Click to verify

Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. In coding theory, the repetition code is one of the most basic linear error-correcting codes. In order to transmit a message over a noisy channel that may corrupt the transmission in a few places, the idea of the repetition code is to just repeat the message several times. The hope is that the channel corrupts only a minority of these repetitions. This way the receiver will notice that a transmission error occurred since the received data stream is not the repetition of a single message, and moreover, the receiver can recover the original message by looking at the received message in the data stream that occurs most often. Because of the bad error correcting performance coupled with the low code rate (ratio between useful information symbols and actual transmitted symbols), other error correction codes are preferred in most cases. The chief attraction of the repetition code is the ease of implementation. Code parameters In the case of a binary repetition code, there exist two code words - all ones and all zeros - which have a length of [math]\displaystyle{ n }[/math]. Therefore, the minimum Hamming distance of the code equals its length [math]\displaystyle{ n }[/math]. This gives the repetition code an error correcting capacity of [math]\displaystyle{ \tfrac{n-1}{2} }[/math] (i.e. it will correct up to [math]\displaystyle{ \tfrac{n-1}{2} }[/math] errors in any code word). If the length of a binary repetition code is odd, then it's a perfect code.[1] The binary repetition code of length n is equivalent to the (n,1)-Hamming code. Example Consider a binary repetition code of length 3. The user wants to transmit the information bits 101. Then the encoding maps each bit either to the all ones or all zeros code word, so we get the 111 000 111, which will be transmitted. Let's say three errors corrupt the transmitted bits and the received sequence is 111 010 100. Decoding is usually done by a simple majority decision for each code word. That lead us to 100 as the decoded information bits, because in the first and second code word occurred less than two errors, so the majority of the bits are correct. But in the third code word two bits are corrupted, which results in an erroneous information bit, since two errors lie above the error correcting capacity. Applications Despite their poor performance as stand-alone codes, use in Turbo code-like iteratively decoded concatenated coding schemes, such as repeat-accumulate (RA) and accumulate-repeat-accumulate (ARA) codes, allows for surprisingly good error correction performance. Repetition codes are one of the few known codes whose code rate can be automatically adjusted to varying channel capacity, by sending more or less parity information as required to overcome the channel noise, and it is the only such code known for non-erasure channels. Practical adaptive codes for erasure channels have been invented only recently, and are known as fountain codes. Some UARTs, such as the ones used in the FlexRay protocol, use a majority filter to ignore brief noise spikes. This spike-rejection filter can be seen as a kind of repetition decoder. See also References ↑ Bossert, Martin (1999). Channel Coding for Telecommunications. Wiley. ISBN 9780471982777. Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. HandWiki Wiki Encyclopedia of Knowledge HandWiki is a wiki encyclopedia for collaborative editing of articles on computing, science, technology and general knowledge. Registered users can post and edit articles, books, manuals and tutorials. Login or request account using the top-right menu. Read more here. Featured Article Carl Jung Synchronicity is a concept first introduced by psychologist Carl G. Jung to describe circumstances that appear meaningfully related yet lack a causal connection. In contemporary research, synchronicity experiences refer to one's subjective experience that coincidences between events in one's mind and the outside world may be causally unrelated to each other yet have some other unknown connection... Read More Did You Know? DesignedWorld is a YouTube channel about unexplained phenomena and strange observations. Submit your favorite remarkable cases related to unexplained, and the channel will create a visual short story. Read more and subscribe You can publish encyclopedic articles and blogs using a decentralized publishing platform called EnHub (enhub.org). The articles can also be saved and downloaded using the ZWI file format. 