



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](#)
[فارسی](#)
[中文](#)

 [Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Adaptive Huffman coding

From Wikipedia, the free encyclopedia

Adaptive Huffman coding (also called **Dynamic Huffman coding**) is an [adaptive coding](#) technique based on [Huffman coding](#). It permits building the code as the symbols are being transmitted, having no initial knowledge of source distribution, that allows one-pass encoding and adaptation to changing conditions in data.

The benefit of one-pass procedure is that the source can be encoded in real time, though it becomes more sensitive to transmission errors, since just a single loss ruins the whole code.

Contents [hide]

- 1 Algorithms
 - 1.1 FGK Algorithm
 - 1.2 Vitter algorithm
 - 1.2.1 Example
- 2 References
- 3 External links

Algorithms [\[edit\]](#)

There are a number of implementations of this method, the most notable are **FGK** ([Faller-Gallager-Knuth](#)) and **Vitter** algorithm.

FGK Algorithm [\[edit\]](#)

It is an online coding technique based on Huffman coding. Having no initial knowledge of occurrence frequencies, it permits dynamically adjusting the Huffman's tree as data are being transmitted. In a FGK Huffman tree, a special external node, called *0-node*, is used to identify a newly-coming character. That is, whenever a new data is encountered, output the path to the 0-node followed by the data. For a past-coming character, just output the path of the data in the current Huffman's tree. Most importantly, we have to adjust the FGK Huffman tree if necessary, and finally update the frequency of related nodes. As the frequency of a datum is increased, the *sibling property* of the Huffman's tree may be broken. The adjustment is triggered for this reason. It is accomplished by consecutive swappings of nodes, subtrees, or both. The data node is swapped with the highest-ordered node of the same frequency in the Huffman's tree, (or the subtree rooted at the highest-ordered node). All ancestor nodes of the node should also be processed in the same manner.

Since the FGK Algorithm has some drawbacks about the node-or-subtree swapping, Vitter proposed another algorithm to improve it.

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

Code is represented as a tree structure in which every node has a corresponding weight and a unique number.

Numbers go down, and from right to left.

Weights must satisfy the sibling property, which states that nodes must be listed in the order of decreasing weight with each node adjacent to its sibling. Thus if A is the parent node of B and C is a child of B, then

$$W(A) > W(B) > W(C)$$

The weight is merely the count of symbols transmitted which codes are associated with children of that node.

A set of nodes with same weights make a **block**.

To get the code for every node, in case of binary tree we could just traverse all the path from the root to the node, writing down (for example) "1" if we go to the right and "0" if we go to the left.

We need some general and straightforward method to transmit symbols that are "not yet transmitted" (NYT). We could use, for example, transmission of binary numbers for every symbol in alphabet.

Encoder and decoder start with only the root node, which has the maximum number. In the beginning it is our initial NYT node.

When we transmit an NYT symbol, we have to transmit code for the NYT node, then for its generic code.

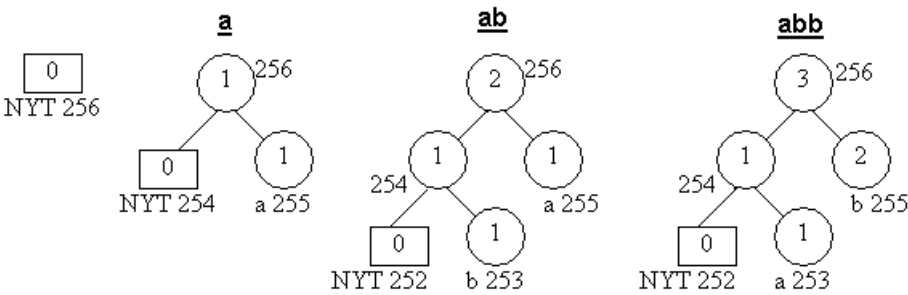
For every symbol that is already in the tree, we only have to transmit code for its leaf node.

For every symbol transmitted both the transmitter and receiver execute the update procedure:

1. If current symbol is NYT, add two child nodes to NYT node. One will be a new NYT node the other is a leaf node for our symbol. Increase weight for the new leaf node and the old NYT and go to step 4. If current symbol is not NYT, go to symbol's leaf node.
2. If this node does not have the highest number in a block, swap it with the node having the highest number, except if that node is its parent^[1]
3. Increase weight for current node
4. If this is not the root node go to parent node then go to step 2. If this is the root, end.

Note: swapping nodes means swapping weights and corresponding symbols, but not the numbers.

Example [\[edit\]](#)



Start with an empty tree.

For "a" transmit its binary code.

NYT spawns two child nodes: 254 and 255. Increase weight for root. Code for "a", associated with node 255, is 1.

For "b" transmit 0 (for NYT node) then its binary code.

NYT spawns two child nodes: 252 for NYT and 253 for leaf node. Increase weights for 253, 254, and root. Code for "b" is 01.

For the second "b" transmit 01.

Go to that leaf node, 253. We have a block of weights of 1 and the biggest number in the block is 255, so swap the weights and symbols of nodes 253 and 255, increase weight, go to root, increase weight for root.

Future code for "b" is 1, and for "a" is now 01, which reflects their frequency.



References [\[edit\]](#)

1. [^] "Adaptive Huffman Coding". [Cs.duke.edu](#). Retrieved 2012-02-26.
- Vitter's original paper: J. S. Vitter, "Design and Analysis of Dynamic Huffman Codes", [Journal of the ACM](#), 34(4), October 1987, pp 825–845.
- J. S. Vitter, "ALGORITHM 673 Dynamic Huffman Coding", [ACM Transactions on Mathematical Software](#), 15(2), June 1989, pp 158–167. Also appears in [Collected Algorithms of ACM](#).
- Donald E. Knuth, "Dynamic Huffman Coding", [Journal of Algorithm](#), 6(2), 1985, pp 163–180.

External links [\[edit\]](#)

- Black, Paul E. "adaptive Huffman coding". *Dictionary of Algorithms and Data Structures*. NIST.
- University of California Dan Hirschberg site
- Cardiff University Dr. David Marshall site
- C implementation of Vitter algorithm
- Excellent description from Duke University

| v · t · e | | Data compression methods | [hide] |
|-----------|-----------------|---|------------------------|
| Lossless | Entropy type | Unary · Arithmetic · Golomb · 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 · μ -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 7 January 2015, at 10:40.

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](#)

