Application of Text File Compression and Decompression Using Huffman Algorithms in C#

Win Win Maw
University of Computer Studies(Mandalay)
winwinmaw.tukse@gmail.com

San San Lwin
Technological University(Kyaukse)
sansanlwinmost@gmail.com

Abstract

In nowadays, for communication or some other purpose a large amount of data is sent from one place to another place. A large amount of data communication can be into high cost and lower down the performance. So,this paper has developed various systems to compress the text data to minimize data transfer cost and to rise the performance of the communication channel. In a process of data compression Text file or audio file or video file may be transformed to another compact file. This file is that original fully file recovers from the original file without any loss of actual information. If one wants to save the storage space, this process will be useful. Also compressed files are much more easily exchanged on the internet since they can be uploaded and downloaded much faster. In this paper, a data compression algorithm is represented using advanced text compression algorithm which uses Huffman Coding to compress and decompress the text data. A technology for reducing the size of data used to describe any content is compression. The number of bits required to store and/or transmit digital media is reduced by compression. Compression technique is used for storing easier for large amount of data. Simple of translating the stream of prefix codes to individual byte values the process of decompression.

Keywords:Huffman, Compression, decompression, text, C#, algorithm

1. Introduction

Compression of multimedia data is intended to conserve storage space or to cram more data into the same amount. Compression of data requires not just compression methods but can operate on various digital data types, such as characters or bytes in data files or whatever. The literature on data compression frequently refers to data compression as data encoding and decompression as decoding. Decompression means the compressed data is returned to its original form which is often referred to as expansion. The term message is used to denote short strings of data. Data compression method is used to save disk space, reducing communication time, or the time needed to transfer data, and more. Data compression technology is seen as a significant element in helping the information system under these circumstances. Data compression technology is used not only in personal computers (PCs), but also in many fields such as modems, routers, digital cameras, facsimiles, compact disks (CDs), mini disk (MDs), video on demand (VOD), television (TV) conference system, wireless telephones and other fields, and yet data compression techniques usually apply to data stored in archives and file transfer over telephone lines. Huffman has been able to design this most effective compression method: no further mapping of specific source symbols to similar strings. A smaller average output size can be created of bits when the actual symbol frequencies suit those used to create the code.

The simplest construction algorithm uses a priority queue in which highest priority is given to the node with the lowest probability:

- Create a leaf node for every symbol, and add it to the priority list.
- If it has more than one node in the queue:
- Delete two highest priority nodes (lowest probability) from the queue
- Create a bigger internal node with these two nodes as children and with a probability equal to the sum of both nodes' probabilities.
- A new node in the queue is added.
- The root node is left, and the tree is complete.

2. Methodology

2.1. Data Compression

Data compression is the process of replacing real data with coded data by using model-based coding paradigm, an input string of symbols, and a model that generates an encoded string that is compressed input version. The decoder that must have the same access Pattern, encoded string regenerates from the same input string. Input symbols are taken from some well-defined set such as the ASCII or binary alphabets; a simple sequence of bits is the encoded string. Compression is achieved by conveying the more likely symbols in fewer bits than the less likely ones. The method is a system to compute the distribution of probabilities in any given context for the next input symbol, and more complex models can provide more precise probabilistic predictions and thus achieve greater compression.

2.1.1. Data Compression Modeling

If data compression is used in a data transmission program the target is speed. The speed of transmission depends on the number of bits sent, the time it takes for the encoder to produce the coded message and the time it takes for the decoder to retrieve the original package. All symbols are individual.The concept entropy has its roots in thermodynamics. The higher the entropy of the

Post, the more instructive it will be.The description entropy has its roots in thermodynamics. The higher the entropy of the Post, the more data it contains. The entropy of a symbol is defined as having negative logarithms as its probabilities. The zero order entropy H (p) can be expressed as n for the determination of the information quality of a message in bits.

H (p) =-Pi log2 Pi bits / symbol respectively (i=1)

Where n= no. separate symbols &

Pi = Probability of symbol occurrence

Entropy determines a limit on the compressive strength of source for a lesser source of memory. Entropy of an entire message is essentially the sum of any single word's entropy. Entropy specifies a limit on compressive strength of the source for a lesser source of memory. Entropy of an entire message is essentially the sum of any single word's entropy.

