Insert here your thesis' task.

Czech Technical University in Prague

Faculty of Information Technology

Department of theoretical computer science

Master's thesis

Example Thesis

Bc. Jan Novák

Supervisor: doc. Ing. Marek Navrátil, Ph.D.

29th April 2012

Acknowledgements

I would like to thank my family and friends for support during writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on 29th April 2012 . . . .

Citation of this thesis

Jan Novák. *Example Thesis: Master's thesis.* Czech Republic: Czech Technical University in Prague, Faculty of Information Technology, 2012.

Abstract

This thesis is concerned with word-based dictionary methods of data compression. The word-based dictionary methods are part of lossless data compression methods. The character-based dictionary compression methods, especially from LZ-algorithms family, are a part of this issue. These compression methods are very important to understand more difficult word-based ones. The main sight of this publication is description of implemented word-based dictionary compression algorithms. Modifications of these methods and results of experiments are included too. There are also details contextual with implementation of the application and its user manual. The objective of this thesis is examination of word-based dictionary data compression methods, possibilities of their improvement and their implementation linked with experiments on well-known data compression corpuses.

Keywords Data compression, entropy, word-based algorithms.

Abstrakt

Tato práce se zabývá slovními slovníkovými metodami komprese dat, specifickou částí bezztrátových metod komprese dat. Součástí publikace jsou rovněž některé znakově orientované slovníkové metody, především z rodiny LZ algoritmů, potřebné k pochopení složitějších slovních metod. Stěžejní dílem práce je popis implementovaných algoritmů pro slovní slovníkovou kompresi dat, jejich modifikací a především výsledky provedených testů, stejně jako jejich vyhodnocení. V publikaci jsou zároveň uvedeny detaily implementace aplikace a její uživatelská příručka.

Klíčová slova Komprese dat, entropie, slovní algoritmy komprese.

Contents

List of Figures

List of Tables

Introduction

This text is based on diploma thesis of Jakub Jaroš.

Motivation and objectives

The amount of information is rapidly growing up. The main technique of reducing loads of space needed to store data or reducing time needed to transmit data is data compression. The size of data reduction is achieved by removing the excessive information. The efficiency of data compression algorithms can be compared by compression ratio, time of compression and/or decompression and size of used memory. Each of these consequences unfortunately mutually interacts. The data compression algorithms can be divided by many factors, for example by the information loss: lossless compression algorithms and lossy ones. Lossless means that the compression processes is fully reversible and decompressed data is identical to the original data. These algorithms are best suited for documents, programs and other data where loss is unacceptable. Lossy algorithms irreversibly remove some parts of data and only an approximation of the original data can be reconstructed. These algorithms achieve better compression efficiency than lossless algorithms, but compression is limited to branches where some loss is acceptable (audio, video, images etc.). This thesis focuses on lossless algorithms especially the dictionary ones, which are established on likenesses of compressed, mostly textual, data, and their properties.

Problem statements

The main sight of the thesis is a part of textual algorithms named word-based which deals with texts in languages (natural, formal etc.). All of texts in natural languages (and also in other languages) have a specific structure. They can be divided into sentences, which can finish by a period, a question mark, or an exclamation mark. Each sentence consists of words that are separated from each other by space and/or punctuation marks. Word-based compression algorithms, where alphabet consists of words, take advantage of these strictly defined structures, repetitions of sequences of words and spaces, repetitions of whole sentences.

State of the Art

There are two basic ways in the world of dictionary word-based algorithms. The both of them are based on dictionary method Lempel Ziv Welch (LZW). The method Word-Based LZW, independently presented by Horspool and Cormack (1), and Jiang and Jones (2) in 1992 as first, is adaptive version of mentioned algorithm. The main sight is recently concentred to word-based context methods of data compression and related word-based *preprocessing* transformations, which reversibly transform a data into another form. The reverse process is called *postprocessing* transformation. Affected data could be compressed with most of existing lossless data compression algorithms with better compression efficiency than it can be achieved using an unaltered data.

Chapter 1

Data compression

This thesis was submitted at Czech Technical University in Prague (see Figure 1.1).

1.1 Notions and definitions

The source *message* consists of *source units*, which can be defined as *alphabet symbols* or sequence of alphabet symbols *(word, string, phrase)*, where alphabet S is a finite and non-empty set of symbols. The *code unit* is defined as a sequence of bits. The empty sequence of symbols is called *empty string* and it is represented by ε. The set of all symbols from alphabet S, free of empty string, is represented by S^+. The *concatenation* of two phrases $x, y \in S$ is represented by $x.y$.

Code is a triple $K = (S, C, f)$, where

- S is a finite set of source units,
- C is a finite set of codewords (code units),
- f is an injective mapping from S to C^+.

