ISSN (Print) : 2319-8613
ISSN (Online) : 0975-4024

Comparison of Text Data Compression Using Run Length Encoding, Arithmetic Encoding, Punctured Elias Code and Goldbach Code

Kenang Eko Prasetyo[1], Tito Waluyo Purboyo [2], Randy Erfa Saputra[3]
Computer Engineering, Faculty of Electrical Engineering, Telkom University, Bandung, Indonesia
E-mail: kenangekoprasetyo@student.telkomuniversity.ac.id[1] ,
titowaluyo@telkomuniversity.ac.id[2], resaputra@telkomuniversity.ac.id[3]

***Abstract*— In computer science, data compression or bit-rate reduction is a way to compress data so that it requires a smaller storage space making it more efficient in storing or shortening the data exchange time. Data compression is divided into 2 parts, Lossless Data Compression and Lossy Data Compression. Examples of Lossless methods are: Run Length, Huffman, Delta and LZW. While the example of Lossy method is: CS & Q (Coarser Sampling and / or Quantization). This paper analyze the Lossless method using Run Length Encoding (RLE) Algorithm, Arithmetic Encoding, Punctured Elias Code and Goldbach Code. This Paper also draw a comparison between the four algorithms in order to determine which algorithm is more efficient in doing data compression.**

Keyword - *Data Compression*, Run Length Encoding, Arithmetic Encoding, Punctured Elias Code, Goldbach Code

I. INTRODUCTION

As the development of science and technology, specifically information technology and telecommunications (ICT) intensifies, the number and size of data files processed or transmitted through computer networks is also increasing. This raises its own problems. Limited bandwidth and speed in retrieving and transmitting data files over the global network of the internet or known as downloads or uploads is often a time-consuming job. Connections are often disconnected and the process of retrieving or transmitting data will have to be repeated.

Various data compression methods have been developed and can generally be grouped into two major groups: lossless data compression and lossy data compression. In lossless data compression, the process of reconstructing the compressed or decompressed file will be able to return it to its original form. This method is used for various data types such as executable code or word processing files, where the slightest difference will result in a fatal error.

While lossy data compression is generally used for data files that represent an image where the reconstruction process does not return the image form to the original form with the same quality so that lossy data compression has a better compression ratio than lossless data compression.

Based on the above background, the authors will make an analysis of the RLE algorithm (Run Length Encoding), Arithmetic Encoding, Punctured Elias Code and Goldbach Code which are included in lossless data compression. After doing the analysis, the authors will do a comparison between the three algorithms to draw the conclusion of which algorithm is more efficient and possesses the greater compression ratio.

II. CONTRIBUTION

Based on paper [2], the selected two methods of lossless data compression that are widely used and used as the basis for the development of other data compression methods are Arithmetic Coding and Run Length Encoding. Implementation of Arithmetic Coding should pay attention to encoder and decoder capabilities, which generally have limited number of mantissa. This can cause errors or errors if an Arithmetic Coding has code with a very long floating point. In implementing the compression program using this RLE algorithm, note that if the original file has the same character with the marker bit, then the character encoding file is written twice in succession. At the time of return to the original file, if it is found that the bits of the marker are lined up twice, that means the character is not the marker bit, but the original character of the original file.

This paper uses 4 methods for data compression. These are Run Length Encoding, Arithmetic Encoding, Punctured Elias Code and Goldbach Code. The Goldbach Codes algorithm is an algorithm that is assumed to use the Goldbach Conjecture theory of "All positive even numbers greater than 2 is the sum of two primes".

The data as much as 40 bits or equal to 5 bytes when compressed using Run Length Encoding algorithm to 80 bits, the second using the compression Arithmetic Encoding with the same character after the compression to produce 32 bits, the third by using the Punctured Elias Code after the compression to produce the number of bits more that 64 bit. Meanwhile, using Goldbach Code produces the same number of bits as the original is 14 bits.