2.1.2. Types of Data Compression

Methods of should data be compressed graded in several ways. One of the most important criteria for classification is whether the compression algorithms remove certain pieces of data which cannot be recovered during decompression. The algorithm which removes some piece of data is called data loss compression. The compression algorithm for loss data is usually used if the origin is in perfect continuity. The algorithm that receives the same after decompression is called lossless compression of data as in. Due to regulatory law, lossless data compression is used in text files, database tables and in medical image.Different lossless algorithms for compression of data were proposed and used. Huffman Coding, Run Length Encoding, Arithmetic Encoding, and Dictionary Based Encoding are some of the main techniques.

2.2. System Methodology

Advanced Algorithm for Intelligent Text Compression operates for the encryption of text data in two processes. In the first phase data is compressed using the dynamic bit reduction process, and in the second phase Huffman coding is used to further compress the data in order to generate the final result. The program must evaluate the number of specific symbols in the input text string in the first step when the user enters an input text data, and then assign numerical code to these unique symbols. It then produces ASCII code from the obtainable binary output. In the second phase Huffman coding will be applied to the first phase output in order to further compact the data and boost the efficiency of the dynamic bit reduction algorithm. Huffman coding follows a top-down approach which means the binary tree is built from top to bottom to achieve the optimum result. Through Huffman Coding, the characters in a data file are translated to binary code and the most common characters in the file have the shortest binary code, and the less common characters would have the longest binary code.

The decompression process functions similarly in reversed order. Huffman Decoder decodes compressed data first and then dynamic text decoder for compression. And next steps are to compress the data with the help of our proposed method.

2.3. Compression Algorithm

Compression algorithm has 8 steps. They are as follows:

- Enter the text data to be compressed.
- Locate the number of specific symbols in the text input.
- Assign the numeric code to special symbols found in step 2.
- Beginning from the first symbol in the input, find the corresponding binary code from the numerical codes assigned to that symbol and concatenates it to obtain the binary output.
- In binary output MSB add 0's until it is divisible by 8.
- Generate and concatenate the ASCII code for every 8 bits of the binary output obtained in step 5 to create input for second phase.
- Give the output produced by Step 6 to Huffman Tree to further compress the data and get the result in a compressed binary output form.
- The final result obtained in phase 7 is shown.

2.4. Algorithm to Create Huffman Tree

There are 7 steps to construct Huffman Tree in algorithm. They are as follows:

- Decode the input and count every symbol that occurs.
- Using the symbol count to determine the probability of occurrence for each symbol.
- Organize the symbols according to their probability of occurrence, putting the most likely first.
- For each symbol, build the leaf nodes, and then add them to the queue.
- Take two less frequently used characters and then logically group them together to get their combined frequency which leads to a binary tree.
- Repeat step 5 until all elements are reached and one parent remains to all nodes known as core.
- Then mark the edges of the parent with the digit 0 to the left child and the edge to the right child with 1. Tracing down the tree leads to "Huffman codes" in which the character is given the shortest higher frequency codes.

2.5. Decompression Algorithm

Decompression algorithm has 8 steps. They are as follows:

- Compressed phase input Final output.

- Assign this input to a Huffman decoder to decompress data compressed with Huffman tree in ASCII format.
- Calculate the binary code which corresponds to the ASCII values obtained in phase II.
- Remove extra bits from binary output inserted during the compression process.
- Measure number code for each 8 bits obtained in Phase IV.
- To get the final decompressed data, consider the corresponding symbol for each numerical value obtained in step V.
- Concatenate the data symbols obtained in step VI, and obtain the final result.
- Show user the end result.

2.6. Performance Evaluation Parameter

The performance evaluation of the proposed algorithm is carried out using two parameters-the compression ratio and the percentage saved. Ratio of compression: The ratio of compression is defined as the size of ratio of the compressed file to the source file size.

Ratio of Compression= C2 / C1

Where C1= Pre-Compression Scale

C2= the compression size after

- Save Percentage: Save Percentage calculates percentage shrinkage of source file.