6174 is the Kaprekar's constant, but it is often called "magic number". At first glance, the number 6174 looks like any other number, but .. Read More Can you hold 70,000 books in one hand?If you are reading this blog, you are a certifiable smart person and probably know about Project Gutenberg (PG)... Read more Advanced PCB Designs for Improved Fault Detection in Electrical InstallationsAdvanced PCB Designs for Improved Fault Detection in Electrical Installations ... Read more Wiki of Our Sponsors DataMelt is a free software for numeric computation, mathematics, statistics, symbolic calculations, data analysis and data visualization. The program runs on Windows/Mac OS/Linux.. Read More.. Mathematics and Computer Science Here are articles on abstract concepts related to computing in a broad sense. Such topics unify foundations of mathematics with computations using computers, and can be applied to many branches of science, finance or technology. Mathematics is an essential tool in computing, but computers are used as a main tool in mathematical problem solving. These topics cover all branches of mathematics, statistics, computer science, data analysis, machine learning and information science. Natural and Formal Sciences The portals "Natural nd Formal Sciences" are dedicated to the study of natural phenomena and physical world. Humanities Here are the portals dedicated to the study of aspects of human society and culture. This group of topics includes arts, practices and procedures used to maintain state of physical, social and mental well-being. Technology, Tools and Finance Here are articles dedicated to various technology topics (building complex devices and machines), software (computer programs and operating systems) and finance (study of money and investments). Companies, Organizations and People Here are articles about organizations, companies and people contributing to research and technology. They serve to support the citations in HandWiki's research articles and books. Books, Monographs, Manuals and Tutorials Here are books, monographs and tutorials. Books and manuals are organized as collections of articles, and typically span many wiki pages. A monograph discusses a detailed study of a single specialized subject. Monographs use a single wiki page. Topical in-depth encyclopedias Here are in-depth encyclopedias organized according the scientific fields supported in HandWiki. Help This section contains the description of the HandWiki encyclopedia project. This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources: "Repetition code" – news · newspapers · books · scholar · JSTOR (May 2021) (Learn how and when to remove this message) In coding theory, the repetition code is one of the most basic linear error-correcting codes. In order to transmit a message over a noisy channel that may corrupt the transmission in a few places, the idea of the repetition code is to just repeat the message several times. The hope is that the channel corrupts only a minority of these repetitions. This way the receiver will notice that a transmission error occurred since the received data stream is not the repetition of a single message, and moreover, the receiver can recover the original message by looking at the received message in the data stream that occurs most often. Because of the bad error correcting performance coupled with the low code rate (ratio between useful information symbols and actual transmitted symbols), other error correction codes are preferred in most cases. The chief attraction of the repetition code is the ease of implementation. In the case of a binary repetition code, there exist two code words - all ones and all zeros - which have a length of n {\displaystyle n} . Therefore, the minimum Hamming distance of the code equals its length n {\displaystyle n} . This gives the repetition code an error correcting capacity of n − 1 2 {\displaystyle {\tfrac {n-1}{2}}} (i.e. it will correct up to n − 1 2 {\displaystyle {\tfrac {n-1}{2}}} errors in any code word). If the length of a binary repetition code is odd, then it's a perfect code.[1] The binary repetition code of length n is equivalent to the (n, 1)-Hamming code. A (n, 1) BCH code is also a repetition code. Consider a binary repetition code of length 3. The user wants to transmit the information bits 101. Then the encoding maps each bit either to the all ones or all zeros code word, so we get the 111 000 111, which will be transmitted. Let's say three errors corrupt the transmitted bits and the received sequence is 111 010 100. Decoding is usually done by a simple majority decision for each code word. That lead us to 100 as the decoded information bits, because in the first and second code word occurred less than two errors, so the majority of the bits are correct. But in the third code word two bits are corrupted, which results in an erroneous information bit, since two errors lie above the error correcting capacity. Despite their poor performance as stand-alone codes, use in Turbo code-like iteratively decoded concatenated coding schemes, such as repeat-accumulate (RA) and accumulate-repeat-accumulate (ARA) codes, allows for surprisingly good error correction performance. Repetition codes are one of the few known codes whose code rate can be automatically adjusted to varying channel capacity, by sending more or less parity information as required to overcome the channel noise, and it is the only such code known for non-erasure channels. Practical adaptive codes for erasure channels have been invented only recently, and are known as fountain codes. Some UARTs, such as the ones used in the FlexRay protocol, use a majority filter to ignore brief noise spikes. This spike-rejection filter can be seen as a kind of repetition decoder. Block code Turbo code ^ Bossert, Martin (1999). Channel Coding for Telecommunications. Wiley. ISBN 9780471982777. Retrieved from " This is a preview for the new MathML rendering mode (with SVG fallback), which is availble in production for registered users. If you would like use the MathML rendering mode, you need