Design of Adaptive Compression Algorithm Elias Delta Code and Huffman

[1] Eko Hariyanto, [2] Andysah Putera Utama Siahaan
Faculty of Science and Technology, Universitas Pembangunan Panca Budi, Medan, Indonesia
Email: [1] eko.hariyanto@dosen.pancabudi.ac.id, [2] andiesiahaan@gmail.com

Abstract: *Compression aims to reduce data before storing or moving it into storage media. Huffman and Elias Delta Code are two algorithms used for the compression process in this research. Data compression with both algorithms is used to compress text files. These two algorithms have the same way of working. It starts by sorting characters based on their frequency, binary tree formation and ends with code formation. In the Huffman algorithm, binary trees are formed from leaves to roots and are called tree-forming from the bottom up. In contrast, the Elias Delta Code method has a different technique. Text file compression is done by reading the input string in a text file and encoding the string using both algorithms. The compression results state that the Huffman algorithm is better overall than Elias Delta Code.*

Key Words: *Huffman, Elias Delta Code, adaptive, compression.*

1. INTRODUCTION:

Data Compression is a branch of computer science derived from Information Theory. Information Theory itself is a branch of Mathematics that developed around the end of the 1940s. The main character of Information Theory is Claude Shannon from Bell Laboratory. Information Theory focuses on various methods of information including message storage and processing. Information theory also learns about redundancy in useless messages. The more redundancy the more significant the size of the message, the effort to reduce redundancy is what ultimately gives birth to the subject matter of Data Compression. Information theory uses entropy terminology as a measure of how much information can be retrieved from a message. The word "entropy" comes from thermodynamics. The higher the entropy of a message the more information contained in it. The entropy of a symbol is defined as the negative logarithm value of the probability of its occurrence. The following formula can be used to determine the information content of a message in the number of bits.

$$number\ of\ bits\ =\ -log\ base\ 2\ (probability)$$

The entropy of the whole message is the sum of the whole entropy of the whole symbol.

Data is an important thing that must be protected and safeguarded [1]–[7]. Data compression is the process that converts an input in the form of the source or raw data stream into another data stream. Based on the possibility of data after being compressed it can be reconstructed back to the data original, data compression techniques are divided into two parts, lossless compression, and lossy compression, Lossless compression allows data to be returned to the original data in full or without any information missing in the data, while Lossy compression cannot restore data that has been completely compressed from the original data during the decompression process [8].

Huffman is a method designed by Peter Fenwick in an experiment to improve the performance of the Burrows-Wheeler transform [9]. The term adaptive compression comes from the Error Control Codes (ECC). It consists of the original data plus some check bits [10]. If several check bits are removed, to shorten the series of codes, the results of the code are intended as Punctured. Adaptive coding is a variation of entropy encoding. Adaptive coding is suitable for data streams because it is dynamic and adapts to changes in data characteristics. Adaptive coding requires more complicated encoders and decoders to maintain a synchronized state, and also more computing power. Adaptive coding utilizes a model that is owned by a data compression method, which is a prediction of the composition of the data. Encoder transmits data contents by utilizing references to the model.

2. THEORIES:

2.1 Data Compression

Data compression in the context of computer science is a science or art in representing information contained in data into a denser form. The development of computers and multimedia has resulted in data compression is very important and useful in technology today. The definition of data compression is a process that converts an input in the form of a data stream into another data stream which has a smaller size. Data flow can be a file or buffer in memory. Data in the context of data compression encompasses all digital forms of information, which can be processed by a

computer program. The form of information can be broadly classified as text, sound, images, and video. There are many data compression software that is often used by many people, such as Winzip, Winrar, 7-Zip and others to reduce the size of the data before they store and send the data to a storage media. It is done to make it easier to manage files and minimize the use of storage media. Besides that, it also helps in accelerating the process of sending files from one media to another media where at this time cloud computing has been developed that is a storage media contained in online media that can be accessed anywhere by requiring an internet connection. With the data compression, the data that has been compressed will reduce the quota or bandwidth in sending data over the internet.

