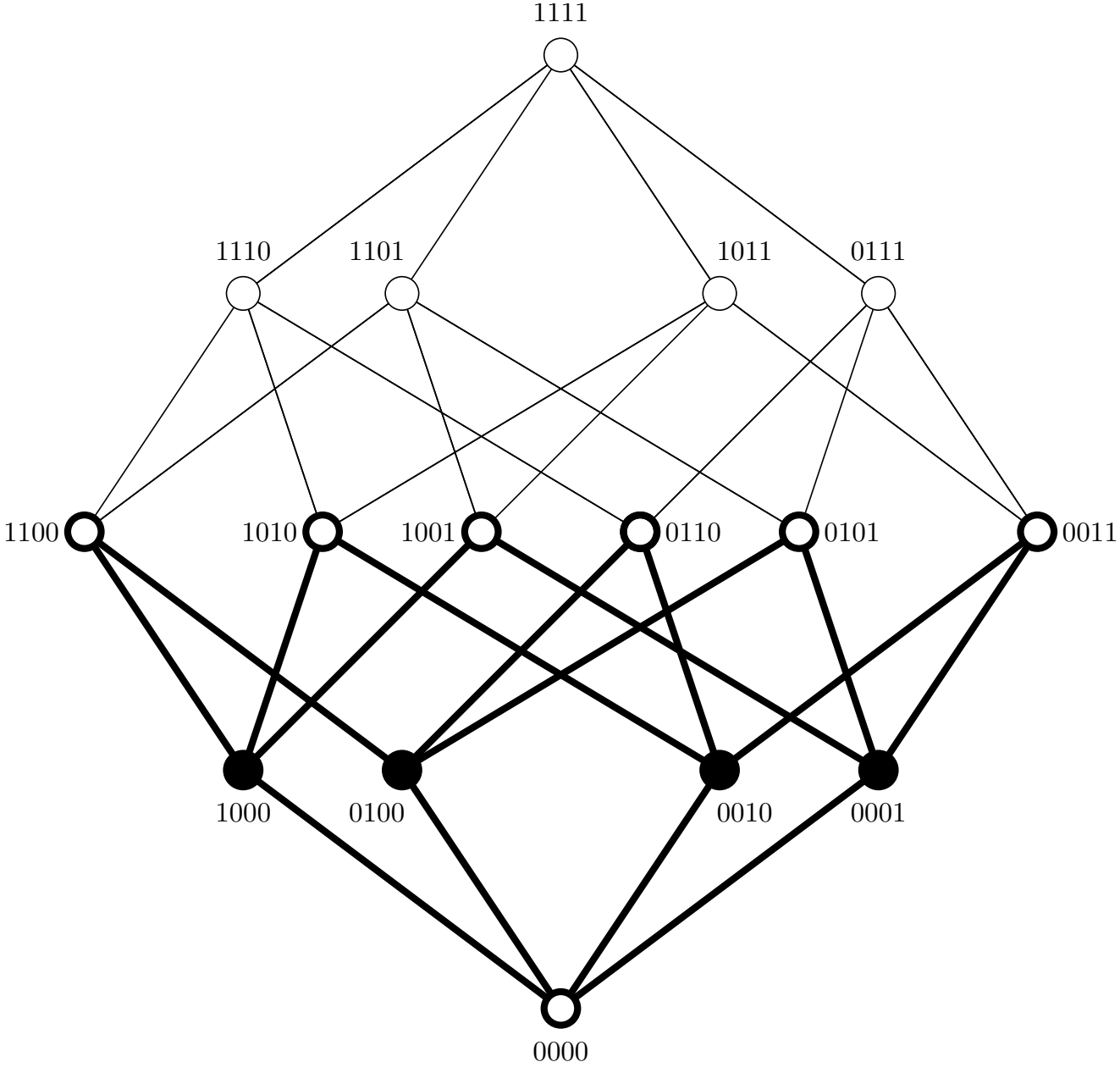
Example 6.5: $D = \{0000, 1100, 0011, 1111\}$.



Example 6.5: $D = \{0000, 1100, 0011, 1111\}$.



Example 6.5: $E = \{1000, 0100, 0010, 0001\}$.



Using Equivalences to Find Largest Sizes of Codes

We will use equivalences of codes to find $A_2(8, 5)$.

In the next lemma, 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 $\mathrm{wt}(u) \geq r$ and $\mathrm{wt}(w) \geq s$. If $r + s \geq n$ then

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

Theorem 6.7

$A_2(8, 5) = 4$.

Other values of $A_n(n, d)$ for small n.

For background, the table below shows some other values of $A_2(n, d)$. 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

§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.

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 $Y_{ij} = b$.

Example 7.4

Two mutually orthogonal Latin squares (MOLs) over the alphabet $\{0, 1, 2, 3\}$ are shown below.

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

To show that these squares are orthogonal we form a new square whose entries are pairs of entries from the two squares,

$$\begin{array}{cccc} 00 & 11 & 22 & 33 \\ 12 & 03 & 30 & 21 \\ 23 & 32 & 01 & 10 \\ 31 & 20 & 13 & 02 \end{array}$$

and then check that each of the 16 pairs $00, 01, \ldots, 33$ appears exactly once.

Hunting for MOLs

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

Remark 7.5

There are no MOLs of order 6. *This was conjectured by Euler, but not proved until 1900.*

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.*

Connection between MOLs and 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.

Example 7.8

(Variation on example in notes.) Given the 4-ary $(4, 16, 3)$-code with codewords

0030	0101	0212	0323	1002	1133	1220	1311
2013	2122	2231	2300	3021	3110	3203	3332

construct a pair of MOLs of order 4. Conversely, given this pair of MOLs we can recover the code.

Sheet 4: Question 1 on Equivalences of Codes

It is very tempting to argue that if C and C' are codes and $u \in C$ is a codeword such that $u \in C'$, then any equivalence of C with C' must send u to itself.

However this is false in general.

For example, consider the binary $(4, 3, 2)$-codes

$$C = \{0000, 1100, 1111\}$$
$$C' = \{0000, 1100, 0011\}.$$

Then C and C' are equivalent.

Sheet 4: Question 1 on Equivalences of Codes

It is very tempting to argue that if C and C' are codes and $u \in C$ is a codeword such that $u \in C'$, then any equivalence of C with C' must send u to itself.

