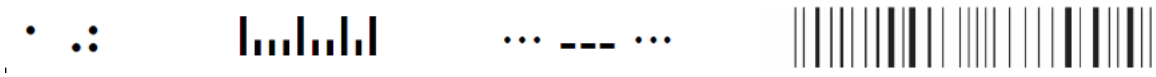
In this chapter, we learn how data can be encoded so that errors can be found. These are samples of some codes that you may recognize.



1. Binary codes

Definition 1.1.
The numbers we typically use are in base ten and are called *decimal numbers.* Each digit can be 10 possible numbers: 0, 1, 2, ... 9.

Computers use a system to transmit information where each digit can only be a 0 or a 1. These are called base two numbers, or binary numbers. Typically, in a computer, a 0 represents "off" and a 1 represents "on."

Definition 1.2.
A _____is short for binary digit and it is the smallest unit of information on a computer. It holds one of two possible values: typically 0 and 1 (which is what we will use).

Example 1.3.
How many different binary strings are there using only 1 bit?

$\cdots$ 2 bits? $\cdots$ 3 bits?

$\cdots$ 4 bits? $\cdots n$ bits?

Example 1.4.
How many bits would we need to code the alphabet? In other words, if we want each letter of the alphabet to be represented by a unique binary string, how many bits would we need?

Example 1.5.
A string of 8 bits is called a byte. How many pieces of information can we code with a byte?

In base ten, each place value is a power of ten. For example, 5472 can be written as

$$5000 + 400 + 70 + 2 \text{ or}$$

$$5(1000) + 4(100) + 7(10) + 2(1) \text{ or}$$

$$5(10^3) + 4(10^2) + 7(10^1) + 2(10^0).$$

In base two, each place value is a power of two.

Example 1.6.
Convert the binary (base two) number 1011 to a decimal (base ten) number.

Example 1.7.
Convert the binary number 11010 to a decimal number.

Example 1.8.
Convert the decimal (base ten) number 88 to a binary (base two) number.

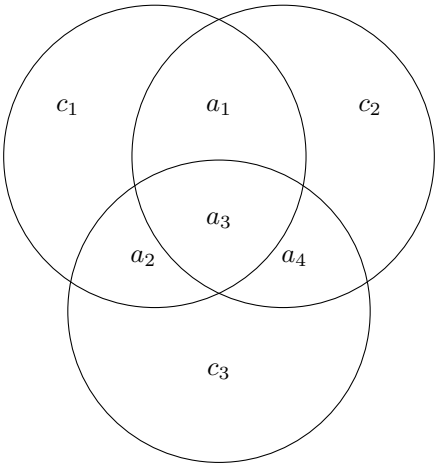
Example 1.9.
Convert the decimal number 137 to a binary number.

2. Parity-Check Sums

Definition 2.1.
When dealing with binary strings with 4 bits, we can use a method involving a circle diagram to create check digits to help catch errors in data transmission. The 4-bit string together with the three check digits is called a *code word.*

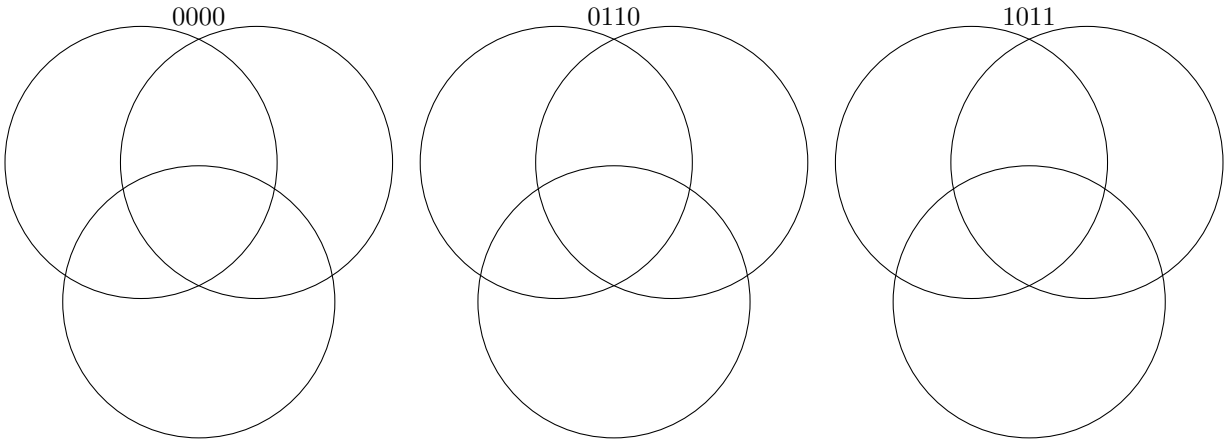
Given word : $a_1a_2a_3a_4$, Codeword:$a_1a_2a_3a_4c_1c_2c_3$

Let c_1, c_2, and c_3 be check digits found in the following manner:



(1) Place the message a_1, a_2, a_3, and a_4 in the overlapping parts of the circles as shown above.
(2) Choose the values of c_1, c_2, and c_3 so that the sum of the numbers in each circle is an even number. Parity refers to whether a number is odd or even, so we could also say that we want to choose c_i so the sum of the numbers in each circle has even parity.

