Hamming code

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Català
Čeština
Deutsch
Español
Euskara
فارسی
Français
한국어
Italiano
עברית
Монгол
Nederlands
日本語
Polski
Português
Русский
Simple English
Српски / srpski
Svenska
Türkçe
Українська
Tiếng Việt
中文
Edit links

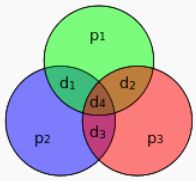
> This article includes a list of references, but **its sources remain unclear** because it has **insufficient inline citations**. Please help to improve this article by introducing more precise citations. *(March 2013)*

In telecommunication, **Hamming codes** are a family of linear error-correcting codes that generalize the Hamming(7,4)-code, and were invented by Richard Hamming in 1950. Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors. By contrast, the simple parity code cannot correct errors, and can detect only an odd number of bits in error. Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance of three.[1]

In mathematical terms, Hamming codes are a class of binary linear codes. For each integer $r \geq 2$ there is a code with block length $n = 2^r - 1$ and message length $k = 2^r - r - 1$. Hence the rate of Hamming codes is $R = k / n = 1 - r / (2^r - 1)$, which is the highest possible for codes with minimum distance of three (i.e., the minimal number of bit changes needed to go from any code word to any other code word is three) and block length $2^r - 1$. The parity-check matrix of a Hamming code is constructed by listing all columns of length r that are non-zero, which means that the dual code of the Hamming code is the punctured Hadamard code. The parity-check matrix has the property that any two columns are pairwise linearly independent.

Due to the limited redundancy that Hamming codes add to the data, they can only detect and correct errors when the error rate is low. This is the case in computer memory (ECC memory), where bit errors are extremely rare and Hamming codes are widely used. In this context, an extended Hamming code having one extra parity bit is often used. Extended Hamming codes achieve a Hamming distance of four, which allows the decoder to distinguish between when at most one one-bit error occurs and when any two-bit errors occur. In this sense, extended Hamming codes are single-error correcting and double-error detecting, abbreviated as **SECDED**.

Binary Hamming Codes



The Hamming(7,4)-code (with $r = 3$)

Named after	Richard W. Hamming
Classification	
Type	Linear block code
Parameters	
Block length	$2^r - 1$ where $r \geq 2$
Message length	$2^r - r - 1$
Rate	$1 - \frac{r}{(2^r - 1)}$
Distance	3
Alphabet size	2
Notation	$[2^r - 1, 2^r - r - 1, 3]_2$-code
Properties	
perfect code	

v · t · e

Contents [hide]

History [edit]

Richard Hamming, the inventor of Hamming codes, worked at Bell Labs in the 1940s on the Bell Model V computer, an electromechanical relay-based machine with cycle times in seconds. Input was fed in on punched cards, which would invariably have read errors. During weekdays, special code would find errors and flash lights so the operators could correct the problem. During after-hours periods and on weekends, when there were no operators, the machine simply moved on to the next job.

Hamming worked on weekends, and grew increasingly frustrated with having to restart his programs from scratch due to the unreliability of the card reader. Over the next few years, he worked on the problem of error-correction, developing an increasingly powerful array of algorithms. In 1950, he published what is now known as Hamming Code, which remains in use today in applications such as ECC memory.

Codes predating Hamming [edit]

A number of simple error-detecting codes were used before Hamming codes, but none were as effective as Hamming codes in the same overhead of space.

Parity [edit]

Main article: Parity bit

Parity adds a single bit that indicates whether the number of ones (bit-positions with values of one) in the preceding data was even or odd. If an odd number of bits is changed in transmission, the message will change parity and the error can be detected at this point; however, the bit that changed may have been the parity bit itself. The most common convention is that a parity value of one indicates that there is an odd number of ones in the data, and a parity value of zero indicates that there is an even number of ones. If the number of bits changed is even, the check bit will be valid and the error will not be detected.

Moreover, parity does not indicate which bit contained the error, even when it can detect it. The data must be discarded entirely and re-transmitted from scratch. On a noisy transmission medium, a successful transmission could take a long time or may never occur. However, while the quality of parity checking is poor, since it uses only a single bit, this method results in the least overhead.

