




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   
Čeština  
Deutsch  
Español  
Français  
한국어  
日本語  
Polski  
Português  
Русский  
Українська  
中文


 Edit links

Create account Log in

Article Talk

Read Edit

More▼

Search 

# Golomb coding

From Wikipedia, the free encyclopedia  
(Redirected from [Rice coding](#))

**Golomb coding** is a [lossless data compression](#) method using a family of [data compression](#) codes invented by [Solomon W. Golomb](#) in the 1960s. Alphabets following a [geometric distribution](#) will have a Golomb code as an optimal [prefix code](#), making Golomb coding highly suitable for situations in which the occurrence of small values in the input stream is significantly more likely than large values.

**Rice coding** (invented by [Robert F. Rice](#)) denotes using a subset of the family of Golomb codes to produce a simpler (but possibly suboptimal) prefix code. Rice used this set of codes in an [adaptive coding](#) scheme; "Rice coding" can refer either to that adaptive scheme or to using that subset of Golomb codes. Whereas a Golomb code has a tunable parameter that can be any positive integer value, Rice codes are those in which the tunable parameter is a power of two. This makes Rice codes convenient for use on a computer, since multiplication and division by 2 can be implemented more efficiently in [binary arithmetic](#).

Rice coding is used as the [entropy encoding](#) stage in a number of lossless [image compression](#) and [audio data compression](#) methods.

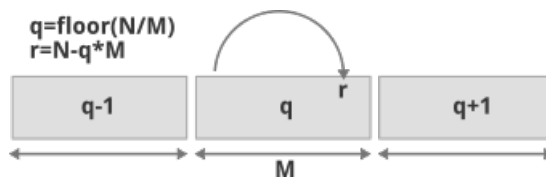
## Contents [hide]

- Overview
  - Construction of codes
  - Use with signed integers
- Simple algorithm
- Example
- Use for run-length encoding
- Applications
- References

## Overview [edit]

### Construction of codes [edit]

Golomb coding uses a tunable parameter *M* to divide an input value *N* into two parts: *q*, the result of a division by *M*, and *r*, the remainder. The quotient is sent in [unary coding](#), followed by the remainder in [truncated binary encoding](#). When *M* = 1 Golomb coding is equivalent to unary coding.



Golomb–Rice codes can be thought of as codes that indicate a number by the position of the *bin* (*q*), and the *offset* within the bin (*r*). The above figure shows the position *q*, and offset *r* for the encoding of integer *N* using Golomb–Rice parameter *M*.

Formally, the two parts are given by the following expression, where *x* is the number being encoded:

$$q = \left\lfloor \frac{(x-1)}{M} \right\rfloor \text{ and } r = x - qM - 1 \text{ The final result looks like: } (q+1)r$$

Note that *r* can be of a varying number of bits, and is specifically only *b* bits for Rice code, and switches between *b*-1 and *b* bits for Golomb code (i.e. *M* is not a power of 2): Let *b* =  $\lceil \log_2(M) \rceil$ . If

$0 \leq r < 2^b - M$ , then use *b*-1 bits to encode *r*. If  $2^b - M \leq r < M$ , then use *b* bits to encode *r*. Clearly, *b* =  $\log_2(M)$  if *M* is a power of 2 and we can encode all values of *r* with *b* bits.

The parameter *M* is a function of the corresponding [Bernoulli process](#), which is parameterized by  $p = P(X = 0)$  the probability of success in a given [Bernoulli trial](#). *M* is either the median of the distribution or the median +/- 1. It can be determined by these inequalities:

$$(1-p)^M + (1-p)^{M+1} \leq 1 < (1-p)^{M-1} + (1-p)^M.$$

Golomb states that, for large  $M$ , there is very little penalty for picking  $M = \left\lceil \frac{-1}{\log_2(1-p)} \right\rceil$ .

The Golomb code for this distribution is equivalent to the [Huffman code](#) for the same probabilities, if it were possible to compute the Huffman code.

### Use with signed integers [\[edit\]](#)

Golomb's scheme was designed to encode sequences of non-negative numbers. However it is easily extended to accept sequences containing negative numbers using an *overlap and interleave* scheme, in which all values are reassigned to some positive number in a unique and reversible way. The sequence begins: 0, -1, 1, -2, 2, -3, 3, -4, 4 ... The  $n^{\text{th}}$  negative value (i.e.,  $-n$ ) is mapped to the  $n^{\text{th}}$  odd number  $(2n-1)$ , and the  $m^{\text{th}}$  positive value is mapped to the  $m^{\text{th}}$  even number  $(2m)$ . This may be expressed mathematically as follows: a positive value  $x$  is mapped to  $(x' = 2|x| = 2x, x \geq 0)$ , and a negative value  $y$  is mapped to  $(y' = 2|y| - 1 = -2y - 1, y < 0)$ . This is an optimal prefix code only if both the positive and the magnitude of the negative values follow a geometric distribution with the same parameter.

