[5.1A.11]

Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong
prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

1

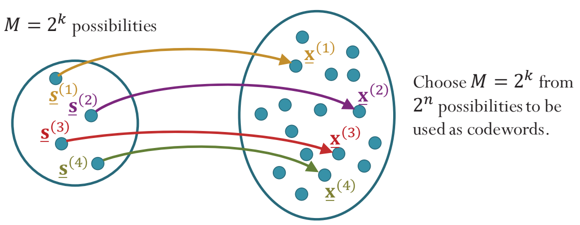
Error Control, Error Detection, Error Correction

2

Two types of **error control**:
1. **error detection**
2. **error correction**

Error Detection

- **Error detection**: the determination of whether errors are present in a received word
  - usually by checking whether the received word is one of the valid codewords.



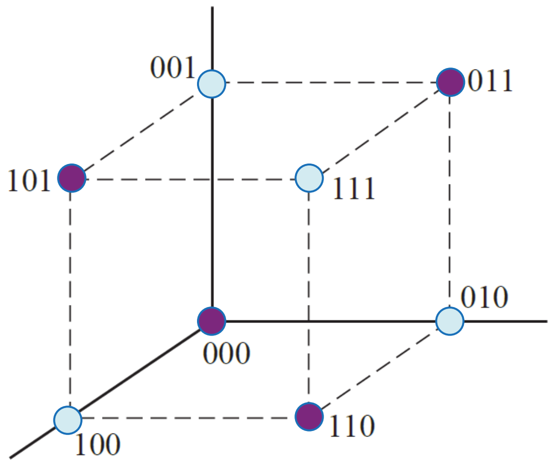
- When a two-way channel exists between source and destination, the receiver can request **retransmission** of information containing detected errors.
  - This error-control strategy is called **automatic-repeat-request (ARQ)**.
- An error pattern is **undetectable** (unnoticed) if and only if it causes the received word to be a valid codeword other than that which was transmitted.
  - Ex: In single-parity-check code, error will be undetectable when the number of bits in error is even.

n k

Example: (3,2) Single-parity-check code

000
011
101
110

- If we receive 001, 111, 010, or 100, we know that something went wrong in the transmission.
- Suppose we transmitted 101 but the error pattern is 110.
  - The received vector is 011
  - 011 is still a valid codeword.
  - The error is undetectable.



Error Correction

- In **FEC** (**forward error correction**) system, when the decoder detects error, the arithmetic or algebraic **structure** of the code is used to determine which of the valid codewords was transmitted.

- It is possible for a detectable error pattern to cause the decoder to select a codeword other than that which was actually transmitted. The decoder is then said to have committed a **decoding error**.

Square array for error correction by parity checking.

$\underline{\mathbf{b}} = [b_1, b_2, \ldots, b_9]$

- The codeword is formed by arranging k message bits in a square array whose rows *and* columns are checked by $2\sqrt{k}$ parity bits.
- A transmission error in one message bit causes a row and column parity failure with the error at the intersection, so single errors can be corrected.

b_1	b_2	b_3	p_1
b_4	b_5	b_6	p_2
b_7	b_8	b_9	p_3
p_4	p_5	p_6	

$\underline{\mathbf{x}} = [b_1, b_2, \ldots, b_9, p_1, p_2, \ldots, p_6]$

[Carlson & Crilly, p 594]

Example: square array

- $k = 9$
- $2\sqrt{9} = 6$ parity bits.

$\underline{\mathbf{b}} = [b_1, b_2, \ldots, b_9]$

$= 101110100$

$\underline{\mathbf{x}} = [b_1, b_2, \ldots, b_9, p_1, p_2, \ldots, p_6]$

$= 101110100\,001111$

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

b_1	b_2	b_3	p_1
b_4	b_5	b_6	p_2
b_7	b_8	b_9	p_3
p_4	p_5	p_6	

$\underline{y} = 10\overset{1}{\cancel{0}}110100001111$

1	0	0	0	× parity failure
1	1	0	0	✓
1	0	0	1	✓
1	1	1		
✓	✓	×		

[Carlson & Crilly, p 594]

ECS 452: In-Class Exercise # 16

Instructions

1. Working alone is always permitted. However, working in groups is also allowed if social distancing can be used (via, e.g., online group chat/call). For group work, **the group cannot be the same as any of your former group after the midterm.**
2. Only one submission is needed for each group.
3. [**ENRE**] Explanation is not required for this exercise.
4. **Do not panic.**

Date: 31 / 3 / 2020			
Name	ID (last 3 digits)		

1. Consider a linear block code that uses *parity checking on a square array*:

First, we use the provided definition to write down the equations that produce the parity bits. This definition is exactly the same as the one given in lecture when we defined parity checking on a square array

b_1	b_3	p_1
b_2	b_4	p_2
p_3	p_4	

$p_1 = b_1 \oplus b_3$

$p_2 = b_2 \oplus b_4$

$p_3 = b_1 \oplus b_2$

$p_4 = b_3 \oplus b_4$

Each parity bit p_i is calculated such that the corresponding row or column has even parity.

Suppose the following bits arrangement is used in the codeword:

$$\underline{\mathbf{x}} = \begin{pmatrix} b_1 & b_2 & p_1 & p_2 & b_3 & p_3 & b_4 & p_4 \end{pmatrix}.$$

a. Find the generator matrix $\mathbf{G}$.

$p_1 = b_1 \oplus b_3$ $p_2 = b_2 \oplus b_4$ $p_3 = b_1 \oplus b_2$ $p_4 = b_3 \oplus b_4$

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

Recall that the 1s and 0s in the j^{th} column of **G** tells which positions of the data bits are combined ($\oplus$) to produce the j^{th} bit in the codeword.

b. Find the codeword for the message $\underline{\mathbf{b}} = \begin{bmatrix} \overset{b_1}{1} & \overset{b_2}{0} & \overset{b_3}{1} & \overset{b_4}{0} \end{bmatrix}$.

Method 1: First, we fill out the array above with the message. Then, we calculate the parity bits.

1	1	p_1
0	0	p_2
p_3	p_4	

1	1	0
0	0	0
1	1	

The codeword can be read directly from the array: $\underline{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$.

