When Good Data GOES BAD

Error-Correcting Codes in Theory and in Practice

By Allen Smith

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Barcodes

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Data that always seems to go bad when the line is longest.

8 21935 11112 4

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Parity Checks

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- Computers use binary numbers
- Each characters is represented by a series of zeroes and ones.

  'a' = 110 0001
- Set the parity so that the number of ones is even:

  'a' => 1100 0011

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Error-Correcting Codes

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Tack on some extra bits which carry redundant information.

Triple-Repetition Code:	1	0	1
	↓	↓	↓
Corrupted:	111	010	100
	↓	↓	↓
Repaired:	111	000	000
	↓	↓	↓
Decoded:	1	0	0

010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Linear Codes

010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Codewords are produced by multiplying data by a generator matrix.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\underbrace{(\ 0\ 0\ 1\ 1\ )}_{\text{Original Data}} H = \underbrace{(\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ )}_{\text{Codeword}}$$

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Linear Codes: Terminology

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- *Weight*
  - the number of 1s in a word
  - `(0 0 1 1)` has weight 2
- *Hamming Distance*
  - the number of coordinates in which two words differ
  - `(0 0 1 1)` and `(0 0 0 1)` have distance 1
- *Sum*
  - the vector sum modulo 2

`(0 0 1 1) + (0 0 0 1) = (0 0 1 0)`

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

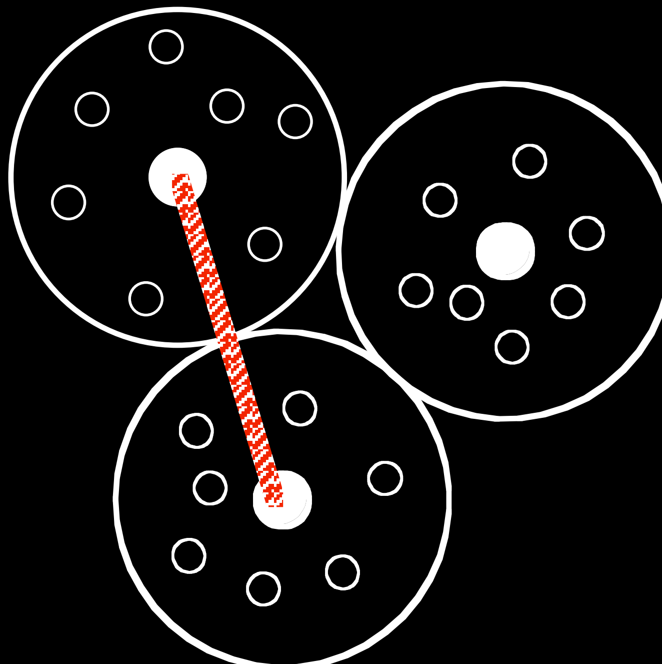
Distance

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

When **u**, **v**, and **w** are binary words,

$$d(\mathbf{u},\mathbf{u}) = 0$$
$$d(\mathbf{u},\mathbf{v}) = d(\mathbf{v},\mathbf{u})$$
$$d(\mathbf{u},\mathbf{v}) \leq d(\mathbf{u},\mathbf{w}) + d(\mathbf{w},\mathbf{v})$$

This is a mathematical distance function; more specifically, a metric of binary n-tuples.



