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

LZWL

From Wikipedia, the free encyclopedia

This article has multiple issues. Please help **improve it** or discuss these issues on the **talk page**. [hide]

- This article **needs additional citations for verification**. *(January 2013)*
- This article's **tone or style may not reflect the encyclopedic tone used on Wikipedia**. *(August 2009)*

LZWL is a syllable-based variant of the character-based LZW compression algorithm[1][2] that can work with syllables obtained by all algorithms of decomposition into syllables. The algorithm can be used for words too.

Algorithm [edit]

Algorithm LZWL can work with syllables obtained by all algorithms of decomposition into syllables. This algorithm can be used for words too.

In the initialization step the dictionary is filled up with all characters from the alphabet. In each next step it is searched for the maximal string **S**, which is from the dictionary and matches the prefix of the still non-coded part of the input. The number of phrase **S** is sent to the output. A new phrase is added to the dictionary. This phrase is created by concatenation of string S and the character that follows **S** in file. The actual input position is moved forward by the length of **S**. Decoding has only one situation for solving. We can receive the number of phrase, which is not from the dictionary. In this case we can create that phrase by concatenation of the last added phrase with its first character.

The syllable-based version works over an alphabet of syllables. In the initialization step we add to the dictionary the empty syllable and small syllables from a database of frequent syllables. Finding string **S** and coding its number is similar to the character-based version, except that string **S** is a string of syllables. The number of phrase **S** is encoded to the output. The string **S** can be the empty syllable.

If **S** is the empty syllable, then we must get from the file one syllable called K and encode K by methods for coding new syllables. Syllable K is added to the dictionary. The position in the file is moved forward by the length of **S**. In the case when S is the empty syllable, the input position is moved forward by the length of K.

In adding a phrase to the dictionary there is a difference to the character-based version. The phrase from the next step will be called S1. If **S** and S1 are both non-empty syllables, then we add a new phrase to the dictionary. The new phrase is created by the concatenation of S1 with the first syllable of **S**. This solution has two advantages: The first is that strings are not created from syllables that appear only once. The second advantage is that we cannot receive in decoder number of phrase that is not from dictionary.

References [edit]

1. ^ http://www.cs.vsb.cz/dateso/2005/slides/slides6.pps
2. ^ "Handbook of Data Compression - David Salomon, D. Bryant, Giovanni Motta - Google Books". Books.google.com. 2010-01-18. Retrieved 2014-07-11.

External links [edit]

- Detailed description

v · t · e **Data compression methods** [show]

Categories: Lossless compression algorithms

This page was last modified on 11 July 2014, at 22:17.

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.