UDC 621.327:681.5

A.V. Yakovenko, candidate of technical sciences, senior staff scientist,

S.N. Pugachov,

V.V. Tverdokhlieb

ANALYSIS OF THE CHARACTERISTICS OF THE LZW DATA COMPRESSION ALGORYTHM

LZW data compression algorythm is examined. It is described in detail how it works. The basic steps of coding are described for example, and for general understanding of the process the results of data compression are showed. The decoding process is briefly described, the principles of the dictionary coscruction in the process of encoding and decoding are considered, as well as its organization and units construction are shown. In conclusion, several issues, related to the method which represent space for upgrading and improvement of the LZW algorithm, are stated.

Keywords: *LZW coding, lossless data compression, construction of LZW dictionary, weaknesses of the LZW data compression algorithm.*

Introduction

Messages sent using channels of communication (speech, music, images, etc.), for the most part, intended for direct human organs of sense perception and usually ill-suited for their effective transfer via communication channels. So they are in the process of transferring is usually subjected to encoding. Message encoding may pursue different objectives. For example, this encoding for classification of information transmitted. Another example of encoding is the conversion of discrete messages from one numeral system to another (from decimal to binary, octal, etc., from the sign-value in position, the conversion letter alphabet in digital, etc.). Coding in the channel or the interference-free coding can be used to reduce the number of errors that occur when transmitting on the channel interference. Finally, the message encoding can be done to reduce the amount of information and increasing the speed of transmission or reduce the bandwidth required for transmission. This encoding is called non-surplus, or economical, efficient coding, and data compression.

And as with a lot of information we have, the more expensive it is. Unfortunately, much of the data that you want to transmit electronically and save, not the highest compact representation. Rather, this information is stored in the form of providing them the most simple usage, for example: plain text, ASC II codes, book text editors, binary codes of computer data, separate counts of signals in data collection systems, etc. but the most easy-to-use reporting requires twice-three times, and sometimes hundreds of times more space for their preservation and bandwidth for transmission than they actually need. Therefore, data compression is one of the most actual directions of modern science.

There are two types of compression: lossless compression of information system (non-destructive compression) and lossy compression system information (destructive compression). In lossless compression decoder restores data source precisely, and in a lossy compression systems (or destruction) of the encoding is done in such a way that the decoder is not able to recover the data source in its original form. The choice of the system of non-destructive or destructive compression depends on the type of data to be compressed. Therefore, the aim of the research article is to analyse the basic vocabulary of the algorithm of data compression without loss of information

Main Part

Dictionary compression information are less mathematically justified than for example the Shannon-Fano, Huffman or but more practical. They do not consider the statistical features of the data being encoded and not use a variable-length codes. Instead, they choose some sequences of characters, store them in a dictionary, and all are encoded as sequences of labels, using a dictionary. Decoding is the reverse rotation of the index for the appropriate phrase from the dictionary. The dictionary is a collection of phrases, which we believe will occur in the sequence being processed. Indexes phrases must be constructed in such a way that, on average, they take up less space than require replaceable strings. Due to this, and compression. The dictionary can be static or dynamic (Adaptive). The first is the constant, sometimes in it add the new sequence, but never removed. Dynamic dictionary contains the sequence,

previously received from the input, it is allowed and adding, and deleting data from the dictionary when reading the input file. In General, the use of dynamic dictionary is preferred. This method begins with a blank or a little original, appends the words as they arrive from the input file and deletes the old words, because the dictionary is made slowly.

The most perfect representative of this group of words is the LZW algorithm, developed in 1984 by Terry Welch. This is a very popular variant of LZ78 algorithm. Its main feature is a short label, which consists only of a pointer to a place in the dictionary. LZW efficiency depends on the structural characteristics of the processed data. The algorithm is based on the principle of eliminating structural redundancies caused by the presence of identical chains of elements. The LZW method begins the initialization of a dictionary of all the characters of the source alphabet. Because the dictionary has been partially filled by the first available characters will always be found in the dictionary, so that a label can consist of only a pointer, and there is no need to further keep the character code as LZ77 and LZ78 algorithms.