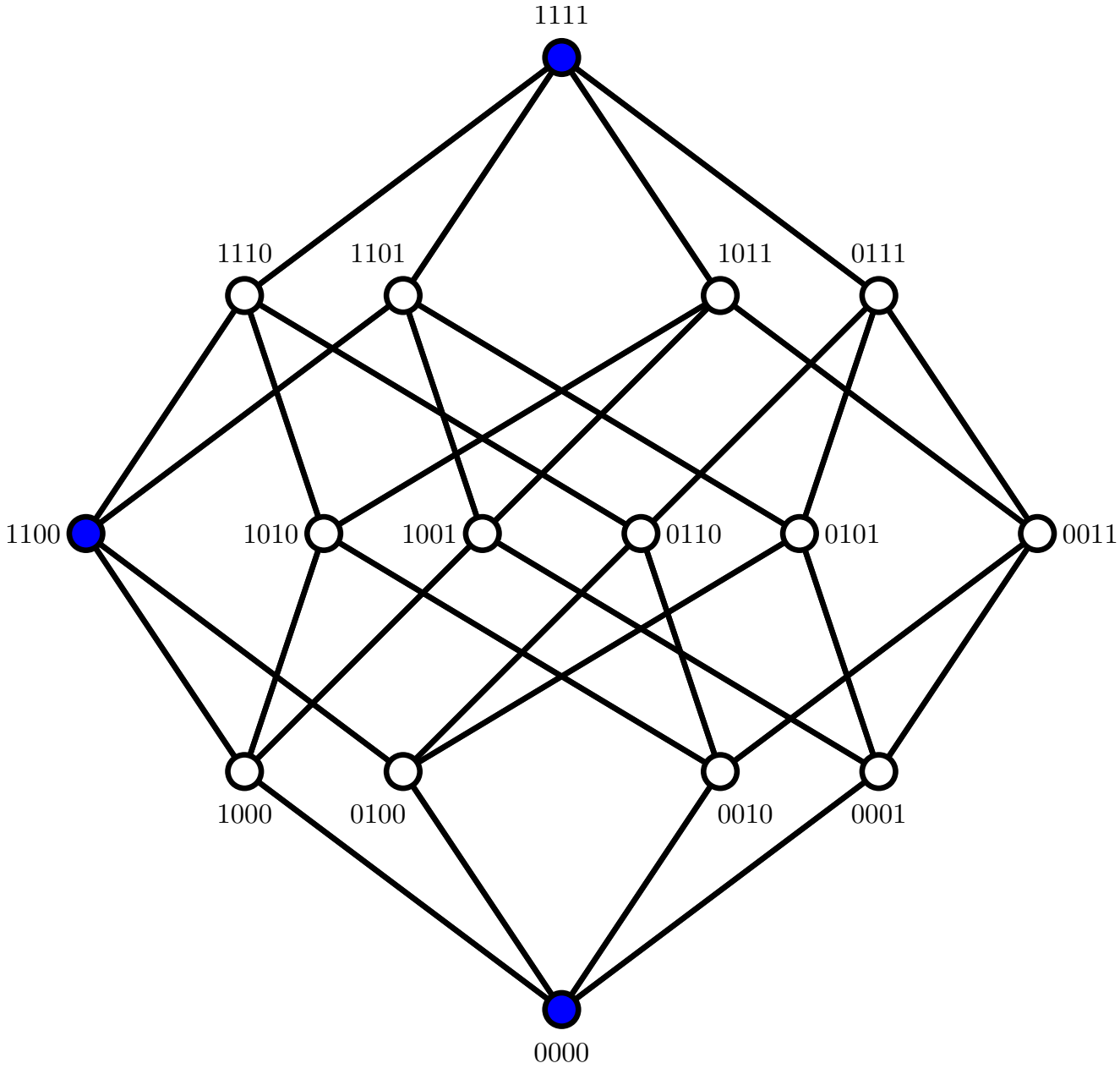
However this is false in general.

For example, consider the binary $(4, 3, 2)$-codes

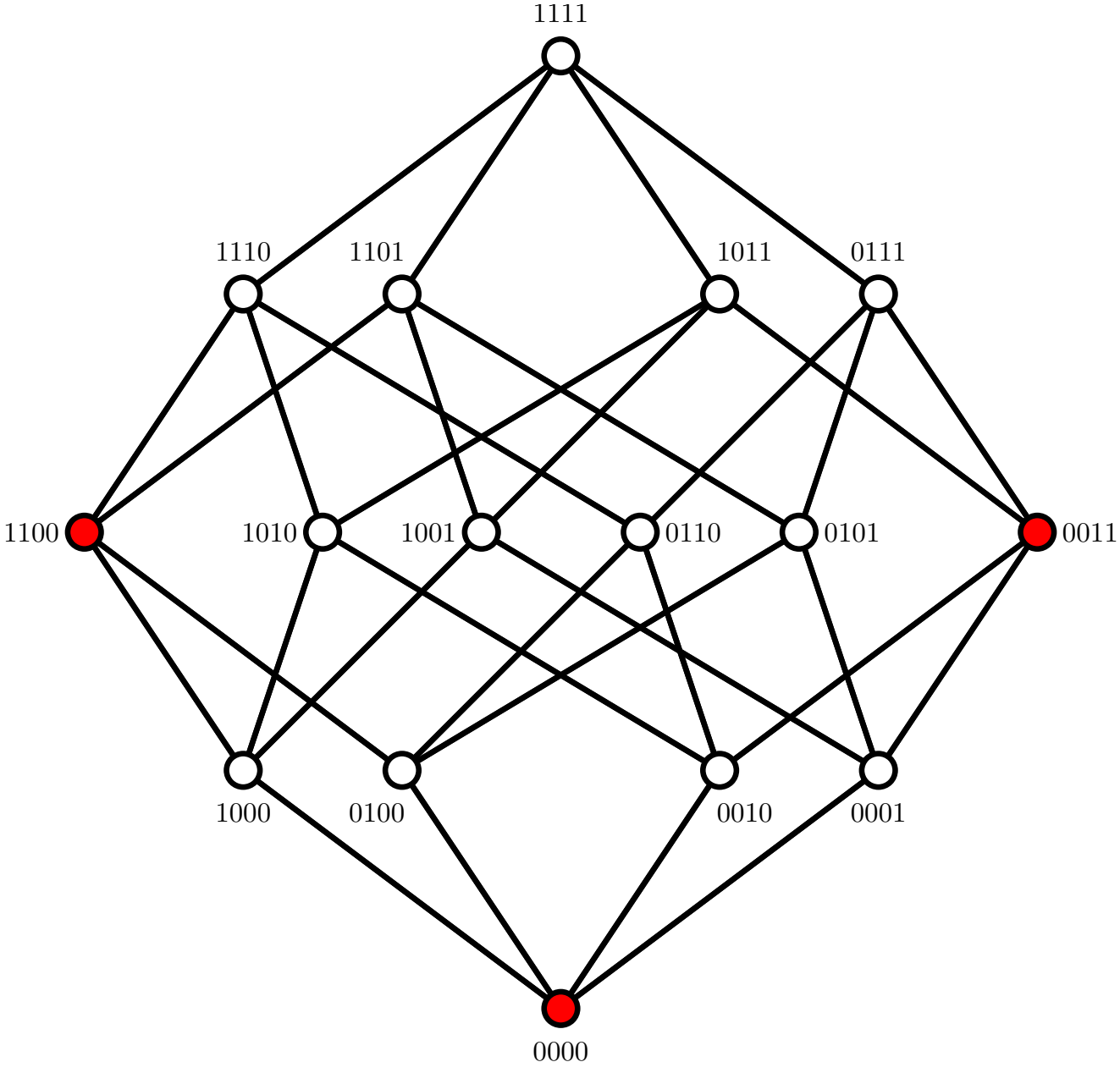
$$C = \{0000, 1100, 1111\}$$
$$C' = \{0000, 1100, 0011\}.$$

Then C and C' are equivalent. In C the unique pair of codewords at distance 4 is 0000 and 1111, and in C' the unique such pair is 1100 and 0011. So any equivalence must send 0000 to either 1100 or 0011.