Two-out-of-five code [edit]

Main article: Two-out-of-five code

A two-out-of-five code is an encoding scheme which uses five bits consisting of exactly three 0s and two 1s. This provides ten possible combinations, enough to represent the digits 0–9. This scheme can detect all single bit-errors, all odd numbered bit-errors and some even numbered bit-errors (for example the flipping of both 1-bits). However it still cannot correct for any of these errors.

Repetition [edit]

Main article: Triple modular redundancy

Another code in use at the time repeated every data bit multiple times in order to ensure that it was sent correctly. For instance, if the data bit to be sent is a 1, an $n = 3$ *repetition code* will send 111. If the three bits received are not identical, an error occurred during transmission. If the channel is clean enough, most of the time only one bit will change in each triple. Therefore, 001, 010, and 100 each correspond to a 0 bit, while 110, 101, and 011 correspond to a 1 bit, as though the bits count as "votes" towards what the intended bit is. A code with this ability to reconstruct the original message in the presence of errors is known as an *error-correcting* code. This triple repetition code is a Hamming code with $m = 2$, since there are two parity bits, and $2^2 - 2 - 1 = 1$ data bit.

Such codes cannot correctly repair all errors, however. In our example, if the channel flips two bits and the receiver gets 001, the system will detect the error, but conclude that the original bit is 0, which is incorrect. If we increase the number of times we duplicate each bit to four, we can detect all two-bit errors but cannot correct them (the votes "tie"); at five repetitions, we can correct all two-bit errors, but not all three-bit errors.

Moreover, the repetition code is extremely inefficient, reducing throughput by three times in our original case, and the efficiency drops drastically as we increase the number of times each bit is duplicated in order to detect and correct more errors.

Hamming codes [edit]

If more error-correcting bits are included with a message, and if those bits can be arranged such that different incorrect bits produce different error results, then bad bits could be identified. In a seven-bit message, there are seven possible single bit errors, so three error control bits could potentially specify not only that an error occurred but also which bit caused the error.

Hamming studied the existing coding schemes, including two-of-five, and generalized their concepts. To start with, he developed a nomenclature to describe the system, including the number of data bits and error-correction bits in a block. For instance, parity includes a single bit for any data word, so assuming ASCII words with seven bits, Hamming described this as an *(8,7)* code, with eight bits in total, of which seven are data. The repetition example would be *(3,1)*, following the same logic. The code rate is the second number divided by the first, for our repetition example, 1/3.

Hamming also noticed the problems with flipping two or more bits, and described this as the "distance" (it is now called the *Hamming distance*, after him). Parity has a distance of 2, so one bit flip can be detected, but not corrected and any two bit flips will be invisible. The (3,1) repetition has a distance of 3, as three bits need to be flipped in the same triple to obtain another code word with no visible errors. It can correct one-bit errors or detect but not correct two-bit errors. A (4,1) repetition (each bit is repeated four times) has a distance of 4, so flipping three bits can be detected, but not corrected. When three bits flip in the same group there can be situations where attempting to correct will produce the wrong code word. In general, a code with distance k can detect but not correct $k - 1$ errors.

Hamming was interested in two problems at once; increasing the distance as much as possible, while at the same time increasing the code rate as much as possible. During the 1940s he developed several encoding schemes that were dramatic improvements on existing codes. The key to all of his systems was to have the parity bits overlap, such that they managed to check each other as well as the data.

General algorithm [edit]

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.
    1. Parity bit 1 covers all bit positions which have the least significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.
    2. Parity bit 2 covers all bit positions which have the second least significant bit set: bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.
    3. Parity bit 4 covers all bit positions which have the third least significant bit set: bits 4–7, 12–15, 20–23, etc.
    4. Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 8–15, 24–31, 40–47, etc.
    5. In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

The form of the parity is irrelevant. Even parity is simpler from the perspective of theoretical mathematics, but there is no difference in practice.

This general rule can be shown visually:

Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		**p1**	**p2**	**d1**	**p4**	**d2**	**d3**	**d4**	**p8**	**d5**	**d6**	**d7**	**d8**	**d9**	**d10**	**d11**	**p16**	**d12**	**d13**	**d14**	**d15**	
Parity bit coverage	**p1**	X		X		X		X		X		X		X		X		X		X		
	p2		X	X			X	X			X	X			X	X			X	X		...
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

Shown are only 20 encoded bits (5 parity, 15 data) but the pattern continues indefinitely. The key thing about Hamming Codes that can be seen from visual inspection is that any given bit is included in a unique set of parity bits. To check for errors, check all of the parity bits. The pattern of errors, called the error syndrome, identifies the bit in error. If all parity bits are correct, there is no error. Otherwise, the sum of the positions of the erroneous parity bits identifies the erroneous bit. For example, if the parity bits in positions 1, 2 and 8 indicate an error, then bit 1+2+8=11 is in error. If only one parity bit indicates an error, the parity bit itself is in error.

