International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 23 (2017) pp. 13618-13622

Comparison of Text Data Compression Using Huffman, Shannon-Fano, Run Length Encoding, and Tunstall Methods

Dea Ayu Rachesti
College Student, Faculty of Electrical Engineering,
Telkom University, Bandung, Indonesia.
Jl. Telekomunikasi No. 01, Terusan Buah Batu, Sukapura, Dayeuhkolot, Sukapura, Dayeuhkolot, Bandung, Jawa Barat 40257.

Orcid Id : 0000-0002-1836-2339

Tito Waluyo Purboyo
Lecturer, Faculty of Electrical Engineering,
Telkom University, Bandung, Indonesia.
Jl. Telekomunikasi No. 01, Terusan Buah Batu, Sukapura, Dayeuhkolot, Sukapura, Dayeuhkolot, Bandung, Jawa Barat 40257.

Orcid Id : 0000-0001-9817-3185

Anggunmeka Luhur Prasasti
Lecturer, Faculty of Electrical Engineering,
Telkom University, Bandung, Indonesia.
Jl. Telekomunikasi No. 01, Terusan Buah Batu, Sukapura, Dayeuhkolot, Sukapura, Dayeuhkolot, Bandung, Jawa Barat 40257.

Orcid Id : 0000-0001-6197-4157

Abstract

Data compression is a way to condense a data so that data storage is more efficient and requires only smaller storage space. In addition, with data compression can shorten the time of data exchange. Currently there are many methods that can be used to compress data. And each method has different results and ways. In this paper we will discuss the comparison of data compression using 4 different algorithms, there are using Shannon-Fano Algorithm, Huffman Algorithm, Run Length Encoding Algorithm and the last Tunstall Algorithm.

Keywords: Data Compression, Huffman Algorithm, Shannon Fano Algorithm, Run Length Algorithm, Tunstall Algorithm

INTRODUCTION

The problem of data compression is one of the important aspects in the development of information technology. Data compression is a process of resizing a file or document to be smaller in size. Along with the development of hardware and software technology is increasingly sophisticated and complex that demands the efficiency in terms of data storage and memory. Therefore, data compression is important in the process of transfer and acceleration in data transmission as well as the efficiency of data storage capacity or documents. This type of data compression is divided into 2 parts, namely: Lossy compression and Lossless compression. Lossy compression is a type of compression that can cause data changes after the compression process. While Lossless compression is where there is no change in the data after the compression process. Examples of these lossless compression algorithms are the Huffman Algorithm, then the Dynamic Markov algorithm, Run Length Encoding, LZW, Wheeler Transform Burrows, Shannon Fano, Tunstall and PPM (Prediction by Partial Matching) algorithms. This compression process begins with input in the form of context or data to be processed into a modeling. Then from the modeling stage will be distributed a probability of the characters / symbols that appear. After that, the symbol / character that appears will be encoded according to the selected algorithm type, depending on whether the algorithm is two-pass or one-pass, lossy or lossless, symbolwise or dictionary. And from this code is formed bit bits are simpler than the symbol or characters are inputted.

RELEVANCE OF RESEARCH

Data compression is used for smaller storage space. Basically on the Huffman algorithm and Shannon algorithm it uses the same method in making short code [3]. Initially, this algorithm creates a tree in the form of a leaf node and its children which has a

probability of the frequent appearance of the character in the text. Then the second process is the encoding process. Of the tree, each character will have the identity of a binary number to store memory. The process of formation of the character into a binary is called the encoding process. While the Run Length Encoding algorithm is not good to use if the data / sentence contains meaning. Because Run Length Encoding will result in a larger bit value if it is not used in repeated words.

Basic Theory

Data compression is a way to compress data so that it only requires smaller storage space so it is more efficient in storing it or shorten the time of data exchange [10].

Data Compression

Data compression has two types of data compression, ie lossless data compression and lossy data compression. In this paper we will explain the comparison of data compression in text using 4 algorithms, Huffman algorithm, Shannon algorithm, Run Length Encoding algorithm, and Tunstall algorithm [4] [5].

