International Journal of Scientific Research in Computer Science and Engineering

Research Paper

Vol.8, Issue.1, pp.37-44, February (2020)

E-ISSN: 2320-7639

Analysis of Huffman Coding and Lempel–Ziv–Welch (LZW) Coding as Data Compression Techniques

Gajendra Sharma

Kathmandu University, School of Engineering, Department of Computer Science and Engineering, Dhulikhel, Kavre, Nepal

Author's Mail id: gajendra.sharma@ku.edu.np

Available online at: www.isroset.org

Received: 10/Jan/2020, Accepted: 22/Jan/2020, Online: 28/Feb/2020

Abstract- Huffman Coding is a statistical data compression providing to reduce code length that is used to represent the symbols of an alphabet. This is a standard technique for the creation of Minimum-Redundancy Codes. LZW is a dictionary based compression tool which is widely popular. This implies that instead of tabulating character counts and building trees LZW encodes data by referencing a dictionary. Compared to any adaptive and dynamic compression method, the concept is to initiate with an initial model, read data and update the framework and data encoding. In this paper, we investigated the following question: Which coding, LZW or Huffman, is more suitable compared to each other? The implemented results show that LZW requires no prior information about the input data stream, and also LZW can compress the input stream in one single pass allowing fast execution. LZW coding is more feasible when the high compression ratio and less compression-decompression time are needed.

Keywords- Digital data, input symbols, compression, minimum-redundancy, LZ Wcoding, Huffman coding.

I. INTRODUCTION

Data compression is a procedure by which different files such as text, audio, and video can be transformed to another file, so that the original file is fully recovered from the original file without any loss of original information. This process is significant if it is needed to save the storage space. Compression is significant as it reduces resources required to disseminate and store data. Computational resources are broadly used in the compression process in the reversal of the decompression process. Data compression is space–time complexity trade-off. The design of data compression schemes includes trade-offs among various components, involving the level of compression, the quantity of distortion as well as the computational resources required to compress and decompress data. To evaluate coding performance, monochrome images are encoded by the baseline JPEG algorithm using Huffman coding [12, 14] and compressed by the offered model and lossless re-encoding mechanisms. StuffIt is data compression software introduced by Smith Micro and its recent version that provides an ability to compress JPEG files [1, 4, 13].

A. Types of Data Compression

Two fundamental data compression techniques are used widely: (1) lossy data compression technique, used to compress image data files and (2) lossless data compression technique used to transmit or store binary files. The two types of data compression mechanism are lossy compression and lossless Compression. The data compression is the phenomena of encoding information

using fewer bits than an unencoded representation, through implication of encoding schemes. Compressions rely on two major strategies: Eliminating redundant information and getting rid of irrelevant [2, 8].

B. Lossy Data Compression

The data retrieves after decompression that is not identical as the original data, but it is close to be useful for precise purpose. It uses lossy data compression to a message and the message can not be recovered. It will not provide back the initial message when the compressed message is decoded. Data was lost in this case as lossy compression cannot be decoded to facilitate the exact original message, for example, text data. It is useful for Digitally Sampled Analog Data (DSAD). DSAD includes sound, video, graphics and pictures. The sound file has very high and very low frequencies.

C. Lossless Data Compression

Lossless data compression is a technique to use data compression algorithms to compress the text data and permitting the original data to be re-structured from the compressed data. The ZIP file is used to compress data files. This is the use of lossless data compression. Lossless compression is used when the original data and decompressed data are analogous. Lossless text data compression algorithms use statistical redundancy to characterize the sender's data concisely without loss of information.

JPEG uses lossy compression whose strategy is based on the characteristics of human visual insight. JPEG is optimized for tone images and photographs including different colors [3, 6, 15]. People easily differentiate brightness than color, JPEG focuses compression on the color information.

II. HUFFMAN CODING

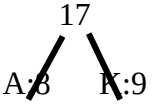
A complicated and lossless compression technique is called Huffman coding. The characters in a data file are converted to a binary code, where the common characters in the file have shortest binary codes. To observe Huffman coding function, a text file should be compressed, and the characters in the file have following frequencies:

Table 1. Huffman Coding Table

Symbols	frequencies
A	8
B	10
E	32
G	12
K	9
M	66

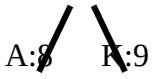
Step 1: Arranging in the ascending order A:8 K:9 B:10 G:12 E:32 M:66

