Error-Correcting Codes

Prepared by Mark on June 26, 2023

Part 1: Error Detection

An ISBN[1] is a unique numeric book identifier. It comes in two forms: ISBN-10 and ISBN-13. Naturally, ISBN-10s have ten digits, and ISBN-13s have thirteen. The final digit in both versions is a *check digit*.

Say we have a sequence of nine digits, forming a partial ISBN-10: $n_1 n_2 ... n_9$.
The final digit, n_{10}, is chosen from $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ so that:

$$\sum_{i=1}^{10} (11 - i) n_i \ (mod\ 11) = 0$$

If n_{10} is equal to 10, it is written as `X`.

Problem 1:
Only one of the following ISBNs is valid. Which one is it?

- `0-134-54896-2`
- `0-895-77258-2`

[1] International Standard Book Number

Problem 2:
Take a valid ISBN-10 and change one digit. Is it possible that you get another valid ISBN-10? Provide an example or a proof.

Problem 3:
Take a valid ISBN-10 and swap two adjacent digits. When will the result be a valid ISBN-10? This is called a *transposition error*.

Problem 4:
ISBN-13 error checking is slightly different. Given a partial ISBN-13 $n_1 n_2 n_3 ... n_{12}$, the final digit is given by

$$n_{13} = \left[\sum_{i=1}^{12} n_i \times (2 + (-1)^i) \right] \bmod 10$$

What is the last digit of the following ISBN-13?
`978-0-380-97726-?`

Problem 5:
Take a valid ISBN-13 and change one digit. Is it possible that you get another valid ISBN-13? If you can, provide an example; if you can't, provide a proof.

Problem 6:
Take a valid ISBN-13 and swap two adjacent digits. When will the result be a valid ISBN-13?
Hint: The answer here is more interesting than it was last time.

Problem 7:
`978-0-08-2066-46-6` was a valid ISBN until I changed a single digit.
Can you find the digit I changed? Can you recover the original ISBN?

Part 2: Error Correction

As we saw in Problem 7, the ISBN check-digit scheme does not allow us to correct errors.
QR codes feature a system that does.

Odds are, you've seen a QR code with an image in the center. Such codes aren't "special"—they're simply missing their central pixels. The error-correcting algorithm in the QR specification allows us to read the code despite this damage.

Definition 8: Repeating codes
The simplest possible error-correcting code is a *repeating code*. It works just as you'd expect:
Instead of sending data once, it sends multiple copies of each bit.
If a few bits are damaged, they can be both detected and repaired.
For example, consider the following three-repeat code encoding the binary string 101:

$$111\ 000\ 111$$

If we flip any one bit, we can easily find and fix the error.

Problem 9:
How many repeated digits do you need to...
- detect a transposition error?
- correct a transposition error?

Definition 10: Code Efficiency
The efficiency of an error-correcting code is calculated as follows:

$$\frac{\text{number of data bits}}{\text{total bits sent}}$$

For example, the efficiency of the three-repeat code above is $\frac{3}{9} = \frac{1}{3} \approx 0.33$

Problem 11:
What is the efficiency of a k-repeat code?

As you just saw, repeat codes are not a good solution. You need many extra bits for even a small amount of redundancy. We need a better system.

Part 3: Hamming Codes

Say we have a 16-bit message, for example `1011 0101 1101 1001`.
We will number its bits in binary, from left to right:

Bit	1	0	1	1	0	1	0	1
Index	0000	0001	0010	0011	0100	0101	0110	01

and so on...

Problem 12:
In this 16-bit message, how many message bits have an index with a one as the last digit?
(i.e, an index that looks like `***1`)

Problem 13:
Say we number the bits in a 32-bit message as above.
How many message bits have an index with a one as the n^{th} digit?

We now want a way to detect errors in our 16-bit message. To do this, we'll replace a few data bits with parity bits. This will reduce the amount of information we can send, but will also improve our error-detection capabilities.

Let's arrange our message in a grid. We'll make the first bit (currently empty, marked `X`) a parity bit. Its value will depend on the content of the message: if our message has an even number of ones, it will be zero; if our message has an odd number of ones, it will be one.
This first bit ensures that there is an even number of ones in the whole message.

Bit Numbering

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Sample Message

X	0	1	1
0	1	0	1
1	1	0	1
1	0	0	1

Problem 14:
What is the value of the parity bit in the message above?

Problem 15:
Can this coding scheme detect a transposition error?
Can this coding scheme detect two single-bit errors?
Can this coding scheme correct a single-bit error?

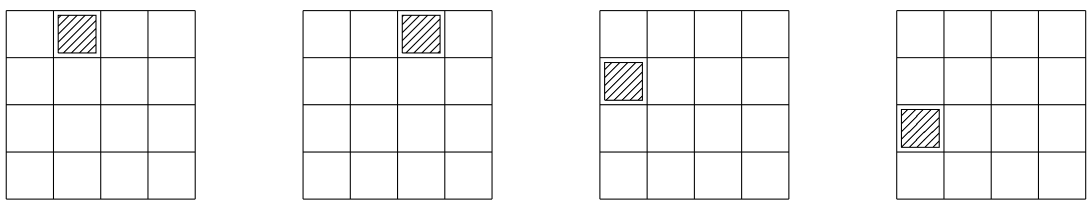
We'll now add four more parity bits, in positions `0001`, `0010`, `0100`, and `1000`:

X	X	X	1
X	1	0	1
X	1	0	1
1	0	0	1

Bit `0001` will count the parity of all bits with a one in the first digit of their index.
Bit `0010` will count the parity of all bits with a one in the second digit of their index.
Bits `0100` and `1000` work in the same way.
Hint: In `0001`, `1` is the first digit. In `0010`, `1` is the second digit.
When counting bits in binary numbers, go from right to left.

Problem 16:
Which message bits does each parity bit cover?
In other words, which message bits affect the value of each parity bit?

Four diagrams are shown below. In each grid, fill in the bits that affect the shaded parity bit.



Problem 17:
Compute all parity bits in the message above.

Problem 18:

Analyze this coding scheme.

- Can we detect one single-bit errors?
- Can we detect two single-bit errors?
- What errors can we correct?

Problem 19:

Each of the following messages has either 0, 1, or two errors.

Find the errors and correct them if possible.

Hint: Bit `0000` should tell you how many errors you have.

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

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

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

Problem 20:
How many parity bits does each message bit affect?
Does this correlate with that message bit's index?

Problem 21:
Say we have a message with exactly one single-bit error.
If we know which parity bits are inconsistent, how can we find where the error is?

Problem 22:
Can you generalize this system for messages of 4, 64, or 256 bits?

Part 4: Bonus

Problem 23:
A stressed-out student consumes at least one espresso every day of a particular year, drinking 500 overall. Show the student drinks exactly 100 espressos on some consecutive sequence of days.