Study on Text Compression

Mohammad Shahriar Reza

011132065

Samrin Ahmed Riya

011142021

Samsey Ara Alam

011132019

Md Al Amin Hossain

011103017

A thesis in the Department of Computer Science and Engineering presented

in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

November 2018

Declaration

We Mohammad Shahriar Reza, Samrin Ahmed Riya, Samsey Ara Alam, and Md Al Amin Hossain declare that this thesis "**Study on Text Compression**" and the work presented in it are our own. We confirm that:

- This work was done wholly or mainly while in candidature for a BSc degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all the main sources of help.
- Where the thesis is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what we have contributed ourselves.

_____

Mohammad Shahriar Reza

ID: 011132065

Department of CSE

_____

Samrin Ahmed Riya

011142021

Department of CSE

_____

Samsey Ara Alam

011132019

Department of CSE

_____

Md Al Amin Hossain

011103017

Department of CSE

Certificate

I do hereby declare that the research works embodied in this thesis entitled "**Study on Text Compression**" is the outcome of an original work carried out by Mohammad Shahriar Reza, Samrin Ahmed Riya, Samsay Ara Alam and Md Al Amin Hossain under our supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of BSc in Computer Science and Engineering.

_____

Dr. Mohammad Nurul Huda

Professor, Dept. of CSE,
United International University

Abstract

In this task, some existing machine learning algorithms for compressing and decompressing texts are analyzed to compare the performances of the algorithms and pick one for further work. This study is done to develop a strategy like Heuristic Algorithm and collaborate with a peer to find a solution to a text compression problem using the Text Compression Widget (Lossless Compression Scheme).

Text compression is a process of reducing the size of the text by encoding it efficiently. In our day to day life, sometimes we reach some points where the physical limit of a sent text or data needs to be faster to work with. This large amount of information is needed to be compressed to achieve a faster transformation. The ways to make text compression should be adaptive and challengeable. A large data of English Language with high redundancy is used to apply the techniques of compression and decompression. Compression of texts reduces the data storage space. With the reduction or elimination of redundancies, data compression is accomplished. The encoding technique uses fewer bits than the original one and unnecessary information is lost in this case. It follows a lossless technique where the number of bits and bytes are cut off to store the information and retrieve some of the physical limits. The decompression technique is a reverse of the compression technique. Therefore, some factors like time, text size and type, efficiency and throughput of an algorithm make the techniques challenging.

Keywords - compression, decompression, lossless technique.

In memory of family/friends:

Mohammad Shahriar Reza – First of all, the entire dedication of this research work goes to the Almighty Allah who always blesses us with blessings that are good for us. I would like to show my gratitude towards my parents who are always with me in every situation with their love and support. My friends and my group mates also deserve this recognition as they inspire me all the way to carry out this work.

Samrin Ahmed Riya – My first and foremost dedication for this research work goes to the Almighty Allah who helped us in the way of accomplishing this work. I also want to dedicate this piece of work to my beloved family for always being there for me with their eternal love and support. Again I want to dedicate this work to my lovable friends for their encouragement. Lastly, the warm dedication goes to my lovely group mates for making this research possible.

Md Al Amin Hossain – I dedicate this thesis to the Almighty Allah, to my family members, to my friends, and to my group mates.

Samsay Ara Alam – I want to dedicate this thesis to the Almighty Allah, to my family for their support, to my sister, to my friends and mates.

Acknowledgement

First of all, we would like to convey our heartiest gratefulness to Dr. Swakkhor Shatabda, Coordinator, Mir Muhammad Munir, Registrar, and Dr. Salekul Islam, Head of Department of Computer Science and Engineering at United International University, Dhaka for allowing us to carry out this work.

We are earnestly thankful to our supervisor Dr. Muhammad Nurul Huda, Associate Professor of Computer Science and Engineering Department at United International University, Dhaka. His constant support, invertible suggestions, and valuable advice help us to carry the research to the peak of completion.

We are also thankful to Mrs. Ayesha Akter, Junior Officer of MSCSE program at United International University, Dhaka for her support and assistance to our work.

We would also like to show our gratitude to each other. We all the members of the group were sincere all the time and put our best foot forward throughout the task till its completion.

Finally, we would like to express our warmth of welcome and admiration to our parents for always there for us with their ultimate support, love, and care throughout the research which takes us to the peak of success.

Contents

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Text Compression

Text compression is a process of reducing the text's size by encoding its information more efficiently. It uses the number of bits and bytes to store the information which is reduced to a certain extent. It needs a lossless technique where no single bit of data is lost when the file is decompressed. This approach takes the text file back to its original state. The techniques for compressing and decompressing texts are intended for natural languages like a large data of English Language with high redundancy and other data with a similar sequential structure such as source code of a program. However, the technique can be applied to any kind of data to achieve some compression.

