Character Encoding

- American Standard Code for Information Interchange (ASCII), is a character encoding standard
- Originally based on the English alphabet, ASCII encodes 128 specified characters into seven-bit integers
- requires all data transmission to send eight bits when seven could suffice
- UTF-8 is a character encoding capable of encoding all possible characters defined by Unicode. The encoding is variable-length and uses 8-bit code units.
- UTF-8 is the dominant character encoding for the World Wide Web, accounting for 87.9% of all Web pages in November 2016
- It was designed for backward compatibility with ASCII
- The first 128 characters of Unicode are encoded using a single octet with the same value as ASCII, so that valid ASCII text is valid UTF-8-encoded Unicode as well

Error detection — parity bit

The redundant 8th bit in ASCII is often used for as a parity bit for error detection

char	ASCII	even parity bit	odd parity bit
A	1000001	0	1
B	1000010	0	1
C	1000011	1	0
D	1000100	0	1
E	1000101	1	0

- The parity bit is the result of dividing the 7-bit code by modulo 2, i.e. 0 if even number of 1s, 1 if an odd number of 1s.
- An error is detected if the result of dividing the 8-bit code (including parity bit) by modulo 2 is 1 if using even parity (0 for odd parity)
- the system could ask for an Automatic Resend Request (ARQ)
- Can only detect a single bit error in a codeword

Error detection

- The probability of k bits being in error in a block of n bits is given by the probability mass function

$$\Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

  where p is the bit error ratio (BER) and the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.
- If $p \ll 1$, then we can approximate $(1-p) \approx 1$.

Error detection

- For a 7-bit word with no parity this gives

$$\Pr_7(X = 1) = \binom{7}{1} p^1 (1-p)^6 \approx 7p$$

- For an 8-bit word with parity, single bit errors are detected, so the probability of error is dominated by the probability of 2 bit errors

$$\Pr_8(X = 2) = \binom{8}{2} p^2 (1-p)^6 \approx 28p^2$$

- The ratio of these is $4p$.
- For example, if BER$= 10^{-4}$ then $\Pr_7(X = 1) = 7 \times 10^{-4}$ and $\Pr_8(X = 2) = 2.8 \times 10^{-7}$.
- the code-rate efficiency for the single parity bit check is 7/8.

Forward Error Correction

- Forward error correction (FEC) or channel coding is a technique used for controlling errors in data transmission over unreliable or noisy communication channels.
- The central idea is the sender encodes the message in a redundant way by using an error-correcting code.
- The American mathematician Richard Hamming pioneered this field in the 1940s and invented the first error-correcting code in 1950.
- The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission. . .
- . . . at the cost of a fixed, higher forward channel bandwidth.

Triple Modular Redundancy

- transmit each data bit 3 times, which is known as a (3,1) repetition code
- an error in any one of the three samples is corrected by “majority vote”
- 111 (error free), 110, 101, or 011 are interpreted as 1
- 000 (error free), 001, 010, or 100 are interpreted as 0
- without FEC, error in 1 bit has probability $\Pr_3(X = 1) = 3p$
- with FEC, error in 2 bits has probability $\Pr_3(X = 2) = 3p^2$
- improved by ratio p
- the code-rate efficiency for triple redundancy is 1/3.
- Though simple to implement and widely used, this triple modular redundancy is a relatively inefficient FEC.

Hamming distance

- In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.
- In another way, it measures the minimum number of substitutions required to change one string into the other, or the minimum number of errors that could have transformed one string into the other.
- The Hamming weight of a code word $\mathbf{c}$ is the number of nonzero components of the code word, denoted $w(\mathbf{c})$.
- The Hamming distance between two code words $\mathbf{c}_i$ and $\mathbf{c}_j$ can be given by $d(\mathbf{c}_i, \mathbf{c}_j) = w(\mathbf{c}_i \oplus \mathbf{c}_j)$ and $\oplus$ represents component-wise modulo 2 addition (XOR).
- The minimum Hamming distance of a code is the minimum Hamming distance between any two different code words $d_{\text{min}} = \min_{i \neq j} d(\mathbf{c}_i, \mathbf{c}_j)$

Hamming distance

- A code with all distinct code words has a minimum Hamming distance of at least 1
- A code which permits detection of up to e errors per word has a minimum Hamming distance of at least (e+1)
- A code which permits correction of up to e errors per word has a minimum Hamming distance of at least (2e+1)