Huffman Algorithm

The Huffman method was made by an MIT student named David Huffman in 1952. The Huffman method is one of the oldest and most famous methods of text compression. There is no method used for lossless type compression, where the compressed data can be restored to its original form intact. This Huffman method works just like a morse code machine, ie every character or symbol is encoded with only a few bits of sequence, where characters that often appear are encoded with short bits of sequences and rarely appearing characters encoded with longer bits. This method belongs to a class that uses static methods. Two-step method: the first step to calculate the probability of occurrence of each symbol and to specify the code map, and second phase to convert the message into a collection of code to be transmitted. While based on symbol coding technique used, Huffman method uses symbolic method. The symbolwise method is a method that displays the appearance of each symbol at a time, where symbols appear more frequently. In general, this method is used for text data compression [2] [3].

Huffman code is basically a prefix code (prefix code). The prefix code is usually represented as a binary tree given a value or label. For the left branch in the binary tree is labeled 0, while on the right branch is labeled 1. The sequence of bits formed on each path from root to leaf is a prefix code for matching characters. This binary tree is called the Huffman tree [6] [9].

Theres a step for data compression using Huffman Algorithm :

1. First, sort symbols or characters based on their probabilities descending
2. If the probability is the same, sort the index of symbols / letters descending as well.
3. Then take the two symbols with the smallest probability, the upper symbol is given the '1' bit, the symbol under bit '0', merge into new symbol, and sum up the probability
4. Rework the symbols like the first step
5. If the probability is the same, the latest symbol is under the old symbol
6. Then repeat steps 2 and 3 repeatedly until the probability sum = 1.0
7. Then specify the codewords of each symbol with binary

Shannon-Fano Algorithm

The Shannon-Fano algorithm was discovered by Claude Shannon (father of information theory) and Robert Fano in 1949. This method was by then the best method, but after the Huffman algorithm, Shannon's algorithm was almost never used and developed. Basically this method replaces each symbol with a binary code whose length is determined based on the probability of the symbol. In the field of data compression, Shannon-Fano coding is a technique for building a prefix code based on a set of symbols and probabilities. However, this algorithm is not able to achieve the code as efficiently as Huffman's algorithm [4] [8].

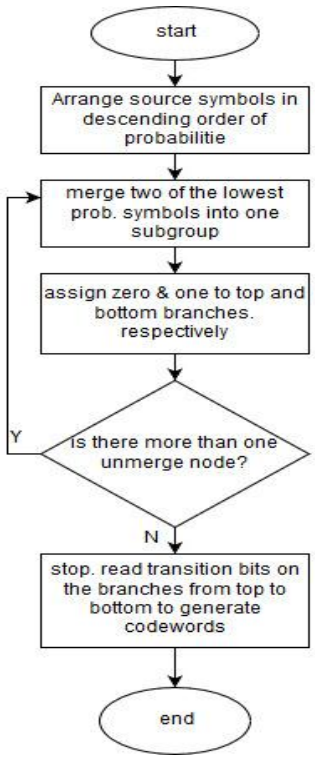

Figure 1: Flowchart of Huffman Algorithm [13]

Here is how to compress data using Shannon Fano Algorithm:

1. First, sort the symbols by descending frequency of occurrences or probabilities
2. If the frequency is the same, then sort the ascending symbol index
3. Then divide the symbols into 2 groups with the minimum difference possible
4. Do keep step 3 so that each group has 1 symbol
5. Once done, then make the code according to the binary tree.

Run Length Encodding Algorithm

RLE (Run Length Encoding) is the easiest form of lossless data compression technique where a series of data with the same value are sequentially stored into a data value and the amount. RLE algorithm is very useful for data that has a lot of data with the same value in sequence such as file icons, line drawings, and animation. RLE algorithm (Run Length Encoding) is an algorithm that can be used to perform data compression so that the resulting data size becomes lower than the actual size. RLE is not very suitable applied to data that has meaning, because it will result in increasing the size of data compression than the initial data [1] [7].

Theres how to compress data using Run Length Encoding :

1. First, see if there are the same sequence of characters in sequentially more than three characters, if they do, compress. Suppose on a row the same characters in sequence as much as 8 character, so more than three and could do compression.
2. Then provide the marker bit in the compression file, bit the markers are 8 rows of bits to choose from just as long as it is used consistently on all bits of compression marker. Bit of this marker-

   serves to mark that the next character is a compression character, so it is not confusing at the time of restoring files that have been compressed into the original file.
3. And then, add a row of bits to declare the amount the same characters in sequence
4. Add a row of bits that represents a repeating character [13].