Figure 1.1: CTU logo

The mapping f does not map two different source units from S to the same codeword from C, as shown Formula 1.1.

$$\forall a_1, a_2 \in S, a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2) \tag{1.1}$$

The string $x \in C^+$ is *uniquely decodable* with f, when Formula 1.2 is true.

$$\forall y_1, y_2 \in S^+, f(y_1) = f(y_2) = x \Rightarrow y_1 = y_2 \tag{1.2}$$

The code $K = (S, C, f)$ is *uniquely decodable*, when all strings $x \in C^+$ are uniquely decodable in f. The code K is called *prefix code*, when none of codewords is a prefix of another codeword. If all codewords are exactly n symbols length in code K, the code K is called *block code.* The prefix codes and block codes are often used by compression algorithms because of their unique decode-ability during the left-to-right reading (decoding).

$$\mathit{Compression\ ratio} = \frac{\mathit{Length\ of\ compressed\ data}}{\mathit{Length\ of\ original\ data}} \tag{1.3}$$

The compression efficiency can be expressed by many units of measure. The amount of data reduction gained by the compression process is *compression ratio.* This compression ratio is a ratio of the length of compressed data to the original size of data (Formula 1.3). For example, the compression ratio is measured in *bpb* (bits per bit), *bpc* (bits per character) or *bpp* (bits per pixel).

The compression algorithms use specific *compression models* to encode the data. For example, these models could follows:

- The algorithm assigns code to each source unit irrespective to its position (statistical compression methods).

- The Markov's model of n-th order look at previous n source units to assign code. The simplist of this codes, 0th order, are mentioned above.

- The models based on finite automata.

1.1.1 Entropy

The entropy is only theoretical minimal length but it is possible to reach this border in some special cases. It is very difficult to measure real entropy of source message in common usage, because not only statistical model of 0th order (context of source units with length 1) exists. For example, the probabilities of appearances of source units pairs (context of source units with length 2) are considered in 1st order statistical model.

Table 1.1: RLE example

Method	Data to encode	Encoded data
RLE 1	*babaaaaabbbaabbbba11aaa*	1*b*1*a*1*b*5*a* 3*b*2*a*4*b*1*a* 2*1*3*a*
RLE 2	*babaaaaabbbaabbbba11aaa*	*bab*@5*a*@3 *baa*@4*ba1 1*@3*a*

1.1.2 Classification

Data compression/decompression is classified by many factors. The first classification depends on information loss during the compression process. The data compression, as data compression algorithms, is divided into two main parts:

- *Lossy*—some information loss is possible. These compression methods achieve higher compression (better compression ratio) but they are useful only in special cases (images, video, voice...).
- *Lossless*—information is acquired in original form. These compression methods are best suited for data where loss is unacceptable (documents, programs, scripts...).

1.2 Elementary methods

Many elementary compression methods are currently known, but only the Run-Length Encoding (RLE) is mentioned on as very simple compression method, to show how compression methods work.

1.2.1 RLE

The RLE technique was created especially for data with strings of repeated symbols (the length of string of repeated symbols is called *run*). The main idea of RLE compression is to encode repeated symbols as a pair—the length of string and the symbol.

There are shown some ways of RLE compression of 23-characters-long string *"babaaaaabbbaabbbba11aaa"* in Table 1.1. The first method (RLE 1) is the simplest way to encode string (20 characters) which exactly follows idea of RLE. The problem is that in worst case the size of output data can be two times longer than size of input data. This problem is solved by second method (20 characters), for simplicity called RLE 2, where only three or more characters long strings are encoded, but we pay for it by another character (@ in example), which precede all of encoded pairs, to differentiate between the length of string and the number as a character. The third method (RLE 3) solves this problem by map of bits (18 characters). RLE compression method is very useful for graphic files coding but it is practically useless in text compressions without using it cooperatively with other methods.

1.3 Dictionary methods

1.3.1 LZ77

It is obvious that the value of offset and the match length have to be limited to some constant. The usually chosen value for the match length is 255 (8 bits) and the offset is commonly encoded on 12–16 bits, so the search buffer is limited to 4 095–65 535. In so far that there is no need to remember more than 65 535 already encoded symbols during compression process.

Chapter 2

Implementation and testing

2.1 Details of realised tests

The scaliness of implemented algorithms were tested on chosen files from Calgary and Canterbury corpus too. The framework to scale the files is different of the testing one. Each file is equally split to 1 000 parts (files with number of lines less than 1 000 are split to 100 parts) by number of lines—the n th part consists of $\frac{n}{1000}$ ($\frac{n}{100}$) lines. Each test runs only 100–10 times (depending up the n—the size of compressed data) because of time complexity. This cycle runs 10–100 times (10 was chosen for this tests) and the minimum time is taken. The file splitting by the number of lines was chosen because of the character of algorithms (word-based)—the splitting by the block of the same size is not so predicative.

