



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

 [Add links](#)

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

Article [Talk](#)

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



Package-merge algorithm

From Wikipedia, the free encyclopedia

The **package-merge algorithm** is an $O(nL)$ -time algorithm for finding an optimal [length-limited Huffman code](#) for a given distribution on a given alphabet of size n , where no [code word](#) is longer than L . It is a [greedy algorithm](#), and a generalization of [Huffman's original algorithm](#). Package-merge works by reducing the code construction problem to the binary **coin collector's problem**.^[1]

Contents [hide]

- [1 The coin collector's problem](#)
- [2 Description of the package-merge algorithm](#)
- [3 Reduction of length-limited Huffman coding to the coin collector's problem](#)
- [4 Performance improvements and generalizations](#)
- [5 External links](#)
- [6 References](#)

The coin collector's problem [\[edit\]](#)

Suppose a coin collector has a number of coins of various denominations, each of which has a [numismatic value](#). The coin collector has run out of money and needs to use some of his coin collection to buy something of cost N . He wishes to select a subset of coins from his collection of minimum numismatic value whose denominations total N .

The binary version of this problem is that all denominations are powers of 2, that is, 1, 1/2, 1/4, etc. dollars.

Description of the package-merge algorithm [\[edit\]](#)

Assume that the largest denomination is 1 dollar, and that N is an integer. (The algorithm works even if these assumptions do not hold, by trivial modifications.) The coin collector first separates his coins into lists, one for each denomination, sorted by numismatic value. He then **packages** the smallest denomination coins in pairs, starting from the pair of smallest total numismatic value. If there is one coin left over, it will be the coin of highest numismatic value of that denomination, and it is set aside and ignored henceforth. These packages are then **merged** into the list of coins of the next smallest denomination, again in order of numismatic value. The items in that list are then **packaged** in pairs, and merged into the next smallest list, and so forth.

Finally, there is a list of items, each of which is a 1 dollar coin or a package consisting of two or more smaller coins whose denominations total 1 dollar. They are also sorted in order of numismatic value. The coin collector then selects the least value N of them.

Note that the time of the algorithm is linear in the number of coins.

Reduction of length-limited Huffman coding to the coin collector's problem

[\[edit\]](#)

Let L be the maximum length any code word is permitted to have. Let p_1, \dots, p_n be the frequencies of the symbols of the alphabet to be encoded. We first sort the symbols so that $p_i \leq p_{i+1}$. Create L coins for each symbol, of denominations $2^{-1}, \dots, 2^{-L}$, each of numismatic value p_i . Use the package-merge algorithm to select the set of coins of minimum numismatic value whose denominations total $n - 1$. Let h_i be the number of coins of numismatic value p_i selected.

The optimal length-limited Huffman code will encode symbol i with a bit string of length h_i . The [canonical Huffman code](#) can easily be constructed by a simple bottom-up greedy method, given that the h_i are known, and this can be the basis for fast [data compression](#).^[2]

Performance improvements and generalizations [\[edit\]](#)

With this reduction, the algorithm is $O(nL)$ -time and $O(nL)$ -space. However, the original paper, "A fast algorithm for optimal length-limited Huffman codes," shows how this can be improved to $O(nL)$ -time and $O(n)$ -space. The

idea is to run the algorithm a first time, only keeping enough data to be able to determine two equivalent subproblems that sum to half the size of the original problem. This is done recursively, resulting in an algorithm that takes about twice as long but requires only linear space.^[1]

Many other improvements have been made to the Package-Merge Algorithm to reduce the [multiplicative constant](#) and to make it faster in special cases, such as those problems having repeated p_j s.^[3] The Package-Merge approach has also been adapted to related problems such as [alphabetic coding](#).^[4]

Methods involving [graph theory](#) have been shown to have better asymptotic space complexity than the Package-Merge Algorithm, but these have not seen as much practical application.

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

- Michael B. Baer. "Twenty (or so) Questions: D -ary Length-Bounded Prefix Coding". [arXiv:cs.IT/0602085](#) [↗](#).
- Alistair Moffat; Andrew Turpin; Jyrki Katajainen (March 1995). *Space-Efficient Construction of Optimal Prefix Codes*. IEEE Data Compression Conference. Snowbird, Utah. doi:10.1109/DCC.1995.515509 [↗](#).
- An implementation of the package-merge algorithm "[2] [↗](#)"

References [\[edit\]](#)

- ↑ ^{*a*} ^{*b*} L. L. Larmore and D. S. Hirschberg. A fast algorithm for optimal length-limited Huffman codes. *Journal of the Association for Computing Machinery*, V 37 No. 3:464–473, 1990.
- ↑ A. Moffat and A. Turpin. On the implementation of minimum redundancy prefix codes. *IEEE Transactions on Communications*. V 45 No. 10:1200–1207, Oct 1997. ([1] [↗](#))
- ↑ I. H. Witten and A. Moffat and T. Bell. *Managing Gigabytes*. Second Edition. Morgan Kaufmann Publishers, 1999. ISBN 978-1-55860-570-1.
- ↑ L. L. Larmore and T. M. Przytycka. A Fast Algorithm for Optimal Height-Limited Alphabetic Binary-Trees. *SIAM Journal on Computing*, V 23 No. 6:1283–1312, 1994.

Categories: [Lossless compression algorithms](#) | [Coding theory](#)

This page was last modified on 23 January 2015, at 03:32.

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](#)