2.2 Lossless Data Compression

It is a data compression technique in which compressed data can be returned to the original data by not reducing the information contained in the data. In this compression technique, the compression process has generated data that is different from the original data. Every bit contained in the data is encoded so that the new bits are shorter. The data is returned to the original data (data before being compressed) during the decompression process. Lossless compression techniques are used in classified data so that there is not much or part of the information missing. Like for example in a text file, if there is a slight change in the data, it will be effortless to see. Examples of lossless algorithms in data compression are Arithmetic Coding, Huffman Coding, and others. Lossless compression techniques can be described as in the following figure [11].

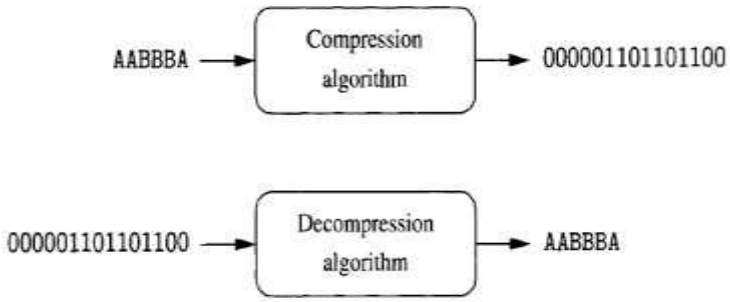

Figure 1. Lossless Data Compression

2.3 Lossy Data Compression

It is a data compression technique where data has little or no loss during the compression process. The data after being compressed the same as the original data only changes the density or resolution of the data. Therefore it is very unlikely and will never even be able to data that has been compressed with a Lossy compression technique can be returned like data before being compressed or the original data. Changes that occur in the data during the compression process are not too visible. This compression technique is usually used in data in the form of images, sound, and video. Because the data is the loss of detailed information that may not be felt by the work system of the senses of human sight and hearing. Examples of Lossy compression algorithms are Fractal Compression, Wavelet Compression, Wyner-Ziv Coding (WZC), and others. The Lossy compression technique can be described as in the following figure [11].

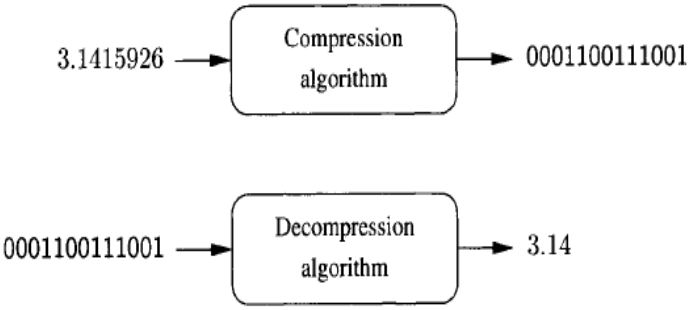

Figure 2. Lossy Data Compression

2.4 Huffman algorithm

David Huffman discovered the Huffman algorithm in 1952. This algorithm uses a coding similar to Morse code. Based on the type of code used by the Huffman algorithm including statistical methods. Whereas based on the coding technique using symbols method [12]. The Huffman algorithm is one of the algorithms used to compress text. The complete Huffman algorithm:

- Select two symbols with the smallest probability (in the example above the symbols B and D). The two symbols are combined as the parent node of the symbol B and D so that it becomes a BD symbol with an opportunity of 1/7 + 1/7 = 2/7, which is the number of opportunities for the two children.
- Next, select the next two symbols, including a new symbol, which has the smallest chance.
- Repeat steps 1 and 2 until all symbols are finished.

For example, in the 7-letter ASCII string code "ABACCDA" requires a representation of 7 × 8 bits = 56 bits (7 bytes), with the following details:

01000001 01000010 01000001 01000011 01000011 010001001000001
A B A C C D A

The length of the code for each character can be shortened to reduce the number of bits needed, especially for characters whose frequency of occurrence is significant. In the string above, the occurrence frequency of A = 3, B = 1, C = 2, and D = 1, so using the algorithm above is obtained by the Huffman code as in Table 1 [13].

