

Binary Numbers, Computers, and Error Correction

Richard Blecksmith
Dept. of Mathematical Sciences
Northern Illinois University
Website: <http://math.niu.edu/~richard/>

BINARY NUMBERS

1. GOAL

Goal: to familiarize you with the mathematical ideas behind some of the technology used in our society.

- Computer architecture uses **binary numbers** in fixed sizes, called **bytes**.
- Megabyte of memory is (approx) one million bytes.
- Gigabyte of memory is (approx) one billion bytes.

3. ERROR-CORRECTING CODES

In **error-correcting** codes, we encode information so that if an error occurs in reading the information, we can still recover the original data.

Compact disk technology utilizes this type of coding, where small scratches on a CD need not damage the sound reproduction.

2. ENCODING INFORMATION

We are particularly interested in the mathematical concepts used to **encode** information.

Two purposes for encoding information:

Purpose 1. **Error-correcting** codes

Purpose 2. **Encryption**

4. SECRET CODES

A second purpose of coding information is to make it **secure**, so that unauthorized persons cannot read private messages.

Secret codes have obvious military, industrial, and personal applications.

Our simple encryption-decryption method relies on some basic facts about “clock-arithmetic” and on a method for solving what we call “the pails-of-water” problem.

5. SECTION 1. BINARY NUMBERS

- Computers store numbers in binary form.
- This means that they are stored in base 2.
- In our familiar base 10 number system, decimal numbers are represented using the powers of 10.
- For example,

$$2047 = 2 \cdot 10^3 + 0 \cdot 10^2 + 4 \cdot 10^1 + 7.$$

6. BINARY REPRESENTATION

- In base 2 the powers of 2 are used instead of the powers of 10.
- Also, the only digits allowed are 0 and 1.
- This restriction is convenient for the computer, which we can think of as using switches to store these binary digits, 1 = on, 0 = off.

7. THE POWERS OF TWO

n	2^n
1	2
2	4
3	8
4	16
5	32

n	2^n
6	64
7	128
8	256
9	512
10	1024

9. BINARY NUMBER CHART

Binary representation of the numbers 1 through 32:

1	1	9	1001	17	10001	25	11001
2	10	10	1010	18	10010	26	11010
3	11	11	1011	19	10011	27	11011
4	100	12	1100	20	10100	28	11100
5	101	13	1101	21	10101	29	11101
6	110	14	1110	22	10110	30	11110
7	111	15	1111	23	10111	31	11111
8	1000	16	10000	24	11000	32	100000

11. COMPUTER LINGO

- each binary digit is called a **bit**,
- 8 consecutive bits are called a **byte**.
- Since computers are binary, not decimal machines,
- the numbers one thousand and one million are not naturally represented.
- However, $2^{10} = 1024$ is close to a thousand and
- $2^{20} = 1,048,576$ is close to a million.

8. BINARY EXAMPLE

The binary number

$$\begin{aligned}
 (110101)_2 &= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \\
 &= 2^5 + 2^4 + 2^2 + 1 \\
 &= 32 + 16 + 4 + 1 = 53.
 \end{aligned}$$

10. TWO KINDS OF PEOPLE

There are

10

kinds of people in the world—

those that can read binary
and those that can't.

12. KILO, MEGA, GIGA, AND TERA

- One kilobyte (K) of memory is actually 1024 bytes,
- although most people think of this as roughly a thousand bytes.
- One megabyte (M) of memory consists of 2^{20} total bytes.
- A gigabyte (G) is 2^{30} bytes, roughly one billion.
- A terabyte (T) is 2^{40} bytes, roughly one trillion.

13. STORING CHARACTERS

- A byte is designed to hold one **character** of information.
- Standard keyboard characters such as letters (lower and upper case), digits, and punctuation symbols are coded by a standard convention called the **ASCII** code.
- ASCII stands for American Standard for Coded Information Interchange.
- In the ASCII code, the letter A is assigned the value 65, B is stored as 66, etc.
- There are $2^8 = 256$ possible characters that can be stored in a byte, well beyond the number of characters on a typewriter keyboard.

15. PARITY CHECK BIT EXAMPLES

What is the parity check bit for $b = 10010110$?
 Answer: 0 **Why?** since b consists of four 1's and 4 is even.