Step 2: making tree of lowest two



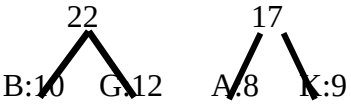
B:10 G:12 E:32 M:66

Step 3: Arranging in the ascending order B:10 G:12



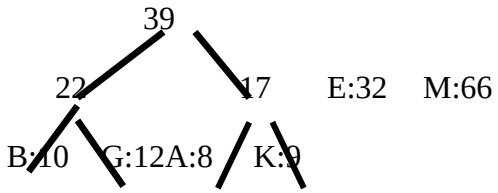
E:32 M:66

Step 4: making tree of lowest two

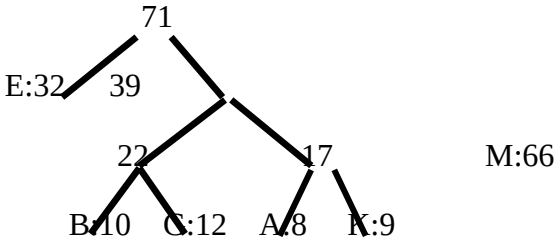


E:32 M:66

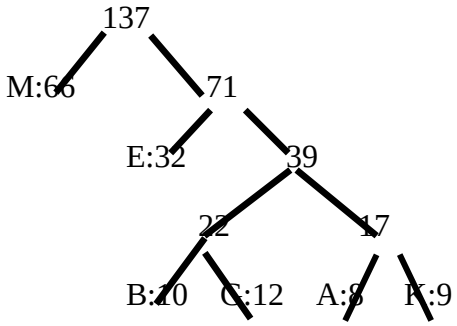
Step 5: making tree of lowest two



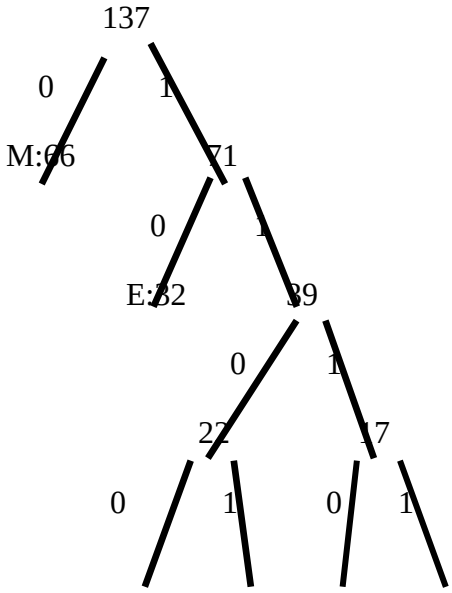
Step 6: making tree of lowest two



Step 7: making tree of lowest two



Step 8: Assigning bit to the tree



B:10 G:12 A:8 K:9

Step 9: Fill bit to the table

Table 2. Huffman Coding Table

Symbols	frequencies	Bit
A	8	1110
B	10	1100
E	32	10
G	12	1101
K	9	1111
M	66	0

The original message had 18-bit string of letters. It was shrunk the number down completely with a binary tree. Two points can be made to shorten the structure of the binary tree:
(1) The shortest frequency symbol is assigned the longest code length. Higher frequency symbols have shorter binary code lengths. (2) There are two longest length codes as binary tree made of two leaves adding to formulate a tree.
A complete image sequence is coded with a single code in coding technique. As such the correlation on pixels is exploited. Arithmetic coding is based on the following principles [5, 10]:

• Finite symbol alphabet
• Symbols sequences with fixed length
• Infinite sequences
• The number of real numbers in the interval [0, 1] is infinite which can assign a subinterval for input

A. *Lempel–Ziv–Welch (LZW) Coding*

The LZW is popularly used compression technique which is used in GIF and in TIFF as well as PDF. LZW is important technique for data compression owing to its flexibility and simplicity. It facilitates to increase the capacity of hard drive twice.

LZW functions by reading symbols, grouping the symbols into strings, and converting the strings into codes. Some features of LZW are as follows:

- LZW compression uses code table, with 4096 common choices. Codes 0-255 in the code table is assigned to represent single bytes from input.
- When encoding starts the code table having only the first 256 entries, with the remainder of the table being blanks. Compression is received by using codes 256 through 4095.
- LZW signifies repeated sequences in the data and adds them to the code table.
- Decoding is performed by taking each code from the compressed file and translating it through the code table to locate characters.

Run-length encoding performs lossless data compression suitable to palette-based iconic images [4, 11] .

B. ***LZW Compression***

Concept of the compression algorithm: As the input data is processed, a dictionary maintains a communication between the longest words and codes. The words are replaced by subsequent codes and thus the input file is compressed. The effectiveness of the algorithm increases as the number of long words increases.

