# Implementation Of Huffman Bigram Compression Algorithm In .Txt Extension Files

Subhan Panji Cipta[1*]
[1]Sari Mulia School of Health Sciences, Banjarmasin, South Kalimantan
* panji@stikessarimulia.ac.id

M. Rizki Ikhsan[1]
[1]Sari Mulia School of Health Sciences, Banjarmasin, South Kalimantan
m.rizkiikhsan@ymail.com

## ABSTRACT

**Objective**: Implements huffman bigram algorithm for data compression process in .txt extension files.
**Method**: This study used literature study to study books relevant to existing literatures, then experiment with data compression process using the huffman bigram algorithm.
Samples of data used as experiments were created in various formats and saved in the .txt extension. Hardware and software used in this experiment is a computer with Intel pentium 4 E 2.8 Ghz Processor, 2 GB RAM, 500 GB Hard Drive, Win XP Pro edition, VB 6.0.
**Results:** The experimental results are measured from the size of the file after compression, then compared to the size of the file before it is compressed. The duration of the compression process is also observed to assess the performance of hardware and software used.
**Conclusion:** Huffman bigram algorithm can compress the size of file extension .txt become smaller than the previous size, more the number of the same pairs of characters in a file then the compression process will be maximized.

**Keywords**: Compression, Compression algorithm, Data compression, Huffman algorithm.

## I. INTRODUCTION

The use of WinZip and WinRar can save space in the storage media [1]. In addition, with a smaller file size makes the process of data transfer becomes faster, either when going to copy the process between storage media, the download process or upload process [2].

Large storage of course requires a large cost and make the data access speed getting slower [3]. The problem of storage and access speed, one of them by compressing the stored information. In recent years very often used by computer users is WinZip and WinRar where the application is very famous for Windows Operating System (OS) [2].

An important problem in the world of information technology is how to manage data from information that is getting bigger and more complex, so much faster in the process of storage and data transfer process. One solution is to compress the information data so that the size is smaller than the original size without reducing the contents of the data. So created a variety of algorithms about data compression and one of the algorithm is Huffman algorithm.

Huffman algorithm is usually a character encoded in ASCII code (8 bits) or

Unicode code (16 bits) that has a fixed length. Unlike ASCII or Unicode encoding, Huffman Coding encoding uses varying bit bits in encoding a character [4].

Huffman Coding uses a tree structure in its processing. Tree is a directed graph that does not contain circuit. Inside the tree structure is known terminology parent and child. Parent is a node that has a path to another node with a level below it. Child is a node that has a path to another node with a level on it [5].

Huffman encoding and decoding is a technique that is used for compressing source data and for lossless transmission between the transmitter and receiver. David A. Huffman and his classmates were asked to find an efficient binary code. Earlier to this, Fano and Claude Shannon constructed a binary tree using top down approach [6] where the choice was made at every step but the final result produced by this code was not optimum. David A. Huffman overcame this problem by using a frequency sorted efficient binary tree in the year 1952 and published a paper "A method for construction of minimum redundancy codes" [7]. Huffman coding was found optimal when compared with fixed length codes and is accepted as a universal coding scheme.

## II. METHOD

### Huffman Algorithm

Huffman Coding is an optimal compression algorithm that uses codes (words or bit sequences) of variable length to replace individual symbols (like characters). The length of Huffman codes is related to the frequency of the characters in the text to be compressed. Symbols that have a high appearance rate are assigned smaller words than those that those with a more rare appearance. Huffman algorithm is based on the prefix property; it designates words so that no two messages will have an identical arrangement of the coding digits and no supplementary information is needed to determine where the message coding begins and ends after the message sequence starting in determined [8]. This is how Huffman Coding ensures that there is no uncertainty when decoding the generated bit stream. Huffman algorithm takes place in the Greedy algorithm class defined by choosing the best solution at any step; because of the non ambiguous code it creates, it is considered a lossless algorithm.

### Programming Platform

The programming scheme was implemented in the Visual Basic Version 6.0.

### Implementation of algorithm

In the compression process, the result of the encoding process in the form of a binary string will be directly saved into the compressed file with the extension *.bhuf. The file storage mechanism is shown in Figure 1.

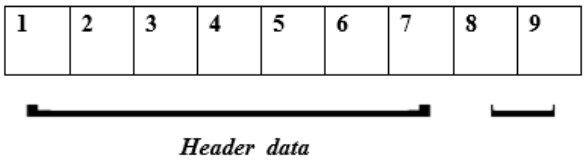


Fig 1. Storage mechanism

To reduce the number of bits required, the length of the code for each character can be shortened, especially for characters with a high chance of occurrence. In the above string, the chance of occurrence is shown in Table 1.

Table 1. Huffman Coding

| Simbol | Frekuensi | Peluang |
|---|---|---|
| E | 1 | 0.1 |
| I | 1 | 0.1 |
| K | 1 | 0.1 |
| M | 2 | 0.2 |
| T | 2 | 0.2 |
| A | 3 | 0.3 |

Table 2. Huffman Coding(1)

| Karakter | Huffman Code |
|---|---|
| E | 000 |
| I | 001 |
| K | 100 |
| M | 101 |
| T | 01 |
| A | 11 |

Using the Huffman code, the string "MATEMATIKA" is represented into a series of bits: 101110100010111010011 0011. The number of bits required is only 25 bits.