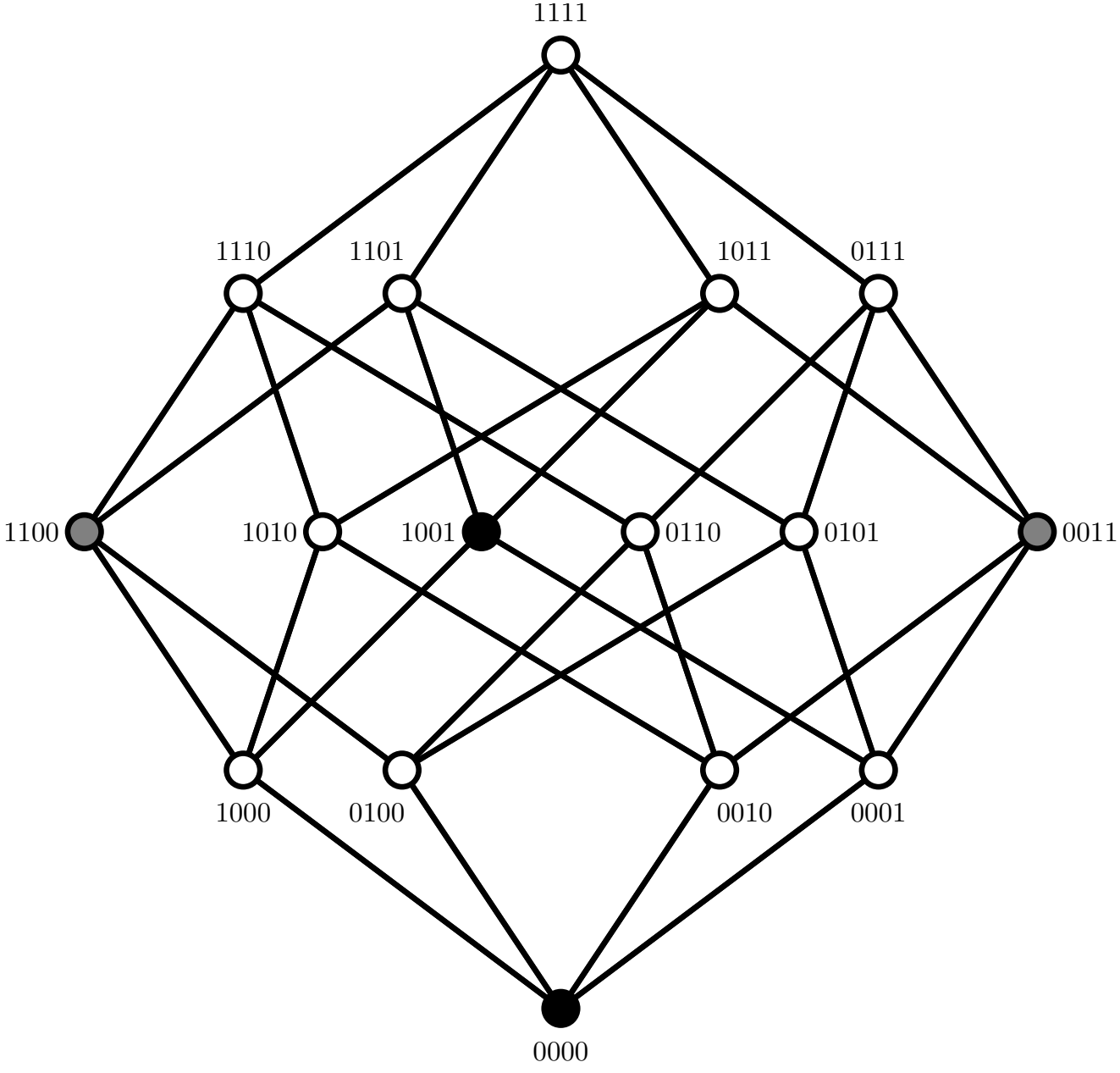
$C = \{0000, 1100, 1111\}$



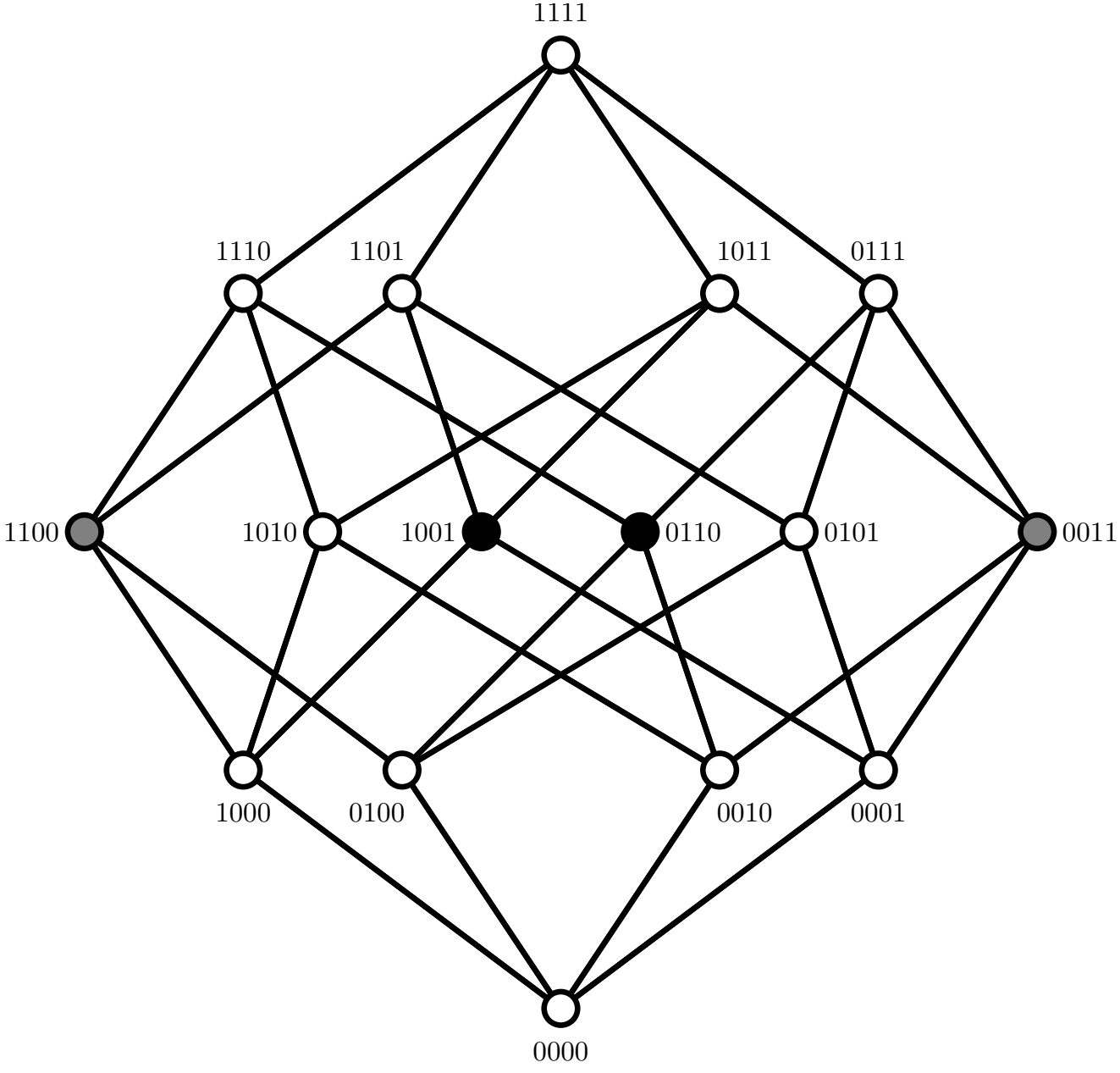
$C' = \{0000, 1100, 0011\}$



Sheet 4, Question 1: $C_4 = \{0000, 1100, 0110, 0011\}$.



Sheet 4, Question 1: $C_5 = \{0110, 1100, 1001, 0011\}$.



§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*.

Example 8.2

Let D be the binary code whose codewords are all binary words of length 4 with an even number of 1s. Let $D^\star$ 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.

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 on Theorem 8.4

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 whenever there is a pair of MOLs of order q we have $A_q(4, 3) = q^2$.

(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 4.3 using the Pigeonhole Principle. The Pigeonhole Principle can be used to prove the general Singleton bound: see Questions 3 and 5 on Sheet 5.

§9 Hadamard Codes and the Plotkin Bound

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.