### Simple algorithm [\[edit\]](#)

Note below that this is the Rice–Golomb encoding, where the remainder code uses simple truncated binary encoding, also named "Rice coding" (other varying-length binary encodings, like arithmetic or Huffman encodings, are possible for the remainder codes, if the statistic distribution of remainder codes is not flat, and notably when not all possible remainders after the division are used). In this algorithm, if the  $M$  parameter is a power of 2, it becomes equivalent to the simpler Rice encoding.

1. Fix the parameter  $M$  to an integer value.
2. For  $N$ , the number to be encoded, find
  1. quotient =  $q = \text{int}[N/M]$
  2. remainder =  $r = N \bmod M$
3. Generate Codeword
  1. The Code format : <Quotient Code><Remainder Code>, where
  2. Quotient Code (in [unary coding](#))
    1. Write a  $q$ -length string of 1 bits
    2. Write a 0 bit
  3. Remainder Code (in [truncated binary encoding](#))
    1. If  $M$  is power of 2, code remainder as binary format. So  $\log_2(M)$  bits are needed. (Rice code)
    2. If  $M$  is not a power of 2, set  $b = \lceil \log_2(M) \rceil$ 
      1. If  $r < 2^b - M$  code  $r$  as plain binary using  $b-1$  bits.
      2. If  $r \geq 2^b - M$  code the number  $r + 2^b - M$  in plain binary representation using  $b$  bits.

### Example [\[edit\]](#)

Set  $M = 10$ . Thus  $b = \lceil \log_2(10) \rceil = 4$ . The cutoff is  $2^b - M = 16 - 10 = 6$

| Encoding of quotient part |             | Encoding of remainder part |        |        |             |
|---------------------------|-------------|----------------------------|--------|--------|-------------|
| $q$                       | output bits | $r$                        | offset | binary | output bits |
| 0                         | 0           | 0                          | 0      | 0000   | 000         |
| 1                         | 10          | 1                          | 1      | 0001   | 001         |
| 2                         | 110         | 2                          | 2      | 0010   | 010         |
| 3                         | 1110        | 3                          | 3      | 0011   | 011         |
| 4                         | 11110       | 4                          | 4      | 0100   | 100         |
| 5                         | 111110      | 5                          | 5      | 0101   | 101         |
| 6                         | 1111110     | 6                          | 12     | 1100   | 1100        |
| ⋮                         | ⋮           | 7                          | 13     | 1101   | 1101        |

|   |                                   |   |    |      |      |
|---|-----------------------------------|---|----|------|------|
| N | $\underbrace{111 \cdots 111}_N 0$ | 8 | 14 | 1110 | 1110 |
|   |                                   | 9 | 15 | 1111 | 1111 |

For example, with a Rice–Golomb encoding of parameter  $M = 10$ , the decimal number 42 would first be split into  $q = 4, r = 2$ , and would be encoded as  $\text{qcode}(q), \text{rcode}(r) = \text{qcode}(4), \text{rcode}(2) = 11110, 010$  (you don't need to encode the separating comma in the output stream, because the 0 at the end of the  $q$  code is enough to say when  $q$  ends and  $r$  begins ; both the qcode and rcode are self-delimited).

### Use for run-length encoding [\[edit\]](#)

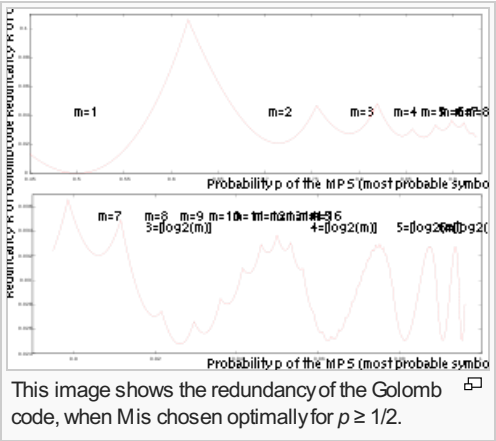
Given an alphabet of two symbols, or a set of two events,  $P$  and  $Q$ , with probabilities  $p$  and  $(1 - p)$  respectively, where  $p \geq 1/2$ , Golomb coding can be used to encode runs of zero or more  $P$ 's separated by single  $Q$ 's. In this application, the best setting of the parameter  $M$  is the nearest integer to  $\frac{-1}{\log_2 p}$ . When  $p = 1/2$ ,  $M = 1$ , and the Golomb code corresponds to unary ( $n \geq 0$   $P$ 's followed by a  $Q$  is encoded as  $n$  ones followed by a zero).