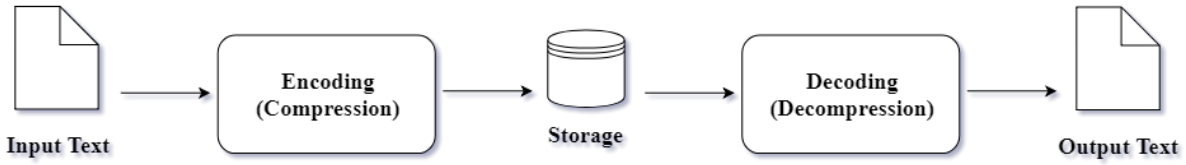

Figure 1. 1 Text Compression Technique

1.2 Importance of Text Compression

There are several significance of compressing text:

- Reduction in storage hardware, data transmission time and communication bandwidth.
- Uses of compressed file can lead to a significant decrease in expense for disk or solid-state-drives.
- Improvement in network bandwidth.
- Cost saving.

1.3 Limitations of Text Compression

- Impact on the performance resulting from CPU and Memory use while compressing and decompressing.

- Not suitable for all input data sets.
- Through put of compression and decompression varies based on algorithms.
- Time consuming for some arbitrary data sets like medical data, or data from bioinformatics field.

1.4 Objectives of the Thesis

- Analyzing some existing machine learning algorithms for compressing and decompressing texts.
- Explain some factors that make compression challenging.
- Describe the purpose and rationale for lossless compression.
- Discover some importance of text compression in real life.

1.5 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 provides the background of the thesis.

Chapter 3 presents the methods used for this thesis.

Chapter 4 discusses the results and experimental analysis.

Chapter 5 presents the conclusions, summaries the thesis contributions, and discusses the future works.

Chapter 2

Background

2.1 Data Compression

The idea of text compression has come from the initiatives of data compression. With the elimination of redundancies in a storage of data, data compression is achieved. The encoding technique takes fewer bits than the original one. Unnecessary information is lost in this case which is a lossless technique. It is lossy when some information is lost. Moreover data compression reduces the data storage space. The decompression technique is just reverse of the compression technique.

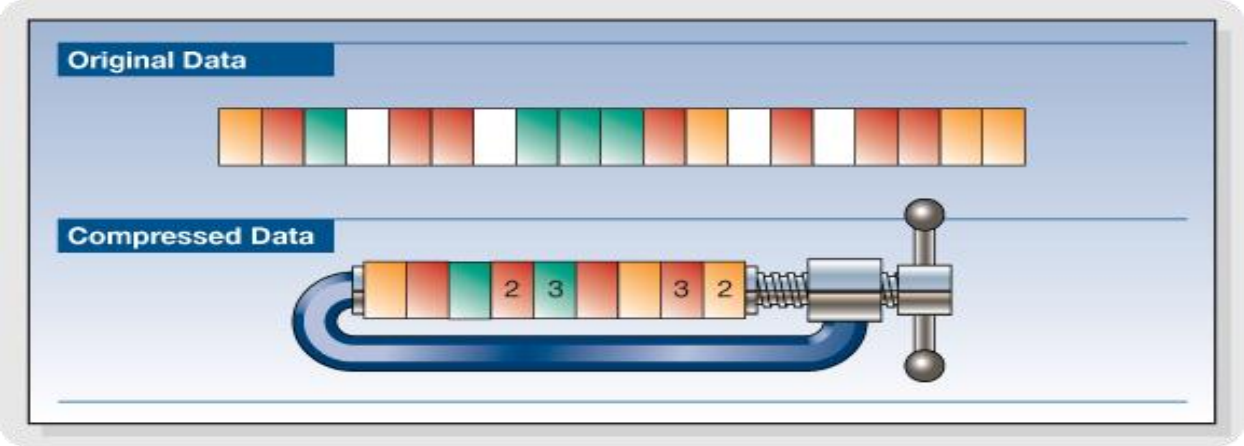

Figure 2. 1 Data Compression Process

2.2 Advantages of Data Compression

- Reduces transmission time needed over the network.
- Data must be decompressed or decoded to be reused.
- Symmetrical or Asymmetrical.
- Software or Hardware.
- Variations of ASCII.

2.3 Fields where data compression uses

- Literary works.

- Product catalogues.
- Genomic databases.
- Raw text databases.

2.4 Methods used for Data Compression

To compress data, there are basically two types of methods. One is the lossless method and another one is the lossy method. In the lossless method, no information or data is lost in the process of compression and decompression. Lossless methods are being used when any text or programs are needed to be compressed and decompressed. On the contrary, data can be lost while compression and decompression in lossy methods. Image, video, and audio are being compressed using lossy methods.

