International Journal of Latest Trends in Engineering and Technology
Special Issue SACAIM 2017, pp. 391-396
e-ISSN:2278-621X

DIGITAL DATA COMPRESSION USING LZW ALGORITHM

DeekshithRai.M S[1], John Paul .A[2] & Mrs. Annapoorna Shetty[3]

Abstract- Lempel–Ziv–Welch (LZW) Data compression algorithm developed by Abraham Lempel, Jacob Ziv, and Terry Welch. LZW compression algorithm is one of the Adaptive Dictionary techniques. Dictionary cannot be transmitted .Dictionary is built while the data being encoded. So encoding is done on the fly. Dictionary built up at receiving end on the fly. If the dictionary overflows we have to reinitialize the dictionary and add a bit to one of the code words. Choosing large dictionary size avoids overflow, but spoils the compressions. A dictionary having the source symbols is developed. For 8-bit images of an monochrome, the first 256 words of the dictionary are allotted to the gray levels 0-255 left over part of the dictionary is filled with sequences of the gray levels.LZW compression algorithm works best when applied images of an monochrome and text files that contain repetitive text/patterns
Keywords– Compression Ratio, Decoding, Encoding

1. INTRODUCTION

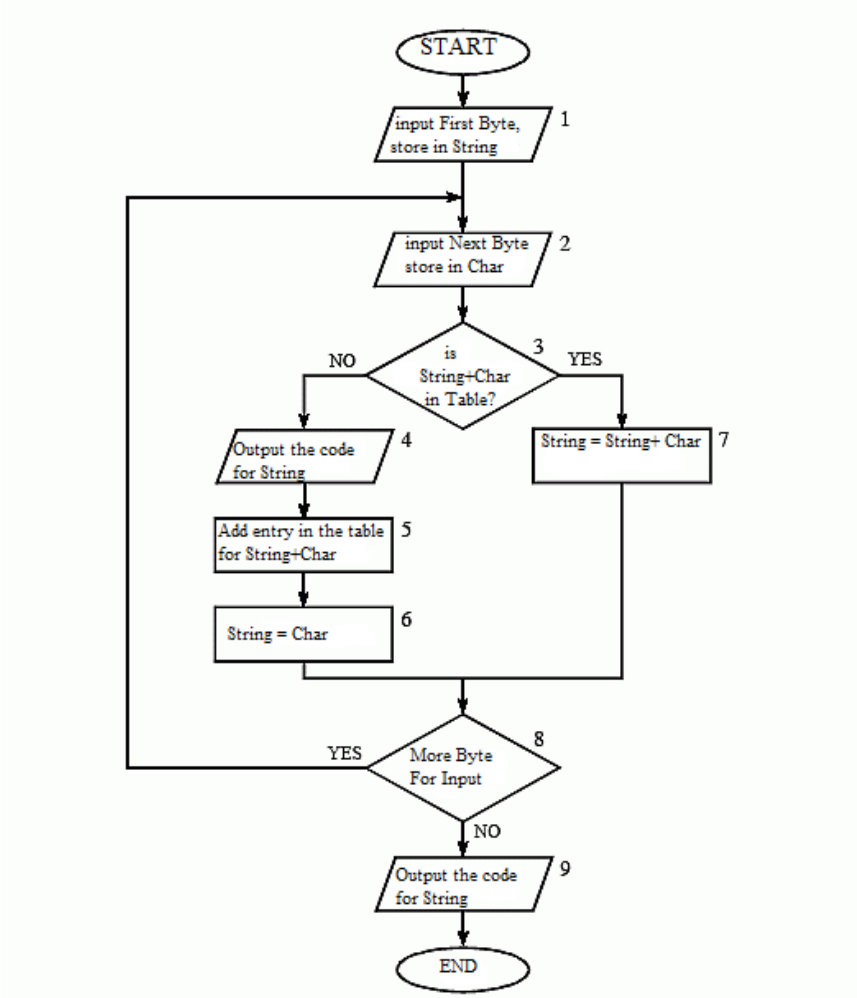
The LZW algorithm is a compression technique. This algorithm is normally utilized in GIF and alternatively in PDF and TIFF. Unix's 'compress' command, among different uses. It is lossless. The algorithm is easy to execute and has the potential for high throughput in hardware implementations. It is the algorithm of the generally utilizedUnix file compression utility compress, and is used in the GIF pictureformat. The Idea reliesupon reoccurring patterns to save data. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility. It is the basis of several PC utilities that claim to "double the limit of your hard drive".LZW compression process by reading symbols in a sequence, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression. Characteristic highlights of LZW includes, LZW compression uses a code table, with 4096 as a typical decision for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input record. When encoding starts the code table contains only the initial 256 passages, with the remainder of the table being blanks. Compression is achieved by utilizing codes 256 through 4095 sequences of bytes is represented. As the encoding continues, LZW identifies sequences which are repeated in the data, and adds them to code table.Decoding is accomplished by taking a code from the compressed file and translating it through the code table to discover what character or characters it represents.