Next experiment with characters equal to 10 bytes or equal to 80 bits. When compressed using the Run Length Encoding algorithm can be compressed into 64 bits to be smaller than the number of bits before being compressed, then using the Arithmetic Encoding the number of bits produced 56 bits, then using Punctured Elias Code Algorithm 75 bits. The next step is to compress using the Goldbach Code Algorithm and generate the number of bits smaller than the original that is 28 bits.

Last experiment uses more bytes of 15 bytes or equal to 120 bits with characters when compressing using the Run Length Encoding algorithm produced is 240 bits become larger using RLE algorithm because the various processes are compressed to vary and do not have the same character so that the number of bits produced more many, then compresses using the Arithmetic Encoding Algorithm 65 bits. Third do the compression with Punctured Elias Code algorithm is 88 bits. Last compression using Goldbach Code the number of bits produces is 49 bits.

Paper [3] selected two methods. These are Punctured Elias Code and Ternary Comma Code. In the Punctured Elias Codes method there are two codes called P1 and P2 codes and which will be discussed in this study only P1 code only. In the compression process, the system initially reads the String contained in the text file and then creates the character table and the frequency of occurrence of that character that has been sorted by the character having the largest frequency. After that the system generates the Punctured P1 code based on the number of characters on which it has been created. Then the system performs the compression process based on the character table and the code table that has been raised it [3].

Paper [4] selected two methods. These are Punctured Elias Code and Huffman Method. Basically these two algorithms have the same way of working. Begin by sorting characters based on their frequency, forming binary trees and ending with code formation. In the Huffman algorithm, the binary tree is formed from the leaves to the roots and is called the formation of the tree from the bottom up. In contrast, the Punctured Elias Codes method has two codes: P1 and P2. Text file compression is done by reading the String input in the text file (* .txt and * .doc) and encoding the String using Punctured Elias Code P1 or P2, then performing the compression process. The end result of compression is a file extension *. The header as the code and character information of the string and * .pec which is a string of compressed decompressable results.

III. RESEARCH METHOD

In this paper, we use 4 different algorithms. These are RLE, Arithmetic Encoding, Punctured Elias Code and Goldbach Code. Run length encoding Algorithm is a form of technique used to compress data containing repetitive characters. Run-Length Encoding is an algorithm that utilizes characters that repeated sequentially in the data by encoding it with a string consisting of the number of character looping that occurs, followed by a repeating character. So the number of recurring characters in the data is the determinant of the success of compression algorithm RLE. In general, optimal RLE algorithm is used on files that have characters that tend to be homogeneous.

In general, data compression algorithms are based on the selection of ways of replacing one or more of the same elements with a particular code. Arithmetic Coding replaces a series of input symbols in a data file with a number using an arithmetic process. The longer and more complex the encoded message, the more bits it takes to compress and decompress the data.

IV. DATA COMPRESSION

In the compression process, in general, the first thing to do is read the required information from a file as an initialization process. Information owned by a file can be a file type, file location and the main one is the file size. From the initial initialization process can be defined information required in the design and development of compression applications.

The data to be created, among other things, the file naming, the location of the file's origin directory, the size of the file, the directory location and the file name of the compressed file are stored. The next process is to search and read all characters and their occurrence frequency in the file.

4.1. Compressing Data Using Run Length Encoding

Compression and decompression using Run Length Encoding is a method used to compress data containing repetitive characters. When the same characters are received consecutively four or more times (more than three), this algorithm compresses the data in a series of three characters. Run-Length Encoding is an algorithm that exploits repetitive characters in sequence in a data by encoding it with a string that consists of the sum of the number of character looping that occurs, followed by a repeating character.

Generally, optimal RLE algorithm is used on files that have characters that tend to be homogeneous. Therefore, if the algorithm is used universally then it is necessary to do the grouping or transformation of similar characters / symbols. The steps of compressing data using Run Length Encoding are:

1. Techniques to reduce the size of data containing Repeating symbols.
2. Repeating symbols are designated: *run*
3. 1 *run* is usually encoded into a 2-byte data.
4. The first byte represents the number of symbols, called *run count*.
5. The second byte represents a repeating symbol, called *run value*.
6. RLE is suitable for compressing text data that contains many symbol repetitions

For example the data AAAAAABBBXXXXXT has the value of 15 Byte, With RLE, it can be encoded into = 6A 3B 5X 1T (8 Byte).