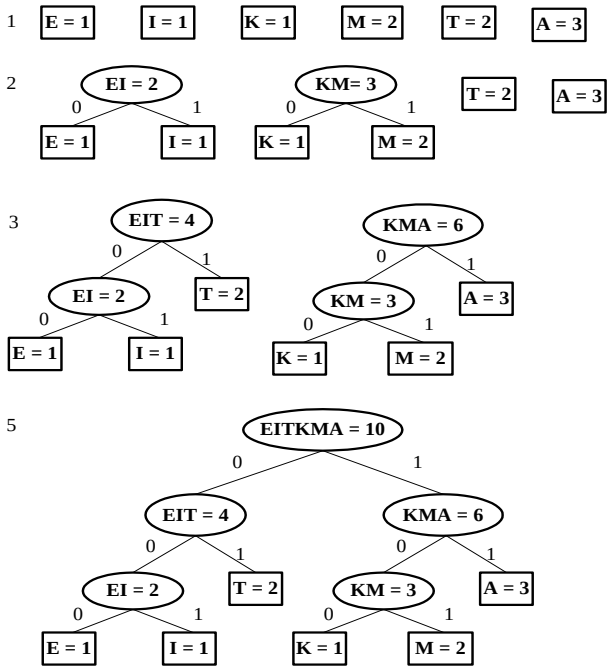


Fig2. Huffman tree (1)

Other Huffman Tree shapes that can be formed for the "MATEMATIKA" string are shown in the figure 3.

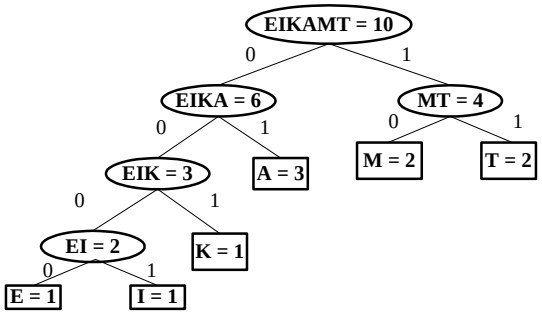


Fig3. Huffman tree (2)

From the Huffman tree it can be concluded that the Huffman code that is formed is in table 1.

Table 3. Huffman Coding

| **Character** | **Huffman Code** |
|---|---|
| E | 0000 |
| I | 0001 |
| K | 001 |
| M | 10 |
| T | 11 |
| A | 01 |

An example of a compressed file storage mechanism from a "MATEMATIKA" file is figure 4.

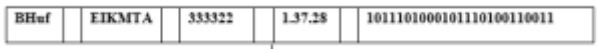


Fig 4. Storage mechanism “MATEMATIKA”

the binary string structure for each character is:

E = 000 I = 001 K = 100 M = 101

T = 01 A = 11

## III. RESULT

In our application measured execution time for Huffman algorithm over 50 iteration, In the application time measured Huffman algorithm execution with more than 50 experiments, the best time obtained from some files that can be seen in the table 4.

Table 4. Result Huffman bigram

| File Name | Size | Lama kompresi (s) | size after compressed |
|---|---|---|---|
| Coba.txt | 4904 byte | 43 second | 4190 byte |
| Test2.txt | 4630 byte | 44 second | 4252 byte |

## IV. DISCUSSION

Testing and operation of the program steps that must be considered in operating the program for files to be decompressed like the file name with the size. Huffman bigram data compression can compress the size of the file .txt extension to be smaller than the previous size but for the future data should be used more and varied, so the results for data compression can be optimized.

## V. CONCLUSION

Huffman algorithm was successfully implemented using. The merging of Huffman algorithm with bigram method can be used for compression technique. Huffman bigram data compression application can compress the size of the file *.txt* to be smaller than the previous size. The more number of characters in the same pair in a file then the compression will be more leverage. In this paper has been obtained, it is expected that the development and refinement of applications that have been made, so that the files in the compression can be more diverse, not limited to the file *.txt* also for the compression results smaller to accelerate the process of data transfer and more efficient in terms of storage media.

## REFERENCE

[1] Radescu R. Transform methods used in lossless compression of text files. Romanian Journal of Information Science and Technology. 2009 Jan 1;12(1):101-15..

[2] Yeo GS, Phan RC. On the security of the WinRAR encryption feature. International Journal of Information Security. 2006 Apr 22;5(2):115-23.

[3] Koets MA, McDaniel LT, Darnell MR, Alvarez JL. Data access architectures for high throughput, high capacity flash memory storage systems. InAerospace Conference, 2017 IEEE 2017 Mar 4 (pp. 1-8). IEEE.

[4] Cormack GV, Horspool RN. Algorithms for adaptive Huffman codes. Information Processing Letters. 1984 Mar 30;18(3):159-65.

[5] Vidhyaa VG, Rajalakshmi SA, Raghavan R, Gopal GV, Gandhiraj R. Huffman encoding and decoding algorithm using IJulia. InCommunication and Signal

Processing (ICCSP), 2016 International Conference on 2016 Apr 6 (pp. 0587-0591). IEEE.

[6] David A. Huffman, A method for the construction of minimum redundancy codes, proceedings of the institute of radio engineers, 40(9):1098-1101, September 1952

[7] Goel R and Gupta R, ”Necessary and sufficient condition for the existence of symmetric reversible variable length codes, based on Kraft’s inequality,” IEEE Recent Advances and Innovations in Engineering Conference, pp 1-3, Jaipur, May 2014

[8] D. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the IRE 40 (9), 1952, pp. 1098–1101.