PSEUDOCODE

```
Initialize table with single character strings
P = first input character
WHILE not end of input stream
   C = next input character
   IF P + C is in the string table
      P = P + C
   ELSE
    output the code for P
    add P + C to the string table
    P = C
   END WHILE
output code for P
```

Run-Length encoding (RLE) is image compression consisting of similar symbols by a pair containing the symbol and the run length [7]. In RLE, runs of data are stored as a single data and count, rather than initial run. This is useful as it contains several runs: for example, graphic images such as icons and animations. It is not

feasible with files which do not include a number of runs as it could increase the file size twice. Run-length encoding performs lossless data compression and feasible to palette-based iconic images [3, 11, 9].

C. LZW Decompression

During decompression the LZW decompressor produces the identical string. It initiates with the first 256 table entries starting to single characters. The string table is updated for character in the input stream, except the first one. Decoding is received by reading codes and translating throughout the code table.

```
PSEUDOCODE
Initialize table with single character strings
OLD = first input code
output translation of OLD
WHILE not end of input stream
   NEW = next input code
   IF NEW is not in the string table
        S = translation of OLD
        S = S + C
   ELSE
        S = translation of NEW
          output S
          C = first character of S
          OLD + C to the string table
          OLD = NEW
END WHILE
```

III. METHODOLOGY

In order to compare between Huffman coding and LZW coding a code for both Huffman and LZW coding using above mentioned algorithm was written using python 3.6 as a programming language. In order to obtain the data, the text size was determined first and then test were carried out accordingly in the Huffman python source code and LZW python source code. Compression time and the decompression time on both the coding techniques were noted for different input text size. Compression ratio in the LZW coding was calculated by the initial text size and the compressed text size for all the input text size. Whereas the compression ratio for the Huffman coding is little different compared to the LZW coding. The initial text was changed into the ASCII format and calculated the total length of the changed text. Then the generated Huffman Code is applied to the initial text now the total compressed text length is divided by the length of the ASCII changed code and multiplied by hundred, hence we get the compression ratio for the Huffman coding.

IV. RESULTS

The results of the algorithms, Huffman and LZW compression time and decompression time are highlighted in Table 3.

Performance Parameters: Performance evaluation of the compression algorithm is performed by using Compression Ratio, Compression Time and Decompression Time.

- **Compression ratio: T**he ratio of size of the input text to the size of the compressed text.
  Compression ratio= (C2/C1) *100%
- **Compression Time:** The total time taken to run the compression algorithm.
- **Decompression Time:** The total time taken to execute the decompression algorithm

C1= Size of input text before compression

C2= Size after text after compression

Table 3. Compression Table

	Huffman Coding			Lempel–Ziv–Welch (LZW) Coding		
Input text size	**Compression Ratio(%)**	**Compression Time**	**Decompression Time**	**Compression Ratio (%)**	**Compression Time**	**Decompression Time**
2.2 KB	54.20	0.0	0.0156	105.56	0.0	0.0
6.7 KB	54.20	0.0	0.069	84.46	0.0	0.0
33.2 KB	54.205	0.015	0.231	49.94	0.0156	0.0156
198.8 KB	54.204	0.0625	1.402	24.219	0.052	0.0159
994.1 KB	54.203	0.601	7.076	12.458	0.284	0.115
3.9 MB	54.203	8.574	30.029	6.319	1.622	1.624
7.8 MB	54.203	34.266	55.879	5.062	3.224	6.852
15.5 MB	54.203	123.664	115.858	3.604	8.560	29.734
31.1 MB	54.203	548.473	237.528	2.566	17.352	103.317
62.1 MB	54.203	1945.389	470.933	1.827	53.198	469.634