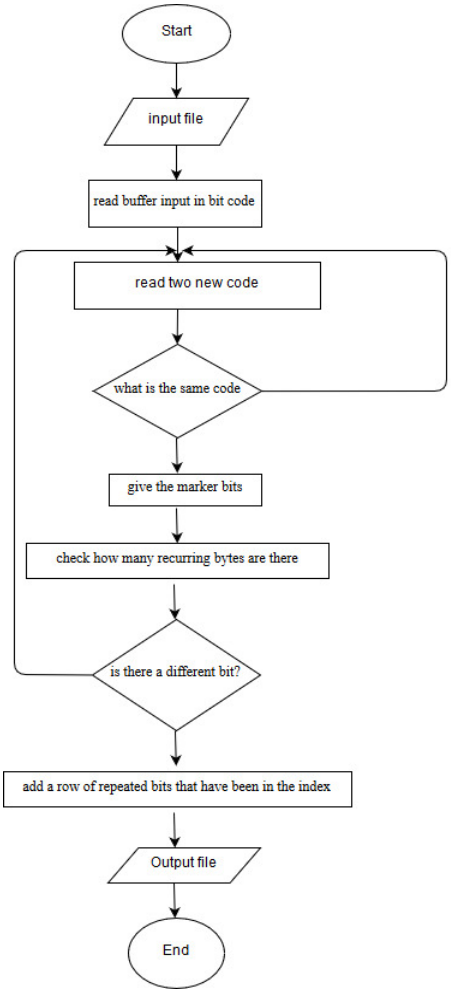

Figure 1. Flowchart Run Length Encoding Algorithm

4.2. Compressing Data Using Arithmetic Encoding

In general, data compression algorithms are based on the selection of ways of replacing one or more of the same elements with a particular code. In contrast, Arithmetic Coding replaces a series of input symbols in a data file with a number using an arithmetic process. The longer and more complex the encoded message, the more bits it takes to compress and decompress the data.

The output of this Arithmetic coding is a number smaller than 1 and greater or equal to 0. This number can be uniquely decompressed to produce a sequence of symbols used to generate that number. To generate the output number, each symbol that will be compressed is assigned a set of probability values. The implementation of Arithmetic Coding should closely regard the capabilities of the encoder and decoder, which generally have a limited number of mantissa. This can cause errors if an Arithmetic Coding has code with a very long floating point.

The steps to compressing data using Arithmetic Encoding algorithm are:

1. Set Current Interval [0,1)
2. Divide the current interval into several sub intervals based on symbol probability; one sub interval represents one symbol.
3. Read the "i" symbol on the message, identify the sub interval belonging to that symbol.
4. Set Current Interval = sub interval symbol to "i".
5. Repeat steps 2-4 until the last symbol on the message.

4.3. Compressing Data Using Punctured Elias Code

Punctured Elias Codes is a method designed by Peter Fenwick in an experiment to improve the performance of the Burrows-Wheeler transform. The term "Punctured" comes from Error Code Memory (ECC). ECC consists of the original data in addition to an array of numbers from check bits. Some check bits are omitted to shorten a series of codes.

The steps to compress data using Punctured Elias Code algorithm are:

1. Take a binary number from "n".
2. Reverse the bit and prepare a flag to show the number of bit that holds the value of 1 in "n".
3. For every 1 bit in "n", we ready a flag from 1 and ending with a flag with 0.
4. Combine the flag with the reversed binary number

Based on Figure 2 it is clear that to compress the .doc file is started by inputting the .doc file then encoded with the code of the Punctured Elias Codes. Next is compressed by the Punctured Elias Codes method will generate output file compression. Then analyzed with parameters RC, CR, Space Savings (SS) and Time Prosses up get results from their respective calculations.