Method 2: It is still true that $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G}$. Therefore, we can still use our old technique: to find $\underline{\mathbf{x}}$ when $\underline{\mathbf{b}} = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$, we simply need to add the first and the third rows of $\mathbf{G}$. This also gives $\underline{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$.

Review: Even Parity

- A binary vector (or a collection of 1s and 0s) has **even parity** if and only if the number of 1s in there is even.
  - Suppose we are given the values of all the bits except one bit.
    - We can force the vector to have even parity by setting the value of the remaining bit to be the sum of the other bits.

Single-parity-check code

[1 0 1 1 0 _]

Square array

1	0	1	_
0	1	1	_
0	0	1	_
_	_	_	

Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Introduction to Minimum Distance

Minimum Distance (d_{min})

The **minimum distance (d_{min})** of a block code is the minimum Hamming distance between all pairs of distinct codewords.

Suppose $|C| = M$. Then, need to calculate $\binom{M}{2} = \binom{2^k}{2}$ distance values

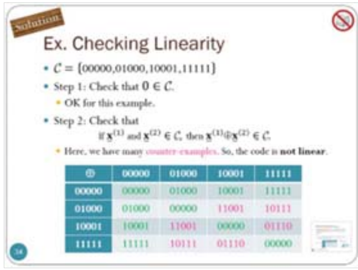
- Ex.:

  A channel encoder map blocks of two bits to five-bit (channel) codewords. The four possible codewords are 00000, 01000, 10001, and 11111. A codeword is transmitted over the BSC with crossover probability $p = 0.1$.

  (a) What is the minimum (Hamming) distance d_{min} among the codewords?

$d_{min} = 1$

d	00000	01000	10001	11111
00000		1	2	5
01000			3	4
10001				3
11111				



- Ex. Repetition code: 000⋯0, 111⋯1 $\quad d_{min} = n$

MATLAB: Distance Matrix and d_{min}

```
function D = distAll(C)

M = size(C,1);
D = zeros(M,M);
for i = 1:M-1
    for j = (i+1):M
        D(i,j) = sum(mod(C(i,:)+C(j,:),2));
    end
end
D = D+D';
```

This can be used to find d_{min} for all block codes. There is no assumption about linearity of the code. Soon, we will see that we can simplify the calculation when the code is known to be linear.

```
function dmin = dmin_block(C)
D = distAll(C);
Dn0 = D(D>0);
dmin = min(Dn0);
```

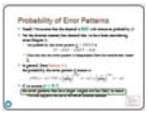
```
>> C=[0 0 0 0 0; 0 1 0 0 0; ...
      1 0 0 0 1; 1 1 1 1 1];
>> distAll(C)
ans =
     0     1     2     5
     1     0     3     4
     2     3     0     3
     5     4     3     0

>> dmin = dmin_block(C)
dmin =
     1
```

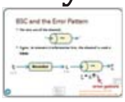
Weight and Distance

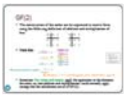
- The **weight** of a vector is the number of nonzero coordinates in the vector.
  - The weight of a vector $\underline{\mathbf{x}}$ is commonly written as $w(\underline{\mathbf{x}})$.
  - Ex. $w(010111) = 4$
  - For BSC with cross-over probability $p < 0.5$, error pattern with smaller weights (less #1s) are more likely to occur.
- The **Hamming distance** between two n-bit blocks is the number of coordinates in which the two blocks differ.
  - Ex. $d(010111, 011011) = 2$

    010111
    011011

    $d(\underline{x}, \underline{y}) = w(\underline{x} \oplus \underline{y})$
  - Note:
    - The Hamming distance between any two vectors equals the weight of their sum.
    - The Hamming distance between the transmitted codeword $\underline{\mathbf{x}}$ and the received vector $\underline{\mathbf{y}}$ is the same as the weight of the corresponding error pattern $\underline{\mathbf{e}}$.







$d_{\min}$ for linear block code

- For any linear block code, the **minimum distance** ($d_{\min}$) can be found from the minimum weight of its nonzero codewords.
  - So, instead of checking $\binom{2^k}{2}$ pairs, simply check the weight of the 2^k codewords.

```
function dmin = dmin_linear(C)
w = sum(C,2);
w = w([w>0]);
dmin = min(w);
```

Proof

Because the code is linear, for any two distinct codewords $\underline{\mathbf{c}}^{(1)}$ and $\underline{\mathbf{c}}^{(2)}$, we know that $\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)} \in \mathcal{C}$; that is $\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)} = \underline{\mathbf{c}}$ for some nonzero $\underline{\mathbf{c}} \in \mathcal{C}$. Therefore,

$$d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) = w\left(\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)}\right) = w\left(\underline{\mathbf{c}}\right) \text{ for some nonzero } \underline{\mathbf{c}} \in \mathcal{C}.$$

This implies

$$\min_{\substack{\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)} \in \mathcal{C} \\ \underline{\mathbf{c}}^{(1)} \neq \underline{\mathbf{c}}^{(2)}}} d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) \geq \min_{\substack{\underline{\mathbf{c}} \in \mathcal{C} \\ \underline{\mathbf{c}} \neq \underline{\mathbf{0}}}} w\left(\underline{\mathbf{c}}\right).$$

Note that inequality is used here because we did not show that $\underline{\mathbf{c}}^{(1)} \oplus \underline{\mathbf{c}}^{(2)}$ can produce all possible nonzero $\underline{\mathbf{c}} \in \mathcal{C}$.

$$\min_{\substack{\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)} \in \mathcal{C} \\ \underline{\mathbf{c}}^{(1)} \neq \underline{\mathbf{c}}^{(2)}}} d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) = \min_{\substack{\underline{\mathbf{c}} \in \mathcal{C} \\ \underline{\mathbf{c}} \neq \underline{\mathbf{0}}}} w\left(\underline{\mathbf{c}}\right)$$

Next, for any nonzero $\underline{\mathbf{c}} \in \mathcal{C}$, note that

$$d\left(\underline{\mathbf{c}}, \underline{\mathbf{0}}\right) = w\left(\underline{\mathbf{c}} \oplus \underline{\mathbf{0}}\right) = w\left(\underline{\mathbf{c}}\right).$$