2. PROPOSED ALGORITHM

2.1 EncodingLZW algorithm –

Encoding is the process of putting a sequence of characters (numbers, letters, certain symbols and punctuation) into a specialized format for efficient transmission or storage. The algorithm works best on data with repeated patterns. Dictionary is initialized to contain the single-character strings corresponding to all the relevant input characters. The LZW algorithm process by checking through the string of an input for successively longer substrings until it finds one that is not in the dictionary. Such a string is found, the index for the string less the last character is retrieved from the dictionary and sent it to output, and the new string is added to the dictionary with the next on hand source code. The finial input character is then used as the next beginning point to check for substrings. In this way, successively longer strings are recorded in the dictionary and made available for subsequent encoding as single output values. So the first phase of a message will have little compression. As the message grows, however, the compression ratio tends core to the maximum.

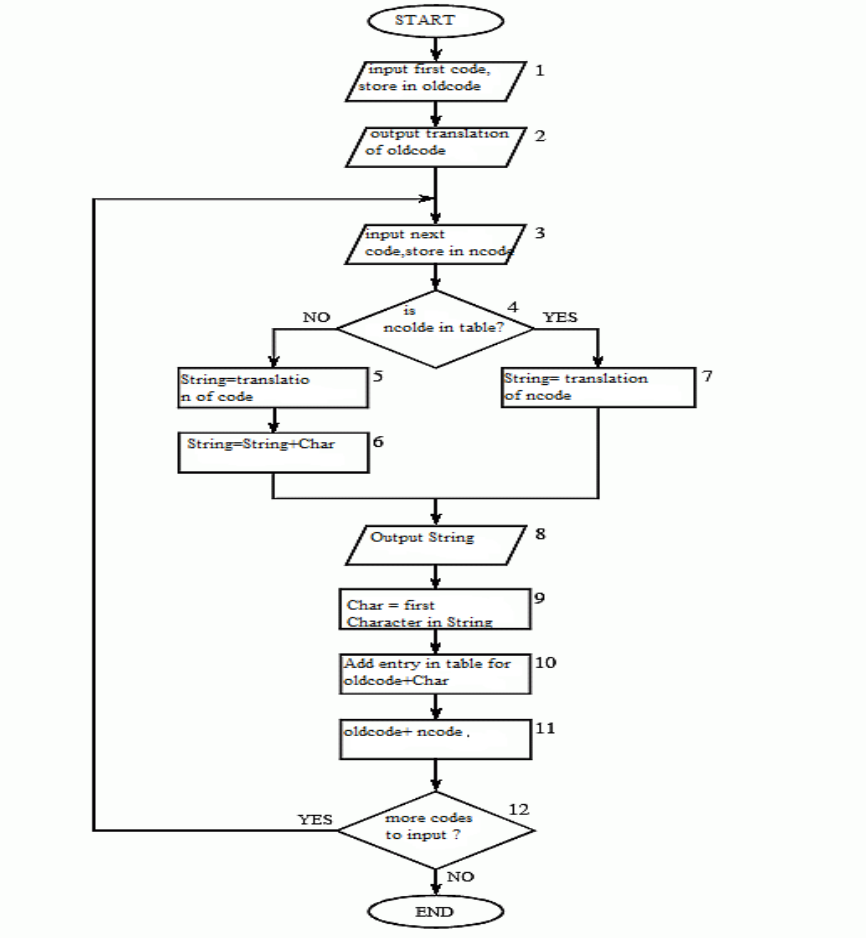
[1] Department of Master of Computer Application, Aloysius Institute of Management and Information Technology, Mangalore, Karnataka, India
[2] Department of Master of Computer Application, Aloysius Institute of Management and Information Technology, Mangalore, Karnataka, India
[3] Assistant Professor, Department of MCA, Aloysius Institute of Management and Information Technology, Mangalore, Karnataka, India



LZW compression algorithm flowchart. Charis a single byte. String is a length of bytes. Data read from input file (Box 1 & 2) as a single byte, and written to the compression file (Box 4) as 12bit codes.

2.2 Decoding LZW algorithm –

Decoding is a process of the conversion of an encoded format back into the original sequence of characters. The decoding algorithm proceed by reading a value from the encoded input and outputting corresponding string from the initialized dictionary from the input .It can get the next value, and it adds it to the dictionary the concatenation of the string just output and by decoding the next input value we can obtain the first character of the string . The decoder proceeds to the next input value and repeats the process until there is no more input, at which point the final input value is decoded without any more additions to the dictionary. In such a way the decoder built up a dictionary which is relevant to that used by the encoder, and uses it to decode subsequent input values. Thus the full dictionary does not need be sent with the encoded data; just the first dictionary containing the single-character strings is sufficient.



