International Journal of Applied Sciences & Engineering (IJASE) **2**(1) : April 2014, 63-82

DOI No. 10.5958/2322-0465.2014.01118.6

A Comparative Analysis of Data Compression Techniques

Boukari Souley[1*], Prodipto Das[2] and Shirley Tanko[3]

[1,3]Mathematical Sciences Programme, Abubakar Tafawa Balewa University (ATBU), Bauchi, Nigeria.
[2]Deptt of Computer Science, Assam University, Assam, India.

*Corresponding author: Boukari Souley, bsouley2001@yahoo.com

ABSTRACT

Data compression is the process that is used to reduce the physical size of a block of information; data compression encodes information using fewer bits to help in reducing the consumption of expensive resources such as disk space or transmission bandwidth. The task of compression consist of two components, an encoding algorithm that takes a message and generates a compressed representation (hopefully with fewer bits), and a decoding algorithm that reconstructs the original message or some approximation of it from the compressed representation. Data Compression is divided into two (2) broad categories namely Lossless compression and lossy algorithms. This paper examined these compression techniques and provided a comparative analysis of three commonly used compression techniques (the Huffman, Lempel-Ziv and RunLength Encoding). The results revealed that compression algorithms can be proven to be more effective on notepad, text, web documents, PDF, Images and sound.

Keywords: Compression, Algorithms, Image and sound

Data compression is the process of encoding information using fewer bits than the original representation will use; it is the process that is used to reduce the physical size of information. Compression is just about everywhere, all images that can be gotten from the web are compressed (Paulus, 2002).

Data Compression is particularly useful in communication because it enables devices to transmit or store the same amount of data in fewer bits. Data Compression is also widely used in File Storage and Distributed Systems, Backup utilities, Spreadsheet applications and Database Management Systems. There are a variety of data compression techniques, but only a few have been standardized. Certain types of data, such as bit-mapped graphics, can be compressed to just a small fraction of their normal size. Other synonyms of Data compression are Source Coding and Bit Rate Reduction. It can also be viewed as a

branch of Information Systems and it is often referred to as Coding in a general term encompassing any special representation of data which satisfies a given need (http://en.wikipedia.org/wiki/)

The concept of Data Compression helps to reduce the consumption of expensive resources such as hard disk space and transmission bandwidth. Although, on the downside, compressed data must be decompressed to be used and this extra processing can be detrimental to some applications. The design of data compression schemes therefore involves tradeoffs among various factors such as the degree of compression, the amount of distortion introduced (in the case of a lossy compression schemes) and the computational resources required to compress and decompress the data, as the case may be. Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent sent data more concisely without error.

The technology behind data compression is to represent an information or data (e.g a data file, a speech signal, an image, or a video signal), as accurately as possible and using the fewest number of bits possible.

Problem

The use of lossless data compression can bring about a number of added advantages in the use of information and communications technology to this era, and also electronic systems. A comparative analysis of Data Compression Techniques helps to figure out how the different compression techniques have an edge over one another.

Goals

The goal of the study is to make comparisons of the lossless data compression techniques so as to establish the following:

- The degree (ratio) of compression that can be achieved with each of the algorithms
- The level of efficiency of the different algorithms
- Comparing the techniques based on the kind of data they are designed to compress

This work seeks to reduce redundancy in stored and communicated data, thereby increasing effective data density. It invariably reduces the number of bits used to store or transmit information; hence it has important application in the area of file storage and distributed systems, since it takes a stream of symbols and transforms them into codes. If the compression is effective, the resulting stream of codes will be smaller than the original symbols.

This work is justified as there is a great need to have an improved and optimal data compression program that will make the use of computers more effective in maximizing memory space and data transmission bandwidth.

Related Woks

Importance of Data Compression

Data Compression seeks to reduce the number of bits used to store or transmit information in a frame. Compression is a way to reduce then the physical size of data but retaining its meaning. It encompasses a wide variety of software and hardware resources. Compression techniques, which can be unlike one another, have little in common except that they compress information bits. The technique is to identify redundancy and to eliminate it.