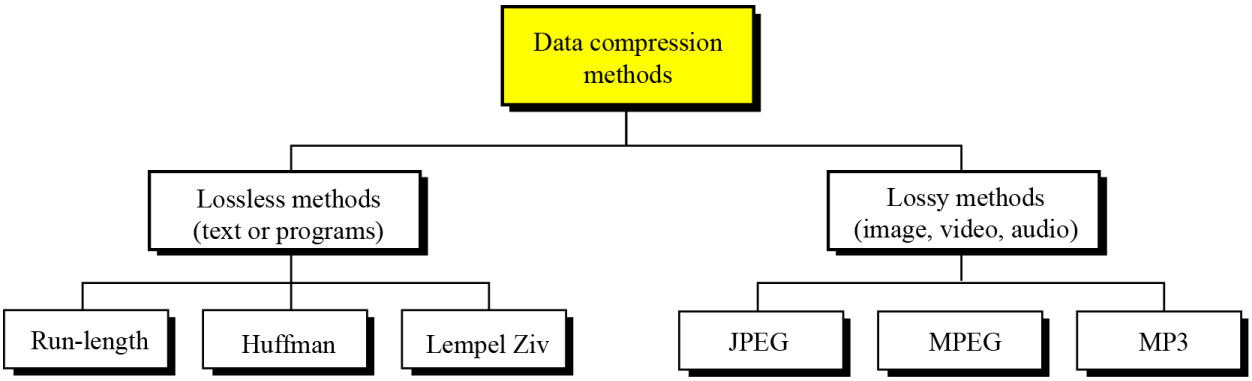

Figure 2. 2 Typical Data Compression Techniques

Chapter 3

Proposed Methods

3.1 Shannon-Fano Coding

The main idea behind the compression is to create such a code, for which the average length of the encoding vector (word) will not exceed the entropy of the original ensemble of messages. In general, the codes used for compression are not uniform. Shannon-Fano encoding constructs reasonably efficient separable binary codes for sources without memory.
Among all the methods, Shannon-Fano encoding is the first established and widely used encoding method. This method and the corresponding code were invented simultaneously and independently of each other by C. Shannon and R. Fano in 1948.

3.1.1 Shannon-Fano Algorithm

Shannon-Fano encoding constructs reasonably efficient separable binary codes for sources without memory. The algorithm works as follows:

1. Start with a list of symbols and their frequencies in the alphabet.
2. Sort the list of symbols according to the frequency in descending order.
3. Split this list into two sections, with the total frequency number of the left section being as close as possible to the total of the right.
4. Assign the binary digit 0 to left section and 1 to another; the symbols in the left part have codes starting with 0 and symbols in the right part with 1.
5. Recursively apply the step 3 & 4 to each of the sections, adding to the codes until no further split can be achieved.

3.1.2 Applications

- Encode and decode JPEG files.
- Low redundant audio files.

3.1.3 Pros and cons

- The transmission of the encoded message is reasonably efficient that is, 1 and 0 appear independently and with “almost” equal probabilities. This ensures transmission of “almost” 1 bit of information per digit of the encoded messages.
- It should be taken into account that the Shannon-Fano code is not unique because it depends on the partitioning of the input set of messages, which, in turn, is not unique.

3.1.4 Example

Input: MOHAMMAD NURUL HUDA

Table 3. 1 Frequency and binary code for each character

Character/Symbol	Frequency	Binary Code
M	3	000
A	3	001
U	3	010
H	2	0110
D	2	0111
_ (space)	2	100
O	1	101
N	1	110
R	1	1110
L	1	1111

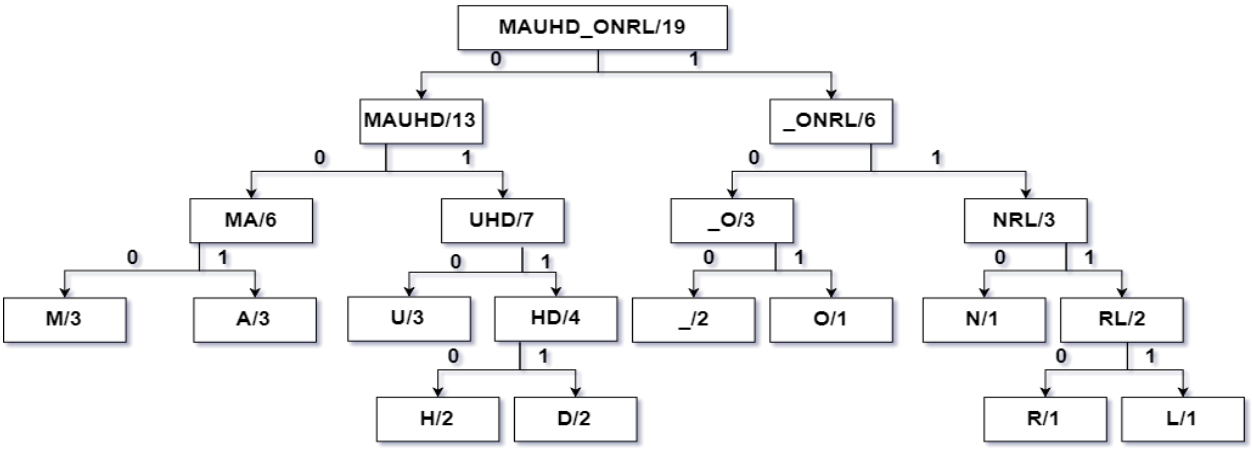

Figure 3. 1 Shannon-Fano Tree

Output: (Compressed string)
000101011000100000000101111001100101110010111110001100100111001