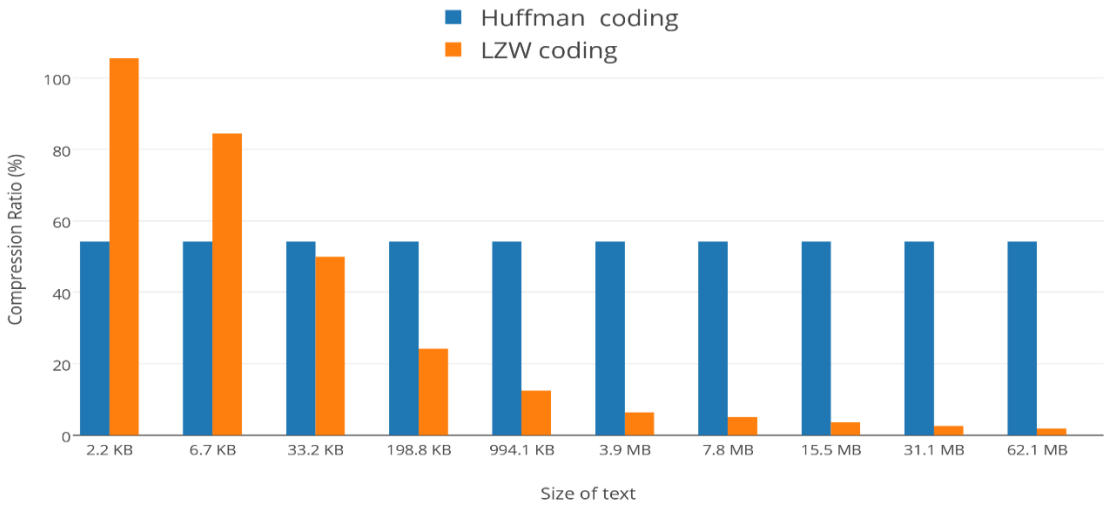


Figure 1. Compression Ratio chart

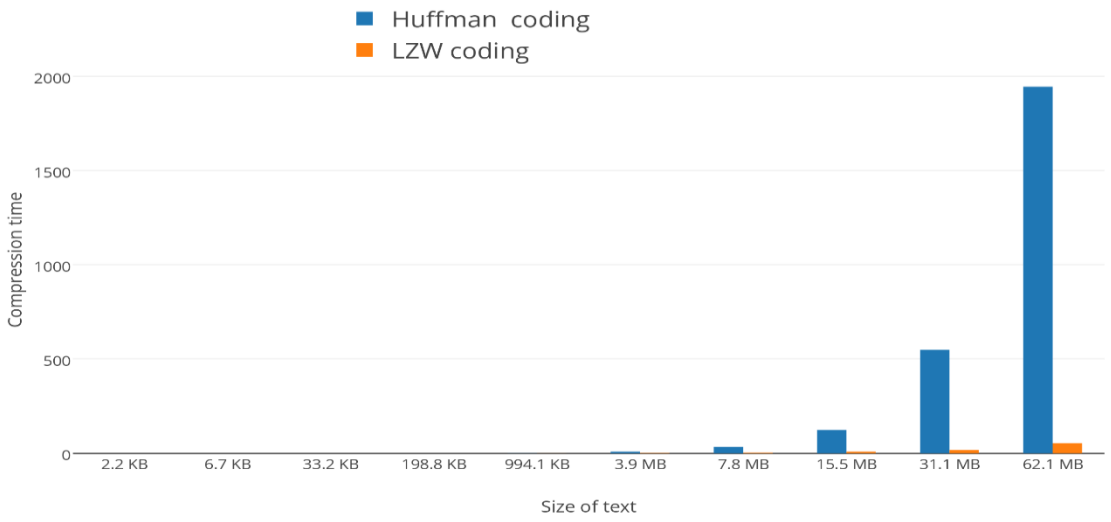


Figure 2. Compression Time chart

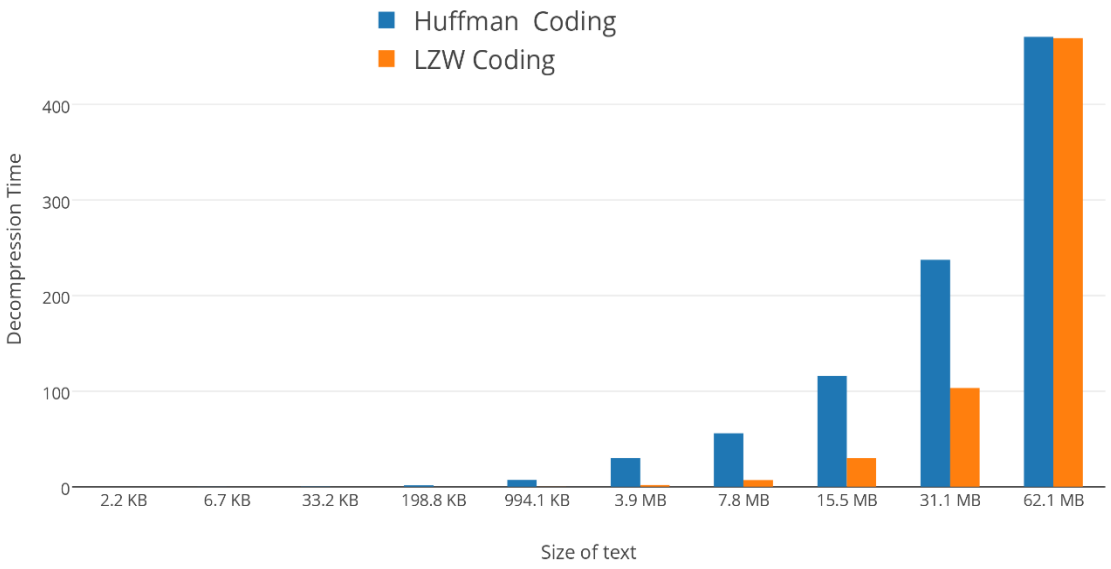


Figure 3. Decompression Time chart

Outcome 1

From the Figure 1 – we can observe that if the size of the input text is small than the compression ratio of the LZW coding is very high comparing to the Huffman coding, whereas when the size of the file is very high than the compression ratio of the LZW coding is very low which suggest that for smaller text size Huffman Coding is appropriate and when the input text size is high than the LZW coding is appropriate.

Outcome 2

From the Figure 2- Compression Time chart, we can get that for the small input text size the time taken for the compression for the both algorithm is same whereas if the input text size is high than the compression time for the Huffman coding is very high, hence LZW coding is appropriate for the big size of text.

Outcome 3

From the Figure 3- Decompression Time chart, we can get that for the small input text size the time taken for the compression for the LZW algorithm is small whereas if the input text size is high than the compression time for both the algorithm is almost same, hence LZW coding is appropriate for the decompression for small size text where as both algorithm perform well for big text size.

V. DISCUSSION

A Huffman encoder accepts input characters with limited length and establishes output bits. It is a fixed-to-variable length code producing a optimal prefix codes O(nlogn). Lempel-Zi is a variable-to-fixed length code. The architecture of the Huffman code is suitable for fixed block length. The Lempel-Ziv code is not suitable for specific source but for a large sources. The Lempel-Ziv algorithm acts as if it was designed for that particular source. Owing to this reason, the Lempel-Ziv code is the used for lossless compression.

LZ compression replaces the repeated patterns. The larger the dictionary size, the greater the bits which are mandatory for the references. The most favorable size of the dictionary changes for different categories of data. As such the more changeable the data, the smaller the optimal size of the directory.

VI. CONCLUSION