The advent of data compression was motivated when the need to maximize computer memory capacity came up. From then, it became a progressive research with each development greater than the one preceding it, in trying to have an optimal data compression program. The concept is often referred to as coding, where coding is a very general term encompassing any special representation of data which satisfies a given need. Information theory is defined as the study of efficient coding and its consequences, in the form of speed of transmission and probability of error. Data compression may be viewed as a branch of information theory in which the primary objective is to minimize the amount of data to be transmitted. The technique is widely used in a variety of programming contexts. All popular operating systems and programming languages have numerous tools and libraries for dealing with data compression of various sorts (http://en.wikipedia.org/wiki/)

Data Compression is a kind of Data encoding that is used to reduce the size of data. Other forms of data encoding are Encryption (Cryptography: coding for purposes of data security and guarantee a certain level of data integrity, and error detection/correction), and Data Transmission.

A simple characterization of data compression is that it involves transforming a string of characters in some representation (such as ASCII) into a new string (of bits, for example) which contains the same information but whose length is as small as possible. Data compression has important application in the areas of data transmission and data storage. Many data processing applications require storage of large volumes of data, and the number of such applications is constantly increasing as the use of computers extends to new disciplines. At the same time, the proliferation of computer communication networks is resulting in massive transfer of data over communication links. Compressing data to be stored or transmitted reduces storage and/or communication costs. When the amount of data to be transmitted is reduced, the effect is that of increasing the capacity of the communication channel. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium. It may then become feasible to store the data at a higher, thus faster, level of the storage hierarchy and reduce the load on the input/output channels of the computer system (Sydeepta, 1999).

Data Compression Techniques

Data compression techniques are the skills, which can be applied to compress files. They are broadly divided into two (2) categories: Lossy compression and Lossless compression

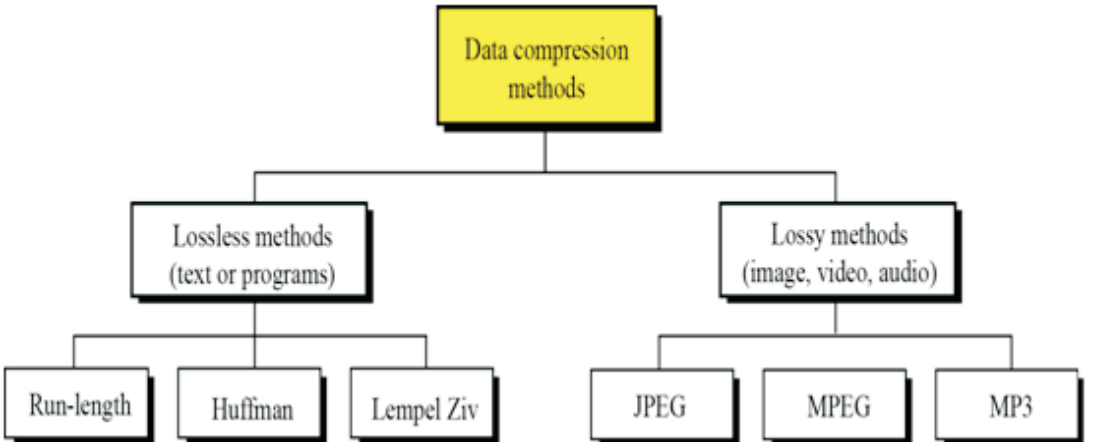


Figure 1. Data Compression Methods

In lossless data compression, the integrity of the data is preserved. The original data and the data after compression and decompression are exactly the same because, for the methods under this subcategory, the compression and decompression algorithms are exact inverses of each other: no part of the data is lost in the process (Salomon, 2004).

In lossy data compression or perceptual coding, the loss of some fidelity is acceptable. The Lossy technique is a data compression method which compresses data by discarding (losing) some of it. The procedure aims to minimize the amount of data that needs to be handled, and/or transmitted by a computer.