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

Connection with Coding Theory

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.

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 $\{+, -\}$.

Example of Theorem 9.4

Example 9.5

Let

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

The construction in Theorem 9.4 gives the binary code with codewords

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

Plotkin Bound

Theorem 9.6 (Plotkin bound)

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

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

For example, taking $n = 8$ and $d = 5$ we get

$$A_2(8, 5) \leq 10/(10 - 8) = 5.$$

By Theorem 6.7, $A_2(8, 5) = 4$ so the Plotkin bound comes close to the strongest possible result.

Exercise: Use the Plotkin bound to give an alternative proof of the result of Question 2(b) on Sheet 4, that $A_2(9, 6) = 4$.

Hadamard Codes are as Large as Possible

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

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*

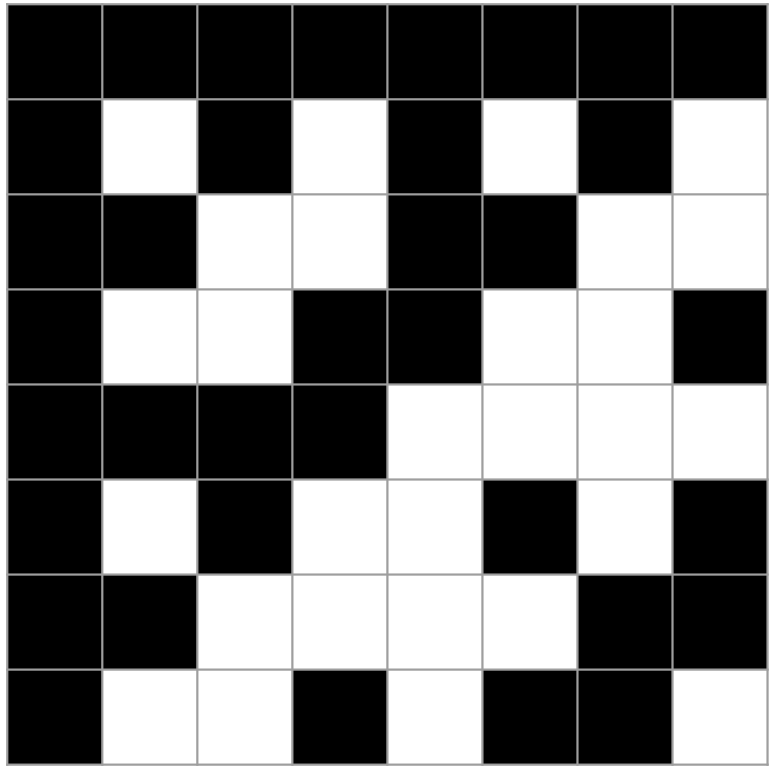
$$A_2(2d, d) = 4d.$$

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: see Question 4 on Sheet 6.

It is a major open problem to show that there are Hadamard matrices of all orders divisible by 4.

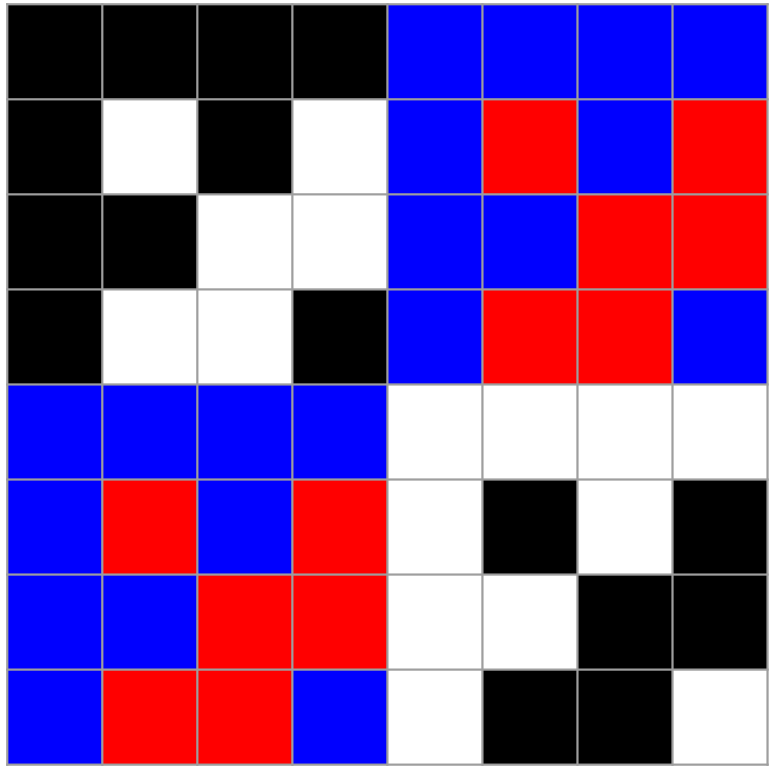
Example of Corollary 9.7

To see how the argument works we take the Hadamard matrix of order 8 shown below. (This comes from the 'doubling' construction on Question 2 Sheet 6.) Recall that black squares show $+1$ and white squares show -1.



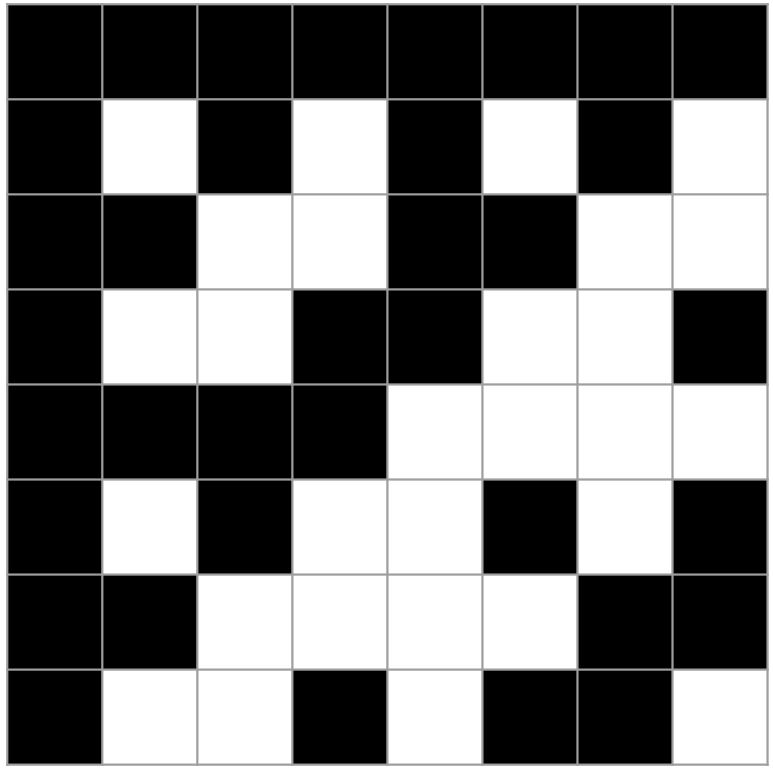
Example of Corollary 9.7

To see how the argument works we take the Hadamard matrix of order 8 shown below. (This comes from the 'doubling' construction on Question 2 Sheet 6.) Recall that black squares show $+1$ and white squares show -1.