Compression is significant technique in multimedia. The data size can be reduced, as such transmitting and storing the reduced data are cheaper and faster. A number of images and video compression such as JPEG, JPEG 2000, and MPEG-2, and MPEG-4 are implemented. In this study, we have focused on these algorithms to clarify differences such as compression ratio, compression time and decompression time. Huffman coding is well-situated than LZW coding. LZW coding facilitates more compression ratio than Huffman algorithm. Huffman coding requires more execution time than the LZW. In some cases time is not important as Huffman coding can be used to obtain high compression ratio. Whereas for some applications the time is important for real-time applications and LZW coding.

REFERENCES

[1] https://en.wikipedia.org/wiki/Huffman_coding

[2] http://www.geeks3.forgeeks.org/lzw-lempel-ziv-welch-compression-technique/

[3] https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node210.html

[4] https://www.researchgate.net/publication/220114874_A_Memory-Efficient_and_Fast_Huffman_Decoding_Algorithm

[5] https://www.researchgate.net/publication/4140913_A_Memory efficient_Huffman_Decoding_Algorithm

[6] "Introduction to Data Compression", by Guy E. Blelloch, Carnegie Mellon University, (October 16, 2001).

[7] M. Rabbani, and P. W. Jones, "Digital Image Compression Techniques", (Edition-4), p. 51, 1991.

[8] S. Saha, "Image Compression – from DCT to Wavelets: A Review", Unpublished.

[9] http://www/acm/org/crossroads/xrds6-3/sahaimgcoding.html.

[10] J.J. Rissanen and Langdon G. G. Jr., "Arithmetic Coding" (PDF), *IBM Journal of Research and Development*, Vol. 23, issue 2, pp. 149-162., 1979

[11] http://researchweb.watson.ibm.com/journal/rd/232/ibmrd2302G.pdf. Retrieved 2007-09-22

[12] S. Kajihara, "On Combining Pinpoint Test Set Relaxation and Run-Length Codes for Reducing Test Data Volume," *In Proceedings of the 21st International Conference on Computer Design (ICCD '03)* , San Jose, Calif, USA, 2003.

[13] ITU-T Rec. T.81 ISO/IEC 10918-1, "Information Technology — Digital Compression and Coding of Continuous-Tone Still Images: Requirements and Guidelines," 1994

[14] http://my.smithmicro.com/win/index.html

[15] http://www.wfu.edu