LZW uncompress ion flowchart. Variables, old code and ncode hold the 12bit codes that is from the compressed file, Char holds a single byte, and String holds a string byte.

3. EXPERIMENT AND RESULT

3.1 Example For Encoding Process-

An example string used to exhibit the LZW pressure calculation is in the outline. The procedures clarify in detail with input and the yield of a procedure. The info string is a short rundown of English words isolated by the '/' character. Experiencing the begin the calculation for this string, you can see initially stage on top of it, a check is performed to check whether the string "/S" is in the table. Since it isn't, the code for '/' is yield, and the string "/S" is added to the table information. Since we have 256 characters effectively characterized for codes 0-255, the code 256 is assigned with first string. After the third letter, 'D', has been perused in, the second string code, "SD" is added to the table, and the yield of the code is letter 'S'. The procedure proceeds until in the second word, the characters '/' and 'S' are perused in, coordinating string number 256. For this situation, the code 256 is yield, and a three character string is added to the string of a table. The procedure proceeds until the point when the string is depleted and
the greater part of the codes have been yield.

Input String = /SDC/SD/SDD/SDA/SDU

Character Input	Code Output	New code value	New String
/S	/	256	/S
D	S	257	SD
C	D	258	DC
/	C	259	C/
SD	256	260	/SD
/	D	261	D/
SDD	260	262	/SDD
/S	261	263	D/S
DA	257	264	SDA
/	A	265	A/
SDU	260	266	/SDU
EOF	U		

The Compression Process

The final output for the string is shown in chart along with the resulting. The string table fills up immediately, since a new string is added to the table each time a code is output.

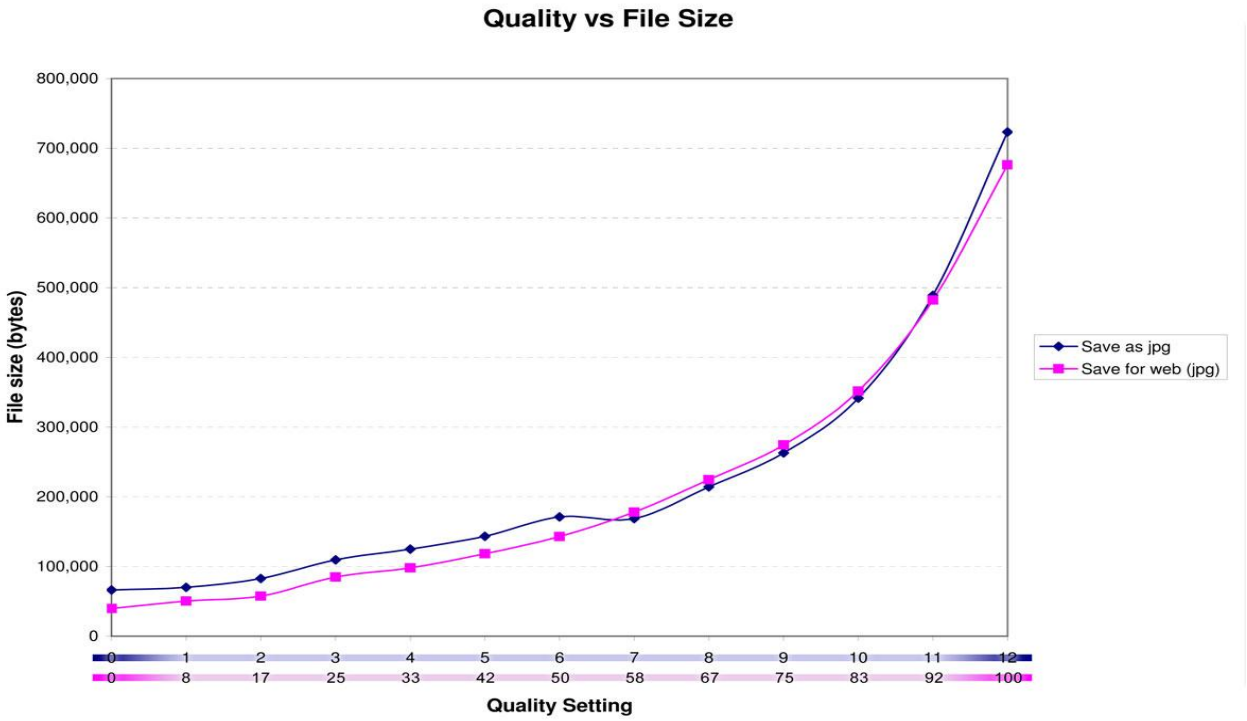
3.2 Example for decoding LZW process-

An example is shown for decoding process of an LZW algorithm. Note the first 256 codes already defined to translate to single character of a strings. Decoding process it adds a new string to the string table each execution it reads in anew code. All it needs to do in addition to that is translate each incoming code into a string and it sends string to the output. Below chart is shows the output of the algorithm given the input created by the compression earlier in the paper. The important thing to note is that the string table ends up looking similar like the table developed up during compression. The output strings are identical to the input string from the compression algorithm.

