Data Compression Using Shannon-Fano Algorithm

Christine Lamorahan, Benny Pinontoan, Nelson Nainggolan

1 Jurusan Matematika, FMIPA, UNSRAT, christine_itin@ymail.com
2 Jurusan Matematika, FMIPA, UNSRAT, bpinonto@yahoo.com
3 Jurusan Matematika, FMIPA, UNSRAT, bapaivana@yahoo.co.id

Abstract

Communication systems in the world of technology, information and communication are known as data transfer system. Sometimes the information received lost its authenticity, because size of data to be transferred exceeds the capacity of the media used. This problem can be reduced by applying compression process to shrink the size of the data to obtain a smaller size. This study considers compression for data text using Shannon – Fano algorithm and shows how effective these algorithms in compressing it when compared with the Huffman algorithm. This research shows that text data compression using Shannon-Fano algorithm has a same effectiveness with Huffman algorithm when all character in string all repeated and when the statement short and just one character in the statement that repeated, but the Shannon-Fano algorithm more effective then Huffman algorithm when the data has a long statement and data text have more combination character in statement or in string/word.

Keywords: Data compression, Huffman algorithm, Shannon-Fano algorithm

1. Introduction

In the world of technology information and communication, the communication process is known as data transfer. In practice, data transfer process does not always finished good, because some parts of data missing during the process. Sometimes the size of data to be transferred is too large, so it exceeds the capacity of media that used. One of possible solutions is shrink the size of data before transferred. To decrease the size of data, there are many methods that can be used and one of them is compression. Compression can be defined as process or way to shrink the size of data, so data has a smaller size than the original data (Salomon, 2004).

To ease the process of compression needed an algorithm, and this research used the Shannon-Fano algorithm to compress data text and also determined how effective the Shannon-Fano algorithm in compressing data text if compared with the Huffman algorithm.

2. Literature Review

2.1. Data Compression

Salomon (2007) says, data compression is the process of converting the input data or the original source data into the data has been compressed to a smaller size. Based on output, there are two types of compression, that is loseless and lossy. Loseless data
compression is a class of data algorithms that follow the exact original data to be reconstructed from the compressed data (Kandaga, 2006). Lossy is data encoding method that compresses data by discarding (losing) some of it (Blelloch, 2010).

2.2. Compression Ratio

The data compression ratio is the ratio between the compressed size and the uncompressed size (Altaraweneh and Altaraweneh, 2011) by following formula:

\[
\text{Compression Ratio} = \frac{\text{Compressed size}}{\text{Uncompressed size}}
\]

2.3. Algorithm

The algorithm is sequence of steps to solve a problem logically arranged. Algorithm presented in the form of picture for example by creating a flowchart and the steps used in the algorithm is sequence, selection and iteration (Sedgewick and Wayne, 2011)

2.4. Shannon-Fano Algorithm

Shannon-Fano algorithm found by Claude Elwood Shannon and Robert Fano in 1949. This algorithm replaces all characters to the binary code whose length determine by the frequency of occurrence of the character (Adhitama, 2009).

The algorithm as following:
1. For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol’s relative frequency of occurrence is know.
2. Sort the list of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.
3. Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.
4. Create a tree, with condition: character in the left part given value 0 and the character in the right part given value 1.
5. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree, (Shamugasundaram and Lourdusamy, 2011)

2.5. Huffman Algorithm

Huffman algorithm invented by David Huffman in 1952. The steps in the Huffman algorithm as follow:
1. Read all characters to calculate the frequency of occurrence of each character. Each character is a tree with a single node and each node is compiled based on the number of frequency of occurrence of each character.
2. Combine every tree which has the lowest frequency of occurrence.
3. Recursively step two until all character are completed encoded
4. The left side of the tree was coded 0 and the right side is coded 1, (Salomon, 2008).

3. Research Methodology

3.1. Procedure of Research

This research used the Shannon-Fano algorithm to compress data text. The steps in this research as follow:
1. Find out the size of the data by ASCII encoding and compare the results with size of data by using Shannon-Fano algorithm.
2. Find out the compression ratio between ASCII coding and Shannon-Fano algorithm.
3. Find out how effective the Shannon-Fano algorithm in compressing the data by comparing the size of data from the Shannon-Fano algorithm with the size of data from the Huffman algorithm.