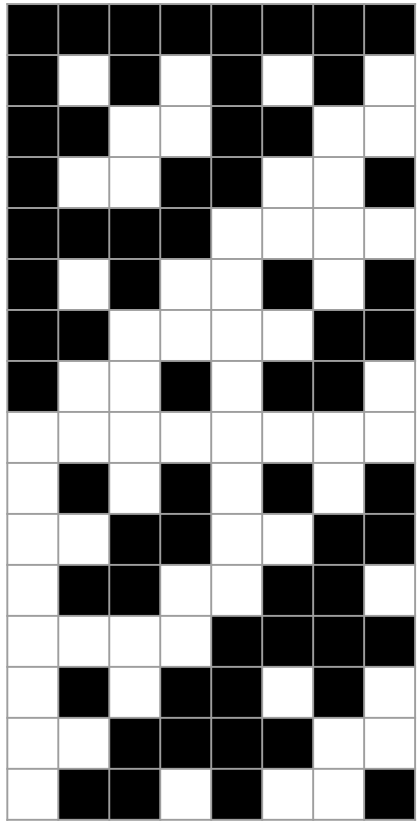
Example of Corollary 9.7

To see how the argument works we take the Hadamard matrix of order 8 shown below. (This comes from the 'doubling' construction on Question 2 Sheet 6.) Recall that black squares show $+1$ and white squares show -1.



Example of Corollary 9.7

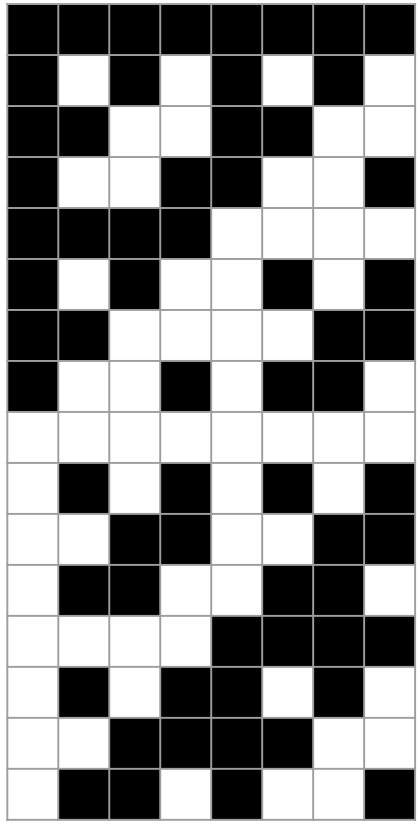
The corresponding code has 16 codewords and minimum distance 4.



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

Example of Corollary 9.7

The corresponding code has 16 codewords and minimum distance 4.

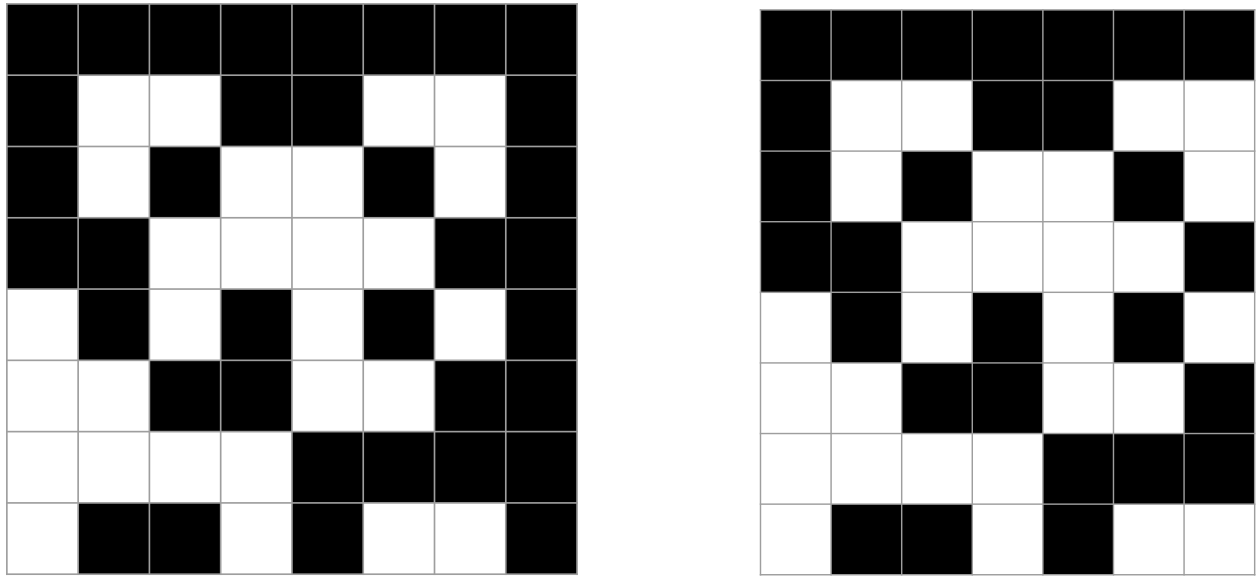


$$\begin{pmatrix}
+ & + & + & + & + & + & + & + \\
+ & - & + & - & + & - & + & - \\
+ & + & - & - & + & + & - & - \\
+ & - & - & + & + & - & - & + \\
+ & + & + & + & - & - & - & - \\
+ & - & + & - & - & + & - & + \\
+ & + & - & - & - & - & + & + \\
+ & - & - & + & - & + & + & - \\
- & - & - & - & - &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

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

Example of Corollary 9.7

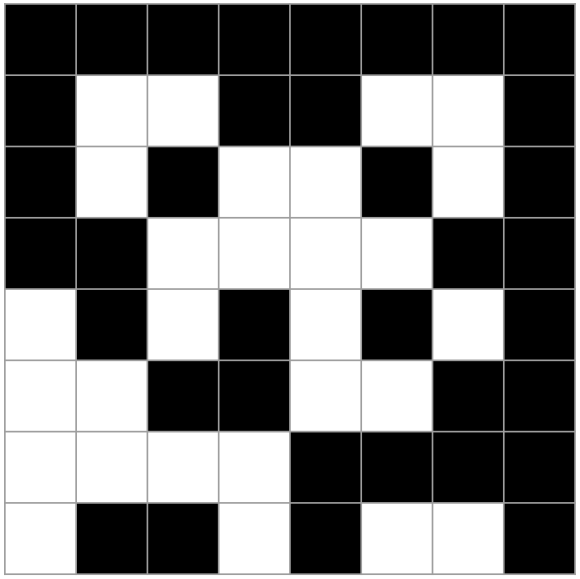
Taking the 8 codewords ending $+1$ (black) gives this code:

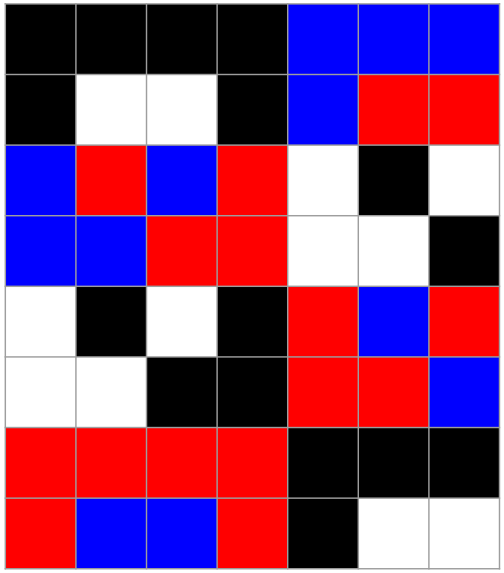


We then puncture to remove the final constant position. In the last step of the proof we applied the Plotkin bound to the resulting code of length 7 and minimum distance 4.

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 applied the Plotkin bound to the resulting code of length 7 and minimum distance 4.