There are parameters of the computer used for tests shown in Table 2.1.

2.2 Integers distribution and encoding

The integers encoding of indexes of phrases from the word (non-word) dictionary is possible cause of the only average results of compression ratio of implemented algorithms. The decision is to get the distribution of indexes during the encoding process (and decoding process too). The length of indexes located in shown graphs is only hypothetical—the binary code with minimal length.

Table 2.1: Testing computer parameters

Part	Description
CPU	2.2 GHz AMD Athlon(tm) 64 Processor 3200+
MEM	2.5 GB
OS	x86_64 GNU/Linux Fedora release 7 (Moonshine)

The graphs of index distribution also shows the differencies between the algorithms with sorted dictionaries (*WLZWS* and *WLZWES*) and the algorithms with unsorted dictionaries (*WLZW* and *WLZWE*). The most frequently used phrases are moved to the front of dictionary in algorithms with sorted dictionary so they get lower indexes. This feature is demonstrated by the growth of number of indexes at the beginning of the distribution. The compression process of algorithms with sorted dictionaries becomes more efficient when the code with variable length of code words (Fibonacci code) is used but the compression efficiency is supposed to be the same at the transition from the *WLZWE2* algorithm to the *WLZWES2* algorithm—the encoding by block code.

Conclusion

The word-based dictionary data compression algorithms (a part of lossless data compression) are the subject of this thesis. The lossless data compression is a very important field of research because the data compression allows to reduce the amount of space needed to store data.

The background of data compression field was presented in Chapter 1. There are basic notions and definitions followed by description of character-based dictionary algorithms. The word-based dictionary compression methods were investigated and discussed at the end of this chapter too.

There is the investigation of index distribution of tested files in Section 2.2. It led to the new modification of semi-adaptive word-based LZW algorithm—*WLZWE2*. The compression efficiency of this algorithm applied to the large files is better than the other implemented algorithms. However, the compression efficiency of *WLZWE2* algorithm is much worse when it is applied to the small files. The experiments with *WLZWE2* and *WLZWES2* algorithms confirm the assumption from Section 2.2—the compression efficiency of version with unsorted dictionaries (*WLZWE2*) is analogous to version with sorted ones (*WLZWES2*).

The testing of memory used during compression and/or decompression process is one of the possibilities of further research. The experiments with files of greater size or multilingual files could be also good opportunity to gain new improvements of algorithms. The static part of dictionaries could improve the compression efficiency too.

The implemented methods achieve fairly good compression ratio (25–30% at large files) with acceptable compression and decompression time. There are possibilities of further improvements especially at semi-adaptive methods. However, the gain of these improvements is not good enough to top the compression efficiency of other lossless data compression methods (context methods from PPM family). The results of implemented algorithms were not as good as it was expected but the work on this thesis showed new ways of possible further research—word-based version of grammar-based compression algorithms and another possibilities in the field of word-based context methods of data compression.

The Gnuplot 4.2 utility was very useful for generation of graphs in this thesis. There was the drawing editor Ipe 6.0 used for figures creation.

Bibliography

(1) Horspool, R. N.; Cormack, G. V.: Constructing Word-Based Text Compression Algorithms. In *Data Compression Conference*, 1992, pp. 62–71.

(2) Jiang, J.; Jones, S.: Word-based dynamic algorithms for data compression. In *Communications, Speech and Vision*, volume 139, December 1992, pp. 582–586.

(3) Powel, M.: The Canterbury Corpus. November 2001, [Cited 2011-10-12]. Available at WWW: <`http://corpus.canterbury.ac.nz/index.html`>

Appendix **A**

Acronyms

Acronyms

LZW Lempel Ziv Welch.

RLE Run-Length Encoding.

Appendix **B**

Content of CD

Visualise the content of enclosed media. Use of `dirtree` is recommended. Note that directories src and text with appropriate content are mandatory.

- `readme.txt` .... the file with CD content description
- `data` .... the data files directory
  - `graphs` .... the directory of graphs of experiments
    - `*.eps` .... the B/W graphs
    - `*.png` .... the color graphs
    - `*.dat` .... the graphs data files
- `exe` .... the directory with executable WBDCM program
  - `wbdcm` .... the WBDCM program executable (UNIX)
  - `wbdcm.exe` .... the WBDCM program executable (Windows)
- `src` .... the directory of source codes
  - `wbdcm` .... the directory of WBDCM program
    - `Makefile` .... the makefile of WBDCM program (UNIX)
  - `thesis` .... the directory of LaTeX source codes of the thesis
    - `figures` .... the thesis figures directory
    - `*.tex` .... the LaTeX source code files of the thesis
- `text` .... the thesis text directory
  - `thesis.pdf` .... the Diploma thesis in PDF format
  - `thesis.ps` .... the Diploma thesis in PS format