The IJICS (International Journal of Informatics and Computer Science)
Vol 4 No 1, March 2020

ISSN 2548-8384 (online)
ISSN 2548-8449 (print)
Page 29 – 34
DOI 10.30865/ijics.v4i1.1387

Comparison Analysis with Huffman Algorithm and Goldbach Codes Algorithm in File Compression Text Using the Method Exponential Comparison

Abu Sani Tanjung, Surya Darma Nasution

Department of Computer Science, STMIK Budi Darma, North Sumatra, Indonesia
Email: Abusanitjn@email.com

Abstract−With the development of technology at this time many people know about compression. In simple compression is a process to shrink the file from its original size. At this time compression applications that are often used are WinZip, WinRar, and 7-Zip, namely with the aim of compressing documents and saving space on memory or data transmission. Compressed data can be in the form of images, audio, video and text. The use of the Huffman algorithm and the Goldbach Codes algorithm in compressing text files is intended to provide enormous benefits in the sending and storage process and requires less memory space compared to uncompressed text. The algorithm starts by providing a string of inputs as input, how to produce an algorithm output in the form of a binary string or code that translates each input string, so that the string has a small number of bits compared to strings that are not compressed. Thus, the problem is how to obtain the code with sorted characters and frequency tables as input and shorter binary code as output. In applying the Huffman algorithm and the Goldbach Codes algorithm in compressing text files is very good, the results were not reduced from the original file or there was no reduction.

Keywords: Huffman, Goldbach Codes, Compression, Exponential Comparison Methods

1. INTRODUCTION

With the development of technology at this time many people know about compression. In simple compression is a process to shrink the file from its original size. At this time compression applications that are often used are WinZip, WinRar, and 7-Zip, namely with the aim of compressing documents and saving space on memory or data transmission. Compressed data can be in the form of images, audio, video and text. Text file is a collection of characters or strings that become a single unit. Text files that contain lots of characters in them always cause problems with the storage media.

The compression method used at this time is by compressing the finished files (original files) and then compressing them and communicating them. After reaching the compressed data on the recipient, then carried out a de-compress to return to its original form, then the file can be used. To overcome the above problem, one solution is to compress the data so that it is smaller than the original size without reducing the content of the data.

In data compression using a compression ratio, the data compression ratio is a measure of the percentage of data that has been successfully compressed. Compression ratio measurement is intended to see how much reduction occurs in the file compression process when compared with the original file. The greater the compression ratio, the more reduction in the size of the compressed file compared to the original file. A compression ratio of 60% means there has been a reduction of 60% from the original file size. Compression ratio can be negative, which means the size of the compressed file is bigger than the original file.

In previous research journals by Nuryasin it can be concluded that the resulting compression ratio varies considerably depending on the type of file being compressed. While the speed of compressing files is obtained that if the file size is getting bigger, the longer the time needed to do the compression process [1].

In previous research journals by Surya Darma Nasution and Mesran it can be concluded that the Goldbach Codes (GC) algorithm is a simple compression algorithm that encodes only positive integers n by converting them into positive integers with 2 (n + 3) and then writing some sum primes in reverse [2].

The use of the Huffman algorithm and the Goldbach Codes algorithm in compressing text files is intended to provide enormous benefits in the sending and storage process and requires less memory space compared to uncompressed text..

2. THEORY

2.1 Data Compression

Data compression (data compression) is a technique to reduce the amount of data size (the result of compression) from the original data. Data compression is generally applied to computer machines, this is done because each symbol that appears on the computer has a different bit value. For example in ASCII each symbol that appears has

ISSN 2548-8384 (online)
ISSN 2548-8449 (print)
Page 29 – 34
DOI 10.30865/ijics.v4i1.1387

a bit length of 8 bits, for example code A on ASCII has a decimal value = 65, if changed in binary numbers to 010000001. Data compression is used to subtract the number of bits generated from each symbol that appears. With this compression is expected to reduce (reduce data size) in storage space [3].