3.2 Huffman Coding

While working on the term paper assigned by the professor Robert M. Fano in 1952, David A. Huffman, the student of MIT discovered this algorithm. He thought of the idea of using a frequency sorted binary tree to find out the most efficient binary code. Huffman went against the

major flaw of Claude Shannon Fano coding and built the tree in bottom up order instead of the top down order. Among lots of compression algorithms, Huffman algorithm is so efficient that it is used to ensure the lossless compression in a sort of smallest application. Huffman Coding is used for image compression for example PNG, JPG. Watching movies or videos in computer, the chances are that its image will be in MPEG format, so in this case static and dynamic Huffman Coding techniques are used for PNG, MPEG and for text compression.
To store information or the strings of data, there are so many ways but the computer scientist are always looking for some new and better ways which use as little space as possible. In this regard, the Huffman coding can be used as a medium of storing the strings of data or information as binary code in an efficient approach. Huffman coding uses "lossless data compression", which means no information is lost. Huffman coding also uses 'variable length coding', which indicates that a binary symbol is prepared based on how often the symbol is used in the text. In short, the symbols in the data are encoded or converted to binary symbol based on the frequency of the symbol. For instance, if a character 'T' is used in the data a lot, the binary symbol represents that it is shorter. Otherwise the representation of the binary symbol is longer in the case of the particular character. However using this method, all the strings of data will take less physical space in the memory when encoded. Each character is given binary code using a method which is based on the tree approach.

3.2.1 Huffman Algorithm

1. Start with a list of symbols and their frequencies in the alphabet.
2. Select two symbols with the lowest frequency.
3. Add their frequencies and reduce the symbols.
4. Repeat the process starting from step-2 until only two values remain.
5. The algorithm creates a prefix code for each symbols back. It assigns 0 and 1 for each frequency value in each phase.

3.2.2 Applications

- Generic file compression.
    - Files: GZIP, BZIP, 7z.
    - Archivers: PKZIP.
    - File Systems: NTFS, HFS+, ZFS.
- Multimedia.
    - Images: GIF, JPEG.
    - Sound: MP3.
    - Video: MPEG, DivX, HDTV.
- Communication.
    - ITU-T T4 Group 3 Fax.
    - V.42bis modem.
    - Skype.
- Databases: Google, Facebook etc.

3.2.3 Pros and Cons

- Time efficiency of building the Huffman tree is important.

- The insert and delete operations each takes log (N) time and they are repeated at most two times. Therefore, the runtime is O (2NlogN) = O (NlogN).

3.2.4 Example

Input: MOHAMMAD NURUL HUDA

Table 3. 2 Frequency and binary code for each character

Character/Symbol	Frequency	Binary Code
L	1	0110
R	1	0111
N	1	0100
O	1	0101
H	2	000
_ (space)	2	001
D	2	100
U	3	101
A	3	110
M	3	111