Note that $\underline{\mathbf{c}}, \underline{\mathbf{0}}$ is just one possible pair of two distinct codewords. This implies

$$\min_{\substack{\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)} \in \mathcal{C} \\ \underline{\mathbf{c}}^{(1)} \neq \underline{\mathbf{c}}^{(2)}}} d\left(\underline{\mathbf{c}}^{(1)}, \underline{\mathbf{c}}^{(2)}\right) \leq \min_{\substack{\underline{\mathbf{c}} \in \mathcal{C} \\ \underline{\mathbf{c}} \neq \underline{\mathbf{0}}}} w\left(\underline{\mathbf{c}}\right).$$

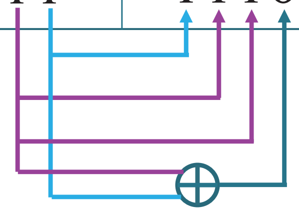
Example

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

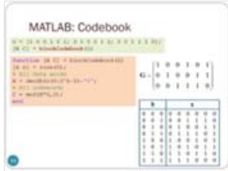
$$\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} b_2 & b_1 & b_1 & b_1 \oplus b_2 \end{bmatrix}$$

$\underline{\mathbf{b}}$	$\underline{\mathbf{x}}$	$w(\underline{x})$
00	0000	~~0~~
01	1001	2
10	0111	3
11	1110	3



$d_{min} = 2$

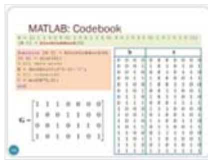


Example

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

$\underline{\mathbf{b}}$			$\underline{\mathbf{x}}$						$w(\underline{\mathbf{x}})$
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	
0	1	0	0	1	0	0	1	1	
0	1	1	0	1	1	1	0	1	
1	0	0	1	0	0	1	0	1	
1	0	1	1	0	1	0	1	1	4
1	1	0	1	1	0	1	1	0	4
1	1	1	1	1	1	0	0	0	3

```
>> G = [1 0 0 1 0 1; 0 1 0 0 1 1; 0 0 1 1 1 0];
>> [B C] = blockCodebook(G);
>> dmin =  dmin_block(C)
dmin =
     3
>> dmin =  dmin_linear(C)
dmin =
     3
```

Example

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

```
>> G = [1 1 1 0 0 0 0; 1 0 0 1 1 0 0;...
        0 0 1 0 1 1 0; 1 0 1 0 1 0 1];
>> [B C] = blockCodebook(G);
>> dmin =  dmin_linear(C)
dmin =
     3
>> dmin =  dmin_block(C)
dmin =
     3
```

$\underline{\mathbf{b}}$				$\underline{\mathbf{x}}$							$w(\underline{\mathbf{x}})$
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	1	4
0	0	1	0	0	0	1	0	1	1	0	3
0	0	1	1	1	0	0	0	0	1	1	3
0	1	0	0	1	0	0	1	1	0	0	3
0	1	0	1	0	0	1	1	0	0	1	3
0	1	1	0	1	0	1	1	0	1	0	4
0	1	1	1	0	0	0	1	1	1	1	4
1	0	0	0	1	1	1	0	0	0	0	3
1	0	0	1	0	1	0	0	1	0	1	3
1	0	1	0	1	1	0	0	1	1	0	4
1	0	1	1	0	1	1	0	0	1	1	4
1	1	0	0	0	1	1	1	1	0	0	4
1	1	0	1	1	1	0	1	0	0	1	4
1	1	1	0	0	1	0	1	0	1	0	3
1	1	1	1	1	1	1	1	1	1	1	7

Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Probability of Error Patterns and Minimum Distance Decoder

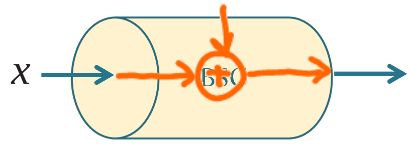
Probability of Error Patterns

- Recall: We assume that the channel is **BSC** with crossover probability $\boldsymbol{p}$.
- For the discrete memoryless channel that we have been considering since Chapter 3,
  - the probability that error pattern $\underline{\mathbf{e}} = 00101$ is
    $$(1-p)(1-p)p(1-p)p.$$
  - Note also that the error pattern is independent from the transmitted vector $\underline{\mathbf{x}}$
- In general, from Section 3.4,
  the probability the error pattern $\underline{\mathbf{e}}$ occurs is
  $$p^{d(\underline{\mathbf{x}},\underline{\mathbf{y}})}(1-p)^{n-d(\underline{\mathbf{x}},\underline{\mathbf{y}})} = \left(\frac{p}{1-p}\right)^{d(\underline{\mathbf{x}},\underline{\mathbf{y}})}(1-p)^n = \left(\frac{p}{1-p}\right)^{w(\underline{\mathbf{e}})}(1-p)^n$$
- If we assume $\boldsymbol{p < 0.5}$,
  the error patterns that have larger weights are less likely to occur.
  - This also supports the use of minimum distance decoder.

BSC and the Error Pattern

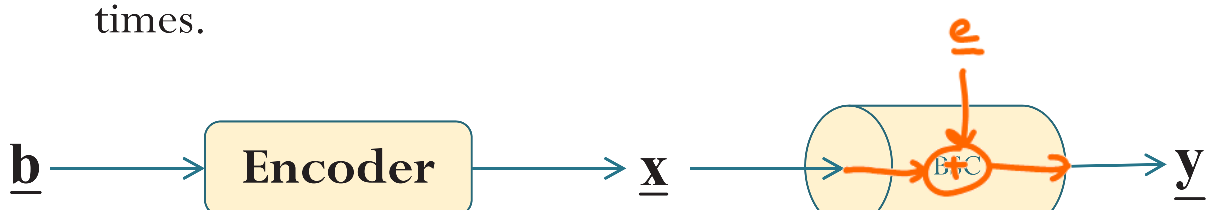
Add e=0: output is the same as input
Add e=1: output is different from input

- For one use of the channel, e



$y = x \oplus e$

- Again, to transmit k information bits, the channel is used n times.