Hamming codes

- Hamming codes are a family of linear binary error-correcting codes
- Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance of three.
- For each integer $r \geq 2$ number of parity bits there is a code with

block length $2^r - 1$
message length $k = 2^r - r - 1$
code-rate efficiency $\frac{k}{n} = 1 - \frac{r}{2^r - 1}$

parity bits	total bits	data bits	name	code-rate efficiency
2	3	1	Hamming(3,1)	$\frac{1}{3} \approx 33\%$
3	7	4	Hamming(7,4)	$\frac{4}{7} \approx 57\%$
4	15	11	Hamming(15,11)	$\frac{11}{15} \approx 73\%$
5	31	26	Hamming(31,26)	$\frac{26}{31} \approx 84\%$

Hamming(3,1) code is equivalent to Triple Modular Redundancy

General Algorithm

The following general algorithm generates a single-error correcting (SEC) code for any number of bits.

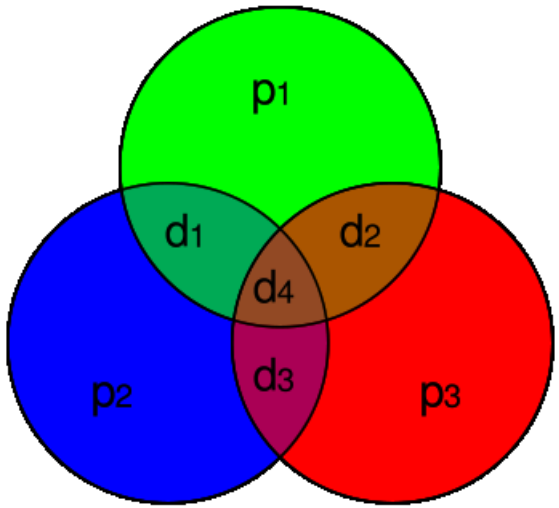
1. Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.
2. Write the bit numbers in binary: 1, 10, 11, 100, 101, etc.
3. All bit positions that are powers of two (have only one 1 bit in the binary form of their position) are parity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000)
4. All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.
5. Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.
6. Parity bit n covers all bit positions which have the nth least significant bit set

Hamming(7,4) code

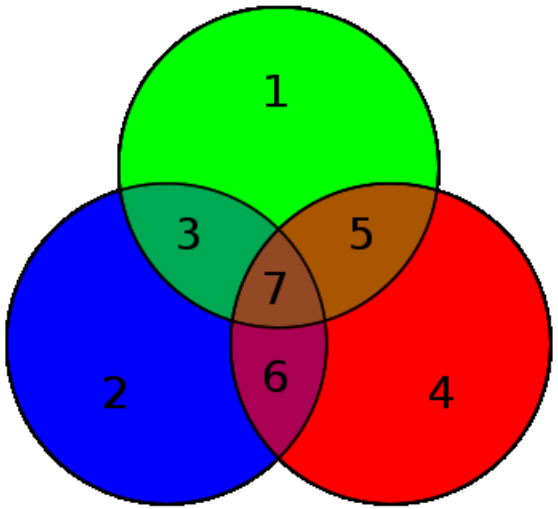
Two Hamming matrices can be defined: the code generator matrix $\mathbf{G}$ and the parity-check matrix $\mathbf{H}$

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

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



data and parity bits



bit position

Generator and Parity Check Matrices

- Here binary matrix multiplications are modulo 2 (i.e. result 0 if even, 1 if odd)
- The Hamming matrices have the property $\mathbf{H} \cdot \mathbf{G}^{\mathrm{T}} = \mathbf{0}$
- The Hamming code word is obtained $\mathbf{x} = \mathbf{p} \cdot \mathbf{G}$ where $\mathbf{p}$ is the original data code word.
- If no error occurs during transmission, the received code word $\mathbf{r} = \mathbf{x}$
- The syndrome $\mathbf{z} = \mathbf{H} \cdot \mathbf{r}^T$, is the null vector if there is no error and $\mathbf{r}$ is a valid code word.
- For a single bit error, $\mathbf{r} = \mathbf{x} + \mathbf{e}_i$, where $\mathbf{e}_i$ has a 1 in the ith position and 0 otherwise.
- The syndrome $\mathbf{z} = \mathbf{H} \cdot \mathbf{r}^T = \mathbf{H} \cdot \mathbf{e}_i^T$ identifies the single bit error from matching the appropriate column of $\mathbf{H}$
- The General Algorithm identifies the bit error as the binary value of $\mathbf{z}$.
- The correction is by flipping the ith bit of $\mathbf{r}$
- Hamming codes will also detect two-bit errors, but these are indistinguishable from single-bit errors and cannot be corrected.