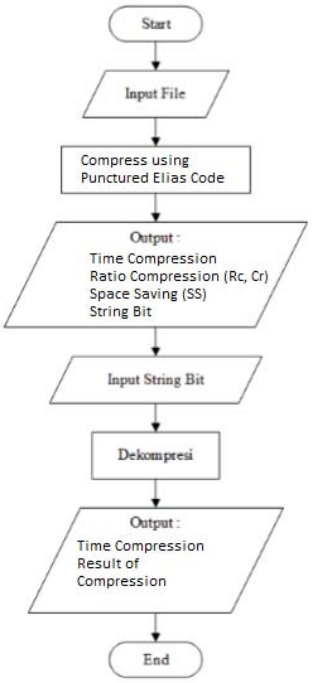

Figure 2. Flowchart Punctured Elias Code Algorithm

4.4. Compressing Data Using Goldbach Code

The Goldbach Codes algorithm is an algorithm that is assumed to use the Goldbach Conjecture theory of "All positive even numbers greater than 2 are the sum of two prime numbers. Goldbach Codes has three codes, the first Goldbach Code is called "G0". G0 encodes the positive integer "n" by converting it into an even positive integer with 2 (n + 3) and then writing the sum of the prime numbers in reverse.

The second Goldbach Codes are called "G1". In principle, G1 is to determine two primes; Pi and Pj (where $i \leq j$), whose sums produce integers n and encode pairs (i, j-i + 1) with gamma codes. While the third G2 or Goldbach Codes is an extension of the G1 Code with cases like the following:

1. The integers 1 and 2 are encoded to 110 and 111 (other than integers 1 and 2 no other encoding starts with 11 …)

2. Even integers are encoded the same as in G1 Code but with an exception; If n = Pi + Pj is determined, it will be encoded by the pair (i +1,j – i + 1) replacing (i,j – i + 1). Thus, if i = 1, it will be encoded to 010, the gamma code of 2. This will guarantee that even integers of G2 Code will not start from 1 and will always have the form 0 ...: 0 ...
3. If n is a prime Pi number, it will be encoded as a gamma code of (i+1)) followed by 1 single to produce 0 …:1.
4. If n is an odd number, then G2 code starts from a single 1, followed by a G2 Code from an even number n-1. Generating a gamma code with the form of 1:0…:0…

V. TESTING AND RESULT

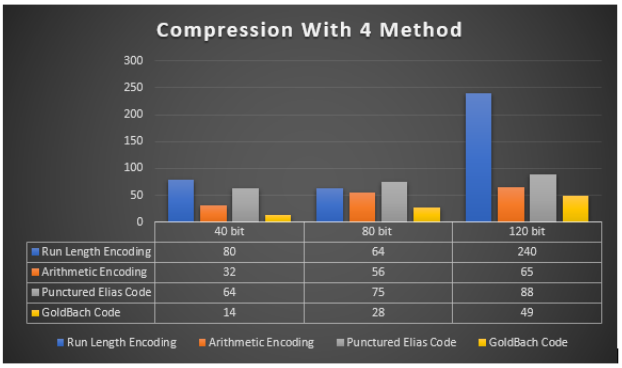

Figure 3. Result of Compression with 4 Methods

In Figure 3 Can be seen the graph for the first experiment. The blue line shows that we compressed the data as much as 40 bits or equal to 5 bytes when compressed using Run Length Encoding algorithm to 80 bits, the second using the compression Arithmetic Encoding with the same character after the compression to produce 32 bits, the third by using the Punctured Elias Code after the compression to produce the number of bits more that 64 bit. Meanwhile, using Goldbach Code produces the same number of bits as the original is 14 bits.

The next experiment with characters equal to 10 bytes or equal to 80 bits when compressed using the Run Length Encoding algorithm can be compressed into 64 bits to be smaller than the number of bits before being compressed, then using the Arithmetic Encoding the number of bits produced 56 bits, then using Punctured Elias Code Algorithm 75 bits. The next step is to compress using the Goldbach Code Algorithm and generate the number of bits smaller than the original that is 28 bits.