Example 2.2.
Find the check digits and write the complete code word for these messages.



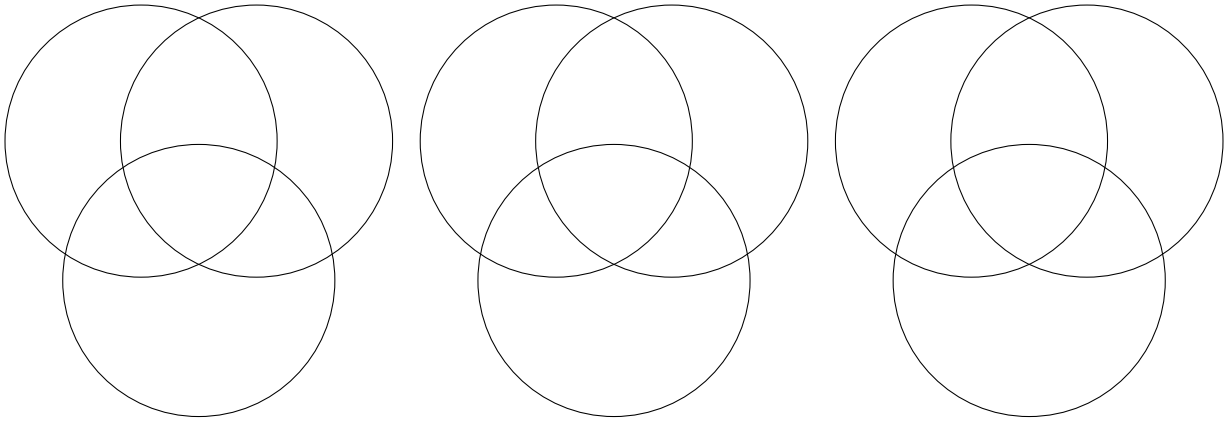
Codeword:

The circle diagram method is useful because it allows us to catch errors (if we assume there is at most one error). We determine which digit has an error by looking at the overlap of the circles that do not have an even number of ones.

Example 2.3.
Fix the error in the following code words, if there is only one error.

Received code word : 0101010 0001101 1001010

Corrected code word :



Unfortunately, the circle diagram method only works for 4-bit strings with the check digits created as above. When we have strings of other lengths, we will be given formulas to use instead.

Definition 2.4 (Hamming Distance)**.** The *distance between two strings of equal length* is the _____of _____in which the strings differ.

Example 2.5.
Find the distance between the given pairs of strings.

(1) $d(1101, 1101) =$
(2) $d(10001, 11001) =$
(3) $d(01010101, 10101010) =$

If we know the entire binary linear code (set of code words for codes of a certain length), we can use the following method to decode a message.

Definition 2.6.

The _____*decoding method* decodes a received message as the code word that agrees with the message in the most positions (has the smallest distance), provided there is only one such message. If there is a tie, don't decode.

Example 2.7.

The table below provides the code words (4-bit strings with the 3 check digits) for all possible 4-bit binary strings, so this is the binary linear code for 4-bit strings. Use this table to decode the received messages, in (a) and (b). In both parts, the chart has been recopied and the message in question has been repeated below each codeword of the chart.

0	0000000	4	0100101	8	1000110	12	1100011
1	0001011	5	0101110	9	1001101	13	1101000
2	0010111	6	0110010	10	1010001	14	1110100
3	0011100	7	0111001	11	1011010	15	1111111

(1) Received message: 0001000

0	0000000		4	0100101		8	1000110		12	1100011	
	0001000			0001000			0001000			0001000	
1	0001011		5	0101110		9	1001101		13	1101000	
	0001000			0001000			0001000			0001000	
2	0010111		6	0110010		10	1010001		14	1110100	
	0001000			0001000			0001000			0001000	
3	0011100		7	0111001		11	1011010		15	1111111	
	0001000			0001000			0001000			0001000	

Decoded code word:

(2) Received message: 0010010

0	0000000		4	0100101		8	1000110		12	1100011	
	0010010			0010010			0010010			0010010	
1	0001011		5	0101110		9	1001101		13	1101000	
	0010010			0010010			0010010			0010010	
2	0010111		6	0110010		10	1010001		14	1110100	
	0010010			0010010			0010010			0010010	
3	0011100		7	0111001		11	1011010		15	1111111	
	0010010			0010010			0010010			0010010	

Decoded code word:

Definition 2.8.
The _____of a binary code is the minimum number of 1's that occur among all non-zero code words (message plus check digits) of that code.