Data Compression Algorithms

The Huffman Compression Technique

The Huffman coding technique assigns shorter codes to symbols that occur more frequently and longer codes to those that occur less frequently. For example, if there is a text file that uses only five characters (A, B, C, D, E). Before we can assign bit patterns to each character, we assign each character a weight based on its frequency of use. In this example, assume that the frequency of the characters is as shown below.

Character	A	B	C	D	E
Frequency	17	12	12	27	32

Figure 2 : Frequency of characters

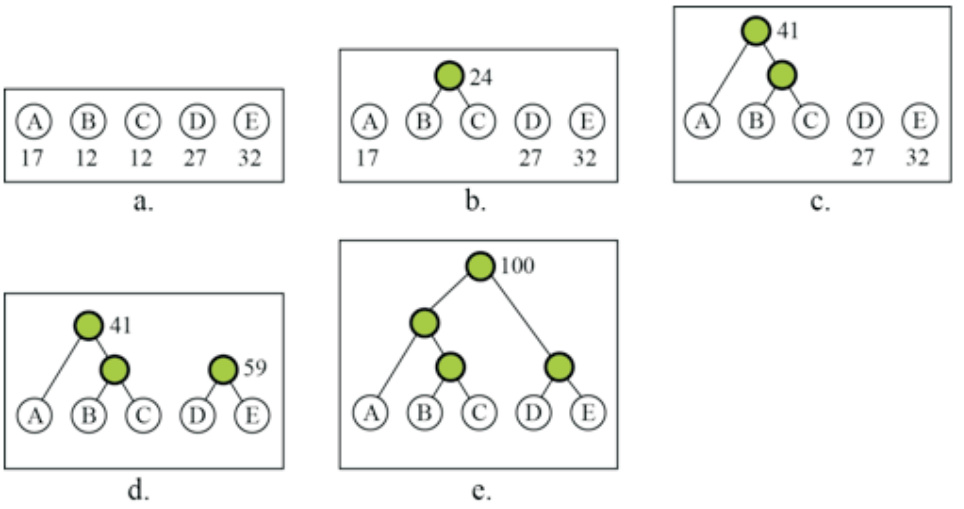


Figure 3 : Huffman Coding

A character's code is found by starting at the root and following the branches that lead to that character. The code itself is the bit value of each branch on the path, taken in sequence.

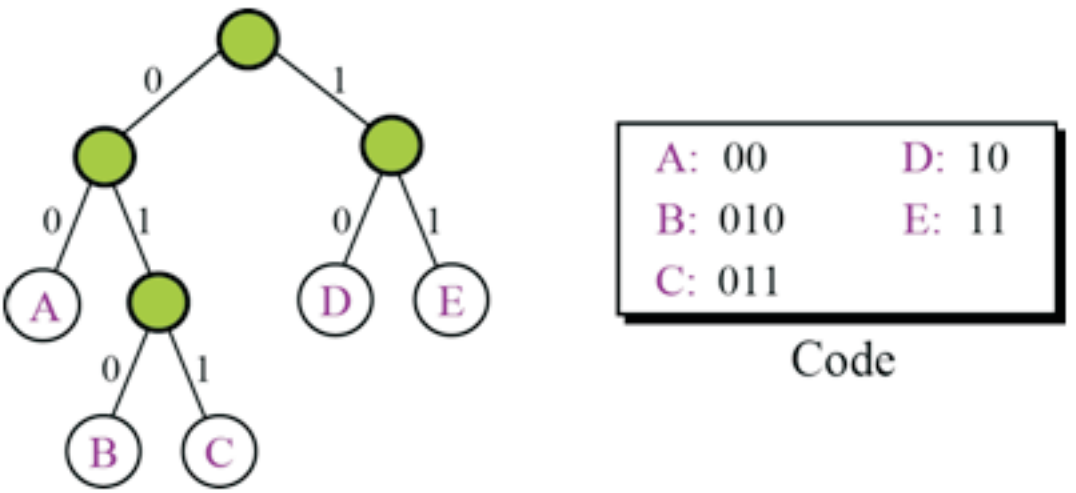


Figure 4 . Reduced Huffman Code

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes", that is, the bit string representing some particular

symbol is never a prefix of the bit string representing any other symbol) that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols. For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding, e.g., ASCII coding.