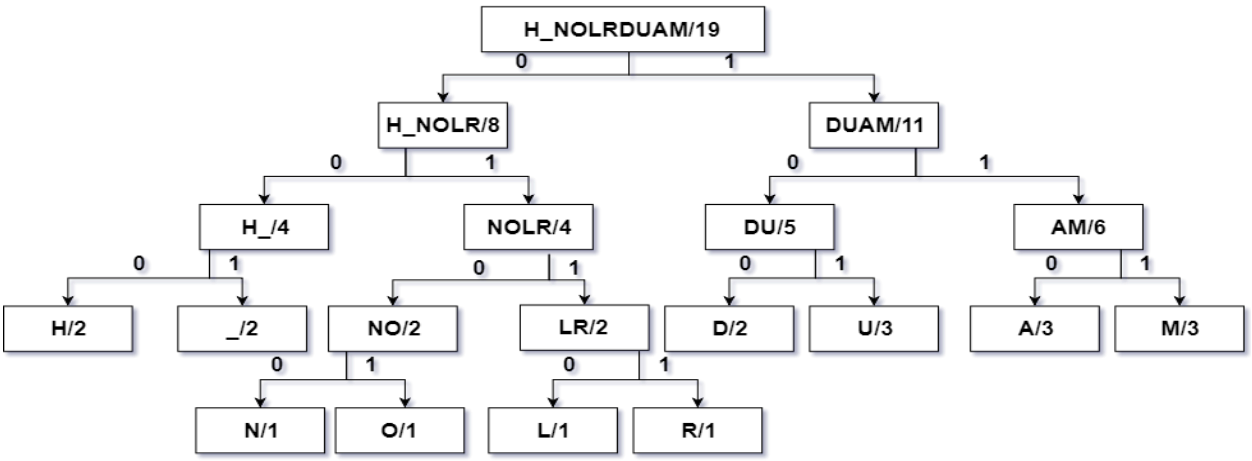

Figure 3. 2 Huffman Tree

Output: (Compressed string)
111010100011011111111010000101001010111101011000100010110011 0

3.3 Repeated Huffman Coding

The purpose of data compression technique is to compress the original strings of data into a form which is condensed into size and moreover the original data can be extracted or decompressed from the compressed one. This technique is used to reduce the storage space as per as requirements and a handsome communication cost over the network. While using the Huffman coding compression technique, the frequency or occurrence of each symbol is countable in the message and then a Huffman tree is constructed based on the symbol probabilities. Based on the tree construction, a code for each input symbol is generated. Then the compressor constructs a compressed output stream using these codes from the Huffman tree input symbols. All the redundant symbols are also countable in this case and these also take extra bits while encoding. If possible, this situation is accountable to be getting ridden off to reduce the text size. Two passes over the text are required in this technique. This creates a hindrance when it is used for network communication. The extra disk access can slow down the operation while using in file compression applications. A one pass scheme is independently proposed by Faller and Gallager, which has been enhanced substantially by Knuth for dynamic Huffman codes, usually known as FGK codes. The dynamic Huffman coding technique has been analyzed by J. S. Vitter and an optimal algorithm has also been proposed by him. He also derived a tight upper bound for the dynamic Huffman codes. Some problems have also been found in using the conventional Huffman coding. One of the most major problems is that the whole stream must be read prior to coding. This creates a foremost problem when the file size is too large. By introducing the block Huffman coding, Mannan and Kaykobad solved the problem of the large file size issue. If the Huffman coding technique can be applied effectively on a file again and again, then it is called repeated Huffman coding. Applying the Huffman Coding again and again, it is expected that the encoded text length would be getting smaller in each iteration of the coding while compressing the tree itself will be an overhead in each pass. So reiteration count will depend upon the efficient use of the Huffman tree representation. If a Huffman tree can be represented efficiently in memory, repeated Huffman coding technique can be applied in an effective number of times. The compression ratio will be increased if the repetition of Huffman coding is performed efficiently.

3.3.1 Repeated Huffman Algorithm

1. Start with a list of symbols and their frequencies in the alphabet.
2. Select two symbols with the lowest frequency.
3. Add their frequencies and reduce the symbols.
4. Repeat the process starting from step-2 until only two values remain.
5. The algorithm creates a prefix code for each symbols back. It assigns 0 and 1 for each frequency value in each phase.
6. As long as the current compression ratio is better than the previous compression ratio, repeat the whole process from step-1 to step-5.

3.3.2 Applications

- Same as Huffman coding.

3.3.3 Pros and cons

- A proficient Huffman tree is very much needed to maintain the performance of repeated Huffman coding and also the recurrence of the algorithm depends on it.

3.3.4 Example

Input: MOHAMMAD NURUL HUDA

Table 3. 3 Frequency and binary code for each character

Character/Symbol	Frequency	Binary Code
L	1	0110
R	1	0111
N	1	0100
O	1	0101
H	2	000
_ (space)	2	001
D	2	100
U	3	101
A	3	110
M	3	111