Example 2.9.
What is the weight of the code we have been using for the 4-bit strings?

$$\begin{aligned} C = \{&0000000, \quad 0001011, \quad 0010111, \quad 0011100, \quad 0100101, \quad 0101110, \\ &0110010, \quad 0111001, \quad 1000110, \quad 1001101, \quad 1010001, \quad 1011010, \\ &1100011, \quad 1101000, \quad 1110100, \quad 1111111\} \end{aligned}$$

We need to determine *in advance* if we want our code to just determine if there are errors (then, if errors are found, ask for retransmission of the data), or if we want our code to be able to correct errors (then, if errors are found, they could be corrected using nearest neighbor decoding).
Consider a code of weight t.

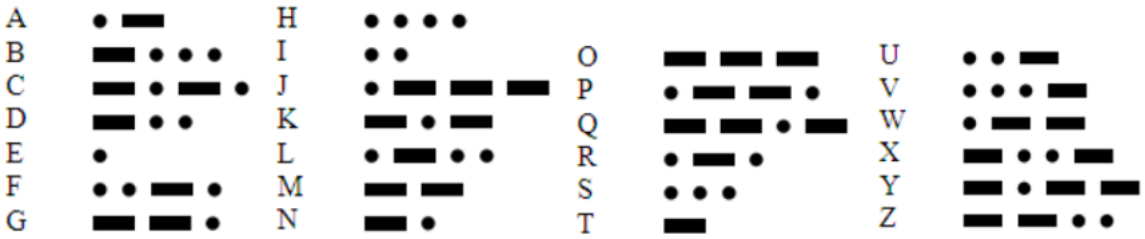
- The code can detect $t-1$ or fewer errors.
- The code can correct half as many as it can detect.
  - $\frac{t-1}{2}$ or fewer errors if t is odd.
  - $\frac{t-2}{2}$ or fewer errors if t is even.

Example 2.10.
For the code we have been using for the 4-bit strings, how many errors can be detected?
How many errors can be corrected?

What would you consider when deciding whether the code should detect or correct errors?

3. Data Compression

Another code is *Morse code.* It uses a dot and a dash in its written form (rather than 0 and 1), as seen below, and uses a space between words.



(Images from learnmorsecode.info)

The table below shows how often various letters occur in a typical text written in English:

	A	B	C	D	E	F	G	H	I	J	K	L	M
Percentage:	8	1.5	3	4	13	2	1.5	6	6.5	0.5	0.5	3.5	3
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Percentage:	7	8	2	0.25	6.5	6	9	3	1	1.5	0.5	2	0.25

What do you notice when you compare the two tables?

Definition 3.1.
Data compression is the process of encoding data so that the _____frequently occurring data are represented by the _____symbols.

Definition 3.2.
A _____converts data from an easy-to-use format to one that is more compact. MP3 is an example.

Definition 3.3.
An _____converts the compressed information back to its original (or approximately original) format.

Example 3.4.
DNA is made from four bases:adenine (A), cytosine (C), guanine (G) and thymine (T).

(1) How could these 4 bases be represented using a binary system?

(2) Using the system from (1), encode the sequence AACGCAT.

(4) Given that A occurs the most often followed by C, T and G, use the encoding

$$A \to 0 \qquad C \to 10 \qquad T \to 110 \qquad G \to 111$$

to encode the sequence AACGCAT.

(5) How much was the data compressed using the scheme in (3) rather than the one we one we used in (2)?

(6) Using the scheme from (3), decode the sequence 1001100111100.

Definition 3.5. _____uses the first value and differences in one value to the next to encode the data.

Example 3.6.
The following data represents the daily high temperatures in College Station for the first 10 days of August:

106 105 107 105 105 102 101 106 110 101

(1) How many characters (other than spaces) were required to write this data?

(2) Use delta function encoding to compress the temperature data.

(3) How many characters (including negative signs but not including spaces) were required to write this encoded data?

(4) What was the space savings for the compression of this data?

Example 3.7.

(1) Given that delta function encoding was used to create the information below, determine the original list of data.

$$54, 9, -3, 5, 7, 8, -2, -5$$

(2) What was the space savings for this compression?

Definition 3.8. _____is a way to assign shorter code words to those characters that occur more often.

Example 3.9.
Consider the case of the following 6 characters that occur with the given probabilities:

G	H	I	J	K	L
0.06	0.13	0.23	0.17	0.20	0.21

Step 1 Arrange these letters from least to most likely.

Step 2 Add the probabilities of the two least likely characters and combine them. Keep the letter with the smaller probability on the left. Arrange the new list from least to most likely.