a wikipedia user account that can be registered here [[1]] Only registered users will be able to execute this rendering mode. Note: you need not enter a email address (nor any other private information). Please do not use a password that you use elsewhere. Registered users will be able to choose between the following three rendering modes: MathML E=mc2 Follow this link to change your Math rendering settings. You can also add a Custom CSS to force the MathML/SVG rendering or select different font families. See these examples. Demos Here are some demos: accessibility: Safari + VoiceOver: video only, File:Voiceover-mathml-example-1.wav, File:Voiceover-mathml-example-2.wav, File:Voiceover-mathml-example-3.wav, File:Voiceover-mathml-example-4.wav, File:Voiceover-mathml-example-5.wav, File:Voiceover-mathml-example-6.wav, File:Voiceover-mathml-example-7.wav Internet Explorer + MathPlayer (audio) Internet Explorer + MathPlayer (synchronized highlighting) Internet Explorer + MathPlayer (braille) NVDA+MathPlayer: File:Nvda-mathml-example-1.wav, File:Nvda-mathml-example-2.wav, File:Nvda-mathml-example-3.wav, File:Nvda-mathml-example-4.wav, File:Nvda-mathml-example-5.wav, File:Nvda-mathml-example-6.wav, File:Nvda-mathml-example-7.wav. Orca: There is ongoing work, but no support at all at the moment File:Orca-mathml-example-1.wav, File:Orca-mathml-example-2.wav, File:Orca-mathml-example-3.wav, File:Orca-mathml-example-4.wav, File:Orca-mathml-example-5.wav, File:Orca-mathml-example-6.wav, File:Orca-mathml-example-7.wav. From our testing, ChromeVox and JAWS are not able to read the formulas generated by the MathML mode. Test pages To test the MathML, PNG, and source rendering modes, please go to one of the following test pages: Displaystyle MathAxisAlignment Styling Linebreaking Unique Ids Help:Formula Inputtypes (private Wikis only) Url2Image (private Wikis only) Bug reporting If you find any bugs, please report them at Bugzilla, or write an email to math_bugs (at) ckurs (dot) de . Not to be confused with Huffman code. Family of linear error-correcting codes This article includes a list of general references, but it lacks sufficient corresponding inline citations. Please help to improve this article by introducing more precise citations. (March 2013) (Learn how and when to remove this message) Binary Hamming codesThe Hamming(7,4) code (with r = 3)Named afterRichard W. HammingClassificationTypeLinear block codeBlock length2r − 1 where r ≥ 2Message length2r − r − 1Rate1 − r/(2r − 1)Distance3Alphabet size2Notation[2r − 1, 2r − r − 1, 3]2-codePropertiesperfect codevte In computer science and telecommunications, Hamming codes are a family of linear error-correcting codes. Hamming codes can detect one-bit and 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] Richard W. Hamming invented Hamming codes in 1950 as a way of automatically correcting errors introduced by punched card readers. In his original paper, Hamming elaborated his general idea, but specifically focused on the Hamming(7,4) code which adds three parity bits to four bits of data.[2] In mathematical terms, Hamming codes are a class of binary linear code. For each integer r ≥ 2 there is a code-word with block length n = 2r − 1 and message length k = 2r − r − 1. Hence the rate of Hamming codes is R = k / n = 1 − r / (2r − 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 2r − 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 shortened Hadamard code, also known as a Simplex 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 (usually RAM), where bit errors are extremely rare and Hamming codes are widely used, and a RAM with this correction system is an ECC RAM (ECC memory). 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. Richard Hamming, the inventor of Hamming codes, worked at Bell Labs in the late 1940s on the Bell Model V computer, an electromechanical relay-based machine with cycle times in seconds. Input was fed in on punched paper tape, seven-eighths of an inch wide, which had up to six holes per row. During weekdays, when errors in the relays were detected, the machine would stop and flash lights so that 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 detected errors. In a taped interview, Hamming said, "And so I said, 'Damn it, if the machine can detect an error, why can't it locate the position of the error and correct it?'".[3] 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. 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. 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. 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 (5 3) = 10 {\displaystyle {\binom {5}{3}}=10} 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 any of these errors. 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, with the greater quantity of digits that are the same ('0' or a '1') indicating what the data bit should be. 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 22 − 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 size of the bit string to four, we can detect all two-bit errors but cannot correct them (the quantity of parity bits is even); at five bits, we can both detect and correct all two-bit errors, but not all three-bit errors. Moreover, increasing the size of the parity bit string is 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. 