![Figure 1. Flowchart of Research Methodology](image)

4. **Result and Discussion**

4.1 **ASCII Coding**

ASCII code save as code 8 bit and each character has a different code.

**Example 1:** CERRIAARREEIICCEEAAIIEECCRRRCR

By ASCII coding obtained the size of data for example 1 as follow:

| Character | Binary Code | Frequency | Counts   |
|-----------|-------------|-----------|----------|
| C         | 01000011    | 6         | 6 x 8 bit = 48 bit |
| E         | 01000101    | 7         | 7 x 8 bit = 56 bit |
| R         | 01010010    | 8         | 8 x 8 bit = 64 bit |
| I         | 01001001    | 5         | 5 x 8 bit = 40 bit |
| A         | 01000001    | 4         | 4 x 8 bit = 32 bit |

Total: 240 bit

**Example 2:** AYOO SEMANGAT!!!

Size of data obtained for this example as follow:

| Character | Binary Code | Frequency | Counts   |
|-----------|-------------|-----------|----------|
| A         | 01000001    | 3         | 3 x 8 bit = 24 bit |
| Y         | 01011001    | 1         | 1 x 8 bit = 8 bit |
| O         | 01001111    | 2         | 2 x 8 bit = 16 bit |
| Spasi     | 01010000    | 1         | 1 x 8 bit = 8 bit |
| S         | 01010011    | 1         | 1 x 8 bit = 8 bit |
| E         | 01000101    | 1         | 1 x 8 bit = 8 bit |
| M         | 01001101    | 1         | 1 x 8 bit = 8 bit |
| N         | 01001110    | 1         | 1 x 8 bit = 8 bit |
| G         | 01000111    | 1         | 1 x 8 bit = 8 bit |
| T         | 01010100    | 1         | 1 x 8 bit = 8 bit |
| !         | 00100001    | 3         | 3 x 8 bit = 24 bit |

Total = 128 bit
Example 3: BUKU ANI
Size of data obtained for example 3 as follow:

| Character | Binary Code | Frequency | Counts       |
|-----------|-------------|-----------|--------------|
| B         | 01000010    | 1         | 1 x 8 bit = 8 bit |
| U         | 01010101    | 2         | 2 x 8 bit = 16 bit |
| K         | 01001011    | 1         | 1 x 8 bit = 8 bit |
| Spasi     | 00100000    | 1         | 1 x 8 bit = 8 bit |
| A         | 01000001    | 1         | 1 x 8 bit = 8 bit |
| N         | 01001110    | 1         | 1 x 8 bit = 8 bit |
| I         | 01001001    | 1         | 1 x 8 bit = 8 bit |
|           |             |           | Total: 64 bit |

| Character | Character | Binary Code | Frequency | Counts       |
|-----------|-----------|-------------|-----------|--------------|
| C         | 10        | 6           | 6 x 2 bit = 12 bit |
| E         | 01        | 7           | 7 x 2 bit = 14 bit |
| R         | 00        | 8           | 8 x 2 bit = 16 bit |
| I         | 110       | 5           | 5 x 3 bit = 15 bit |
| A         | 111       | 4           | 4 x 3 bit = 12 bit |
|           |           |             | Total: 69 bit |

4.2. Shannon-Fano Coding

Example 1: CERRIAARREIICCEEAIIIEECRRRCR

![Shannon-Fano encoding tree for example 1](image)

By using Shannon-Fano algorithm, size of data obtained from the tree is 69 bit, as shown in the following table:

| Character | Binary Code | Frequency | Counts       |
|-----------|-------------|-----------|--------------|
| C         | 10          | 6         | 6 x 2 bit = 12 bit |
| E         | 01          | 7         | 7 x 2 bit = 14 bit |
| R         | 00          | 8         | 8 x 2 bit = 16 bit |
| I         | 110         | 5         | 5 x 3 bit = 15 bit |
| A         | 111         | 4         | 4 x 3 bit = 12 bit |
|           |             |           | Total: 69 bit |