Example: An AAAABBCC string is represented in 8 bytes of data, using the RLE algorithm, encoded into: 3A2B2C = 6 bytes of data.

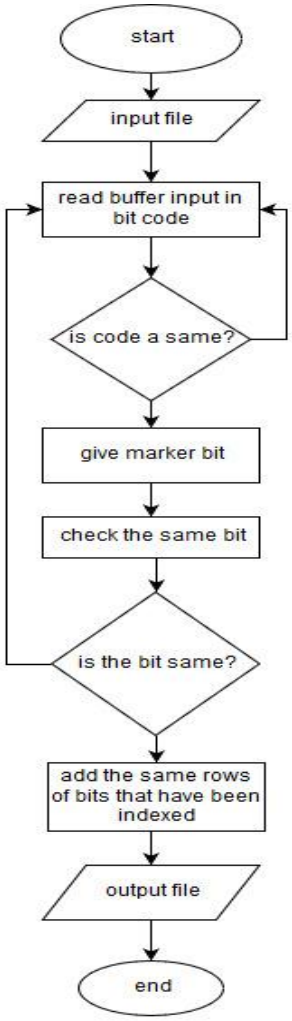

Figure 2: Flowchart of Run Length Encoding Algorithm [13]

Tunstall Algorithm

Tunstall algorithm is one method to compress lossless data. In this algorithm the first step to do is create a table containing symbols, frequency, and probability columns. After that sort the symbol according to the biggest probability. Then do the literacy, to know how many literation to do that is by entering into the formula N + k (N-1) <= 2 ^ N. Then do the literacy in accordance with the results k obtained. To do literacy, first sort the symbols according to the largest probability, then remove the symbol with the highest probability. After that, include the symbol with the symbol in the initial table [12] [14].

EXPERIMENT AND RESULTS

The author has made several experiments using 4 different algorithms.

The Result of Data Compression

There are the result of Data Compression's chart:

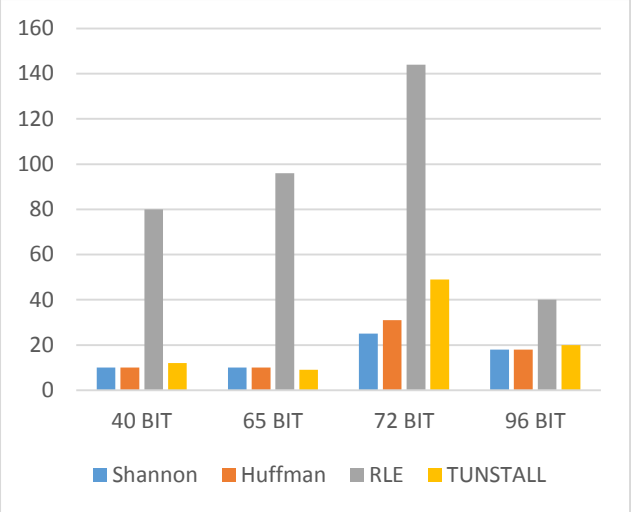

Figure 3: The Result of Chart

From the chart in figure 3, it can be seen, in the first experiment, the author tries to compress the word which has 5 bytes or 40 bits. From that word is compressed by 4 algorithm. the final result is, after compressed by using Shannon-Fano algorithm, we get the final result of 10 bits, for Huffman algorithm we get also the final hash of 10 bits. In the Shannon-Fano and Huffman algorithms obtained a bit smaller than the initial bit. For the Run Length Encoding algorithm obtained results greater than the previous bit is as much as 80 bits, while the Tunstall algorithm obtained results that are smaller than the previous bit is as much as 12 bits.

Then the second experiment is the word which has 7 bytes or 56 bits. In this experiment, the results obtained from the use of Shannon Fano algorithm is as much as 10 bits. Likewise with Huffman algorithm obtained 10 bits. While the Run Length Encoding algorithm obtained the final result as much as 96 bits. And in Tunstall algorithm obtained the final result as much as 9 bits. Means can be concluded, for that word after compressed with 4 algorithms can be concluded that the Run Length Encoding algorithm obtained results greater than the initial bits.