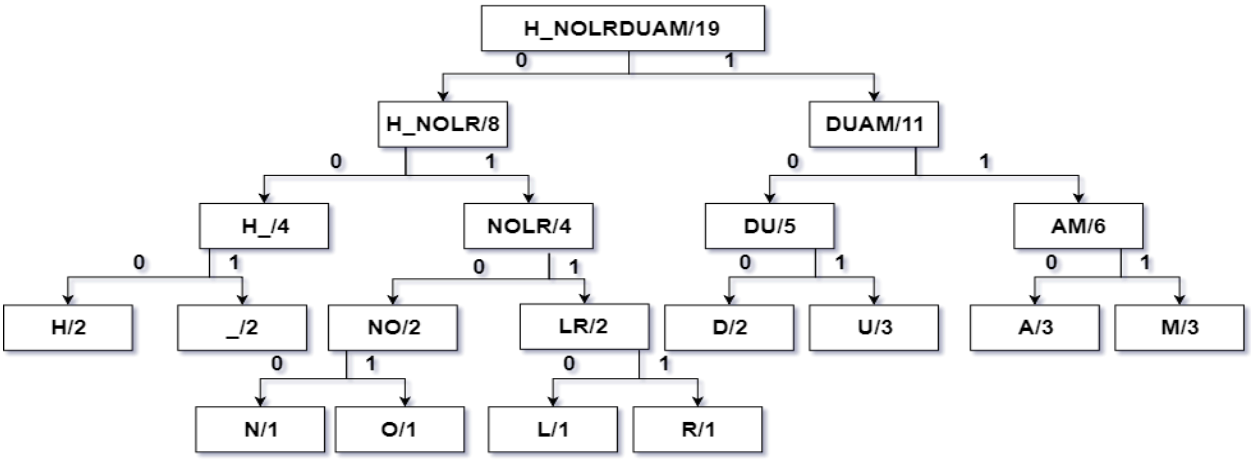

Figure 3. 3 Repeated Huffman Tree

Output: (Compressed string)
11101010001101111111101000010100101011110101100010001011001l0

3.4 Lempel-Ziv-Welch (LZW) Coding

LZW is the first letters of the name of the scientists Abraham Lempel, Jakob Ziv and Terry Welch who developed this algorithm. It is a lossless compression algorithm hence works well for text compression. Lempel Ziv Welch (LZW) encoding is an example of a category of an algorithm called dictionary-based encoding. Usually, it starts with a dictionary of all the single characters and gradually builds the dictionary as the information is sent through. It tries to identify repeated sequences of data and adds them to the code table or dictionary. It is a general compression algorithm capable of working on almost any type of data. Large size Text files in the English language can be typically be compressed to half its size.

3.4.1 Lempel-Ziv-Welch (LZW) Algorithm

```
Read OLD_CODE
Output OLD_CODE
CHARACTER = OLD_CODE
WHERE there are still input characters DO
        Read NEW_CODE
        IF NEW_CODE is not in the translation table THEN
                STRING = get translation of OLD_CODE
                STRING = STRING + CHARACTER
        ELSE
                STRING = get translation of NEW_CODE
        END of IF
        Output STRING
        CHARACTER = first character in STRING
        Add OLD_CODE + CHARACTER to the translation table
        OLD_CODE = NEW_CODE
END of WHILE
```

3.4.2 Applications

- Used in GIF (Graphics Interchange Format) to reduce the size without degrading the visual quality.
- Used in TIFF and PDF files.

3.4.3 Pros and cons

- This is a lossless compression technique as none of the contents in the file are lost during or after compression.
- LZW algorithm is efficient since it doesn't need to send the string table to the decompression code.

- Useful only for a large amount of text data when redundancy is high.

3.4.4 Example

Input: MOHAMMAD NURUL HUDA

Table 3. 4 Initial Dictionary

Index	Character/Symbol
0	M
1	O
2	H
3	A
4	D
5	_ (space)
6	N
7	U
8	R
9	L

Table 3. 5 New Dictionary

Index	Character/Symbol
0	M
1	O
2	H
3	A
4	D
5	_ (space)
6	N
7	U
8	R
9	L
10	MO
11	OH
12	HA
13	AM
14	MM
15	MA
16	AD
17	D_
18	_N
19	NU
20	UR

21	RU
22	UL
23	L_
24	_H
25	HU
26	UD
27	DA

Output: (Compressed string) 0,1,2,3,0,0,3,4,5,6,7,8,7,9,5,2,7,4,3

Chapter 4

Experimental Analysis

4.1 Datasets

In this thesis, as an experiment, 4 compression and decompression techniques: Shanon-Fano coding, Huffman coding, repeated Huffman coding, and Lempel Ziv Welch Coding have been used. 10 different sizes of text file have been used as dataset and the compression ratio has been measured to determine which compression technique is better. And for every individual compression methods, the extensions of the compressed files are different.

Table 4. 1 File Description

File #	Original Size (bytes)	File Extension
1	19	.txt
2	673	.txt
3	2,903	.txt
4	4,923	.txt
5	9,124	.txt
6	14,551	.txt
7	19,889	.txt
8	27,936	.txt
9	47,116	.txt
10	98,733	.txt

Table 4. 2 Compressed File Description

Methods	Compressed File Extension
Shannon-Fano Coding	.sfcf
Huffman Coding	.hcf
Repeated Huffman Conding	.rhcf
Lempel Ziv Welch Coding	.lzwcf