Percentage savings = 100*((C1-C2)/C1) percent

Where C1= Pre-Compression Scale

C2= the size of compression after

3. DESIGN AND BLOCK DIAGRAM OF SYSTEM

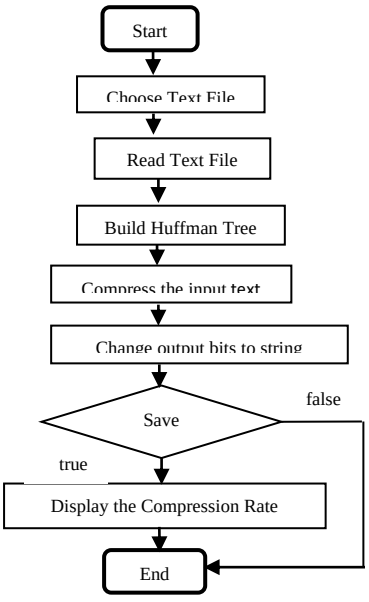

Figure 1. System Design for Sender

At the compression side, user can click compress button and that will open file open dialog and user can choose any text file. Then, Huffman Tree will be build and encoding is done using this Huffman Tree. The output will be bits originally. They are converted to base64 strings and saved in text file. After encoding and string conversion, file save dialog will be opened and output file can be saved in any place user want. The numbers of character in original file, the number of character in compressed file and compression rate are displayed.

At the decompression side, the same user operation is done. All the internal processes are in reverse direction of those of compression.

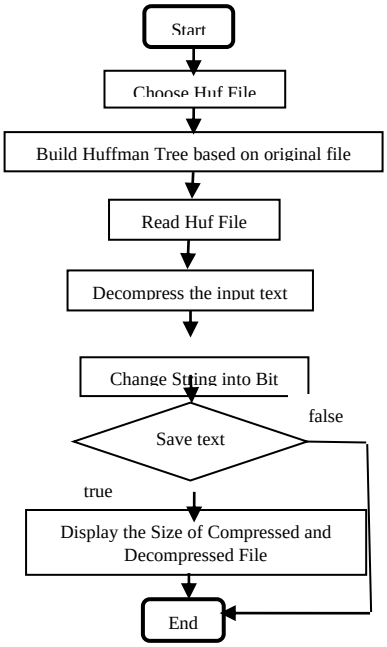

Figure 2. System Design for Receiver

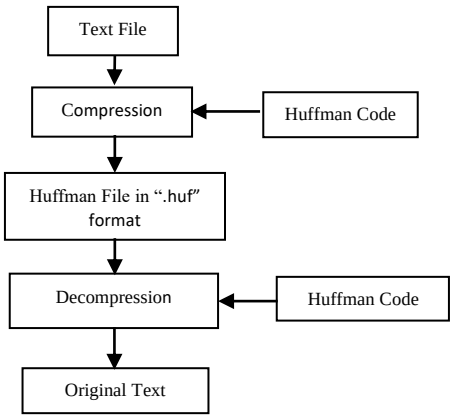

Figure3. Block Diagram of Huffman Text File

4. Implementation of the System

In this paper, the results of the executed program along with GUI window frame will be expressed. The program was written in the Microsoft windows visual C# studio.

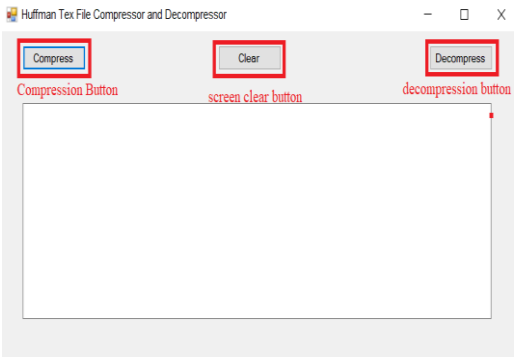

Figure 4. Showing the description of buttons

The compress button is entered when the user want to compress the file. The clear button is used when the user want to clear the window screen.The decompress button is entered when the user want to decompress the file.

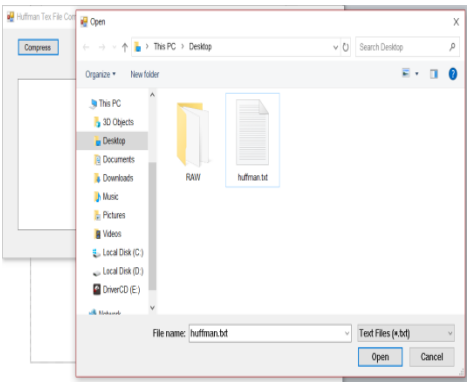

Figure 5. Choosing the file to compress

This file choosing window is opened when the user enters the compress button. The file that can be chose is only text file in the format of .txt. Before choosing the file to compress, it is needed to create the txt file.