§10 Gilbert–Varshamov Bound and Summary

Recall from Definition 4.2 that the Hamming ball of radius t about a binary word $u \in \{0,1\}^n$

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

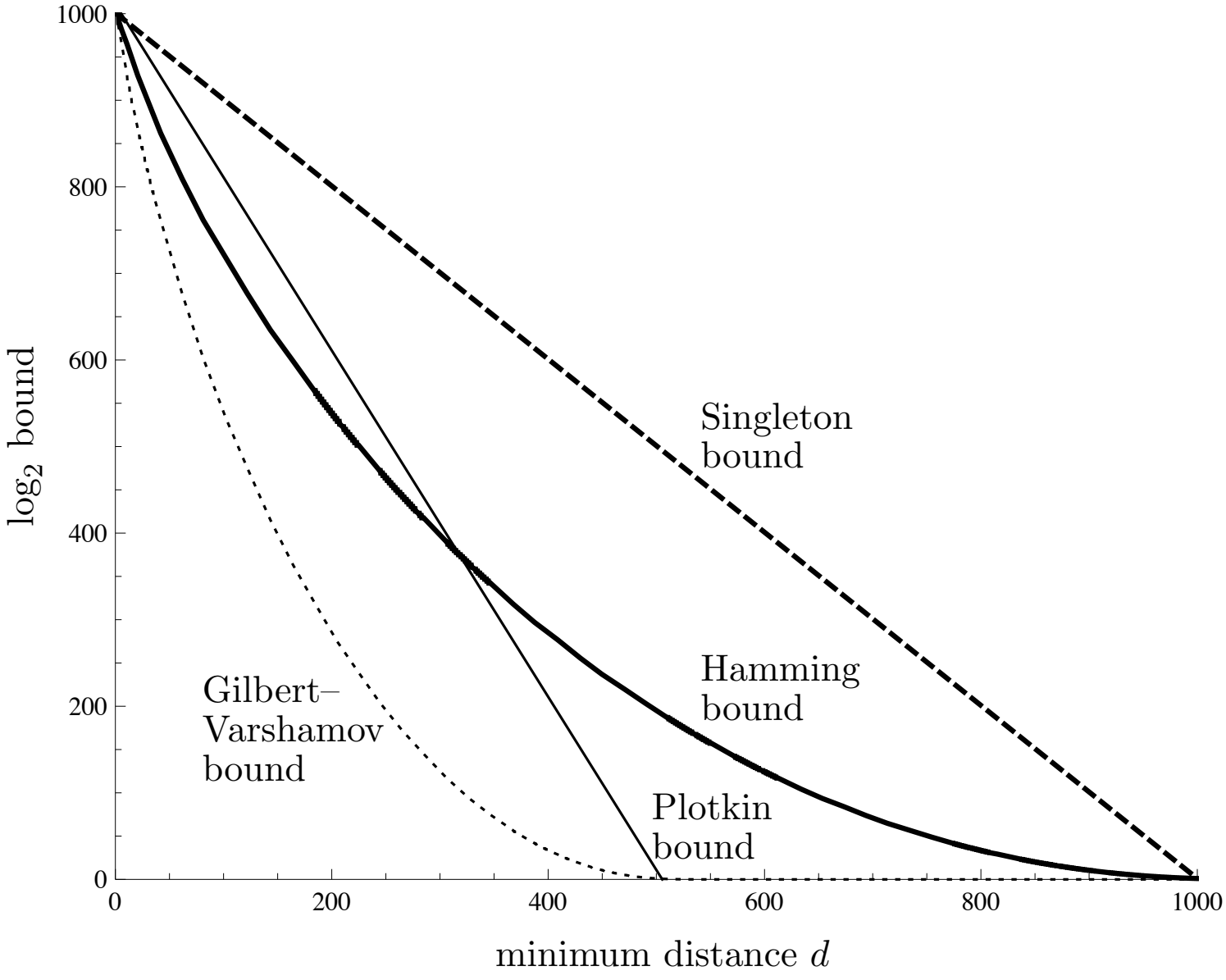
The idea in the next theorem is to construct a large binary code of length n and minimum distance d in the most naïve way possible: we just keep on adding codewords until the Hamming balls of radius $(d-1)$ cover $\{0,1\}^n$, and so every word is distance $\leq (d-1)$ from some codeword.

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 on 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) (**MSc, MSci**) Show that a perfect binary code has odd minimum distance.

Question 1 on Sheet 5

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.

The point of Question 1 on Sheet 5 is to show that given any $(4, q^2, 3)$-code over the alphabet $\{0, 1, \ldots, q-1\}$ we can construct a pair of MOLs, proving '$\Leftarrow$'. (See Example 7.8 for an example.)

Suppose that C is a $(4, q^2, 3)$-code over the alphabet $A = \{0, 1, \ldots, q-1\}$.

(a) Show that if $u = (u_1, u_2, u_3, u_4)$ and $u' = (u'_1, u'_2, u'_3, u'_4) \in C$ are distinct codewords then $(u_1, u_2) \neq (u'_1, u'_2)$.

(b) Deduce that for all $i, j \in A$ there is a unique codeword, say (i, j, X_{ij}, Y_{ij}), whose first two positions are (i, j). [*Hint: C has size q^2.*]

(c) Explain in one sentence why it follows that

$$C = \{(i, j, X_{ij}, Y_{ij}) : i, j \in A\}$$

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

Admin

If you have the sign-up sheet at end of lecture, please give to the lecturer.

Please collect your marked work.

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

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1, \quad 1 + 1 = 0.$$

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

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

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

(2) The binary repetition code of length n is a linear $(n, 2, n)$-code.

(3) 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 2.6, 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 Preliminary Problem Sheet) is linear.

Hamming Distances for Linear Codes

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 **[not 8.6]**) to be the number of positions of u equal to 1. For example, $\mathrm{wt}(11100) = 3$ and $\mathrm{wt}(11011) = 4$.

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 easier to find

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

and then to use 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	$u_3 + u_4$
$u_1 + u_3$	$u_2 + u_4$	

for $u_1, u_2, u_3, u_4 \in \mathbf{Z}_2$. 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 2.6 and Example 11.2(2). (For a related result see Questions 5 and 6 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

Suppose that C be a linear binary (n, M, d)-code. If d is odd then the parity check extension C_{ext} of C is a linear binary $(n+1, M, d+1)$-code.

Example of Theorem 11.6

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

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

and append a final bit to make the weight even.

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

Example of Theorem 11.6

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

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

and append a final bit to make the weight even.

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

The appended bit is equal to the sum of the first 8 bits, so is

Example of Theorem 11.6

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

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

and append a final bit to make the weight even.

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

The appended bit is equal to the sum of the first 8 bits, so is $u_1 + u_2 + u_3 + u_4$. The codewords in C_{ext} can be represented by

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

The minimum distance of C_{ext} is 4.

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?

(a) C only (b) E and F only (c) C and E only (d) E only

§12 Bases, generator matrices and encoding

To proceed any further we need to think of binary words as vectors and linear binary codes as subspaces of $\mathbf{Z}_2^n$.

Definition 12.1

Let C be a linear code. We say that codewords $u(1), \ldots, u(m) \in C$ are:

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

$$x(1)u(1) + \cdots + x(m)u(m) = 0$$

with $x(1), \ldots, x(m) \in \mathbf{Z}_2$ is $x(1) = \ldots = x(m) = 0$.

(b) *span* C if for every $w \in C$ there exist $x(1), \ldots, x(m) \in \mathbf{Z}_2$ such that

$$w = x(1)u(1) + \cdots + x(m)u(m).$$

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

Examples of bases

Example 12.2

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

(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.

Note that a linear binary code may have several different bases. So it is correct to write 'a basis of C' rather than 'the basis of C'.

Exercise: Find another basis for the code C_{ext} in Example 12.2(2).

What Good is a Basis?