Hamming(7,4) code

Using the general algorithm form, we get the linear code:

data	code-word	weight
0000	**000**0000	0
1000	**111**0000	3
0100	**10**0**1**100	3
1100	**01**1**1**100	4
0010	**01**0**1**010	3
1010	**10**1**1**010	4
0110	**11**0**0**110	4
1110	**00**1**0**110	3

data	code-word	weight
0001	**11**0**1**001	4
1001	**00**1**1**001	3
0101	**01**0**0**101	3
1101	**10**1**0**101	4
0011	**10**0**0**011	3
1011	**01**1**0**011	4
0111	**00**0**1**111	4
1111	**111**1111	7

- minimum of 3 bit changes to change code-word, i.e. Hamming distance is 3
- can correct 1 bit error per word, or detect up to 2 bit errors
- syndrome identifies placement of error

Hamming(7,4) in systematic form

The code matrices can be mutated by swapping columns and linear combinations (modulo 2 arithmetic) of rows to give equivalent forms, including the systematic form (data bits followed by parity bits):

$$\mathbf{G} = \left(I_k | -A^T\right) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{H} = \left(A | I_{n-k}\right) = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Relating the syndrome to the error bit can be done by matching the column in $\mathbf{H}$ (lookup table).

Ternary Parity Check Problems

Problem 1

- You are given 9 nominally identical coins
- One coin is a forgery, and weighs more than a genuine coin
- You are given a set of (balance) scales
- In just two weighings, identify the forged coin

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

Problem 2

- You are given 12 nominally identical coins
- One coin may (or may not) be a forgery, which either weighs less or more than a genuine coin
- You are given a set of (balance) scales
- In just three weighings, identify if there is a forgery, which coin it is and whether it is lighter of heavier than a genuine coin