The LZW method stores coming to input characters in the string (I) after each new character was added to the line I, coder looking for (I) in the dictionary. If the string is found, the process of lengthening (I) continues. At some point adding a new symbol x leads to an undetecting line Ix (x symbol has been added to I). Then the encoder writes the output file pointer in the dictionary on line I, stores a row Ix (which would now be called a phrase) in the dictionary on the next valid position and initializes (assigns) line (I) the new value of x. Let's consider job of the algorithm on the example (table 1). Image compression algorithm is dictionary initialization codes of all the colors in the image. For your convenience we take only three primary colors: red, green, blue; Let us denote their respective R, G, B, and also assign codes, "1", "2" and "3". To illustrate the process of coding the code the following string "RGBGRBRBGRGRBGBRGRGR". Get the following steps.

1. The first input element in the "R" is found in the dictionary under number 1 (this number we conditionally appointed). The next input element "G", but "RG" is not in the dictionary. Encoder does the following: (1) it produces the output link 1, (2)

retains the "RG" at the next available position dictionary (record 4), and (3) initializes the I row "G". 2. On the sign is the "B" element, but the string "GB" is not in the dictionary. Coder (1) writes to the output 2 (the code that we have appropriated the "G"), (2) stores in the dictionary "GB" (record 5), and (3) initializes I string "B".

Table 1.

Coding LZW for «RGBGRBRBGRGRBGBRGRGR»

I	In dictionary?	New record	Exit	*I*	In dictionary?	New record	Exit
R	yes			GRG	no	11-GRG	7(GR)
RG	no	4-RG	1(R)	G	yes		
G	yes			GR	yes		
GB	no	5-GB	2(G)	GRB	no	12-GRB	7(GR)
B	yes			B	yes		
BG	no	6-BG	3(B)	BG	yes		
G	yes			BGB	no	13-BGB	6(BG)
GR	no	7-GR	2(G)	B	yes		
R	yes			BR	yes		
RB	no	8-RB	1(R)	BRG	no	14-BRG	9(BR)
B	yes			G	yes		
BR	no	9-BR	3(B)	GR	yes		
R	yes			GRG	yes		
RB	yes			GRGR	no	15-GRGR	11(GRG)
RBG	no	10-RBG	8(RB)	R	yes		
G	yes			R,end	no		1(R)
GR	yes						

The decoder starts work as a coder, with the initialization of the dictionary. In our case there are codes of all colors. Then he reads the input file, which consists of pointers in the dictionary, uses every pointer in order to restore the compressed characters in the dictionary and write them to the output file. In addition he builds a dictionary with the same approach as a coder. In the first step the decoding decoder introduces the first pointer and use it to restore the vocabulary item (I). a string of

characters, and it is written by the decoder in the output file. Next, you write a dictionary of row Ix, but the symbol x is not known; This will be the first character of the next line, extracted from the dictionary.

At each step, the decoder after the first decoding enters the next pointer, retrieves the next row of J from the dictionary, writes it to the output file, retrieves the first character of x, and logs the row in a dictionary on a free position (after checking that the string (Ix) is not in the dictionary). The decoder then moves (J) (I). now it is ready for the next step. In our example, the "RGBGRBRBGRGRBGBRGRGR" as the first character of the input file decoder is a pointer 1. It matches "R", which is derived from a dictionary is stored in (I) and becomes the first row of the output file. The next pointer-2, so J is "G", and the contents of the J is written to the output file. The first character of the string (J) is added to the variable I, forming a string "RG", which is not in the dictionary, so it is added to the dictionary in position 4. The contents of the variable J is transferred into a variable I; now I is "G". The next pointer is 3, so the string is retrieved from the dictionary "B" appears in J and is written to the output file. The variable I is supplemented by up to a value of "GB"; This string is not in the dictionary, so it there is listed at number 5. The variable J is rewritten in (I); now I is equal to "B". In the next step decoder reads the pointer 2, writes the "G" in the dictionary file and stores the string "BG". This continues until the end of the file.

As the encoder and decoder reads the symbols and add them to a string (I) until the I is in the dictionary. At some point the line Ix in the dictionary is not detected, and then row (Ix) is placed in the dictionary. So, when new rows are added to the dictionary is just one character (x). This means that for each row in the dictionary is a dictionary the parent string that exactly one character shorter. Therefore, for the dictionary is a good solution to the LZW dictionary in the form of a tree, when you build the add new lines of Ix is adding just a single character x. It follows that for each row in the dictionary is a dictionary the parent string that exactly one character shorter. Example the LZW dictionary tree shown in Fig. 1.