Hamming Code: Encoding

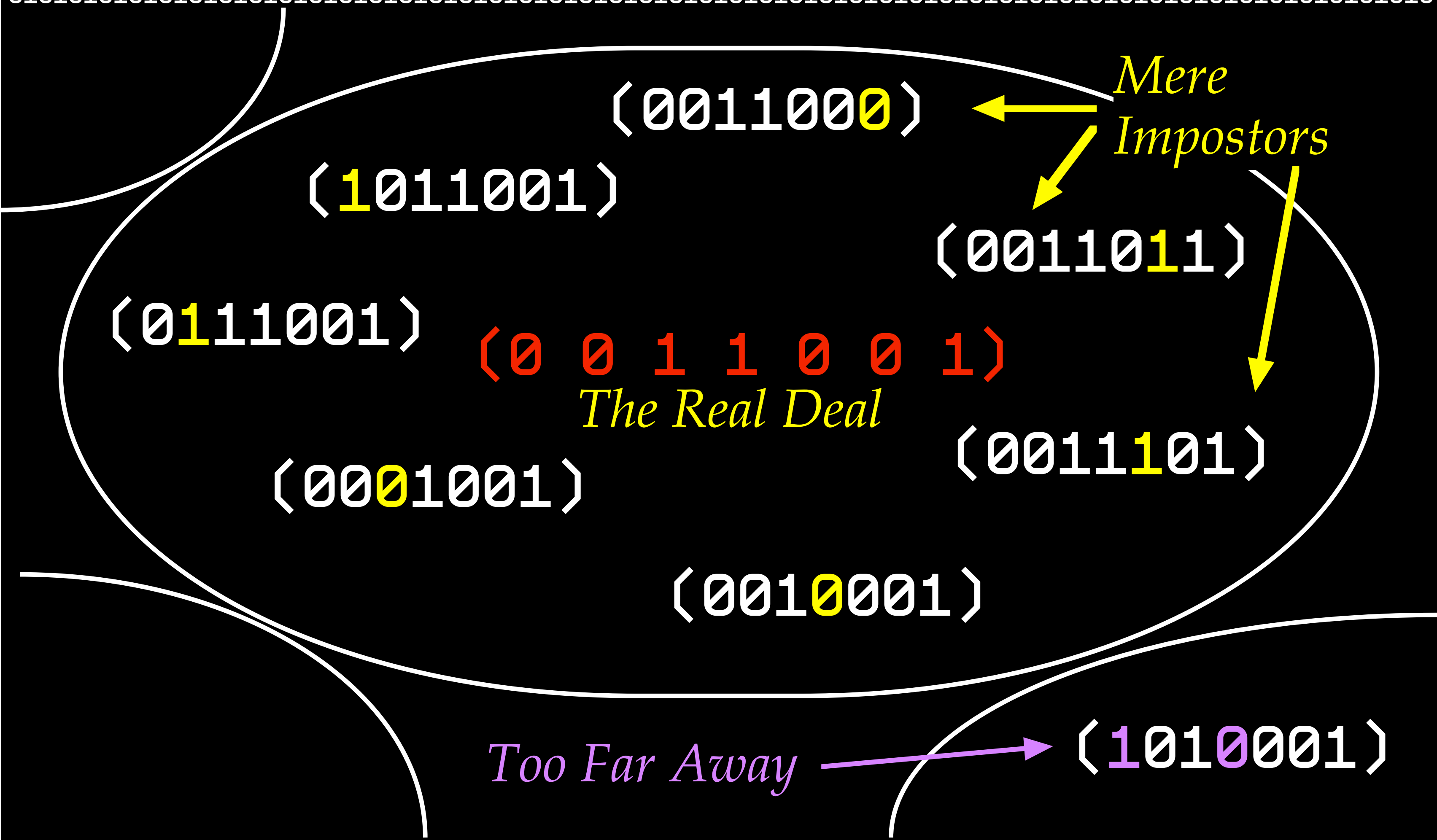
Generator Matrix for the Hamming Code: $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$

Creating Codewords:

$$(0\ 0\ 1\ 1)H = (0\ 0\ 1\ 1\ 0\ 0\ 1)$$

$$(0\ 0\ 1\ 0)H = \underbrace{(0\ 0\ 1\ 0\ 1\ 0\ 0)}_{\text{codewords have distance 3}}$$