Step 3 Repeat Step 2 until all the letters have been combined into one group with probability of 1.

Step 4 To assign a binary code to each letter, display the information in a Huffman tree by undoing the process from Steps 2 and 3. Always keep the smaller probability on the left and assign a 0 to that branch. Assign a 1 to the branch with the higher probability.

Step 5 The 0's and 1's for each path determine the code word for that letter. Read from the top of the chart down to the letter.

G	H	I	J	K	L

Example 3.10.
Using the Huffman code created in the previous example, decode

10110111100110011110.

4. Cryptography

Definition 4.1.
The process of coding information to (hopefully) prevent unauthorized use is called _____. The process of decoding the encrypted message to get back to the original message is called _____.

_____is the study of making and breaking secret codes. While modern encryption schemes are extremely complex, we will look at a few simplified examples to illustrate the fundamental concepts that are involved. To see a coding scheme that is used for quite a few things and is hard to break, read about RSA public key encryption on pages 622-625.

A _____shifts the letters of the alphabet by a fixed number of positions.

Example 4.2.
Create a Caesar cipher that shifts the alphabet by 6 letters to the right, and use it to encrypt the message MATH IS FUN.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Example 4.3.
Use the Caesar cipher above to decrypt QKKV G YKIXKZ.

Note: The Caesar cipher encodes the letter n with $(n+x)$ mod 26 and decodes the letter m with $(m-x)$ "mod" 26.

Definition 4.4.

A _____multiplies the position of each letter by a fixed number k (called the key) and then uses modular arithmetic. To use a decimation cipher,

Step 1 Assign the letters A – Z to the numbers 0 – 25, in order.

Step 2 Choose a value for the key, k, that is an odd integer from 3 to 25 but not 13. (Why can we not use 13?)

Step 3 To find the encrypted letter, x, multiply the value of each original letter, i, by the key, k, and find the remainder when divided by 26. That is, $x \equiv ki \bmod 26$.

Step 4 To decrypt a message, first determine the decryption key, j, such that $kj \equiv 1 \bmod 26$. Then, the original letter will be equal to $xj \bmod 26$.

Math Note: Another way of performing arithmetic mod n is to add or subtract n until you get a number between 0 and n–1. This is especially useful when dealing with negative numbers.

Example 4.5.

(1) 57 mod 26 =
(2) 78 mod 26 =
(3) – 5 mod 26 =
(4) – 38 mod 26 =

Example 4.6.

Encrypt "MATH IS FUN" using a decimation cipher with a key of 7.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

	M	A	T	H		I	S		F	U	N
Position:											
(Pos)∗(e.key):											
Mod 26:											
Code:											

There is a chart on page 618 of your book showing the decryption key, j, for all possible values for the encryption key, k. Pairs of encryption and decryption values include $(3, 9), (5, 21), (7, 15), (11, 19), (17, 23)$, and $(25, 25)$. The decryption key is chosen so that $kj \equiv 1 \bmod 26$. Which also means that $kj \bmod 26 = 1$. You do not need to memorize this chart.

Example 4.7.

Show that 15 is the correct decryption key for an encryption key of 7.

Example 4.8.

The message below was encrypted with the key 21. The decryption key is $j = 5$. What does the message say?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

	S	A	T	I	I	N		I	E	J
Position:										
(Pos)∗(e.key):										
Mod 26:										
Code:										

Definition 4.9.

A _____uses any *key* word to encode the characters.

Example 4.10.

To encrypt the message MATH IS FUN using the key word BOX, translate your message into numbers, add the numbers for successive letters of the key word mod 26, and then translate the results back into letters.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

	M	A	T	H		I	S		F	U	N
Position:											
KeyWord:											
Sum:											
Mod26:											
Code:											

Example 4.11.

To decode the message EMYWGIRII that used the key word RICE, translate the coded message into numbers, _____the numbers for successive letters of the key word mod 26, and then translate the results back into letters.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

	E	M	Y	W	G	I	R	I	I
Position:									
KeyWord:									
Difference:									
Mod 26:									
Message:									

Definition 4.12.

Encrypting Credit Card Data on the Web To carry out addition of binary strings, add each of the digits. If the result is even, enter 0. If the result is odd, enter 1. In other words, take the sum of the digits mod 2.

Example 4.13.

Add the binary strings 100011 and 110010.

Example 4.14.

Add the binary strings 110010 and 110010.

Example 4.15.

What happens when any binary string is added to itself?

This gives us another way to encrypt data.

When you place an order on the web, the company sends your computer a string of random 1's and 0's, called a key, that is the same length as the data you need to send them. Your computer adds the key to your data and transmits the result over the internet.

Example 4.16.
Let's say you want to send a company the information 01100010 and they send your computer the key 10101101, what would your computer send back?

To decode what they receive, the company adds the key to what your computer sent back. What would be the result of the example above?