2.2. Huffman's Algorithm

In 1952 David Huffman introduced a compression algorithm called Huffman Coding. The formation of binary trees in the Huffman algorithm is formed from leaves to roots and is called the formation of trees from the bottom up. This method uses almost all the characteristics of Shannon-Fano Coding. The principle of the Huffman code is that the characters that appear most often in the data are encoded with the shortest codes, while characters that rarely appear are coded with longer codes. The Huffman algorithm builds a binary tree to generate prefix code [6].

2.3. Goldbach Codes Algorithm

In 2001, Peter Fenwick had the idea to use Goldbach's assumption (assuming that was true) to design a completely new class of code based on prime numbers. Prime numbers can function as the basis of a number system, so if we write even integers. This system, its representation will have exactly two. So, even number 20 equals 7 + 13 and can therefore be written 10100, where five bits are given the main weight (from left to right) 13, 11, 7, 5, and 3. Now turn this bit pattern over. The least significant bit becomes 1, producing 00101. Such numbers are easy to read and extract from long string bits. Simply stop reading in the second part. Remember that unary code (the zero sequence ending in single) is read by a similar rule. Stop at the first. So, the Goldbach code can be considered as an extension of a simple unary code [7].

2.4. Exponential Comparison Method

In calculating and comparing the search process of the two algorithms is as follows:
1. Determine alternatives
2. Determine the criteria
3. Grading values on each criterion
4. Calculating values
5. Determine the results or priority decisions [8].

3. RESULT AND DISCUSSION

The analysis carried out in comparing the process of the workings of the Huffman algorithm and the Goldbach Codes algorithm only leads to the compression of text files. Compression is a reduction in the size of a file to a size smaller than the original. Compressing this file is very beneficial when there is a large file and the data in it contains a lot of repetition of characters. The technique of this compression is to replace these repetitive characters with a certain pattern so that the file can minimize its size.

The algorithm starts by giving string strings as input, how to produce algorithmic output in the form of binary strings or code that translates each input string so that the string has a smaller number of bits compared to strings that are not compressed. Thus, the problem is how to obtain the code with sorted characters and frequency tables as input and shorter binary code as output. Text file compression analysis refers to text documents with * .txt and * .doc extensions, with processes that depend on the size of the document and how many characters are in the document. In data compression using a compression ratio, the data compression ratio is a measure of the percentage of data that has been successfully compressed.

Compression ratio measurement is intended to see how much reduction occurs in the file compression process when compared with the original file. The greater the compression ratio, the more reduction in the size of the compressed file compared to the original file. In analyzing the workings of the two algorithms, it is necessary to have an application that will find out how the process produced by each algorithm in doing compression. This designed application is a desktop based application. The tools used to design the comparative analysis application are to use Microsoft Visual Basic.Net 2008 as a tool for design and source code.

3.1 *Goldbach Codes Algorithm*

The following is an example of compression using the Goldbach Codes Algorithm, there is a text file with the extension .txt and the size is 19 bytes containing the string: AKU ANAK MANDAILING, here's how to solve:
1. Read all the characters in the text to calculate the frequency of occurrence of each character, where the character that appears the most is the first (n = 1) and so on.
2. Find the codeword for each character to be encoded, by finding a prime number that can represent the sum of two prime numbers. Primary starts from number 3, 5, 7 and so on.

ISSN 2548-8384 (online)
ISSN 2548-8449 (print)
Page 29 – 34
DOI 10.30865/ijics.v4i1.1387

3. Codeword only has two "1 or 0". Prime number search represents the number in search then stop being given "1" and if you don't find it given "0" and so on until you find what you are looking for. For example the number 24, then 11 + 13 = 24. After getting the numbers of several prime numbers are correct, then the bit pattern is obtained is 11000. Then the bit pattern is reversed towards the very back, producing a bit pattern 00011.