The next experiment uses more bytes of 15 bytes or equal to 120 bits with characters when compressing using the Run Length Encoding algorithm produced is 240 bits become larger using RLE algorithm because the various processes are compressed to vary and do not have the same character so that the number of bits produced more many, then compresses using the Arithmetic Encoding Algorithm 65 bits. Third do the compression with Punctured Elias Code algorithm is 88 bits. Last compression using Goldbach Code the number of bits produces is 49 bits.

VI. RATIO COMPRESSION

The results of the compression ratio performed in the experiments can be seen at Figure 4.

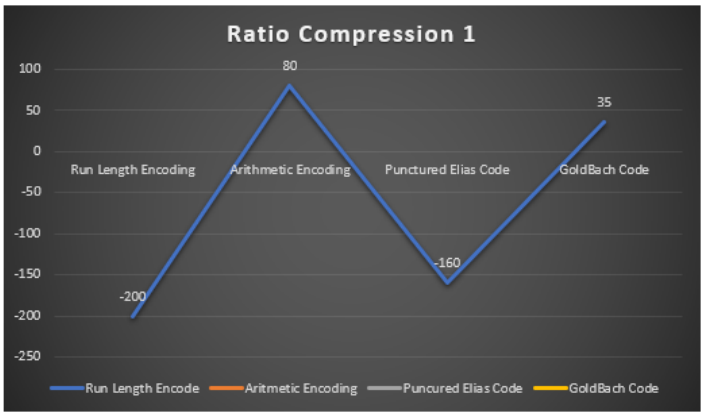

Figure 4. Ratio Compression 1

Figure 4 shows the result of compression ratio using characters with the number of 5 bytes / 40 bits. When using the Run Length Encoding algorithm has number of ratio compression -200% , then the original bit number compressed using the Arithmetic Encoding algorithm yields a compression ratio of 80%. Then the compression using the Punctured Elias Code algorithm yields a -160%. Then using the Goldbach Code has number of ratio Compression 35%.

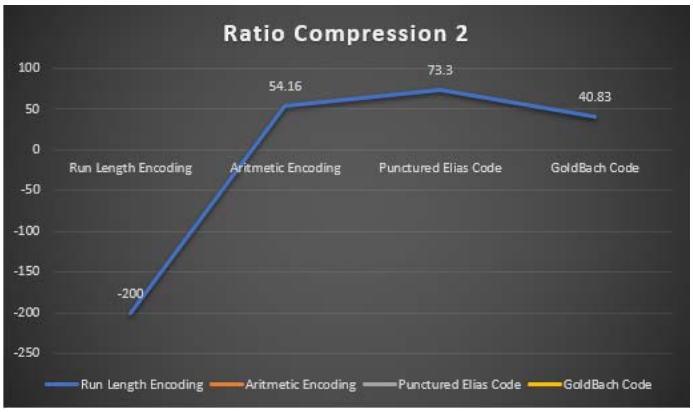

Figure 5. Ratio Compression 2

Figure 5 shows the result of compression ratio using characters with the number of 10 bytes / 80 bits. When using the Run Length Encoding algorithm has number of ratio compression 80%, then the original bit number compressed using the Arithmetic Encoding algorithm yields a compression ratio of 70%. Then compression using the Punctured Elias Code algorithm yields a 93.75%. Then using the Goldbach Code has number of ratio Compression 35%.