As you can see, if you have m parity bits, it can cover bits from 1 up to $2^m - 1$. If we subtract out the parity bits, we are left with $2^m - m - 1$ bits we can use for the data. As m varies, we get all the possible Hamming codes:

Parity bits	Total bits	Data bits	Name	Rate
2	3	1	Hamming(3,1) (Triple repetition code)	1/3 ≈ 0.333
3	7	4	Hamming(7,4)	4/7 ≈ 0.571
4	15	11	Hamming(15,11)	11/15 ≈ 0.733
5	31	26	Hamming(31,26)	26/31 ≈ 0.839
...				
m	$2^m - 1$	$2^m - m - 1$	Hamming$(2^m - 1, 2^m - m - 1)$	$(2^m - m - 1)/(2^m - 1)$

If, in addition, an overall parity bit (bit 0) is included, the code can detect (but not correct) any two-bit error, making a SECDED code. The overall parity indicates whether the total number of errors is even or odd. If the basic Hamming code detects an error, but the overall parity says that there are an even number of errors, an uncorrectable 2-bit error has occurred.

Hamming codes with additional parity (SECDED) [edit]

Hamming codes have a minimum distance of 3, which means that the decoder can detect and correct a single error, but it cannot distinguish a double bit error of some codeword from a single bit error of a different codeword. Thus, they can detect double-bit errors only if correction is not attempted.

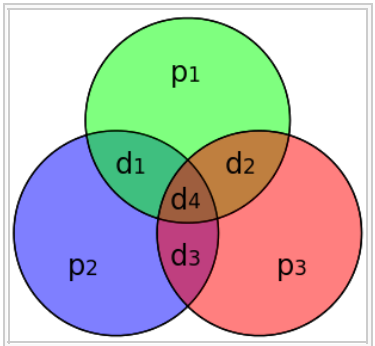
To remedy this shortcoming, Hamming codes can be extended by an extra parity bit. This way, it is possible to increase the minimum distance of the Hamming code to 4, which allows the decoder to distinguish between single bit errors and two-bit errors. Thus the decoder can detect and correct a single error and at the same time detect (but not correct) a double error. If the decoder does not attempt to correct errors, it can detect up to three errors.

This extended Hamming code is popular in computer memory systems, where it is known as *SECDED* (abbreviated from *single error correction, double error detection*). Particularly popular is the (72,64) code, a truncated (127,120) Hamming code plus an additional parity bit, which has the same space overhead as a (9,8) parity code.

[7,4] Hamming code [edit]

Main article: Hamming(7,4)

In 1950, Hamming introduced the [7,4] Hamming code. It encodes four data bits into seven bits by adding three parity bits. It can detect and correct single-bit errors. With the addition of an overall parity bit, it can also detect (but not correct) double-bit errors.



Graphical depiction of the four data bits and three parity bits and which parity bits apply to which data bits

Construction of G and H [edit]

The matrix $\mathbf{G} := \begin{pmatrix} I_k \mid -A^{\mathrm{T}} \end{pmatrix}$ is called a (canonical) generator matrix of a linear (*n*,*k*) code,

and $\mathbf{H} := \begin{pmatrix} A \mid I_{n-k} \end{pmatrix}$ is called a parity-check matrix.

This is the construction of **G** and **H** in standard (or systematic) form. Regardless of form, **G** and **H** for linear block codes must satisfy

$\mathbf{H}\mathbf{G}^{\mathrm{T}} = \mathbf{0}$, an all-zeros matrix.[2]

Since [7, 4, 3] = [*n*, *k*, *d*] = [$2^m - 1$, $2^m - 1 - m$, *m*]. The parity-check matrix **H** of a Hamming code is constructed by listing all columns of length *m* that are pair-wise independent.

Thus **H** is a matrix whose left side is all of the nonzero n-tuples where order of the n-tuples in the columns of matrix does not matter. The right hand side is just the (*n* − *k*)-identity matrix.