4.2 System Diagram

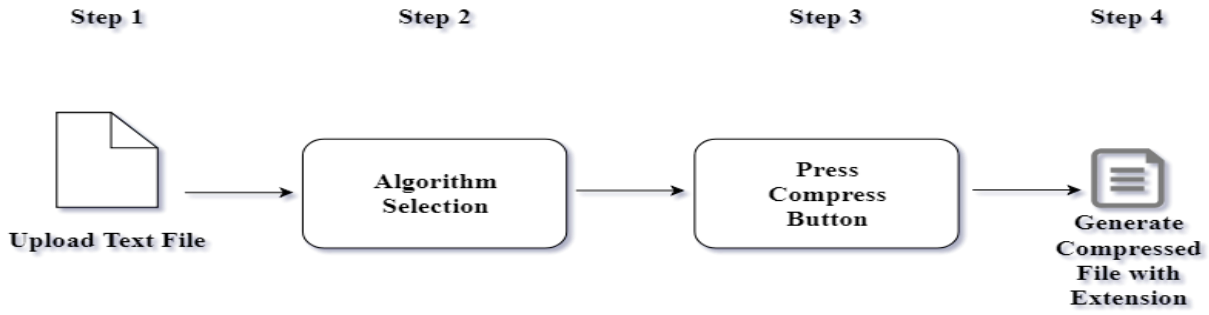

Figure 4. 1 Compression Methods

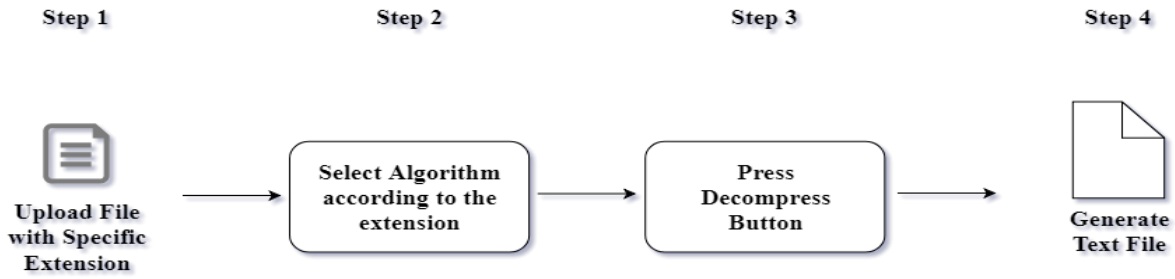

Figure 4. 2 Decompression Methods

4.3 Experimental Results

4.3.1 Shannon-Fano Coding

Table 4. 3 Compression Ratio for Shannon-Fano Coding

Original Size (bytes)	Compressed Size (bytes)	Compression Ratio (%)
19	8.5	57
673	432	35
2,903	2,160.26	24
4,923	3,778.125	22
9,124	7,305.875	19
14,551	11,146.125	23
19,889	15,368	22

27,936	20,009.875	28
47,116	38,701.75	17
98,733	81,141	17

4.3.2 Huffman Coding

Table 4. 4 Compression Ratio for Huffman Coding

Original Size (bytes)	Compressed Size (bytes)	Compression Ratio (%)
19	8.375	58
673	361.75	45
2,903	1,629.25	43
4,923	2,727.625	44
9,124	5,145.125	43
14,551	8,027.625	44
19,889	10,961.625	44
27,936	15,845.5	43
47,116	27,118.5	42
98,733	56,857	42

4.3.3 Repeated Huffman Coding

Table 4. 5 Compression Ratio for Repeated Huffman Coding

Original Size (bytes)	Compressed Size (bytes)	Compression Ratio (%)
19	8.275	58
673	360.75	45
2,903	1,629.25	43
4,923	2,717.625	44
9,124	5,140.15	43
14,551	8,017.62	44
19,889	10,951.625	44
27,936	15,845	43
47,116	27,108.5	42
98,733	56,857	42

4.3.4 Lempel Ziv Welch (LZW) Coding

Table 4. 6 Compression Ratio for Lempel Ziv Welch Coding

Original Size (bytes)	Compressed Size (bytes)	Compression Ratio (%)
19	2.5	86
673	102.25	84
2,903	430.5	85
4,923	698.625	85
9,124	1,329.375	85
14,551	1,905.25	86
19,889	2,546.125	87
27,936	3,130.25	88
47,116	6,192.875	86
98,733	12,470.625	87

4.4 Summery

After analyzing the results, we got that the compression ratio for Lempel Ziv Welch (LZW) coding for all the different files is better than the other compression ratio. It indicates that LZW coding can compress much better than the other three compression methods whether the size of the files is large or small.

Table 4. 7 Comparison of the Compression Ratio

Original Size (bytes)	Compression Ratio			
	Shannon-Fano Coding	Huffman Coding	Repeated Huffman Coding	Lempel Ziv Welch Coding
19	57	58	58	86
673	35	45	45	84
2,903	24	43	43	85
4,923	22	44	44	85
9,124	19	43	43	85
14,551	23	44	44	86
19,889	22	44	44	87
27,936	28	43	43	88
47,116	17	42	42	86
98,733	17	42	42	87

Chapter 5

Conclusion & Future Work

5.1 Conclusion

Text Compression is very important to some extent in real life. It needs lossless technique so that no efficient information is lost while compressing or decompressing the data. These techniques enhance our physical memory storage which is very helpful for the performance of a device as well as of a program. A large text data, in English language with high redundancy should be used to compress. An adaptive, changeable and complex way is sometimes needed to compress text.