Lemma 12.3

Let C be a linear code. The codewords

$$u(1), \ldots, u(m) \in C$$

are a basis for C $\iff$ for all $w \in C$ there exist unique $x(1), \ldots, x(m)$, such that

$$w = x(1)u(1) + \cdots + x(m)u(m).$$

Definition 12.4

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

~~**Correction:** the codes in Example 12.2 are $[5, 1, 3]$ and $[5, 2, 4]$-codes, respectively.~~

Generator Matrices

Definition 12.5

Suppose that C is a linear binary code of length n having $u(1), \ldots, u(m) \in \mathbf{Z}_2^n$ as a basis. The $m \times n$ matrix G with rows $u(1), \ldots, u(m)$ is said to be a *generator matrix* for C.

Example 12.6

(1) Let $C = \{00000, 11100, 00111, 11011\}$. Then a generator matrix for C is

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

Example 12.6 continued

(2) 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

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

(3) The parity check code C_{ext} in Example 12.2(2) has as a generator matrix

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

Encoding for Linear Codes

Let C be a linear code of dimension m with generator matrix G.

We have seen that C has size 2^m, so C can encode 2^m different messages. We will assume that the messages are numbers between 0 and $2^m - 1$.

To encode a message, write its number in binary, say as $b_1 b_2, \ldots b_m$, and encode it as

$$(b_1, b_2, \ldots, b_m)G.$$

If G has rows $u(1), u(2), \ldots, u(m)$ then the message is encoded as

$$b_1 u(1) + b_2 u(2) + \cdots + b_m u(m) \in C.$$

Example of Encoding

Example 12.7

Let C be the linear code in Example 12.6(2). In this example 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 6 we write 6 in binary as 110 and take the codeword

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

In general, the binary number $b_1 b_2 b_3$ is encoded as

$$(b_1, b_2, b_1 + b_2, b_3, b_1 + b_3, b_2 + b_3, b_1 + b_2 + b_3).$$

Quiz on Generator Matrices

As in Example 12.7, let C be the linear binary code with 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 8 codewords, namely

$$\begin{array}{llll} 0000000, & 1010101, & 0110011, & 1100110 \\ 0001111, & 1011010, & 0111100, & 1101001. \end{array}$$

What is the minimum distance of C?

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

How is 3 encoded using G?

(a) 1100110 (b) 0111100 (c) 1011010 (d) 1110000

Suppose you receive 1010010. Decode the message using nearest neighbour decoding.

(a) 3 (b) 5 (c) 6 (d) 7

Quiz on Generator Matrices

As in Example 12.7, let C be the linear binary code with 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 8 codewords, namely

$$\begin{array}{llll} 0000000, & 1010101, & 0110011, & 1100110 \\ 0001111, & 1011010, & 0111100, & 1101001. \end{array}$$

What is the minimum distance of C? Answer (c): 4

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

How is 3 encoded using G? Answer (b): $0110011 + 0001111$

(a) 1100110 (b) 0111100 (c) 1011010 (d) 1110000

Suppose you receive 1010010. Decode the message using nearest neighbour decoding. Answer (b): Nearest to 1011010, encoding 5.

(a) 3 (b) 5 (c) 6 (d) 7

Clarification of Example 12.6(2)

By row operations we reached the matrix

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

Let $u(1), u(2), u(3)$ be the first three rows. I claimed that $u(1)$, $u(2)$, $u(3)$ are linearly independent: i.e. if

$$x(1)(1,0,1,0,1,0,1)+x(2)(0,1,1,0,0,1,1)+x(3)(0,0,0,1,1,1,1) = \mathbf{0}$$

then $x(1) = x(2) = x(3) = 0$. We found that

$$x(1)(1,0,1,0,1,0,1) + x(2)(0,1,1,0,0,1,1) + x(3)(0,0,0,1,1,1,1)$$
$$= (x(1), x(2), ?, x(3), ?, ?, ?)$$

where ? indicates a position not computed in the lecture.

Clarification of Example 12.6(2)

By row operations we reached the matrix

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

Let $u(1), u(2), u(3)$ be the first three rows. I claimed that $u(1)$, $u(2)$, $u(3)$ are linearly independent: i.e. if

$$x(1)(1,0,1,0,1,0,1)+x(2)(0,1,1,0,0,1,1)+x(3)(0,0,0,1,1,1,1) = \mathbf{0}$$

then $x(1) = x(2) = x(3) = 0$. We found that

$$x(1)(1,0,1,0,1,0,1) + x(2)(0,1,1,0,0,1,1) + x(3)(0,0,0,1,1,1,1) = (x(1), x(2), ?, x(3), ?, ?, ?)$$

where ? indicates a position not computed in the lecture. So if the left-hand side is $\mathbf{0} = (0,0,0,0,0,0,0)$ then, **just from this calculation**, we have $x(1) = x(2) = x(3) = 0$.

Clarification of Example 12.6(2)

By row operations we reached the matrix

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

Let $u(1), u(2), u(3)$ be the first three rows. I claimed that $u(1)$, $u(2)$, $u(3)$ are linearly independent: i.e. if

$$x(1)(1,0,1,0,1,0,1)+x(2)(0,1,1,0,0,1,1)+x(3)(0,0,0,1,1,1,1) = \mathbf{0}$$

then $x(1) = x(2) = x(3) = 0$. We found that

$$x(1)(1,0,1,0,1,0,1) + x(2)(0,1,1,0,0,1,1) + x(3)(0,0,0,1,1,1,1)$$
$$= (x(1), x(2), ?, x(3), ?, ?, ?)$$

where ? indicates a position not computed in the lecture. All the positions are shown below:

$$(x(1), x(2), x(1) + x(2), x(3),$$
$$x(1) + x(3), x(2) + x(3), x(1) + x(2) + x(3))$$

Standard Form for Generator Matrices

Definition 12.8

A generator matrix G for a linear binary $[n, m, d]$-code is said to be in *standard* form if

$$G = \begin{pmatrix} I_m & A \end{pmatrix}$$

where I_m is the $m \times m$ identity matrix and A is a $m \times (n - m)$ matrix.

Theorem 12.9

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

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

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

Encoding with Standard Form Generator Matrices

If $G = \begin{pmatrix} I_m & A \end{pmatrix}$ is a generator matrix for a code C in standard form then the message labelled $(b_1, b_2, \ldots, b_m)$ is encoded as

$$(b_1, b_2, \ldots, b_m)G = (b_1, b_2, \ldots, b_m, c_1, \ldots, c_{n-m})$$

for some $c_1, \ldots, c_{n-m} \in \mathbf{Z}_2$. This is convenient because if no errors occur in transmission, then the message can be easily read off from the received word.

Definition 12.10

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

Thus if C is a linear binary $[n, m, d]$-code then C has 2^m codewords and the rate of C is m/n. Roughly put, the rate of a code measures the proportion of bits that give direct information about the encoded message.

Questionnaires

The batch number is 865037.

The additional questions have changed from previous years:

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 Q17: whether you found the speed too fast, too slow, or about right.)

§13 Decoding by standard arrays

In this section we shall see a way to implement nearest neighbour decoding for linear codes.

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 $v = \mathbf{0} + v$ so $v \in C + v$.

Example 13.2

Let C be the linear binary code

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

obtained by puncturing the code in Example 12.2(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 $C + 0000 = C$.

Exercises on Cosets

Exercise: Taking C as in Example 13.2, show that

$$C + 1001 = C + 0100 = \{0100, 1010, 0111, 1001\}.$$

Show that if v is a word in this coset then using nearest neighbour decoding, v is decoded as $v + 0100 \in C$.

It is **very important** to bear in mind that cosets are sets, and that the same coset can be written as $C + v$ for many different words v.

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. **(Minor notation change from notes.)**

Standard arrays

Definition 13.4

Let C be a linear binary code. 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*.