Hamming Code: Distance



0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

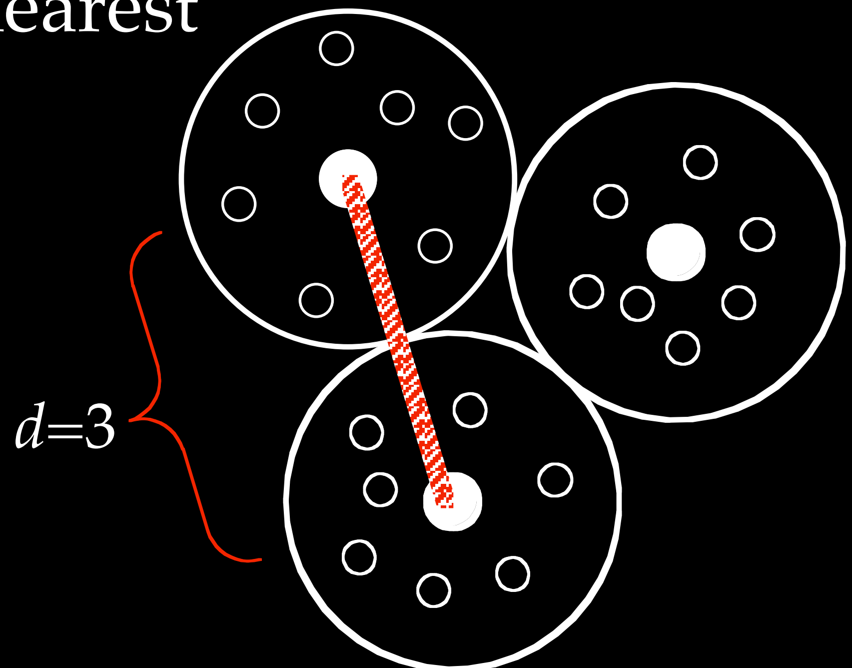
Maximum Likelihood Decoding

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- When a corruption occurs, we correct it to the nearest actual codeword.
- A code can correct

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

or fewer errors.



Minimum Distance

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Hamming Encoding/Checking

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \end{pmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{pmatrix} a_1 & a_2 & a_3 & a_4 & \underbrace{(a_2 + a_3 + a_4) \quad (a_1 + a_3 + a_4) \quad (a_1 + a_2 + a_4)}_{\text{Extra Check Bits}} \end{pmatrix}$$

Received Word:

$$\begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 \end{pmatrix} \rightarrow \begin{array}{l} b_2 + b_3 + b_4 + b_5 \bmod 2 = 0 \\ b_1 + b_3 + b_4 + b_6 \bmod 2 = 0 \\ b_1 + b_2 + b_4 + b_7 \bmod 2 = 0 \end{array}$$

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Hamming Check Matrix

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

$$\left.\begin{aligned} b_2 + b_3 + b_4 + b_5 \bmod 2 = 0 \\ b_1 + b_3 + b_4 + b_6 \bmod 2 = 0 \\ b_1 + b_2 + b_4 + b_7 \bmod 2 = 0 \end{aligned}\right\} P = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Checking Hamming Codewords:

$$(0\ 0\ 1\ 1\ 0\ 0\ 1)P = (0\ 0\ 0)$$

$$(1\ 0\ 1\ 1\ 0\ 0\ 1)P = (0\ 1\ 1)$$

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Hamming on the Computer

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Hamming Parity Check

01010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- Attach a parity bit to the 7-bit codewords

- If the received parity is wrong, 1 error has occurred.
  - Correct using the syndrome

- If the parity is correct, 0 or 2 errors have occurred.
  - If 0 errors, syndrome will be (000)
  - If 2 errors, syndrome will be nonzero

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Hamming Problems!