Table 1. Huffman Code

Character	Frequency	Probability	Huffman Code
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

Using this Huffman code, the string "ABACCDA" is represented as a series of bits: 0 110 0 10 10 111 0. So, the number of bits needed is only 13 bits from which 56 bits should be needed. To decipher the data that has been encoded previously with the Huffman algorithm, the following methods can be used:

1. Read the first bit of the input binary string
2. Traversal the Huffman tree starting from the root according to the bit read. If that bit read is 0, then read the left child, but if the bit read is 1 then read the right child.
3. If the child is from a non-leaf tree (childless node), then read the next bit of the input binary string.
4. This is repeated (traversal) until the leaves are found.
5. On the leaf, the symbol is found, and the decoding process is complete.
6. The decoding process is carried out until the entire input binary string is processed.

2.5 Elias Delta Code

Elias Delta code was discovered by Peter Elias. This code also applies the method on gamma coding, especially in the head [14]. The techniques are as follows:

- Look for the highest rank of the binary, for example decimal 11 if it is fixed to 1011 where the highest rank is 3. So N '= 3.
- Use Gamma Coding to encode the number N where N = N '+ 1. So for the decimal case 11 then we have to make Gamma Coding from 4 that is 00100.
- Add the remainder of the N 'binary to result # 2. So the answer is 00100011.

Next is the Elias Delta Coding code, the principle is the opposite of steps one to three above. For example, we will decode 00100011.

- Find the number of zeros before finding the number one, which is 00, amounting to two. Means there are (2 + 1) numbers that must be considered after these two zeros are 100 which in decimal means 4, so we get N 'with N-1 = 4-1 = 3.
- If N 'has been known, that is 3 then three remaining bits are part of that number, namely 011. So the answer is 1011 which means 11.
- If we are given a binary sequence as follows: 00110000110011110010100111110000 is a row of numbers 35, 101, and 112.

3. RESULT AND DISCUSSION:

3.1 Elias Delta Test

At this stage, compression testing is performed for a simple string. It is "**THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG**." Stages in this section include:

- Calculate the length of the text
- Create and calculate the length of the character set
- Calculating bit length
- Specifies padding bits if necessary

Table 2 shows the original string before the compression process. The string has 43 characters.

Table 2. Original string

No.	Char	ASCII	Binary
1	T	84	01010100
2	H	72	01001000
3	E	69	01000101
4		32	00100000
5	Q	81	01010001
6	U	85	01010101
7	I	73	01001001
8	C	67	01000011
9	K	75	01001011
10		32	00100000
11	B	66	01000010
12	R	82	01010010
13	O	79	01001111
14	W	87	01010111
15	N	78	01001110
16		32	00100000
17	F	70	01000110
18	O	79	01001111
19	X	88	01011000
20		32	00100000
21	J	74	01001010
22	U	85	01010101
23	M	77	01001101
24	P	80	01010000
25	S	83	01010011
26		32	00100000
27	O	79	01001111
28	V	86	01010110
29	E	69	01000101
30	R	82	01010010
31		32	00100000
32	T	84	01010100
33	H	72	01001000
34	E	69	01000101
35		32	00100000

36	L	76	01001100
37	A	65	01000001
38	Z	90	01011010
39	Y	89	01011001
40		32	00100000
41	D	68	01000100
42	O	79	01001111
43	G	71	01000111

The next stage is the formation of character sets. Repeated characters will be eliminated so that only single characters live. After this process, there are only 27 characters left with different frequency of occurrences. The number of bits in this string is 344. The result of the character set process can be seen in Table 3.

Table 3. Elias Delta characters set

No.	Char	ASCII	Binary	Freq.	Bits
1	T	84	01010100	2	16
2	H	72	01001000	2	16
3	E	69	01000101	3	24
4		32	00100000	8	64
5	Q	81	01010001	1	8
6	U	85	01010101	2	16
7	I	73	01001001	1	8
8	C	67	01000011	1	8
9	K	75	01001011	1	8
10	B	66	01000010	1	8
11	R	82	01010010	2	16
12	O	79	01001111	4	32
13	W	87	01010111	1	8
14	N	78	01001110	1	8
15	F	70	01000110	1	8
16	X	88	01011000	1	8
17	J	74	01001010	1	8
18	M	77	01001101	1	8
19	P	80	01010000	1	8
20	S	83	01010011	1	8
21	V	86	01010110	1	8
22	L	76	01001100	1	8
23	A	65	01000001	1	8
24	Z	90	01011010	1	8
25	Y	89	01011001	1	8
26	D	68	01000100	1	8
27	G	71	01000111	1	8
					344