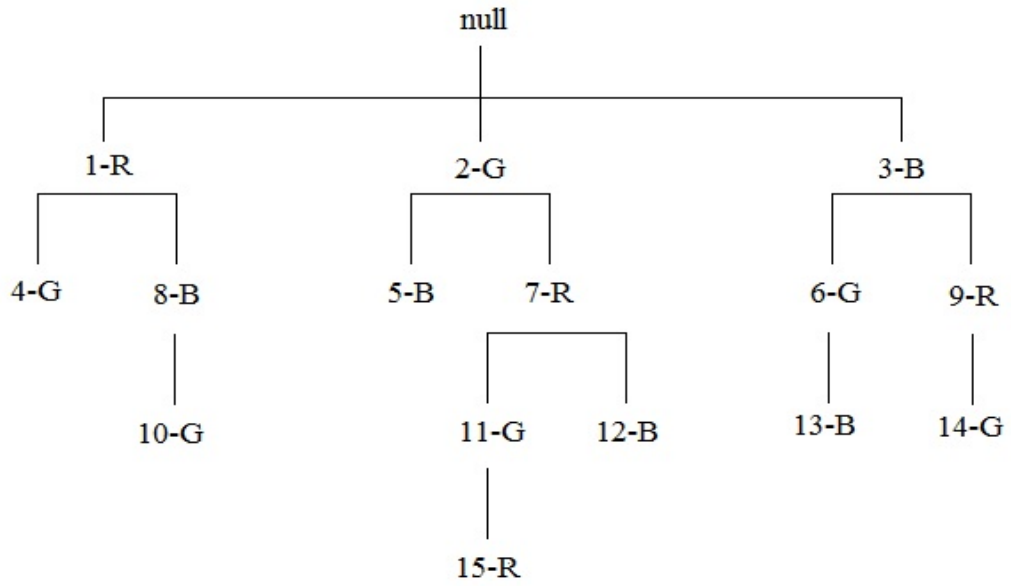

Fig. 1. The tree of dictionary LZW

In practice, the dictionary tree nodes are composed of three fields: a pointer to the parent node pointer (or index), created in the process of hashing, and character code, which is in this site (Fig. 2).

Parent
Index
Symbol

Fig. 2. The structure of the site dictionary LZW

Conclusions

The LZW data compression algorithm uses the same principles as other Dictionary compression methods. It reads a file character by character and adds a phrase to the dictionary. Phrases are individual characters or strings of characters in the input file. When the line of the input file is the same as with some words in the dictionary, the compressed file is recorded the position of the phrase or label. LZW efficiency depends on the structural characteristics of the processed data. The algorithm is based on the principle of eliminating structural redundancies caused by the presence of identical chains of elements. Despite the fact that the LZW is more

effective than not only statistical methods and techniques of his team, he also has a number of weaknesses that leave space for modification and refinement.

The main disadvantages of LZW, which require removal of the compression effect is manifested. Because the strings in the dictionary are longer on one symbol on each step, it takes a long time for the appearance of long lines in the dictionary. As is known, at the very beginning of the coding dictionary initialized codes all file elements, and due to the nature of its replenishment it quickly becomes crowded. Of course, there are solutions, such as a full reset dictionary, or delete old records, but, unfortunately, there is no good algorithm to determine the optimal action which records to delete and which to keep.

LIST OF USED SOURCES

1. *Сэломон Д.* Сжатие данных, изображений и звука / Д. Сэломон. – М. : Техносфера, 2004. – 368 с.
2. *Ватолин Д.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : ДИАЛОГ-МИФИ, 2003. – 384 с.
3. *Миано Дж.* Форматы и алгоритмы сжатия изображения в действии / Дж. Миано. – М. : Издательство «Триумф», 2003. – 336 с.
4. *Лидовский В.В.* Теория информации : Учебное пособие / В.В. Лидовский. – М. : Компания «Спутник+», 2004. – 111 с.