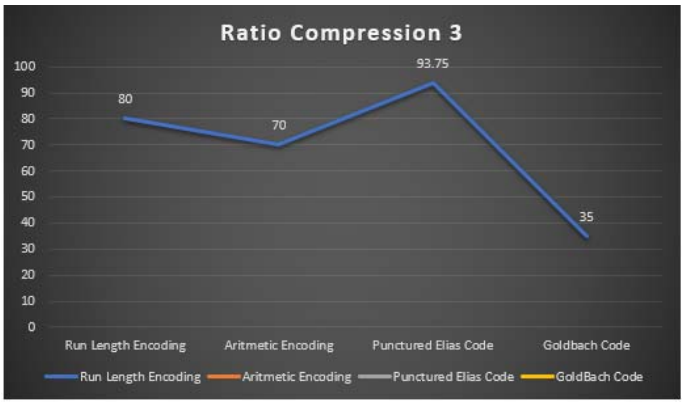

Figure 6. Ratio Compression 3

Figure 6 shows the result of compression ratio using characters with the number of 10 bytes / 80 bits. When using the Run Length Encoding algorithm has number of ratio compression -200%, then the original bit number compressed using the Arithmetic Encoding algorithm yields a compression ratio of 54.26%. Then compression using the Punctured Elias Code algorithm yields a -73.3%. Then using the Goldbach Code has number of ratio Compression 40.83%.

VII. CONCLUSION

Data compression algorithm can be applied to text, image and video files. This compression algorithm can reduce the file size. This is useful because the small file size can save storage space on the computer. When compressing using the Arithmetic Encoding and Goldbach Code result in the number of bits smaller, reducing lightweight storage and performing data transmission easily. However, when compressing using the RLE method the number of bits in the compression will become larger because there is redundancy in the uncompressed text.

REFERENCE

[1] David Solomon. Variable-length Codes for Data Compression. California, Springer Science+Bussiness Media. 2007. Page 120
[2] Desy H Husni. IMPLEMENTATION AND ANALYSIS OF COMPRESSION ALGORITHMS PUNCTURED ELIAS CODES AND TERNARY COMMA CODE IN .DOC FILE. Universitas Sumatera Utara. 2016
[3] Cut Try Utari, " Implementasi Algoritma Run Length Encoding Untuk Perancangan Aplikasi Kompresi dan Dekompresi File Citra", Jurnal TIMES, Vol.V No.2, 24-31, 2016.
[4] Monica Borda. Fundamentals In Information Theory and Coding. Romania. Springer-Verlag Berlin Heidelberg. Page 95
[5] Peter Fenwick. Variable-Length Integer Codes Based on the Goldbach Conjecture, and Other Additive Codes. IEEE Transactions on Information Theory, Vol. 48, No. 8, pages 2412-2417, 2017
[6] Eric Bodden, Malte Clasen, and Joachim Kneis. Arithmetic Coding Revealed. Sable Technical Report, No. 5, 2007.
[7] Paul G Howard and Jeffrey Scott Vitter. Analysis Of Arithmetic Coding For Data Compression. Brown University. 1992.
[8] Peter Fenwick. Punctured Elias Code for Variable-Length Coding of the Integer. Auckland University. 1996
[9] Senthil Shanmugasundaram, Robert Lourdusamy "A Comparative Study Of Text Compression Algorithms". International Journal of Wisdom Based Computing, Vol. 1 (3), September 2017
[10] Haroon A, Mohammed A "Data Compression Techniques on Text Files: A Comparison Study" International Journal of Computer Applications (0975 – 8887) Volume 26– No.5. September 2017
[11] Ritu Antil, Sandeep Gupta. Analysis and Comparison of Various Lossless Compression Techniques. Deenbandhu Chotu Ram University. 2014
[12] Eric Bodden, Malte Clasen, and Joachim Kneis. Arithmetic Coding Revealed. Sable Technical Report, No. 5, 2007.

AUTHOR PROFILE

Kenang Eko Prasetyo is a student of Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University.

Tito Waluyo Purboyo is currently a lecturer and researcher at Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University. He received his Master degree in Mathematics from Institut Teknologi Bandung (2009). He received his Doctoral degree in Electrical Engineering and Informatics from ITB (2016). He is also member of IEEE. His research interest include Security, Cryptography, Mathematics, Steganography and Information theory.

Randy Erfa Saputra is a lecturer at Department of Computer Engineering, Faculty of Electrical Engineering, Telkom University since January 2015. He received his Master degree in Computer Engineering from Institut Teknologi Bandung (2012). His research interest include Network and Multimedia .