According to the arrangement of bits of the characters in the Elias Delta table, the result of compression of the previous string is 248 as seen in Tabel 4. This result is obtained based on the use of bits according to the frequency of appearance of the characters. The space character is the one that most often appears in the string; There are eight occasions. There is a character that appears three and four times. Four characters appear twice and 20 characters each appearing just once.

Table 4. Elias Delta result

No.	Char	Freq.	Binary	Length	Bits
1		8	1	1	8
2	O	4	0100	4	16
3	E	3	0101	4	12
4	H	2	01100	5	10
5	U	2	01101	5	10
6	R	2	01110	5	10
7	T	2	01111	5	10
8	C	1	00100000	8	8
9	K	1	00100001	8	8
10	B	1	00100010	8	8
11	Q	1	00100011	8	8
12	I	1	00100100	8	8
13	W	1	00100101	8	8
14	N	1	00100110	8	8
15	F	1	00100111	8	8
16	X	1	001010000	9	9
17	J	1	001010001	9	9
18	M	1	001010010	9	9
19	P	1	001010011	9	9
20	S	1	001010100	9	9
21	V	1	001010101	9	9
22	L	1	001010110	9	9
23	A	1	001010111	9	9
24	Z	1	001011000	9	9
25	Y	1	001011001	9	9
26	D	1	001011010	9	9
27	G	1	001011011	9	9
					248

Bit Sequence:

0111101100010110010001101101001001000010000000100001100100010011100100001001010010011010010011101000010100001001010001011010010100100010100110010101001010000101010101010111010111101100010110010101011000101011100101100000101100110010110100100001011011**00110000**

TB = 248

TC = $\frac{TB}{8}$

= $\frac{248}{8}$

= 31

Padding = $TB \ mod \ 8$

= $248 \ mod \ 8$

= 0

Total Bits = 248 + 8

= 256

The bold character at the end of the bit sequence is the number of padding bits. For the above calculation, it is obtained that Padding = 0. Character "0" is on the order of 48 in ASCII code which after converted to binary will produce 00110000. This binary number will be added to the compression result, so the binary number becomes 248 + 8 = 256 bits. After the 256 bit is converted to a sequence of characters, this will generate the string **{? FÒB ????????? N ?? E ¥ "?? R UuìYX®X, ËH [0** as many as 32 characters.

$$\text{Compression Ratio} = \frac{after\ compression}{before\ compression} * 100\%$$
$$= \frac{256}{344} * 100\%$$
$$= 74.41860465116279\%$$

$$\text{Redudancy} = \frac{before - after\ compression}{before\ compression} * 100\%$$
$$= \frac{344-256}{344} * 100\%$$
$$= \frac{88}{344} * 100\%$$
$$= 25.58139534883721\%$$

The compression process has saved data of 25.58139534883721% of the original data. The savings rate depends on the order and character pattern of the original message.

3.2 Huffman Test

Assume the sentence is "LIKA-LIKU LAKI-LAKI TAK LAKU-LAKU". The text above is 33 length. It must be categorized base on the character frequency. The *Character_Set* function is to determine how many times each character appears.

Original Text : LIKA-LIKU LAKI-LAKI TAK LAKU-LAKU
Replaced Text : LIKA-###U ##########T#############
Character Set : LIKA-U T

The above procedure calculates the frequency of the character occurrence. First, the character set function must be run to obtain a series of a single character used; then the first character until the last character must be compared to the original text to get the frequency. The result will be sent to the node after getting incremented. Figure 1 illustrates the characters and their frequency obtain from first phase. After the table is fully sorted, it is time to face the most difficult step; that is to make the Huffman tree. Each node must be categorized and put it on the linked list. The Greedy algorithm takes apart at this section. It needs to combine two nodes and make a new node and make it as a parent of those earlier nodes. Let's see the illustration below:

T	-	U		I	L	A	K
1	3	3	3	4	6	6	7

Draw the first two nodes, and release from the table. Make a new node which will be their parent.