$1 \times k$ $\qquad$ $1 \times n$

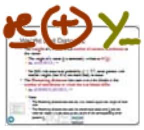
$$\underline{\mathbf{y}} = \underline{\mathbf{x}} \oplus \underline{\mathbf{e}}$$

Ex. $\underline{x} = 01010$
$\oplus$
$\underline{y} = 01111$

$\underline{e} = 00101$

$\underline{y} = \underline{x} \oplus \underline{e}$

$\underline{x} \oplus \underline{y} = \underbrace{\underline{x} \oplus \underline{x}}_{0} \oplus \underline{e} = \underline{e}$



error pattern

Its nonzero elements mark the positions of transmission error in $\underline{\mathbf{y}}$

GF(2)

- The construction of the codes can be expressed in matrix form using the following definition of addition and multiplication of bits:

$\oplus$	0	1
0	0	1
1	1	0

$\bullet$	0	1
0	0	0
1	0	1

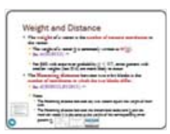
Checking for difference:
$x \oplus y = 1$ iff $x \neq y$
$x \oplus y = 0$ iff $x = y$

- Note that

$x \oplus 0 = x$ $\qquad$ $0 \oplus 0 = 0$, $1 \oplus 0 = 1$

$x \oplus 1 = \bar{x}$ $\qquad$ $0 \oplus 1 = 1$, $1 \oplus 1 = 0$

$x \oplus x = 0$ $\qquad$ $0 \oplus 0 = 0$, $1 \oplus 1 = 0$



The property above implies $\underbrace{-x}_{} = x$ ⇒ "subtraction = addition"

By definition, "$-x$" is something that, when added with x, gives 0.

- Extension: For vector and matrix, apply the operations to the elements the same way that addition and multiplication would normally apply (except that the calculations are all in GF(2)).

Digital Communication Systems EES 452

Asst. Prof. Dr. Prapun Suksompong
prapun@siit.tu.ac.th

5.1 Binary Linear Block Codes

Properties of $d_{\min}$

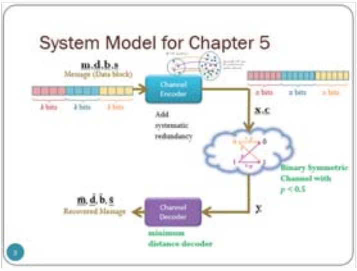
$d_{\min}$: two important facts

- For any linear block code, the **minimum distance** ($d_{\min}$) can be found from the minimum weight of its nonzero codewords.
  - So, instead of checking $\binom{2^k}{2}$ pairs, simply check the weight of the 2^k codewords.

- A code with minimum distance $d_{\min}$ can
  - detect all error patterns of weight $w \le d_{\min}-1$.
  - correct all error patterns of weight $w \le \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$.

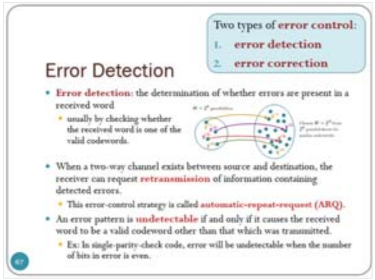
the floor function

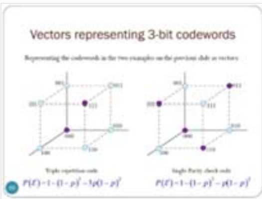
Visual Interpretation of $d_{\min}$

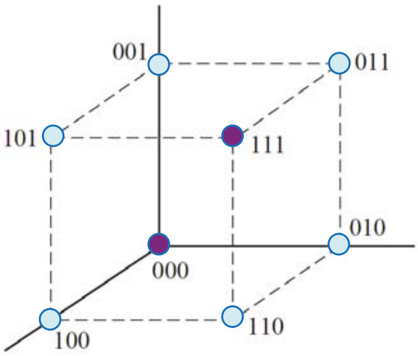


Recall: Codebook construction

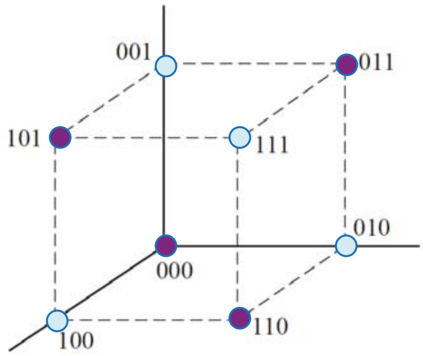
Choose $M = 2^k$ from 2^n possibilities to be used as codewords.