What is the parity check bit for $b = 10010111$?
 Answer: 1 **Why?** since b consists of five 1's and 5 is odd.

17. EXERCISES

- (1) Write the following numbers in binary:
21, 57, 99
- (2) How can you tell if a binary number is even or odd?
- (3) Write the following binary numbers in decimal: 101010, 1001001, 11111111.
- (4) Add the following binary numbers:
 $1101101 + 1010101$
 $10111011 + 10001000$
 $11111111 + 1$
- (5) What is the parity check bit for the binary numbers in Exercise 3?

14. PARITY CHECK BIT

Many computers contain a 9th bit, which you never see, called the **parity check bit**, for every byte in memory.

Suppose b is a byte of memory.

This means that b consists of 8 consecutive bits.

The parity check bit is

- 0 if the number of 1's in b is even;
- 1 if the number of 1's in b is odd.

16. PARITY CHECK BIT AND HARDWARE CRASH

- The number of 1's in each byte plus the parity check bit is always even.
- If the hardware of a computer detects an odd number of 1's in a byte plus its parity check bit,
- the computer will crash immediately and report a hardware malfunction.

ERROR CORRECTING CODES

18. COMMUNICATION ERRORS

In daily communications, errors can and do occur.

Words are misspelled.

Wrong letters are typed.

Data is damaged, like by scratches on a CD.

20. ERROR CORRECTING CODES

Suppose we wish to encode a half byte (4 bits) so that if an error occurs in transmitting this data we could detect the error.

One simple minded way would be to just repeat the 4 bits.

For example, we could encode $m = 1011$ as

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix},$$

where the two rows are identical.

22. AN ERROR CORRECTING CODE

Goal: Use the parity check bit

- to **detect** errors and also
- to **correct** them.

Our Code is called the **Tic-Tac-Toe Code**

19. CONSEQUENCES

Usually the errors can be identified and corrected.

Like when Google says “Did you mean ...”

But even a one-letter change can have serious consequences.

Like when **NOT** is changed to **NOW**.

21. FINDING THE ERROR

If the message $M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$, is sent over a communications channel and the message

$$M' = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

is received, we know that an error occurred in the third bit.

But we don't know whether the third bit was originally a 0 and the first line is wrong, or whether the third bit was originally a 1 and the second line is wrong.

You can **find** the error— you just can't **fix** it.

23. THE TIC-TAC-TOE CODE

Store the original message $m = b_1b_2b_3b_4$ in a 3 by 3 tic-tac-toe board:

$$M = \left[\begin{array}{cc|c} b_1 & b_2 & d_1 \\ b_3 & b_4 & d_2 \\ \hline c_1 & c_2 & c_3 \end{array} \right].$$

- d_1 is the parity check bit for row 1
- d_2 is the parity check bit for row 2
- c_1 is the parity check bit for column 1
- c_2 is the parity check bit for column 2
- c_3 in the lower right hand corner (row 3, col 3) is the parity check bit for row 3 or for column 3. You get the same answer either way. Why?

24. EXAMPLE

The message $m = 1011$ is encoded:

$$M = \begin{bmatrix} 1 & 0 & ?1 \\ 1 & 1 & ?0 \\ ?0 & ?1 & ?1 \end{bmatrix}.$$

Note: the numbers of 1's in each column and each row is always even.

If the receiver gets a message with an odd number of 1's in a row or column, she knows an error occurred somewhere in that row or column.

26. CORRECTING THE ERROR

error in row 2

1	0	1
0	1	0
0	1	1

error in
col 1

Change the 0 in row 2, column 1 to a 1.

We thus recover the correct transmitted message M .

To obtain the original four-bit message m , simply peel off the 5 parity check bits from M .

$$m = 1\ 0\ 1\ 1$$

28. EXERCISE 2

Decode the following messages using the Tic-tac-toe code:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

25. EXAMPLE CONT'D

Suppose she receives

$$M' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

- Both the second row and the first column contain a single 1.
- Since the number 1 is odd, an error must have occurred in
row 2, column 1.
- Not only have we detected an error,
- but we know where it occurred!

27. EXERCISE 1

Encode the following messages in the Tic-tac-toe code:

1011

0110

1111.

29. QUESTION

Using the Tic-tac-toe code, can you detect two errors?

Are you able to correct both errors?

Given examples to support your answers.