The Run-Length Compression Technique

Run-length encoding is probably the simplest method of compression. It can be used to compress data made of any combination of symbols. It does not need to know the frequency of occurrence of symbols and can be very efficient if data is represented as 0s and 1s (Dipperstain, 1998).

The general idea behind this method is to replace consecutive repeating occurrences of a symbol by one occurrence of the symbol followed by the number of occurrences. The method can be even more efficient if the data uses only two symbols (for example 0 and 1) in its bit pattern and one symbol is more frequent than the other.

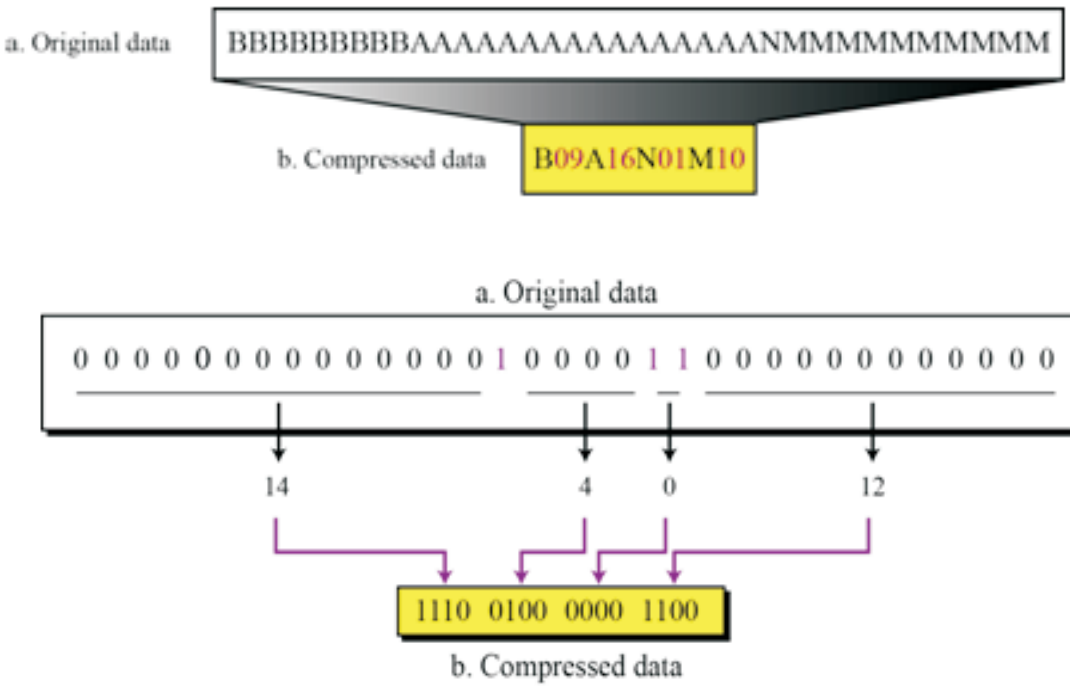


Figure 5: Run-length Encoding for two Symbols

The Lempel-Ziv Compression Technique

The Lempel-Ziv (LZ) compression methods are among the most popular algorithms for lossless The code that the LZ algorithm outputs can be of any arbitrary length, but it must have more bits in it than a single character. The first 256 codes (when using eight bit characters) are by default assigned to the standard character set. The remaining codes are assigned to strings as the algorithm proceeds (McNaughton, 2001).