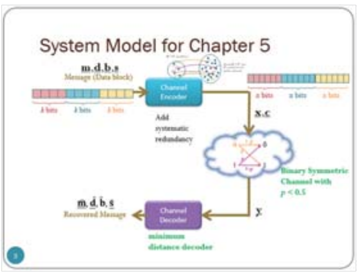


Triple-repetition code



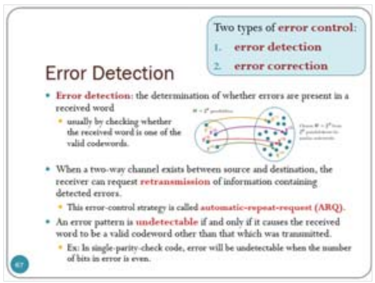
Single-Parity-check code

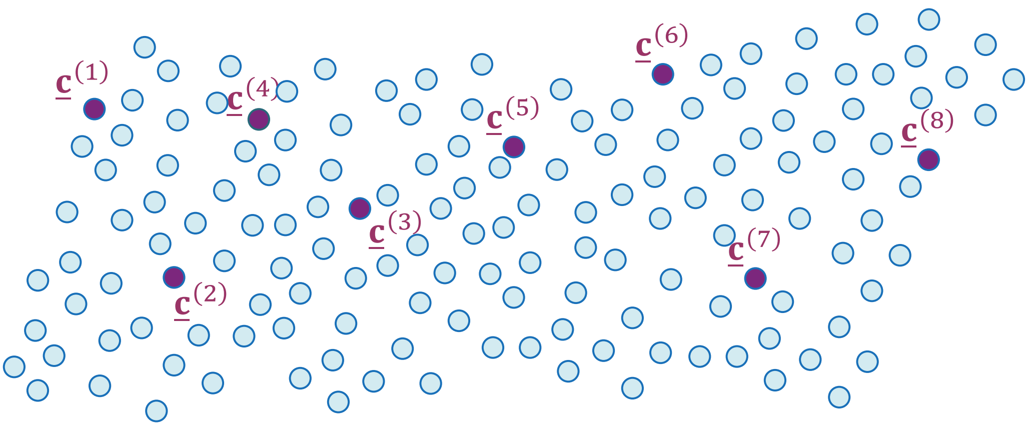
Visual Interpretation of $d_{\min}$



Recall: Codebook construction

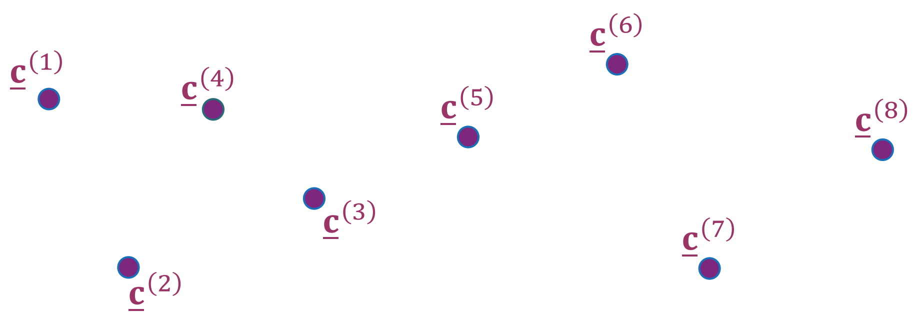
Choose $M = 2^k$ from 2^n possibilities to be used as codewords.





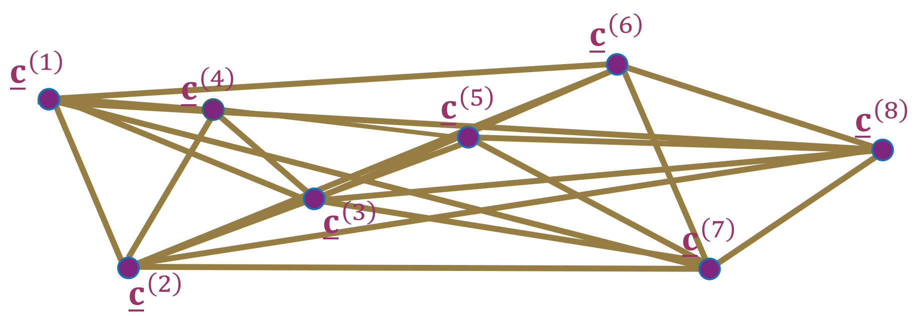
Visual Interpretation of $d_{\min}$

- Consider all the (valid) codewords (in the codebook).



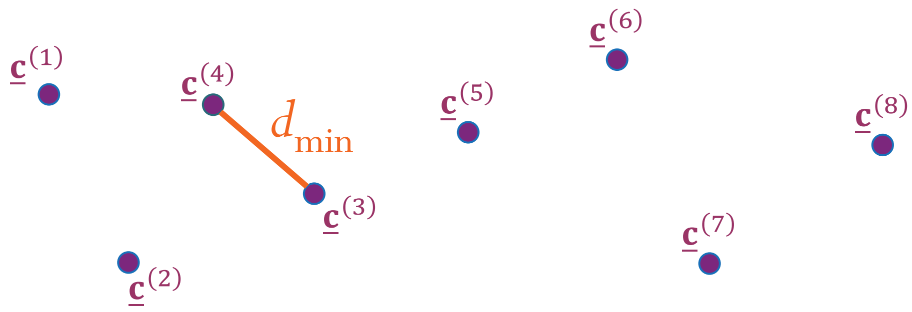
Visual Interpretation of $d_{\min}$

- Consider all the (valid) codewords (in the codebook).
- We can find the distances between them.



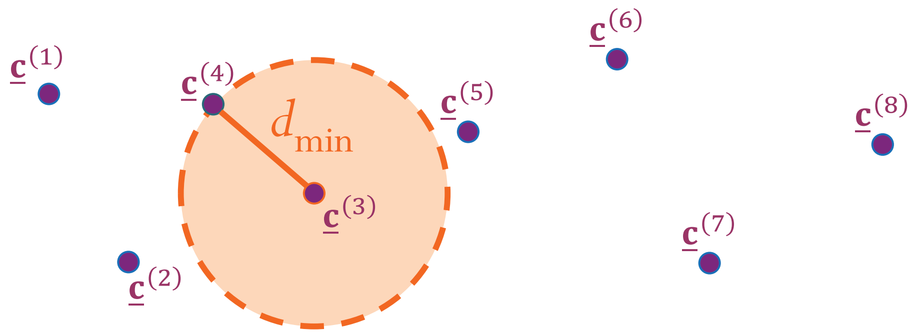
Visual Interpretation of $d_{\min}$

- Consider all the (valid) codewords (in the codebook).
- We can find the distances between them.
- We can then find $d_{\min}$.



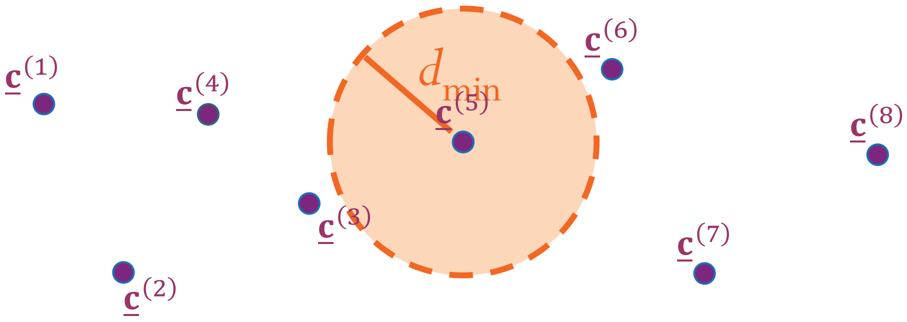
Visual Interpretation of $d_{\min}$

- When we draw a circle (sphere, hypersphere) of radius $d_{\min}$ around any codeword, we know that there can not be another codeword inside this circle.
- The closest codeword is at least $d_{\min}$ away.