Example 2: AYOO SEMANGAT!!!
By using Shannon-Fano algorithm, size of data obtained from the tree is 53 bit, as shown in the following table:

| Character | Binary Code | Frequency | Counts      |
|-----------|-------------|-----------|-------------|
| A         | 00          | 3         | 3 x 2 bit = 6 bit |
| Y         | 1000        | 1         | 1 x 4 bit = 4 bit |
| O         | 011         | 2         | 2 x 3 bit = 6 bit |
| Spasi     | 1001        | 1         | 1 x 4 bit = 4 bit |
| S         | 1010        | 1         | 1 x 4 bit = 4 bit |
| E         | 1011        | 1         | 1 x 4 bit = 4 bit |
| M         | 1100        | 1         | 1 x 4 bit = 4 bit |
| N         | 1101        | 1         | 1 x 4 bit = 4 bit |
| G         | 1110        | 1         | 1 x 4 bit = 4 bit |
| T         | 1111        | 1         | 1 x 4 bit = 4 bit |
| !         | 010         | 3         | 3 x 3 bit = 9 bit |
|           |             |           | Total = 53 bit |

Example 3: BUKU ANI

![Shannon-Fano encoding tree for example 3](image)

By using Shannon-Fano algorithm, size of data obtained from the tree is 22 bit, as shown in the following table:

| Character | Binary Code | Frequency | Counts      |
|-----------|-------------|-----------|-------------|
| B         | 010         | 1         | 1 x 3 bit = 3 bit |
| U         | 00          | 2         | 2 x 2 bit = 4 bit |
| K         | 011         | 1         | 1 x 3 bit = 3 bit |
| Spasi     | 100         | 1         | 1 x 3 bit = 3 bit |
| A         | 101         | 1         | 1 x 3 bit = 3 bit |
| N         | 110         | 1         | 1 x 3 bit = 3 bit |
| I         | 111         | 1         | 1 x 3 bit = 2 bit |
|           |             |           | Total: 22 bit |

### 4.4. Compression Ratio of ASCII Coding and Shannon-Fano Coding

1. **Compression Ratio for example 1**

\[
\text{Compression Ratio} = \frac{\text{Compressed size}}{\text{Uncompressed size}}
\]
\[
= \frac{69 \text{ bit}}{240 \text{ bit}} \\
= 0.287 \times 100\% \\
= 31\%
\]

2. Compression Ratio for example 2

\[
\text{Compression Ratio} = \frac{\text{Compressed size}}{\text{Uncompressed size}} \\
= \frac{53 \text{ bit}}{128 \text{ bit}} \\
= 0.414 \times 100\% \\
= 42\%
\]

3. Compression Ratio for example 3

\[
\text{Compression Ratio} = \frac{\text{Compressed size}}{\text{Uncompressed size}} \\
= \frac{22 \text{ bit}}{64 \text{ bit}} \\
= 0.343 \times 100\% \\
= 34\%
\]

4.5. Huffman Coding

Example 1: CERRIAARREEIICCEEAAIIEECCRRRCR

By using Huffman algorithm, size of data obtained from the tree is 69 bit, as shown in the following table:

| Character | Binary Code | Frequency |
|-----------|-------------|-----------|
| C         | 11          | 6         |
| E         | 10          | 7         |
| R         | 00          | 8         |
| I         | 101         | 5         |
| A         | 011         | 4         |

| Counts   |
|----------|-----------|
| 6 x 2 bit = 12 bit |
| 7 x 2 bit = 14 bit |
| 8 x 2 bit = 16 bit |
| 5 x 3 bit = 15 bit |
| 4 x 3 bit = 12 bit |
| Total: 69 bit |
Example 2: AYOO SEMANGAT!!!