Table 1. Before Compression

Character	Frequency	Ascii Desimal	Ascii Biner	Bit	Bit x Freq
U	1	85	01010101	8	8
M	1	77	01001101	8	8
D	1	68	01000100	8	8
L	1	76	01001100	8	8
G	1	71	01000111	8	8
K	2	75	01001011	8	16
Sp	2	83	01010011	8	16
I	2	73	01001001	8	16
N	3	78	01001110	8	24
A	5	65	01000001	8	40
Total Bit x Freq					152

Tabel 2. Sesudah Di Dekompresi

Character	Freq	n	2 (n+3)	Bil. Prima	*Codeword*	Bit	Bit x Freq
A	5	1	8	3 + 5	11	2	10
N	3	2	10	3 + 7	101	3	9
K	2	3	12	5 + 7	011	3	6
K	2	3	12	5 + 7	011	3	6
Sp	2	4	14	3 + 11	1001	4	8
I	2	5	16	5 + 11	0101	4	8
U	1	6	18	7 + 11	0011	4	4
M	1	7	20	7 + 13	00101	5	5
D	1	8	22	5 + 17	010001	6	6
L	1	9	24	11 + 13	00011	5	5
G	1	10	26	7 + 19	0010001	7	7
Total Bit x Freq							68

The size of the text file before compression with ASCII code (8 bits) is 19 bytes or 19 x 8 bits = 152 bits. Thus, to save a text file consisting of 19 characters using the Goldbach Codes algorithm requires 68 bytes.

3.2 *Huffman's Algorithm*

The Huffman algorithm uses a table of occurrence of characters for the frequencies of two trees combined. Therefore, the total cost of forming a Huffman tree is the sum of all the combined leaves. Huffman provides an algorithm to construct a Huffman code with the input text string S = {s1, s2, ..., sn} and the frequency of occurrence of characters F = {f1, f2, ..., fn}, resulting in the form of binary string C = {c1 , c2, ..., cn} or called the Huffman code.

Compression steps for the Huffman algorithm:

1. Data are analyzed first by making a frequency table for each ASCII symbol to appear, the frequency table has attributes in the form of ASCII symbols and frequencies.
2. The two data that have the smallest occurrence frequency are selected as the first node in the Huffman tree.
3. From these two nodes, a parent node is made which records the number of frequencies of the first two nodes.
4. Then the two nodes removed from the table are replaced by the parent node earlier. This node is then used as a reference to form a tree.
5. Steps 3-5 are repeated until the contents of the table are only one, this data will be the free node or the root node.
6. Each node located on the left branch (node with greater frequency) is given a value of 0 and a node located on the right branch (node with smaller frequency) is given a value of 1.
7. The reading is done from the root node towards the leaf node by observing the value of each branch.

Steps to complete the Huffman algorithm:

1. Read a bit from the binary code set
2. Starting from the root of the binary tree.
3. For each bit in step 1, traverse the corresponding branch.

The IJICS (International Journal of Informatics and Computer Science)
Vol 4 No 1, March 2020
ISSN 2548-8384 (online)
ISSN 2548-8449 (print)
Page 29 – 34
DOI 10.30865/ijics.v4i1.1387

4. Repeat steps 1, 2 and 3 until they meet the leaves. Codecode a series of bits that have been read with characters on the leaf.

Repeat from step 1 until there are no more bits in the binary code set.

Based on the description above given the string "I AM CHILDREN MANDAILING" as input to the compression of the text file. This input consists of alphabetic characters that often and do not appear frequently S = {U, M, D, L, G, K, Sp, I, N, A} with the frequency of occurrence of characters F = {1, 1, 1, 1, 1, 2, 2, 2, 3, 5}. To minimize the number of bits needed, the length of the code for each character should be shortened as much as possible, especially for each character whose frequency occurs frequently.