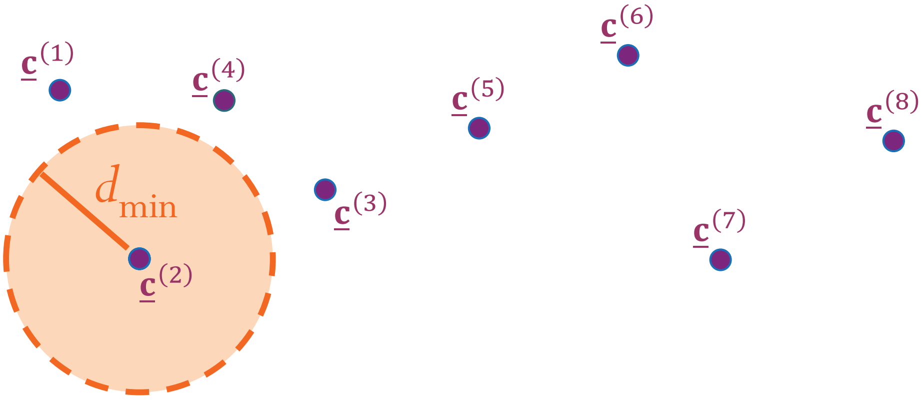
Visual Interpretation of d_{min}

- When we draw a circle (sphere, hypersphere) of radius d_{min} around any codeword, we know that there can not be another codeword inside this circle.
- The closest codeword is at least d_{min} away.



Visual Interpretation of d_{min}

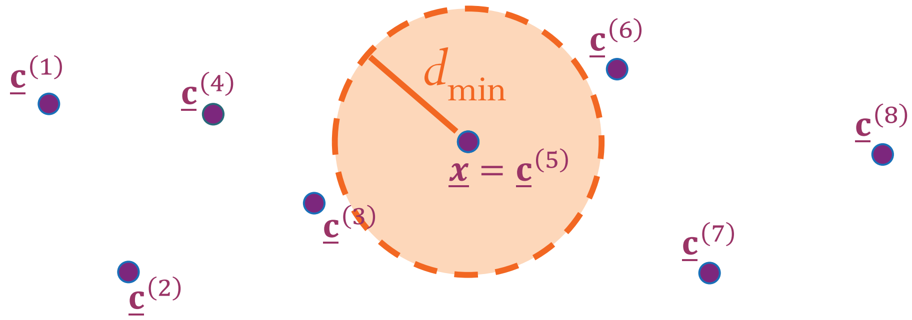
- When we draw a circle (sphere, hypersphere) of radius d_{min} around any codeword, we know that there can not be another codeword inside this circle.
- The closest codeword is at least d_{min} away.



$d_{\min}$ and Error Detection

- Suppose codeword $\underline{\mathbf{c}}^{(5)}$ is chosen to be transmitted; that is

$$\underline{\boldsymbol{x}} = \underline{\mathbf{c}}^{(5)}.$$



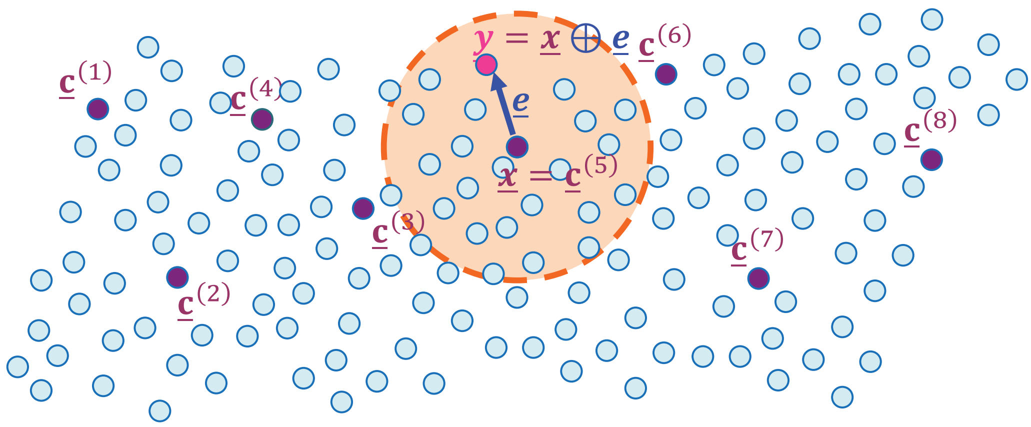
$d_{\min}$ and Error Detection

- Suppose codeword $\underline{\mathbf{c}}^{(5)}$ is chosen to be transmitted; that is

$$\underline{\boldsymbol{x}} = \underline{\mathbf{c}}^{(5)}.$$

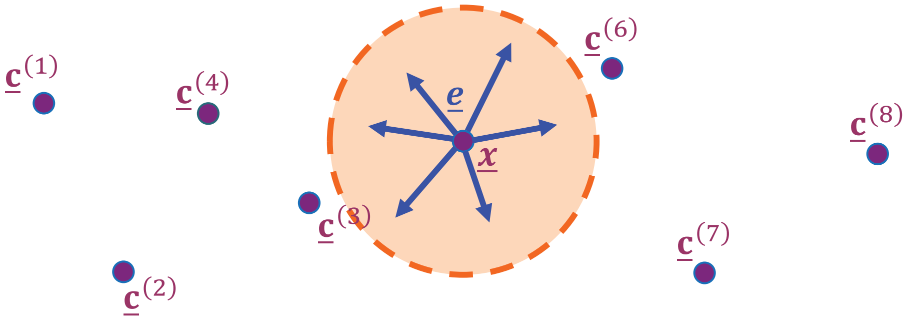
- The received vector $\underline{\boldsymbol{y}}$ can be calculated from

$$\underline{\boldsymbol{y}} = \underline{\boldsymbol{x}} \oplus \underline{\boldsymbol{e}}.$$



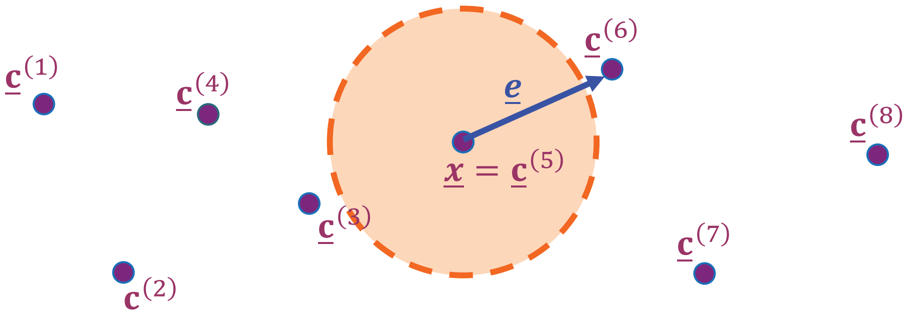
d_{min} and Error Detection

- When $d_{\text{min}} > w$, there is no way that w errors can change a valid codeword into another valid codeword.