	*	
	4	
T		-
1		3

U		*	I	L	A	K
3	3	4	4	6	6	7

The parent node will be inserted to the table in ascending order using insertion algorithm. The first node is now replaced by the third node before. Moreover, It has to do the same way again until the table consists of one node.

	*	
	6	
U		
3		3

*	I	*	L	A	K
4	4	6	6	6	7

	*	
	8	
*		I
4		4

*	L	A	K	*
6	6	6	7	8

	*	
	12	
*		L
6		6

A	K	*	*
6	7	8	12

	*	
	13	
A		K
6		7

*	*	*
8	12	13

	*	
	20	
*		*
8		12

	*	
	33	
*		*
13		20

*	*
13	20

*
33

T	-	U		*	I	L	A	K						
1	3	3	3	4	4	6	6	7						
T	-	U		*	I	*	L	A	K					
1	3	3	3	4	4	6	6	6	7					
T	-	U		*	I	*	L	A	K	*				
1	3	3	3	4	4	6	6	6	7	8				
T	-	U		*	I	*	L	A	K	*	*			
1	3	3	3	4	4	6	6	6	7	8	12			
T	-	U		*	I	*	L	A	K	*	*	*		
1	3	3	3	4	4	6	6	6	7	8	12	13		
T	-	U		*	I	*	L	A	K	*	*	*	*	
1	3	3	3	4	4	6	6	6	7	8	12	13	20	
T	-	U		*	I	*	L	A	K	*	*	*	*	*
1	3	3	3	4	4	6	6	6	7	8	12	13	20	33

Finally, the array consists of 15 nodes.

In this step, there are two models of tree.
- Double Linked List.
- Binary Tree Linked List.

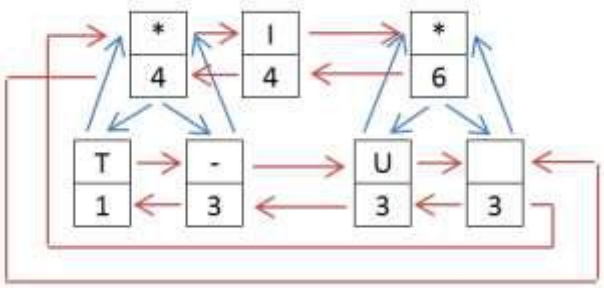

Figure 3. Double & Binary Tree Linked List

Figure 3 shows the hierarchy of the array. The * indicates the node has a parent. Figure 4 is the complete diagram of the Huffman tree.

The aim using two models of linked list is to avoid the searching procedures. The search can be either breadth-first or depth-first search to form the bit code, but it takes time and advanced programming technique. However, a list can be used to replace the backtracking procedure. The procedure track what the parent is. Huffman Tree procedure is used to form the Huffman tree by processing the earlier linear tree. The nodes must be marked "what is on the left" and "what is on the right". It is done by adding a sign 0 or 1 to the node field. The first two nodes must be combined. The first node will be marked as '0' since it is on the left and the second node will be marked as '1' since it is one the right. Moreover, both nodes have the same parent, and the parent has two children, the nodes. After the parent node is created, it must be inserted into the linked list by combining the value of the node and the value of the linked list. The node must be added to the left of the larger or same value. However, if the parent node is greater than every node, it has to be inserted after the last node.

Each node has consisted of code. This phase, everything that has been coded is converted to a Huffman table. Table 1 shows the priority of characters. The character "K" has the most appearance while the character "T" has the less one. Characters with the highest emergence is having the shortest binary code as seen on Table 5.