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 it can 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. The following general algorithm generates a single-error correcting (SEC) code for any number of bits. The main idea is to choose the error-correcting bits such that the index-XOR (the XOR of all the bit positions containing a 1) is 0. We use positions 1, 10, 100, etc. (in binary) as the error-correcting bits, which guarantees it is possible to set the error-correcting bits so that the index-XOR of the whole message is 0. If the receiver receives a string with index-XOR 0, they can conclude there were no corruptions, and otherwise, the index-XOR indicates the index of the corrupted bit. An algorithm can be deduced from the following description: Number the bits starting from 1: bit 1, 2, 3, 4, 5, 6, 7, etc. Write the bit numbers in binary: 1, 10, 11, 100, 101, 110, 111, etc. All bit positions that are powers of two (have a single 1 bit in the binary form of their position) are parity bits: 1, 2, 4, 8, etc. (1, 10, 100, 1000) All other bit positions, with two or more 1 bits in the binary form of their position, are data bits. 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. 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. Parity bit 2 covers all bit positions which have the second least significant bit set: bits 2–3, 6–7, 10–11, etc. Parity bit 4 covers all bit positions which have the third least significant bit set: bits 4–7, 12–15, 20–23, etc. Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 8–15, 24–31, 40–47, etc. In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero. If a byte of data to be encoded is 10011010, then the data word (using _ to represent the parity bits) would be __1_001_1010, and the code word is 011100101010. The choice of the parity, even or odd, is irrelevant but the same choice must be used for both encoding and decoding. 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 Paritybitcoverage p1 p2 p4 p8 p16 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. With m parity bits, bits from 1 up to 2 m − 1 {\displaystyle 2^{m}-1} can be covered. After discounting the parity bits, 2 m − m − 1 {\displaystyle 2^{m}-m-1} bits remain for use as 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 6 63 57 Hamming(63,57) 57/63 ≈ 0.905 7 127 120 Hamming(127,120) 120/127 ≈ 0.945 8 255 247 Hamming(255,247) 247/255 ≈ 0.969 9 511 502 Hamming(511,502) 502/511 ≈ 0.982 ... m n = 2 m − 1 {\displaystyle n=2^{m}-1} k = 2 m − m − 1 {\displaystyle k=2^{m}-m-1} Hamming (2 m − 1 , 2 m − m − 1) {\displaystyle (2^{m}-1,2^{m}-m-1)} (2 m − m − 1) / (2 m − 1) {\displaystyle (2^{m}-m-1)/(2^{m}-1)} 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, some double-bit errors will be incorrectly decoded as if they were single bit errors and therefore go undetected, unless no correction is 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 reliably detect triple bit errors. If the decoder does correct errors, some triple errors will be mistaken for single errors and "corrected" to the wrong value. Error correction is therefore a trade-off between certainty (the ability to reliably detect triple bit errors) and resiliency (the ability to keep functioning in the face of single bit errors). This extended Hamming code was popular in computer memory systems, starting with IBM 7030 Stretch in 1961,[4] where it is known as SECDED (or SEC-DED, abbreviated from single error correction, double error detection).[5] Server computers in 21st century, while typically keeping the SECDED level of protection, no longer use Hamming's method, relying instead on the designs with longer codewords (128 to 256 bits of data) and modified balanced parity-check trees.[4] The (72,64) Hamming code is still popular in some hardware designs, including Xilinx FPGA families.[4] Main article: Hamming(7,4) Graphical depiction of the four data bits and three parity bits and which parity bits apply to which data bits In 1950, Hamming introduced the [7,4] Hamming code. It encodes four data bits into seven bits by adding three parity bits. As explained earlier, it can either detect and correct single-bit errors or it can detect (but not correct) both single and double-bit errors. With the addition of an overall parity bit, it becomes the [8,4] extended Hamming code and can both detect and correct single-bit errors and detect (but not correct) double-bit errors. The matrix G := (I k − A T) {\displaystyle \mathbf {G} :={\begin{pmatrix} {\begin{array}{c|c}I_{k}&-A^{\text{T}}\\\end{array}}\end{pmatrix}}} is called a (canonical) generator matrix of a linear (n,k) code, and H := (A I n − k) {\displaystyle \mathbf {H} :={\begin{pmatrix}{\begin{array}{c|c}A&I_{n-k}\\\end{array}}\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 H G T = 0 {\displaystyle \mathbf {H} \,\mathbf {G} ^{\text{T}}=\mathbf {0} } , an all-zeros matrix.[6] Since [7, 4, 3] = [n, k, d] = [2m − 1, 2m − 1 − m, 3]. 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 G {\displaystyle \mathbf {G} } and the parity-check matrix H {\displaystyle \mathbf {H} } are: G := (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) 4 , 7 {\displaystyle \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 H := (1 1 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1