So **G** can be obtained from **H** by taking the transpose of the left hand side of **H** with the identity k-identity matrix on the left hand side of **G**.

The code generator matrix $\mathbf{G}$ and the parity-check matrix $\mathbf{H}$ are:

$$\mathbf{G} := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}_{4,7}$$

and

$$\mathbf{H} := \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}_{3,7}.$$

Finally, these matrices can be mutated into equivalent non-systematic codes by the following operations:[2]

- Column permutations (swapping columns)
- Elementary row operations (replacing a row with a linear combination of rows)

Encoding [edit]

Example

From the above matrix we have $2^k = 2^4 = 16$ codewords. The codewords $\vec{x}$ of this binary code can be obtained from $\vec{x} = \vec{a}G$. With $\vec{a} = a_1a_2a_3a_4$ with a_i exist in F_2 (A field with two elements namely 0 and 1).

Thus the codewords are all the 4-tuples (*k*-tuples).

Therefore,

(1,0,1,1) gets encoded as (1,0,1,1,0,1,0).

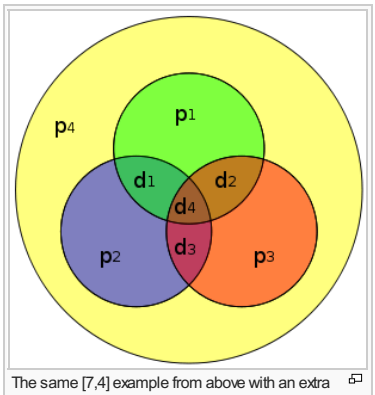
[7,4] Hamming code with an additional parity bit [edit]

The [7,4] Hamming code can easily be extended to an [8,4] code by adding an extra parity bit on top of the (7,4) encoded word (see Hamming(7,4)). This can be summed up with the revised matrices:

$$\mathbf{G} := \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}_{4,8}$$

and

$$\mathbf{H} := \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}_{4,8}.$$



The same [7,4] example from above with an extra parity bit. This diagram is not meant to correspond to the matrix H for this example.

Note that H is not in standard form. To obtain G, elementary row operations can be used to obtain an equivalent matrix to H in systematic form:

$$\mathbf{H} = \left(\begin{array}{cccc|cccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)_{4,8}.$$

For example, the first row in this matrix is the sum of the second and third rows of H in non-systematic form. Using the systematic construction for Hamming codes

from above, the matrix A is apparent and the systematic form of G is written as

$$\mathbf{G} = \left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right)_{4,8}.$$

The non-systematic form of G can be row reduced (using elementary row operations) to match this matrix.

The addition of the fourth row effectively computes the sum of all the codeword bits (data and parity) as the fourth parity bit.

For example, 1011 is encoded into 01100110 where blue digits are data; red digits are parity from the [7,4] Hamming code; and the green digit is the parity added by [8,4] code. The green digit makes the parity of the [7,4] code even.

Finally, it can be shown that the minimum distance has increased from 3, as with the [7,4] code, to 4 with the [8,4] code. Therefore, the code can be defined as [8,4] Hamming code.

See also [edit]

- Coding theory
- Golay code
- Reed–Muller code
- Reed–Solomon error correction
- Turbo code
- Low-density parity-check code
- Hamming bound


Hamming distance
Computer science portal

Notes [edit]

1. ^ See Lemma 12 of
2. ^ [a] [b] Moon T. Error correction coding: Mathematical Methods and Algorithms. John Wiley and Sons, 2005.(Cap. 3) ISBN 978-0-471-64800-0

References [edit]

- Moon, Todd K. (2005). *Error Correction Coding*. New Jersey: John Wiley & Sons. ISBN 978-0-471-64800-0.
- MacKay, David J.C. (September 2003). *Information Theory, Inference and Learning Algorithms*. Cambridge: Cambridge University Press. ISBN 0-521-64298-1.
- D.K. Bhattacharryya, S. Nandi. "An efficient class of SEC-DED-AUED codes". *1997 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '97)*. pp. 410–415. doi:10.1109/ISPAN.1997.645128.
- "Mathematical Challenge April 2013 Error-correcting codes" (PDF). swissQuant Group Leadership Team. April 2013.

External links [edit]

- CGI script for calculating Hamming distances (from R. Tervo, UNB, Canada)

Categories: American inventions | Coding theory | Error detection and correction | Computer arithmetic