5.2 Future Work

In future, we will try to develop a strategy (Heuristic Algorithm) for compressing text. We will also try to collaborate with a peer to find a solution to a text compression problem using the Text Compression Widget (Lossless Compression Scheme). Our main motto will be to make compression easier ever than before.

Bibliography

[1] Senthil Shanmugasundaram, Robert Lourdusamy, "A Comparative Study Of Text Compression Algorithms", *International Journal of Wisdom Based Computing, Vol. 1 (3), December 2011*.

[2] Mohammed Al-laham, Ibrahiem M. M. El Emary, "Comparative Study between Various Algorithms of Data Compression Techniques", *IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.4, April 2007.*

[3] S.R. Kodituwakku, U. S. Amarasingh, "Comparison of lossless data compression algorithms for text data*", S.R. Kodituwakku et. al. / Indian Journal of Computer Science and Engineering, Vol 1 No 4 416-426.*

[4] R. N. Williams, "An extremely fast Ziv-Lempel data compression algorithm", *[1991] Proceedings. Data Compression Conference*, Snowbird, UT, USA, 1991, pp. 362-371.

[5] H. Kruse and A. Mukherjee, "Preprocessing text to improve compression ratios", *Proceedings DCC '98 Data Compression Conference (Cat. No.98TB100225)*, Snowbird, UT, USA, 1998, pp. 556.

[6] R. Horspool and G. Cormack, "Constructing word-based text compression algorithms", *1992 Data Compression Conference(DCC), Snowbird, UT, USA, , pp. 62-71.*

[7] H. Yokoo, "Improved variations relating the Ziv-Lempel and Welch-type algorithms for sequential data compression*", in* IEEE Transactions on Information Theory*, vol. 38, no. 1, pp. 73-81, Jan. 1992.*

[8] T. Mantoro, M. A. Ayu and Y. Anggraini, "The performance of text file compression using Shannon-Fano and Huffman on small mobile devices", *2017 International Conference on Computing, Engineering, and Design (ICCED)*, Kuala Lumpur, 2017, pp. 1-5.

[9] Amit Jain, Kamaljit I. Lakhtaria, "Comparative Study of Dictionary based Compression Algorithms on Text Data", *IJCSNS International Journal of Computer Science and Network Security, VOL.16 No.2, February 2016*.

[10] Mamta Sharma, "Compression Using Huffman Coding", *IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010*.

[11] J. B. Connell, "A Huffman-Shannon-Fano code", *in* Proceedings of the IEEE*, vol. 61, no. 7, pp. 1046-1047, July 1973.*

[12] Robbi Rahim, Muhammad Dahria, Muhammad Syahril, Badrul Anwar, "Combination of the Blowfish and Lempel-Ziv-Welch algorithms for text compression", *World Transactions on Engineering and Technology Education Vol.15, No.3, 2017*.

[13] W. D. Maurer, "File Compression Using Huffman Coding", *in Computing Methods in Optimization, L. A. Zadeh et al., eds., Vol. 2, Academic Press, New York, 1969, pp. 247-256.*

[14] P. G. Sorenson and A. Listoe, "Compression Code Techniques for Textual Data Bases", *Proc. CIPS 76, 1976, pp.78-101.*

[15] S. S. Ruth and P. J. Kreutzer, "Data Compression for Large Business Files", *Datamation, Sept. 1972, pp. 62-66.*

[16] Md. Bahlul Haider, Md. Jahid Hossain, Shafi Ullah Afrad, S. M. Faisal Farman, Dr. M. Kaykobad, "A Study on Repeated Huffman Coding".

[17] Chung, K.L., "Efficient Huffman decoding*", Inform. Process. Lett. 61 (1997) 9799.*

[18] Chowdhury, Rezaul Alam, Kaykobad, M. and King Irwin, "An efficient decoding technique for Huffman codes", *Inform. Process. Lett. 81 (2002) 305-308.*

[19] Chen, H.-C., Wang, Y.-L. and Lan, Y.-F., "A memory efficient and fast Huffman decoding algorithm", *Inform. Process. Lett. 69 (1999) 119-122.*

[20] Kashfia Sailunaz, Mohammed Rokibul Alam Kotwal, Mohammad Nurul Huda, "Data Compression Considering Text Files", *International Journal of Computer Applications (0975 – 8887) Volume 90 – No 11, March 2014.*

[21] Yogesh Rathore, Manish k. Ahirwar, Rajeev Pandey, "A Brief Study of Data Compression Algorithms", *(IJCSIS) International Journal of Computer Science and Information Security, Vol. 11, No. 10, October 2013.*