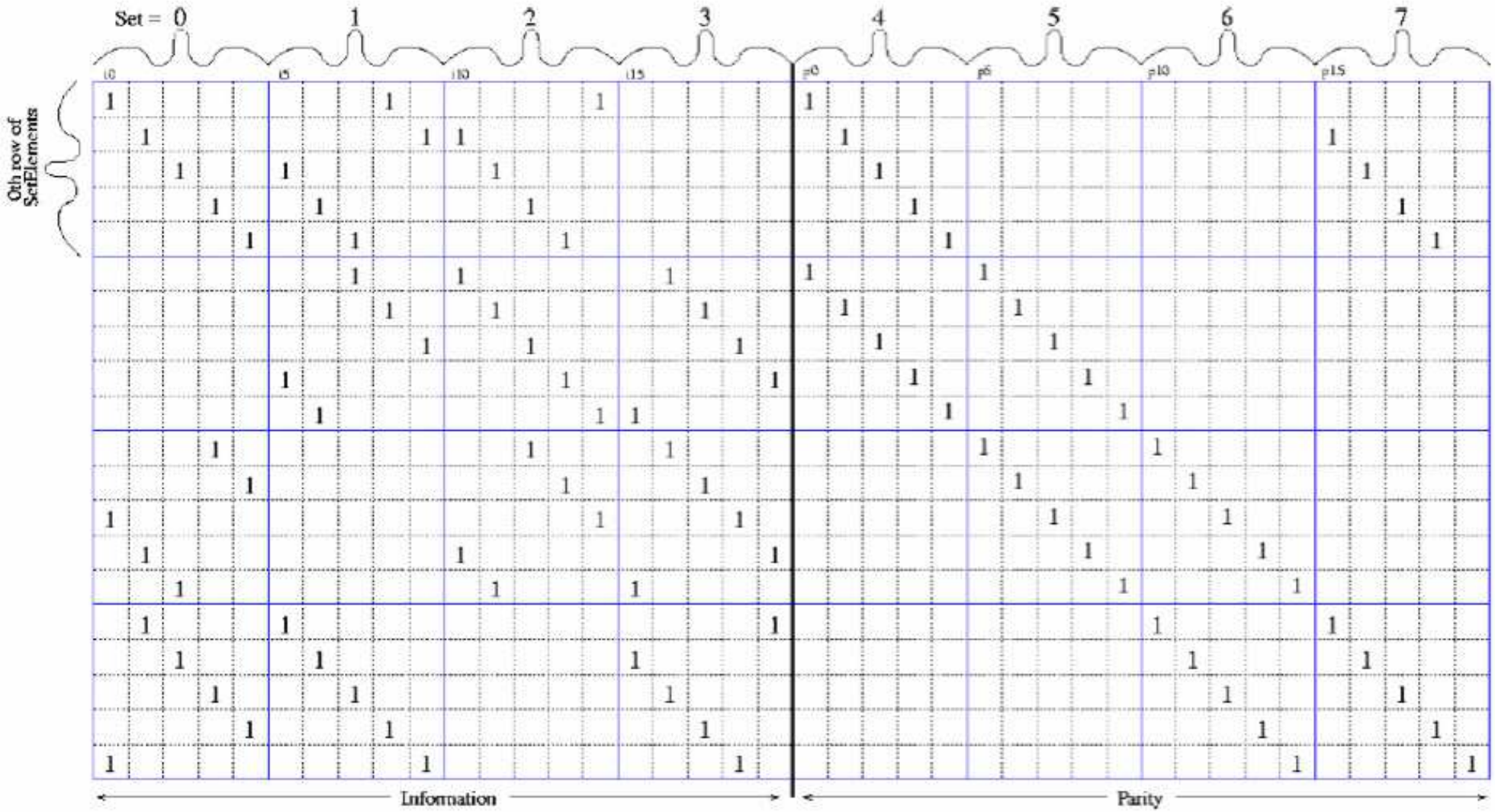
Low-density parity-check code

- a low-density parity-check (LDPC) code, also known as Gallager codes in honor of Robert G. Gallager who developed the LDPC concept, is a linear error correcting code
- An LDPC is constructed using a sparse bipartite graph
- LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close to the Shannon limit
- LDPC codes functionally are defined by a sparse parity-check matrix. This sparse matrix is often randomly generated, subject to the sparsity constraints

Low-density parity-check code

- Using iterative belief propagation techniques, LDPC codes can be decoded in time linear to their block length
- view each parity check that makes up the LDPC as an independent single parity check (SPC) code. Each SPC code is decoded separately using soft-in-soft-out (SISO) techniques
- The soft decision information from each SISO decoding is cross-checked and updated with other redundant SPC decodings of the same information bit, and iterated until a valid code word is achieved
- In a practical LDPC decoder implementation, sets of SPC codes are decoded in parallel to increase throughput

Example LDPC parity-check matrix



Interleaving

- Many communication channels are not memoryless: errors often occur in bursts rather than independently.
- If the number of errors within a code word exceeds the error-correcting code's capability, it fails to recover the original code word.
- Interleaving ameliorates this problem by shuffling source symbols across several code words, thereby creating a more uniform distribution of errors.
- Simple example:

source code	aaaaaabbbbbbccccccddddddeeeeeefffffffgggggghhhhhh
interleaved	abcdefghabcdefghabcdefghabcdefghabcdefghabcdefgh
burst error	abcdefghabc??????bcdefghabcdefghabcdefghabcdefgh
deinterleaved	aa?aaabbbbbbccccccd?dddde?eeeef?ffffg?ggggh?hhhh

Use of interleaving increases total delay as the entire interleaved block must be received before the packets can be decoded

Interleaving Example

- An interleaver of depth m reads m code words of length n.
- Arrange these as an $m \times n$ block: read in by row, read out by column.
- As an example, take m=8 with a Hamming(15,11) code, i.e. n=15
- data is 88 bit, and transmitted is 120 bit
- interleaved block is 8×15
- any burst of errors of length m (8 in this example) or less will result in at most one bit error per code word and therefore can be corrected

Latency and Interleaving

- A feature of the interleaving code is that there is a delay associated with the decode – because the decode can not proceed until a full block has been received – this delay is called the latency.
- The bigger the data block the better the protection against burst error but the longer the latency. The size of the data block is called the interleave depth.
- In some systems the latency does not matter too much for Digital Audio Broadcast (DAB) and if you listen to the same station on FM and DAB you will notice a delay of around 2 s, partly due to the interleaved coding and associated latency that used in DAB.
- For some systems it does matter, for example, GSM, in a mobile phone conversation a latency of more than 330 ms or so is unacceptable because longer delays are confusing for the users. So the interleave depth of the speech channels on the GSM system is limited by this consideration.