In the third experiment is the word which has 9 bytes or equivalent to 72 bits. On the use of shannon fano algorithm obtained as much as 25 bits, for Huffman algorithm obtained as much as 31 bits. As for the Run Lenght Encoding algorithm got the final result as much as 144 bits, and on the Tunsstall algorithm obtained results as much as 49 bits. In this third experiment it can be concluded that just like the first and second experiments, on the Shnnon Fano algorithm, Huffman and Tunstall have bits smaller than the initial bit. As for the Run Length Encoding algorithm has a bit larger than the initial bit.

In the last experiment, used the word which has 12 bytes of byte or equivalent to 96 bits. Just like in previous experiments, the word is compressed using 4 algorithms. In the Shannon Fano algorithm obtained the final result as much as 18 bits, for Huffman algorithm obtained as much as 18 bits as well. As for the Run Length Encoding algorithm of 40 bits, and for Tunstall algorithm obtained as much as 20 bits. In this last experiment, it is concluded that word after compression by 4 algorithms get smaller result than the initial bit.

The Result of Compression Ratio

In this experiment, the authors also calculate the value of compression ratio with the formula:

$$\frac{\text{(Initial bit - bit after compressed)}}{\text{Initial bit}}$$

And there are a chart of Compression Ratio :

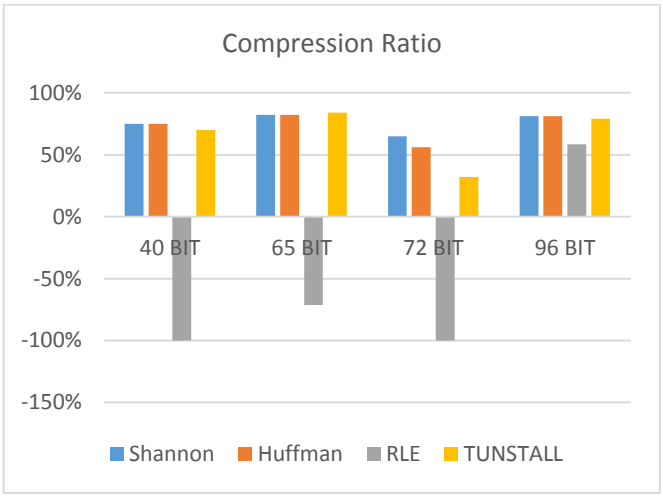

Figure 4 : The Result of Compression Ratio

In figure 4, it can be seen, that in the first experiment which has 40 bytes of byte by using shannon algorithm get 75% result, and for Huffman algorithm obtained result of 75%, while at Run Length Encoding algorithm got result as much -100%. This is because the resulting bit is greater than the initial bit. Then on the Tunstall algorithm obtained as much as 70%.

Then in the second experiment, which has 7 bytes or 56 bits. In the Shannon-Fano algorithm obtained 82, 14% as well as the Huffman algorithm obtained results as much as 82.14%. Then on Run Length Encoding algorithm got the final result of -71,4% while in tunstall algorithm got result as much 83,92%.

In the third experiment, which has bytes as much as 9 bytes or equivalent to 72 bits. On the use of shannon fano algorithm obtained as much as 65%, for Huffman algorithm obtained as much as 56% As for the Run Lenght Encoding algorithm got the end result of -100%, and on the Tunsstall algorithm obtained results as much as 31,94%. In this third experiment it can be concluded that just like the first and second experiments, on the Shnnon Fano algorithm, Huffman and Tunstall have a considerable percent value compared to the Run Length Encoding algorithm.

And in the last experiment, which has 12 bytes of byte or

equivalent to 96 bits. In the Shannon-Fano algorithm obtained the final result of 81.25%, for Huffman algorithm obtained results as much as 81.25%. As for the Run Length Encoding algorithm of 58.33%, and for the Tunstall algorithm obtained as much as 79.17%.

ANALYSIS AND DISCUSSION

From the chart in figure 3, it can be seen that the results of the Hufman algorithm, Shannon algorithm, and Tunstall Algorithm can produce smaller bit values. Unlike the Run Length Encoding algorithm. If used on a sentence that has a mean, the resulting bit will be larger than the initial bit. Therefore why the Run Lenght Encoding algorithm is not recommended for a sentence that has meaning. On the contrary, if used on a recurring word, it will produce smaller bits than before. Can be seen in a 96 bit in the last experiment.