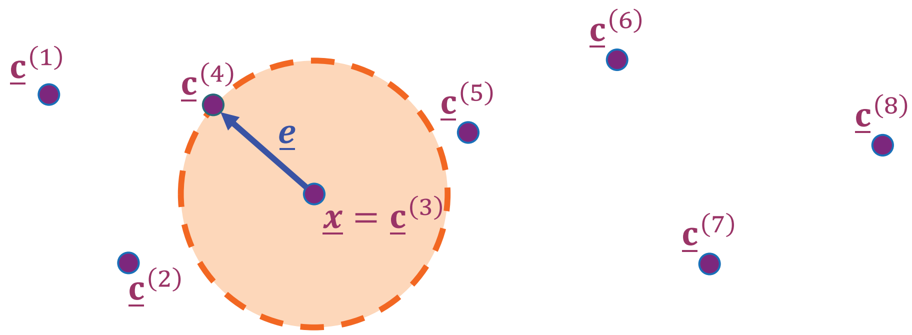
d_{min} and Error Detection

- When $d_{\text{min}} < w$, it is possible that w errors can change a valid codeword into another valid codeword.



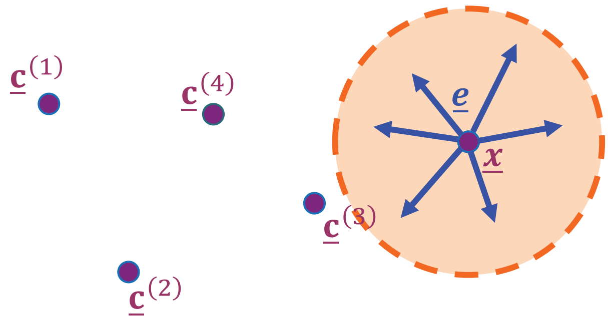
d_{min} and Error Detection

- For some codewords,
  when $d_{\text{min}} = w$, it is possible that w errors can change a valid codeword into another valid codeword.



d_{min} and Error Detection

- To be able to **detect** *all* w-bit errors, we need $d_{\text{min}} \geq w + 1$.
  - With such a code there is no way that w errors can change a valid codeword into another valid codeword.
  - When the receiver observes an illegal codeword, it can tell that a transmission error has occurred.

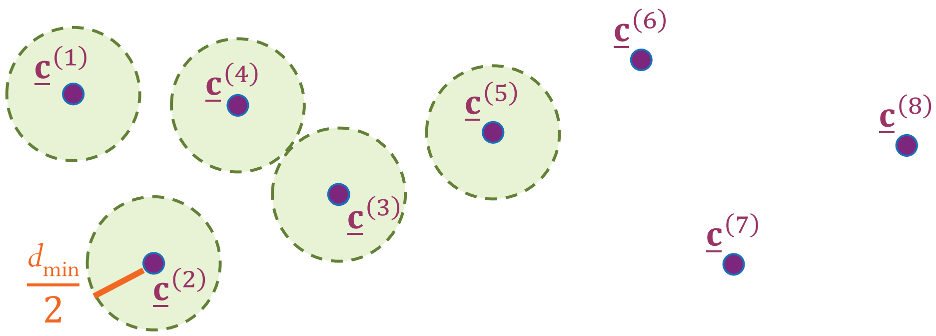


When $d_{\text{min}} > w$, there is no way that w errors can change a valid codeword into another valid codeword.

When $d_{\text{min}} \leq w$, it is possible that w errors can change a valid codeword into another valid codeword.

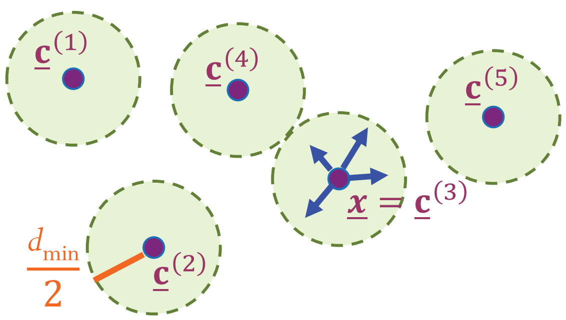
$d_{\min}$ and Error Correction

- To be able to **correct** *all* w-bit errors, we need $d_{\min} \geq 2w + 1$.
  - This way, the legal codewords are so far apart that even with w changes, the original codeword is still ***closer*** than any other codeword.



$d_{\min}$ is an important quantity

- To be able to **correct** *all* w-bit errors, we need $d_{\min} \geq 2w + 1$.
  - This way, the legal codewords are so far apart that even with w changes, the original codeword is still ***closer*** than any other codeword.



$= \min_{i \neq j} d(\underline{x}^{(i)}, \underline{x}^{(j)})$

$d_{\min}$: two important facts

- For any linear block code, the **minimum distance** ($d_{\min}$) can be found from the minimum weight of its nonzero codewords. $d_{\min} = \min_{\underline{x} \neq \underline{0}} w(\underline{x})$
  linear
  - So, instead of checking $\binom{2^k}{2}$ pairs, simply check the weight of the 2^k codewords.

True for nonlinear codes as well

- A code with minimum distance $d_{\min}$ can
  - detect all error patterns of weight w $\leq d_{\min}$-1. $w(\underline{e})$
  - correct all error patterns of weight w $\leq \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$.