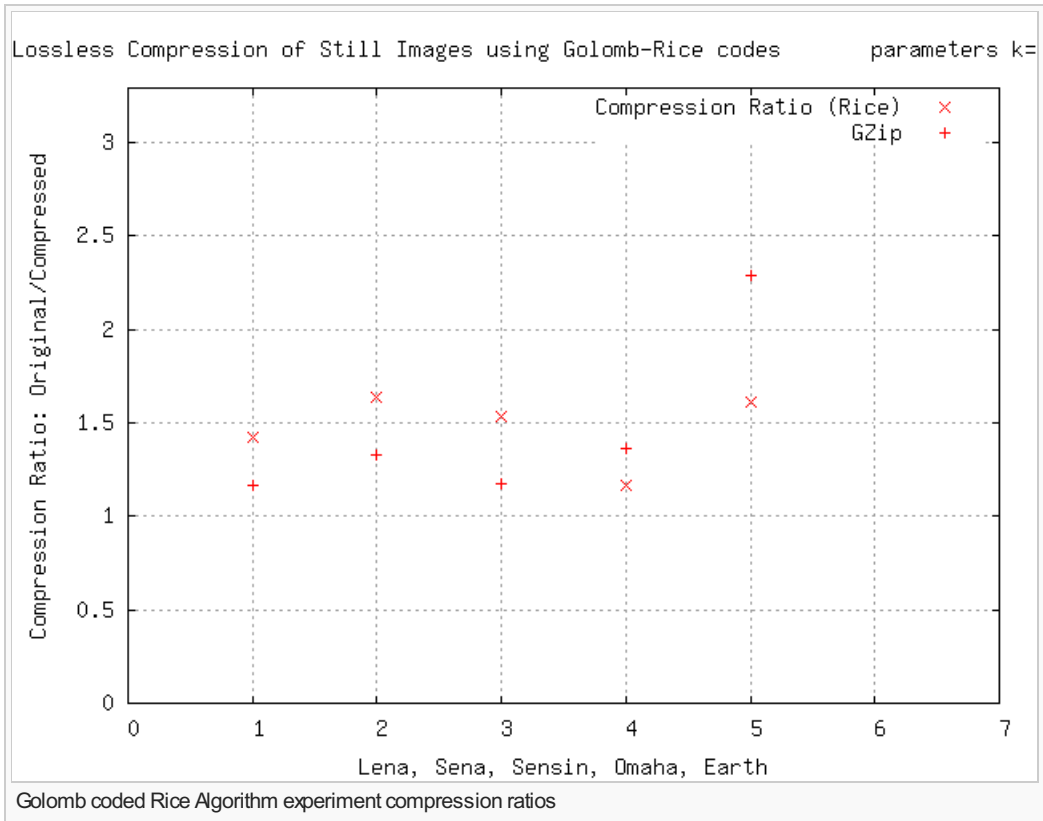
### Applications [\[edit\]](#)

Numerous signal codecs use a Rice code for [prediction](#) residues. In predictive algorithms, such residues tend to fall into a two-sided [geometric distribution](#), with small residues being more frequent than large residues, and the Rice code closely approximates the Huffman code for such a distribution without the overhead of having to transmit the Huffman table. One signal that does not match a geometric distribution is a [sine wave](#), because the differential residues create a sinusoidal signal whose values are not creating a geometric distribution (the highest and lowest residue values have similar high frequency of occurrences, only the median positive and negative residues occur less often).

Several lossless [audio codecs](#), such as [Shorten](#),<sup>[1]</sup> [FLAC](#),<sup>[2]</sup> [Apple Lossless](#), and [MPEG-4 ALS](#), use a Rice code after the [linear prediction step](#) (called "adaptive FIR filter" in Apple Lossless). Rice coding is also used in the [FELICS](#) lossless image codec.