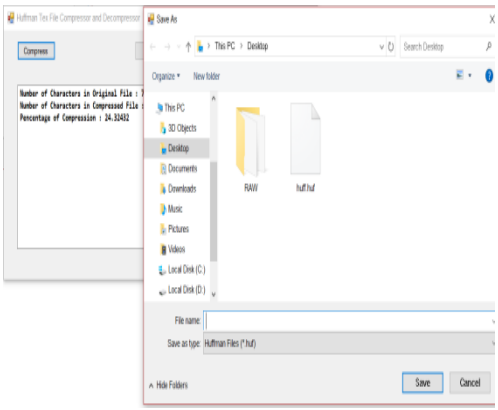

Figure 6. Compressed file saving

After the compression of the file, the system will ask the compressed file saving path and its name. The compressed file is in the format of .huf. In window screen, the number of characters in original file, the number of characters in compressed file and percentage of compression are shown in program window.

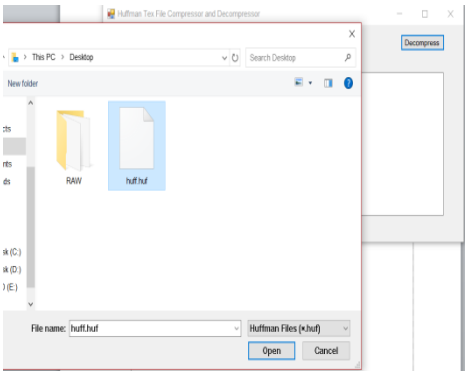

Figure 7. File Decompression

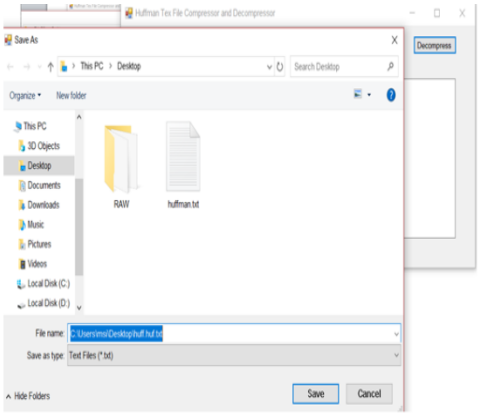

Figure 8. Decompressed file saving

When the user clicked on the decompress button, the file choosing window will also open and the user needed to choose the file in the format of .huf. After the decompression, the system will ask the path where the file is used to save.

5. Conclusion

An advanced dynamic text compression algorithm is described in this system to compress and decompress the text data based on lossless approach to data compression. Specific tests were performed on various datasets, such as the Random, Alphanumeric, Numeral and Special Character datasets. From the results analysis it is concluded that the proposed method shows very good compression performance in terms of compression ratio and percentage savings compared to current techniques for all the data sets that were considered. Advanced Dynamic Text Compression Algorithm only deals for single-language text data that can also be evaluated for multi-lingual data compression, i.e. text data written in multiple languages in multiple languages.

Acknowledgements

The authors wish to express their gratitude to the University Journal of Created and InnovativeResearch (UJCIR_2020) committee for accepting this paper.The author is very thankful to the Editorial Board for editing this paper and effective true guidelines to publish this paper.

References

[1]M. N. Huda,"Study on Huffman Coding," Graduate Thesis, (2004).
[2] S.R. Kodituwakku and U. S. Amarasinghe, "Comparison of Lossless Data Compression Algorithms for Text Data", *Indian Journal of Computer Science and Engineering*, vol. I (4), (2007),pp. 416-426.
[3]David Salomon, "Data Compression Fourth edition", (Dec, 2006)
[4]Mark Neison and Jean, "Data Compression", second edition, M&T books, (1996)
[5]Elsevier, K. Sayood,"Introduction to Data Compression," 4th ed., (2012).
[6]M. R. Hasan, M. I. Ibrahimy, S. M. A. Motakabber, M. M. Ferdaus and M. N. H. Khan, "Comparative data compression techniques and multi-compression results," IOP Conference, (2013).
[7]C. Lamorahan, B. Pinontoan and N. Nainggolan, "Data Compression Using Shannon-FanoAlgorithm," JdC, Vol. 2, No. 2, 2013 September, pp. 10-17.
[8]John sharp, "Microsoft Visual C# Step by Step, eight editions",(2010).
[9] M. A. Khan, "Evaluation of Basic Data Compression Algorithms in a Distributed Environment," *Journal of Basic & Applied Sciences*, Vol. 8,(2012), pp. 362-365.
[10] https://www.techiedelight.com/Huffman coding
[11] http://en.wikipedia.org/wiki/huffman coding