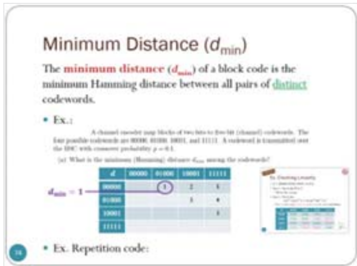
using min dist. dec. (same as min error weight)

the floor function

Usually denoted by t

Error correction capability of a code

Example



Repetition code with $n = 5$
- We have seen that it has $d_{\min} = 5$. $= n =$
- It can detect (at most) <u>4</u> errors.

$d_{\min} - 1 = 5 - 1 = 4$

- It can correct (at most) <u>2</u> errors.

$t = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor = \left\lfloor \frac{5-1}{2} \right\rfloor = \left\lfloor \frac{4}{2} \right\rfloor = \lfloor 2 \rfloor = 2$

Example

Consider the code

$\mathcal{C} \in \{0000000000,\ 0000011111,\ 1111100000, \text{ and } 1111111111\}$

- Is it a linear code?

① $\underline{0} \in \mathcal{C}$ ✓

② $\underline{c}^{(i)} \oplus \underline{c}^{(j)} \in \mathcal{C}$, $i \neq j$ $(\neq \underline{0})$

	$\oplus$	$\underline{c}^{(1)}$	$\underline{c}^{(2)}$	$\underline{c}^{(3)}$	$\underline{c}^{(4)}$
0000000000	$\underline{c}^{(1)}$				
0000011111	$\underline{c}^{(2)}$				
1111100000	$\underline{c}^{(3)}$				
1111111111	$\underline{c}^{(4)}$				

- $d_{\min} =$

- It can detect (at most) ____ errors.
- It can correct (at most) ____ errors.

Example

Consider the code

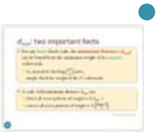
$\mathcal{C} \in \{0000000000,\ \underbrace{0000011111}^{w=5},\ \underbrace{1111100000}^{w=5}, \text{ and } \underbrace{1111111111}^{w=10}\}$

- Is it a linear code?

Yes

	$\oplus$	$\underline{c}^{(1)}$	$\underline{c}^{(2)}$	$\underline{c}^{(3)}$	$\underline{c}^{(4)}$
0000000000	$\underline{c}^{(1)}$	$\underline{c}^{(1)}$	$\underline{c}^{(2)}$	$\underline{c}^{(3)}$	$\underline{c}^{(4)}$
0000011111	$\underline{c}^{(2)}$	$\underline{c}^{(2)}$	$\underline{c}^{(1)}$	$\underline{c}^{(4)}$	$\underline{c}^{(3)}$
1111100000	$\underline{c}^{(3)}$	$\underline{c}^{(3)}$	$\underline{c}^{(4)}$	$\underline{c}^{(1)}$	$\underline{c}^{(2)}$
1111111111	$\underline{c}^{(4)}$	$\underline{c}^{(4)}$	$\underline{c}^{(3)}$	$\underline{c}^{(2)}$	$\underline{c}^{(1)}$

- $d_{\min} = \min_{\underline{c} \neq \underline{0}} w(\underline{c}) = 5$



- It can detect (at most) __4__ errors.

$d_{min} - 1 = 5 - 1 = 4$

- It can correct (at most) __2__ errors.

$\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$

ECS 452: In-Class Exercise # 15

Instructions

1. Working alone is always permitted. However, working in groups is also allowed if social distancing can be used (via, e.g., online group chat/call). For group work, **the group cannot be the same as any of your former group after the midterm.**
2. Only one submission is needed for each group.
3. **Do not panic.**

Date: 27 / 3 / 2020			
Name	ID (last 3 digits)		

1. Consider a linear block code whose generator matrix is

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

a. Find the length k of each message block

$\mathbf{G}$ has 3 rows. Therefore, $k = 3$.

b. Find the code length n

$\mathbf{G}$ has 5 columns. Therefore, $n = 5$.

c. In the table below, list all possible data (message) vectors $\underline{\mathbf{b}}$ in the leftmost column (one in each row). Then, find the corresponding codewords $\underline{\mathbf{x}}$ and their weights in the second and third columns, respectively.

$\underline{\mathbf{b}}$			$\underline{\mathbf{x}}$					$w(\underline{\mathbf{x}})$
b_1	b_2	b_3	x_1	x_2	x_3	x_4	x_5	
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	2
0	1	0	0	1	1	0	1	3
0	1	1	0	0	1	1	1	3
1	0	0	1	0	0	0	1	2
1	0	1	1	1	0	1	1	4
1	1	0	1	1	1	0	0	3
1	1	1	1	0	1	1	0	3

First, we list all possible $\underline{\mathbf{b}}$.

Next, from $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G}$, we can calculate the codeword $\underline{\mathbf{x}}$ corresponding to each $\underline{\mathbf{b}}$ one by one. Alternatively, by considering $\underline{\mathbf{b}} = [b_1 b_2 b_3]$ and carrying out the multiplication $\underline{\mathbf{x}} = [b_1 b_2 b_3]\mathbf{G}$, we have

$$\underline{\mathbf{x}} = [b_1, b_2 \oplus b_3, b_2, b_3, b_1 \oplus b_2].$$

So, each "column" of the answer for $\underline{\mathbf{x}}$ can be calculated accordingly. In particular,

- the 1st, 3rd, and 4th columns are simply copied from the columns for b_1, b_2, and b_3 respectively,
- the 2nd column is simply the sum of the columns for b_2 and b_3
- the 5th column is simply the sum of the columns for b_1 and b_2.

d. Find the minimum distance $d_{\min}$ for this code.

Because the code is linear,

$$d_{\min} = \min_{\underline{\mathbf{x}} \neq \underline{\mathbf{0}}} w(\underline{\mathbf{x}}) = 2.$$

e. What is the maximum number of bit errors that this code can guarantee to **detect**?

$$d_{\min} - 1 = 1$$

f. What is the maximum number of bit errors that this code can guarantee to **correct**?

$$\left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{1}{2} \right\rfloor = 0$$

2. Consider a linear block code whose generator matrix is

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

Suppose the minimum distance $d_{\min}$ for this code is $d_{\min} = 8$.

a. What is the maximum number of bit errors that this code can guarantee to **detect**?

$$d_{\min} - 1 = 7$$

b. What is the maximum number of bit errors that this code can guarantee to **correct**?

$$\left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{7}{2} \right\rfloor = 3$$