1. Formation of Character Tables
   The characters are sorted in the table from the smallest to the largest frequency.

Table 3. Formation of Character Tables

Frequency	Character
U	1
M	1
D	1
L	1
G	1
K	2
Sp	2
I	2
N	3
A	5
Total	19

2. Formation of the Huffman Tree
   Each character is described as a single knotted leaf or tree. Then the combination of two leaves that have the smallest frequency of appearance of characters to form roots. The root is the sum of the frequencies of the two leaves they compose. This iteration is carried out until a binary tree is formed. Label 0 and 1 on each side of the binary tree. The left side is labeled with 0 and the right side is labeled with 1. The process of forming a binary tree to form a Huffman tree can be seen in the picture below:

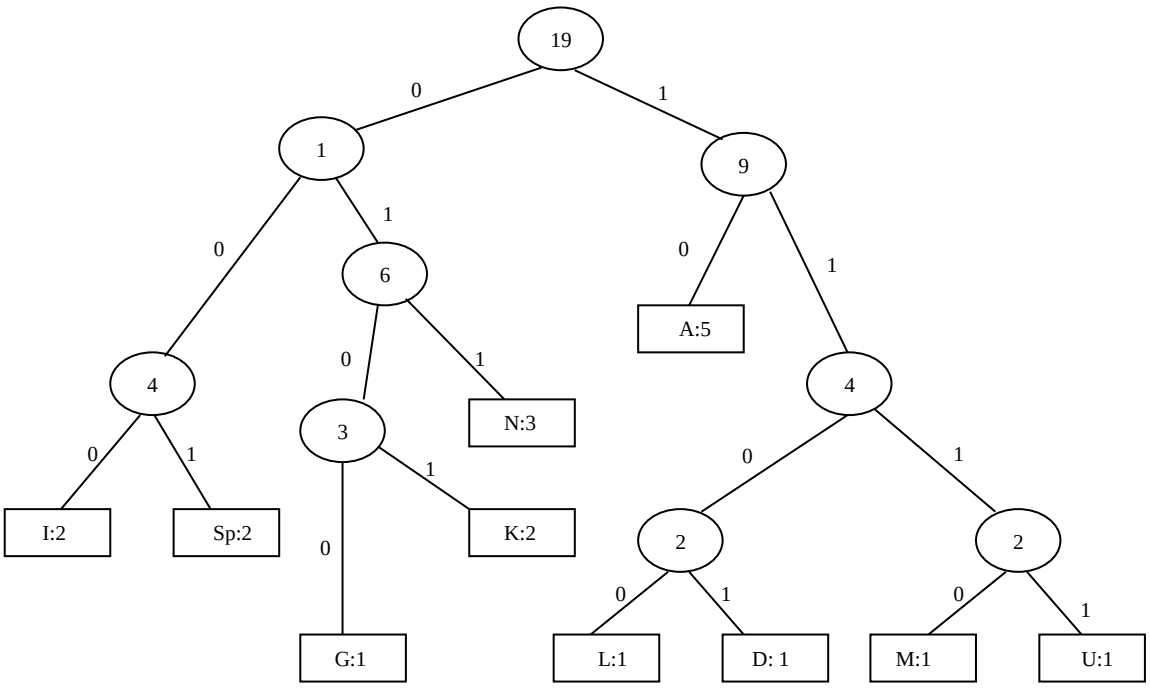

Figure 1. Huffman Tree That Has Formed

3. Formation of the Huffman Code

ISSN 2548-8384 (online)
ISSN 2548-8449 (print)
Page 29 – 34
DOI 10.30865/ijics.v4i1.1387

Rows of numbers 0 and 1 on the sides of the tree from root to leaf represent the Huffman code for the corresponding character. Browse the binary tree from root to leaf to form the Huffman code.

Table 4. Huffman codes that have been formed

Character	Frequency	Huffman Code	Total Code Bits	Total Code Bits * Frequency
U	1	1111	4	4
M	1	0111	4	4
D	1	1011	4	4
L	1	0011	4	4
G	1	0010	4	4
K	2	1010	4	8
Sp	2	100	3	6
I	2	000	3	6
N	3	110	3	9
A	5	01	2	10
Total	19		35	59

3.3 *Exponential Comparison Method*

In calculating and comparing the search process of the two algorithms is as follows:

1. Determine alternatives
   To analyze the speed ratio between the Huffman algorithm and the Goldbach Codes algorithm in compression it is necessary to determine which algorithm will be used as the compression algorithm.
2. Determine the criteria
   To be able to compare the two algorithms, the next step is to determine criteria in analyzing the process and how it works. The criteria can be seen in the following table:

Table 5. Determination of Criteria

Criteria	Description
Time Compression (TM)	Calculation of time is obtained when the algorithm performs compression or decompression.
Ratio Of Compression (RC)	The value of the comparison between the size of the data bit before it is compressed to the size of the data bit that has been compressed
Compression Ratio (CR)	Percentage of comparison between compressed data and uncompressed data.
Space Saving (SS)	Difference between uncompressed data and large compressed data

3. Grading values on each criterion
   The keriteria that has been formed must be given a value. This value can be seen in the example below where the value is taken based on the analysis of the Huffman algorithm and the previous Goldbach Codes algorithm.

Table 6. Scoring on each criterion

Alternative	Example	Criteria				
		RC		CR		SS
		SB	SD	SD	SB	
Algorithm *Huffman*	AKU ANAK MANDAILING	152 (bit)	59 (bit)	59 (bit)	152 (bit)	1-CR
Algorithm *Goldbach Codes*		152 (bit)	68 (bit)	68 (bit)	152 (bit)	1-CR

Information:
TM: Time Compression
RC: Ratio of Compression
CR: Compression Ratio
SS: Space Saving
SB: Before Compressed
SD: After being compressed

5. Determine the results or priority decisions
   After obtaining the final value or the total value of each alternative, then the next step that needs to be done is to determine the priority of the decision based on the value of each alternative. The results of priority decisions can be seen in the table below:

ISSN 2548-8384 (online)
ISSN 2548-8449 (print)
Page 29 – 34
DOI 10.30865/ijics.v4i1.1387

Table 7. Priority Decisions

Alternative	RC	CR	SS	Rank
Algoritma *Huffman*	2,57	38 %	62 %	1
Algoritma *Goldbah Codes*	2,23	44 %	56 %	2

In the table above explains that the alternative that has the highest total value in the SS table (Space Saving) will be positioned first rank, this is because the greater the percent (%), the better and faster compression. Based on this analysis, the Huffman algorithm is the best algorithm for the compression process.

4. CONCLUSION

From this study it can be concluded as follows:

1. In applying the Huffman algorithm and Goldbach Codes algorithm in compressing text files is very good, the results were not reduced from the original file or there was no reduction (the contents of the file).
2. The designed application has been able to compress text files using the Huffman algorithm and the Goldbach Codes algorithm, and decompress the compression results with the Huffman algorithm and the Goldbach Codes algorithm, into the original file.
3. The results of the comparison of text file compression between the Huffman algorithm and the Goldbach Codes algorithm are very different results, in the application of the Exponential Comparison Method with four criteria, the better the compression result is the Huffman algorithm.

REFERENCES

[1] Nuryasin, "Perbandingan Kompresi File Data Dengan Algoritma Huffman, Helf Byte Dan Run Length," Studi Informatika: Jurnal Sistem Informasi, Vol. 7, No. 1979-0767, Pp. 1-6, Februari 2014.
[2] Mesran Surya Darma Nasution, "Goldbach Codes Algorithm For Text Compression," IJournals: International Journal Of Software & Hardware Research In Engineering, Vol. 4, No. 11 November, 2016 , Pp. 43-46, November 2016.
[3] Ari Wibowo, "Kompresi Data Menggunakan Metode Huffman," Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2012 (Semantik 2012), vol. Juni, pp. 47-51, Juni 2012.
[4] Abdul Kadir, Algoritma & Pemrograman Menggunakan C & C ++. Yogyakarta: Penerbit Andi, 2012.
[5] Abdul Kadir, Pengenalan Algoritma, Pendekatan Secara Visual Dan Interaktif Menggunakan Raptor. Yogyakarta: Penerbit Andi, 2013.
[6] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proceedings of the IRE, vol. September, pp. 1098-1101, September 1952.
[7] D. Salomon G. Motta, Handbook Of Data Compression, Springer, Fifth Edition ed. Springer London Dordrecht Heidelberg New York, 2010.
[8] Marimin, Teknik dan Aplikasi Pengambilan Keputusan Kriteria Majemuk. Jakarta: Grassindo, 2004.