Example 13.5

A standard array for the code C in Example 10.2 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 [**please correct handout**]

$$0001 \quad \mathbf{1111} \quad \mathbf{0010} \quad 1100$$

with 0001 as the coset leader. (Both 0010 and 0001 have weight 1.) The other coset leaders 0000, 1000 and 0100 are uniquely determined by their cosets.

Decoding using a standard array

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 $u \in C$.

If C is e-error correcting and the coset leader x in the **theorem** above has weight $\leq e$ then

$$d(v + x, v) = d(x, 0) = \mathrm{wt}(x) \leq e.$$

So $v + x$ must be the *unique* nearest codeword to v, and decoding using the standard array finds this codeword. This shows that standard arrays give an efficient way to implement nearest neighbour decoding.

§14 Parity check matrices and syndrome decoding

The square code can be defined by

$$S = \left\{ (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8) : \begin{array}{l} u_1 + u_2 = u_5, \;\; u_3 + u_4 = u_6 \\ u_1 + u_3 = u_7, \;\; u_2 + u_4 = u_8 \end{array} \right\}$$

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 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 using nearest neighbour decoding.

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10000001

Give an example of a received word for which all four equations fail.

Quiz on square code

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 using nearest neighbour decoding.

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10000001

Give an example of a received word for which all four equations fail.

Quiz on square code

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 using nearest neighbour decoding.

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10000001

Give an example of a received word for which all four equations fail.

(i) Equations $u_3 + u_4 = u_6$ and $u_1 + u_3 = u_7$ fail. Decoding by flipping bit 3 to get 10101100.

Quiz on square code

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 using nearest neighbour decoding.

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10000001

Give an example of a received word for which all four equations fail.

(i) Equations $u_3 + u_4 = u_6$ and $u_1 + u_3 = u_7$ fail. Decoding by flipping bit 3 to get 10101100.

(ii) Equation $u_1 + u_2 = u_5$ is the only one that fails. Flip bit 5 to get 11000011.

Quiz on square code

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 using nearest neighbour decoding.

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10000001

Give an example of a received word for which all four equations fail.

(i) Equations $u_3 + u_4 = u_6$ and $u_1 + u_3 = u_7$ fail. Decoding by flipping bit 3 to get 10101100.

(ii) Equation $u_1 + u_2 = u_5$ is the only one that fails. Flip bit 5 to get 11000011.

(iii) Equations $u_1 + u_3 = u_7$ and $u_2 + u_4 = u_8$ fail. Nearest neighbour decoding fails, since 11000000 is equidistance from 00000000 and 11000011.

Quiz on square code

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 using nearest neighbour decoding.

(i) 10001100 (ii) 11001011 (iii) 11000000 (iv) 10000001

Give an example of a received word for which all four equations fail.

(i) Equations $u_3 + u_4 = u_6$ and $u_1 + u_3 = u_7$ fail. Decoding by flipping bit 3 to get 10101100.

(ii) Equation $u_1 + u_2 = u_5$ is the only one that fails. Flip bit 5 to get 11000011.

(iii) Equations $u_1 + u_3 = u_7$ and $u_2 + u_4 = u_8$ fail. Nearest neighbour decoding fails, since 11000000 is equidistance from 00000000 and 11000011.

(iv) All equations except $u_3 + u_4 = u_6$ fail. Nearest neighbour decoding fails.

Parity Check Matrices

Definition 14.1

Let C be a linear binary code of length n and dimension m. A *parity check matrix* for C is an $(n-m) \times n$ matrix H with linearly independent rows such that for each $u \in \mathbf{Z}_2^n$ we have

$$u \in C \iff uH^{tr} = \mathbf{0}.$$

Example 14.2

(1) The code C_{ext} defined above 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

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

Parity Check Matrices

Theorem 14.3

Let C be a linear binary code of length n and dimension m. Then C has a parity check matrix. Moreover, if C has a generator matrix G in standard form $G = \begin{pmatrix} I_m & A \end{pmatrix}$ then

$$\begin{pmatrix} A^{tr} & I_{n-m} \end{pmatrix}$$

is a parity check matrix for C.

I claimed that Example 14.6 would show how to reduce to the case where G has generator matrix in standard form. I'd rather not do this today: instead see Question 1 on Sheet 9.

Dual Codes

Definition 14.4

Let C be a linear binary code of length n 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 $n - m$ with generator matrix H.

Example 14.5

Let C_{ext} be as in Example 14.2(1). Then

$$C_{\text{ext}}^{\perp} = \{00000, 11111\}$$

is the binary repetition code of length 5, and

$$\{00000, 11111\}^{\perp} = C_{\text{ext}}.$$

Hamming $[7, 4, 3]$-code

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, so to show C has minimum distance 3, it suffices to show there are no codewords of weight 1 or 2.

Quiz on Linear Codes

Decide which of the following statements are true, and which are false.

(1) There is a linear binary code of size 5.

(2) There is a linear binary code of size 4.

(3) The code with generator matrix $G = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$ has size 2.

(4) The binary code

$$C = \left\{(x_1, x_2, x_3) \in \mathbf{Z}_2^3 : x_1(x_1 + 1) + x_2 + x_3 = 0\right\}$$

is linear.

Common Mistake on Sheet 8

It is not possible to read off the minimum distance (or minimum weight) of a linear binary code from its generator matrix in any easy way.

For example, let $C = \{0000, 1110, 0001, 1111\}$. Then C has as as a basis 1110, 1111, and so a generator matrix for C is

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

The codewords in the generator matrix have weight ≥ 3, but C has minimum distance 1.

Common Mistake on Sheet 8

It is not possible to read off the minimum distance (or minimum weight) of a linear binary code from its generator matrix in any easy way.

For example, let $C = \{0000, 1110, 0001, 1111\}$. Then C has as as a basis 1110, 1111, and so a generator matrix for C is

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

The codewords in the generator matrix have weight ≥ 3, but C has minimum distance 1.

Note also that the generator matrix of a linear binary code is usually not unique. For instance, another generator matrix for C is

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

Syndromes

In standard array decoding one has to hunt through the entire standard array to find the coset of the code in which a received word lies. We end with an improved method that uses parity check matrices.

Theorem 14.7

Let C be a linear binary code of length n and dimension m 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 $\iff$ $vH^{tr} = v'H^{tr}$.

Definition 14.8

Let C be a linear binary code of length n and dimension m with parity check matrix H. The *syndrome* of a word $v \in \mathbf{Z}_2^n$ is defined to be $vH^{tr} \in \mathbf{Z}_2^{n-m}$.

Example 14.9

Let $C = \{0000, 1110, 0011, 1101\}$ be the code used as an example in §13. Then C has parity check matrix

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

By Theorem 14.7, 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 $C + 1000$.

Example 14.9 [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

A defect of the code C is that $C + 0010 = C + 0001$ and so the single bit errors 0010 and 0001 have the same syndrome. If C had to be used in practice, one possibility would be to use *incomplete decoding* and request retransmission whenever a received word has syndrome 01.

Syndrome decoding for Hamming $[7, 4, 3]$-code

Example 14.10

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, so by Lemma 13.3 and Theorem 14.7 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 so we decode v as $v + e(i)$.

A 7 Question Strategy from the Hamming Code

Recall the Liar Game, in which the questioner must discover a secret number by asking questions to someone who is allowed to lie at most once.

The Hamming code gives a questioning strategy *with fixed questions* that is optimal. Question i can be stated more concisely as 'when your number is encoded in the Hamming $[7, 4, 3]$-code, is the ith bit of the codeword equal to 1?'

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 $\{1, 3, 5, 7, 9, 11, 13, 15, 17\}$?