Input Codes: /S D C 256 D 260 261 257 A 260 U

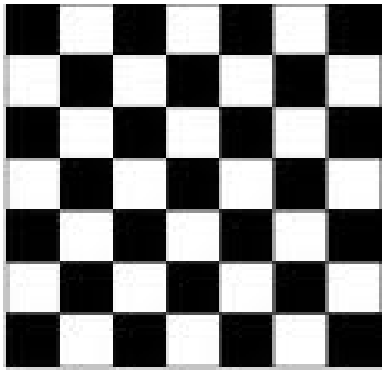
Input New Code	Old Code	STRING Output	CHARACTER	New Table entry
/	/	/		
S	/	S	S	256=/S
D	S	D	D	257=SD
C	D	C	C	258=DC
256	C	/S	/	259=C/
D	256	D	D	260=/SD
260	D	/SD	/	261=D/
261	260	D/	D	262=/SDD
257	261	SD	S	263=D/S
A	257	A	A/	264=SDA
260	A	/SD	/	265=A/
U	260	U	U	266=/SDU

3.3 Quality of the image VS image file size grsph-



(Image quality in x-axis, file size in y-axis)

3.4 Checker board image-



For example, Using LZW compression algorithm, a board image consisting of repetitive white and black patterns can be compressed up to 70% of its original file size. A high compression ratio can be achieved by using LZW compression algorithm.

LZW compression algorithm works well it is applied on images (monochrome images) and text files that contain repetitive text/patterns.

3.5 Compression Ratio-

The compression ratio conveys the variation between the file size of an uncompressed image, and the file size of the similar image when compressed. The compression ratio is balanced to the original image size is divided by the compressed image size. This ratio gives an hint of how much compression is attained for a particular image. Most algorithms have a common range of compression ratios that they can attain over a collection of images. Because of this normally more useful to look at an average compression ratio for a particular method. The picture quality is affected by compression ratio. In general, the higher the compression ratio, the poorer the quality of an image. The balance between picture quality and compression ratio is an important one to consider when compressing images.

$$\text{Compression Ratio} = \frac{\text{Size of an image (original)}}{\text{Size of an image (compressed)}}$$

Using LZW, 60-70 % of compression ratio can be attained images like monochrome and text files with repeated patterns.

4. CONCLUSION

In this paper we have described LZW compression algorithms. As discussed LZW dictionary based algorithm, which is lossless in nature and incorporated as the standard algorithm for compression. The LZW has found various Applications fromtext compression to multimedia but Still associated with several drawbacks which we have discussed previous.Compression and decompression time is determined as the amount of time needed to encode and decode a digital data. LZW has various advantages when being used to compress large text data, in English language which has high redundancy.

5. REFERENCES

[1] A Comparative Study Of Text Compression Algorithms", International Journal of Wisdom Based Computing, Senthil Shanmugasundaram, et.al,Vol. 1 (3), December 2011.

[2] M.K. Ibrahim ,L. Ghouti, A. Bouridane and S. Boussakta, IEEE Trans. Signal Process,"Digital image watermarking using balanced multiwavelets", Process., 2006, Vol. 54, No. 4, pp. 1519-1536.

[3] A. Neogi; Tzi-cker ChiuehPerceptual preprocessing techniques applied to video compression: some result elements and analysis

[4] IEEE, "Computer",Welch, T.A.,"A Technique for High-Performance Data Compression, Vol. 17, No. 6, pp. 8 ,19, June 1984.

[5] R. Nigel Horspool,"Improving LZW" , Dept. of Computer Science, University of Victoria P.O. Box 3055, Victoria, B.C., Canada V8W 3P6.

[6] Enhanced LZW (Lempel-Ziv-Welch) Algorithm by Binary Search with Multiple Dictionary to Reduce Time Complexity for Dictionary Creation in Encoding and Decoding",Nishad PM ,et.al. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, Issue 3, March 2012.

[7] Mark Daniel Ward,"Exploring Data Compression via Binary Trees", Purdue University.

[8] Design and Implementation of LZW Data Compression Algorithm", International Journal of Information Sciences and Techniques (IJIST) Simrandeep Kaur, et.al,"Vol. 2, No. 4, July 2012. The Data Compression Book by Mark Nelson W. KinsnerDept. of Electr. & Comput. Eng., Manitoba Univ., Winnipeg, Man., Canada.