Table 5. Huffman Table

Char	Freq.	Code	Bit Len.	Code Len.
T	1	1000	4	4
-	3	1001	4	12
U	3	1100	4	12
	3	1101	4	12
I	4	101	3	12
L	6	111	3	18
A	6	00	2	12
K	7	01	2	14
				96

Each code represents the character. It consists of a few digit of the binary string. The previous text will be transformed into the new binary set. The old eight-bit binary is replaced by the new one. Let's see the example below:

Original Text : LIKA-LIKU LAKI-LAKI TAK LAKU-LAKU

Original Code :

111 101 01 00 1001 111 101 01 1100 1101 111 00 01 101 1001 111 00 01 101 1101 1000 00 01 1101 111 00 01 1100 1001 111 00 01 1100

Bit Code :

11110101 00100111 11010111 00110111 10001101 10011110 00110111 01100000 01110111 10001110 01001111 00011100

Decimal Code :

254, 39, 215, 55, 141, 158, 55, 96, 119, 142, 79, 28

4. CONCLUSION:

Huffman and Elias Delta code test results have several conclusions which state the compression quality of both algorithms. In Compression using the Huffman code will be more optimal if the variation of characters from the information is not too much even though the occurrence frequency is high because the Huffman tree that will be formed is not too long, so the Huffman code that represents the character becomes shorter. Based on the compression testing of

text files with non-repetitive characters, it can be concluded that the compression with the Huffman method is much more effective than the Elias Delta Code method.

REFERENCES:

1. V. Tasril, M. B. Ginting, Mardiana, dan A. P. U. Siahaan, "Threats of Computer System and its Prevention," *Int. J. Sci. Res. Sci. Technol.*, vol. 3, no. 6, hal. 448–451, 2017.
2. I. Sumartono, A. P. U. Siahaan, dan Arpan, "Base64 Character Encoding and Decoding Modeling," *Int. J. Recent Trends Eng. Res.*, vol. 2, no. 12, hal. 63–68, 2016.
3. M. D. L. Siahaan, Elviwani, A. B. Surbakti, A. H. Lubis, dan A. P. U. Siahaan, "Implementation of Simple Additive Weighting Algorithm in Particular Instance," *Int. J. Sci. Res. Sci. Technol.*, vol. 3, no. 6, hal. 442–447, 2017.
4. A. P. U. Siahaan, "Adler-32 Integrity Validation in 24bit Color Image."
5. S. Haryati, A. Ikhwan, D. Arisandi, Fadlina, dan A. P. U. Siahaan, "Quality Assurance in Knowledge Data Warehouse," *Int. J. Sci. Res. Sci. Technol.*, vol. 3, no. 6, hal. 239–242], 2017.
6. S. Ramadhani, Y. M. Saragih, R. Rahim, dan A. P. U. Siahaan, "Post-Genesis Digital Forensics Investigation," *Int. J. Sci. Res. Sci. Technol.*, vol. 3, no. 6, hal. 164–166, 2017.
7. A. P. U. Siahaan, "A Three-Layer Visual Hash Function Using Adler-32," *Int. J. Comput. Sci. Softw. Eng.*, vol. 5, no. 7, hal. 142–147, 2016.
8. D. A. Yansyah, "Perbandingan Metode Punctured Elias Code dan Huffman Pada Kompresi File Text," *J. Ris. Komput.*, vol. 2, no. 6, hal. 33–36, 2015.
9. P. Fenwick, "Burrows—Wheeler Compression," in *Lossless Compression Handbook*, Elsevier, 2003, hal. 169–193.
10. Suherman dan A. P. U. Siahaan, "Huffman Text Compression Technique," *Int. J. Comput. Sci. Enginee ring*, vol. 3, no. 8, hal. 103–108, 2016.
11. D. Putra, *Pengolahan Citra Digital*. Yogyakarta: Andi Offset, 2010.
12. R. Mustafa, "An Improved Decoding Technique for Efficient Huffman Coding," *J. Comput. Sci. Appl. Inf. Technol.*, vol. 2, no. 1, hal. 1–5, Feb 2017.
13. Blogryanpn, "Kompresi Data," 2012. [Daring]. Tersedia pada: https://blogryanpn.wordpress.com/2012/04/20/kompresi-data/. [Diakses: 01-Okt-2018].
14. L. Marlina, A. P. U. Siahaan, H. Kurniawan, dan I. Sulistianingsih, "Data Compression Using Elias Delta Code," *Int. J. Recent Trends Eng. Res.*, vol. 3, no. 8, hal. 210–217, Agu 2017.