By using Huffman algorithm, size of data obtained from the tree is 54 bit, as shown in the following table:

Table 8 Huffman coding for example 2

| Character | Binary Code | Frequency | Counts          |
|-----------|-------------|-----------|-----------------|
| A         | 000         | 3         | 3 x 3 bit = 9 bit |
| Y         | 0010        | 1         | 1 x 4 bit = 4 bit |
| O         | 101         | 2         | 2 x 3 bit = 6 bit |
| Spasi     | 0011        | 1         | 1 x 4 bit = 4 bit |
| S         | 110         | 1         | 1 x 3 bit = 3 bit |
| E         | 111         | 1         | 1 x 3 bit = 3 bit |
| M         | 010         | 1         | 1 x 3 bit = 3 bit |
| N         | 0101        | 1         | 1 x 4 bit = 4 bit |
| G         | 0110        | 1         | 1 x 4 bit = 4 bit |
| T         | 111         | 1         | 1 x 3 bit = 3 bit |
| !         | 100         | 3         | 3 x 3 bit = 9 bit |
|           |             |           | Total: 54 bit    |

Example 3: BUKU ANI

By using Huffman algorithm, size of data obtained from the tree is 22 bit, as shown in the following table:

Table 9 Huffman coding tree for example 3

| Character | Binary Code | Frequency | Counts          |
|-----------|-------------|-----------|-----------------|
| B         | 010         | 1         | 1 x 3 bit = 3 bit |
| U         | 00          | 2         | 2 x 2 bit = 2 bit |
| K         | 011         | 1         | 1 x 3 bit = 3 bit |
| Spasi     | 100         | 1         | 1 x 3 bit = 3 bit |
| A         | 101         | 1         | 1 x 3 bit = 3 bit |
To determine how effectiveness Shannon-Fano algorithm in data compression, then size of data from Shannon-Fano coding compared with size of data from Huffman coding and obtained the following result:

| Example                     | Shannon-Fano | Huffman |
|-----------------------------|--------------|---------|
| CERIA                       | 75 bit       | 69 bit  |
| AYOO SEMANGAT!!!           | 53 bit       | 54 bit  |
| BUKU ANI                    | 23 bit       | 22 bit  |
| PROGRAM STUDI MATEMATIKA   | 88 bit       | 99 bit  |
| JAGALAH KEBERSIHAN         | 52 bit       | 54 bit  |
| WAHANA                      | 11 bit       | 12 bit  |

Table 10 Comparison of Shannon-Fano and Huffman

Based on the result of compression in example 1 and 3 can be said that the Shannon-Fano algorithm has same effectiveness with Huffman algorithm when all character in string all repeated and when the statement short and just one character in the statement that repeated, but the Shannon-Fano algorithm more effective then Huffman algorithm when the data has a long statement and data text have more combination character in statement or in string/word.

5. CONCLUSIONS

According to result and discussion, some conclusions are taken such as:
1. Compression using Shannon – Fano algorithm in data text resulting a smaller size of data than the size of the data by ASCII encoding.
2. Compression ratio between ASCII and Shannon-Fano algorithm explains that some data with a large size can be some data with a small size, through the compression process.
3. When compared with the Huffman algorithm, when all character in string all repeated and when the statement short and just one character in the statement that repeated, but the Shannon-Fano algorithm more effective then Huffman algorithm when the data has a long statement and data text have more combination character in statement or in string/word.

5. REFERENCES

[1] Adhitama, G. 2009. Perbandingan Algoritma Huffman dengan Algoritma Shannon- Fano. Program Studi Teknik Informatika, Institut Temologi Bandung. Elangga, Jakarta.
[2] Altraweneh, H and M. Altarweneh. 2011. Data Compression Techniques on Text File: A Comparison Study, International Journal of Computer Applications. Vol 26 (5).
[3] Blelloch, G. E. 2010. Introduction to Data Compression. Computer Science Department, Carnegie Mellon University.
[4] Kandaga, Tjatur. 2006. Analisis Penerapan Kompresi dan Dekompresi Data dengan Menggunakan Metode Statistik dan Kamus. Jurnal Informatika. Vol 2 (2).
[5] Salomon, David. 2004. Data Compression 3rd Edition: The Complete Reference, Springer–Verlag, New York.
[6] Salomon, David. 2007. Variable – Length Codes for Data Compression, Springer – Verlag, London.
[7] Salomon, David. 2008. A Concise Introduction to Data Compression, Springer – Verlag, London.
[8] Sedgewick, R and K. Wayne. 2011. Algorithms Fourth Edition. Princeton University. Addison-Wesley, Boston.
[9] Shanmugasundaram and Lourdusamy. 2011. A Comparative Study of Text Compression Algorithms. International Journal of Wisdom Based Computing. Vol 11 (3).