The Golomb–Rice coder is used in the entropy coding stage of [Rice Algorithm](#) based *lossless image codecs*. One such experiment yields a compression ratio graph given below. See other entries in this category at the bottom of this page. in those compression, the progressive space differential data yields an alternating suite of positive and negative values around 0, which are remapped to positive-only integers (by doubling the absolute value and adding one if the input is negative), and then Rice–Golomb coding is applied by varying the divisor which remains small.<sup>[*citation needed*]</sup>





In those results, the Rice coding may create very long sequences of one-bits for the quotient; for practical reasons, it is often necessary to limit the total run-length of one-bits, so a modified version of the Rice–Golomb encoding consists of replacing the long string of one-bits by encoding its length with a recursive Rice–Golomb encoding; this requires reserving some values in addition to the initial divisor  $k$  to allow the necessary distinction.

The [JPEG-LS](#) scheme uses Rice–Golomb to encode the prediction residuals.

## References [\[edit\]](#)

- Golomb, S.W. (1966). , [Run-length encodings](#). *IEEE Transactions on Information Theory*, *IT--12*(3):399--401
- R. F. Rice (1971) and R. Plaunt, , ["Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data"](#), *IEEE Transactions on Communications*, vol. 16(9), pp. 889–897, Dec. 1971.
- R. F. Rice (1979), , ["Some Practical Universal Noiseless Coding Techniques"](#), *Jet Propulsion Laboratory, Pasadena, California, JPL Publication 79—22*, Mar. 1979.
- Witten, Ian Moffat, Alistair Bell, Timothy. ["Managing Gigabytes: Compressing and Indexing Documents and Images."](#) Second Edition. Morgan Kaufmann Publishers, San Francisco CA. 1999 [ISBN 1-55860-570-3](#)
- David Salomon. ["Data Compression"](#),[ISBN 0-387-95045-1](#).
- S. Büttcher, C. L. A. Clarke, and G. V. Cormack. [Information Retrieval: Implementing and Evaluating Search Engines](#) . MIT Press, Cambridge MA, 2010.
- 1  ^ man shorten
- 2  ^ FLAC documentation: format overview

| v · t · e |                 | Data compression methods   | <span>[hide]</span> |
|-----------|-----------------|--|---------------------|
| Lossless  | Entropy type    | Unary · Arithmetic · <b>Golomb</b> · Huffman (Adaptive · Canonical · Modified) · Range · Shannon · Shannon–Fano · Shannon–Fano–Elias · Tunstall · Universal (Exp-Golomb · Fibonacci · Gamma · Levenshtein) |                     |
|           | Dictionary type | Byte pair encoding · DEFLATE · Lempel–Ziv (LZ77 / LZ78 (LZ1 / LZ2) · LZJB · LZMA · LZO · LZRW · LZS · LZSS · LZW · LZWL · LZX · LZ4 · Statistical)   |                     |
|           | Other types     | BWT · CTW · Delta · DMC · MTF · PAQ · PPM · RLE  |                     |
| Audio     | Concepts        | Bit rate (average (ABR) · constant (CBR) · variable (VBR)) · Companding · Convolution · Dynamic range · Latency · Nyquist–Shannon theorem · Sampling · Sound quality · Speech coding · Sub-band coding     |                     |
|           | Codec parts     | A-law · <span>μ</span> -law · ACELP · ADPCM · CELP · DPCM · Fourier transform · LPC (LAR · LSP) · MDCT · Psychoacoustic model · WLPc   |                     |
| Image     | Concepts        | Chroma subsampling · Coding tree unit · Color space · Compression artifact · Image resolution · Macroblock · Pixel · PSNR · Quantization · Standard test image   |                     |

|   |  |  |
|---|--|--|
|   | Methods  | Chain code · DCT · EZW · Fractal · KLT · LP · RLE · SPIHT · Wavelet  |
| Video   | Concepts   | Bit rate (average (ABR) · constant (CBR) · variable (VBR)) · Display resolution · Frame · Frame rate · Frame types · Interlace · Video characteristics · Video quality |
|   | Codec parts  | Lapped transform · DCT · Deblocking filter · Motion compensation   |
| Theory  | Entropy · Kolmogorov complexity · Lossy · Quantization · Rate–distortion · Redundancy · Timeline of information theory |  |
| 📄 Compression formats · 📄 Compression software (codecs) |  |  |

Categories: [Lossless compression algorithms](#)

This page was last modified on 13 August 2015, at 14:07.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