From the chart In figure 4, can be seen, overall that always get higher percentage is Shannon-Fano and Huffman Algorithm. And that has the smallest percentage is Run Length Encoding. That's because the bit that is run by Run Length Encoding has a larger result than the initial bit. So the value of the resulting percentage is smaller than other algorithms.

CONCLUSION

After doing 4 experiments above, it can be concluded that, in Shannon-Fano, Huffman, and Tunstall algorithm always get smaller result than previous bit. As for the Run Length Encoding algorithm, it depends on the sentence used. If the sentence used is a sentence that has a meaning, usually the algorithm is always obtained results greater than the previous bit. Whereas if the word used is a word that has a loop, it can produce a smaller end result than the previous bit. So it all depends on the data used.

Similarly, the Compression Ratio, the percent value obtained by the Shannon-Fano algorithm, the Huffman algorithm, and the Tunstall algorithm always have a high percentage or high enough value. Unlike the Run Length Encoding algorithm, the algorithm if the sentence contains meaning, always get a relatively small percentage. Because the result of the calculation is greater than the initial bit value. Thus affecting the percent value of the compression ratio.

REFERENCE

[1] M.VidyaSagar, J.S, Rose Victor, "Modified Run Length Encoding Scheme for High Data Compression Rate", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Vijayawada, December 2013.

[2] K. Ashok Babu and V. Satish Kumar, "Implementation of Data Compression Using Huffman Coding", International Conference on Methods and Models in Computer Science, India, 2010.

[3] Harry Fernando, "Kompresi data dengan algoritma Huffman dan algoritma lainnya", ITB, Bandung.

[4] Mohammed Al-laham1 & Ibrahiem M. M. El Emary, "Comparative Study between Various Algorithms of Data Compression Techniques", IJCSNS International Journal of Computer Science and Network Security, Jordan, April 2007.

[5] S.R.Kodituwakku and U.S.Amarasinghe, "Comparison of Lossless Data Compression Algorithms for Text", Indian Journal of Computer Science and Engineering, Sri Lanka.

[6] Rhen Anjerome Bedruz and Ana Riza F. Quiros, "Comparison of Huffman Algorithm and Lempel-Ziv Algorithm for Audio, Image and Text Compression", IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM). Philippines, 9-12 Decmber 2015.

[7] C. Oswald, Anirban I Ghosh and B.Sivaselvan, "Knowledge Engineering Perspective of Text Compression", IEEE INDICON, India, 2015.

[8] Ardiles Sinaga, Adiwijaya and Hertog Nugroho, "Development of Word-Based Text Compression Algorithm for Indonesian Language Document", International Conference on Information and Communication Technology (ICoICT),Indonesia, 2015

[9] Manjeet Kaur, "Lossless Text Data Compression Algorithm Using Modified Huffman Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, india, July 2015

[10] Tanvi Patel, Kruti Dangarwala,Judith Angela, and Poonam Choudhary, "Survey of Text Compression Algorithms", International Journal of Engineering Research & Technology (IJERT), india, March 2015

[11] Shmuel T. Klein and Dana Shapira, "On Improving Tunstall Codes", Information Processing & Management, Israel, September 2011.

[12] Mohammad Hosseini, "A Survey of Data Compression Algorithms and their Applications", Applications of Advanced Algorithms, At Simon Fraser University, Canada, January 2012

[13] Maria Roslin Apriani Neta, "Perbandingan Algoritma Kompresi Terhadap Objek Citra Menggunakan JAVA", Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2013 (SEMANTIK 2013), Semarang, November 2013.

[14] Dr. Shabana Mehfz1, Usha Tiwad, "A Tunstall Based Lossless Compression Algorithm for Wireless Sensor Networks", India Conference (INDICON), 2015 Annual IEEE, India, 2015.

[15] Dr. Ahmad Odat, Dr. Mohammed Otair and Mahmoud Al-Khalayleh, "Comparative Study between LM-DH Technique and Huffman Coding Technique", International Journal of Applied Engineering Research, India, 2015