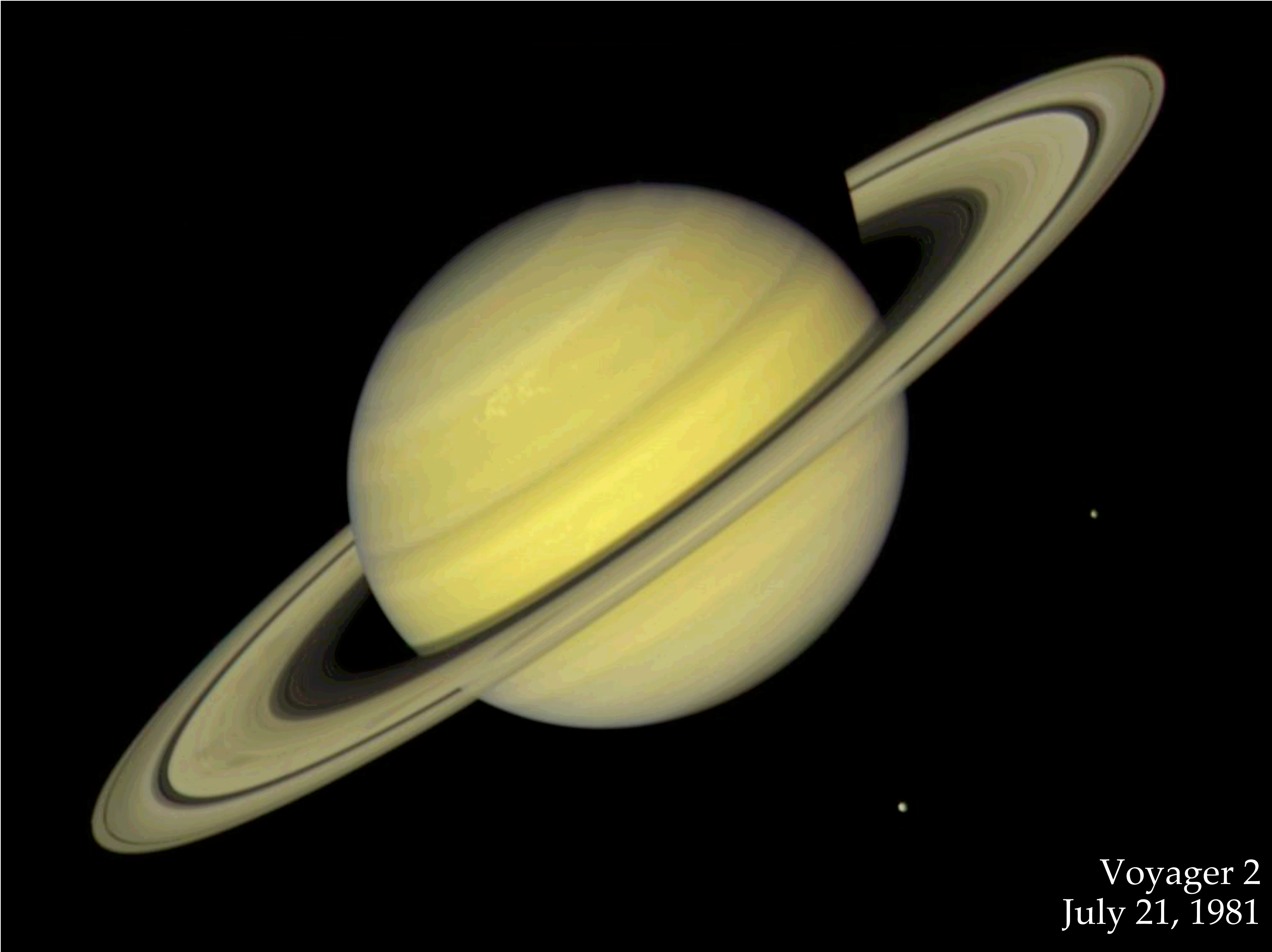
0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- `0000 0000` is the string termination symbol in C.
- Guess what happens to the letter P:

`'P' = 0101 0000`

$$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix} \cdot \begin{bmatrix}0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1\end{bmatrix} = \begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0\end{pmatrix}$$

- Even parity produces `0000 0000`
- Odd parity produces `0000 0001`


Voyager 2
July 21, 1981

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Error Vectors

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- Error vectors identify the position in which errors occurred.

original word: (0 1 1 0)

received word: (0 0 1 0)

error vector: (0 1 0 0)

- The sum of an error vector and the received word gives the correct word.

received word: (0 0 1 0)

error vector: +(0 1 0 0)

original word: (0 1 1 0)

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Golay-23

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

- Every Golay codeword has even parity.
  - Thus, one bit is extra
- By tossing out one check bit, we maintain the code structure.
- The parity information can be used to detect 4 errors.

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010

Error Correction and You

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010