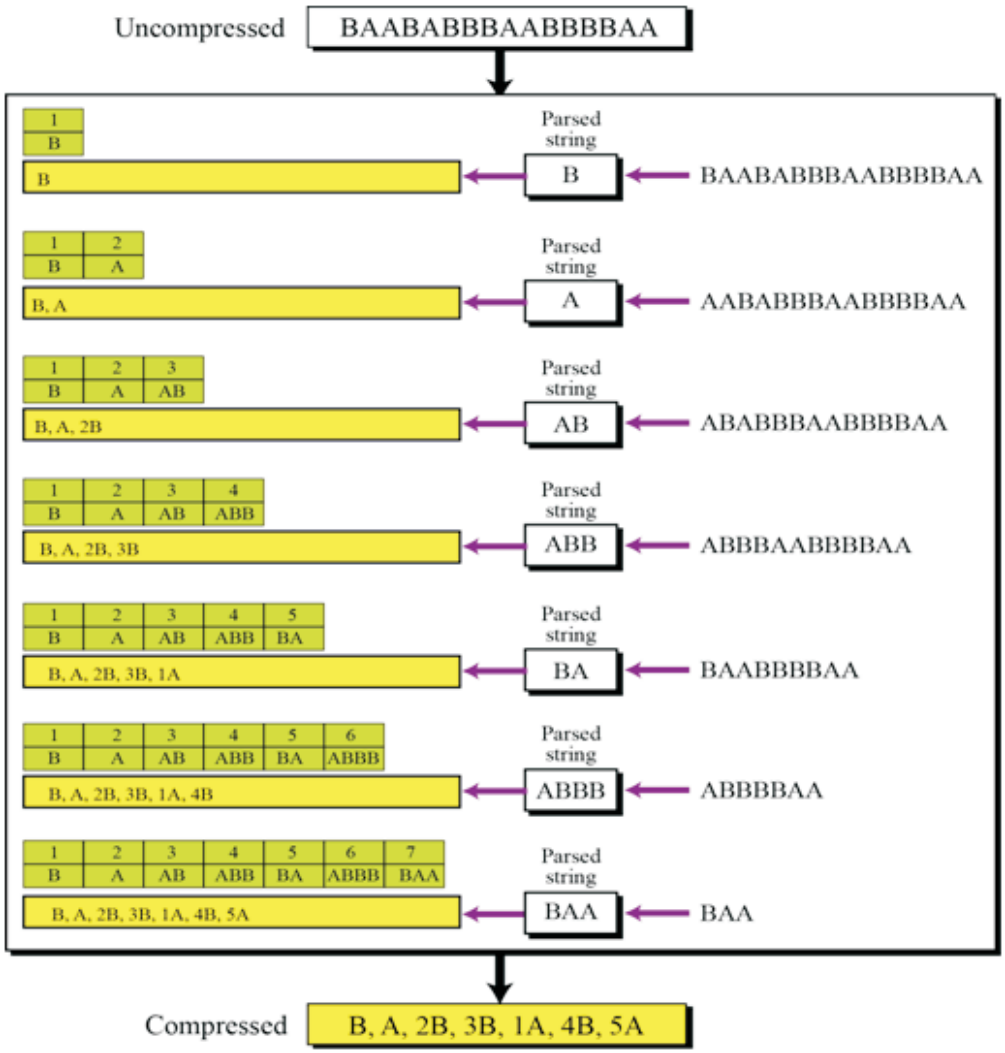


Figure 6. Lempel-Ziv Encoding (Source: Belloch ,2010)

LZ encoding is an example of a category of algorithms called dictionary-based encoding. The idea is to create a dictionary (a table) of strings used during the communication session. If both the sender and the receiver have a copy of the dictionary, then previously-encountered strings can be substituted by their index in the dictionary to reduce the amount of information transmitted. There are two concurrent events: building an indexed dictionary and compressing a string of symbols. The algorithm extracts the smallest substring that cannot be found in the dictionary from the remaining uncompressed string. It then stores a copy of this substring in the dictionary as a new entry and assigns it an index value. Compression occurs when the substring, except for the last character, is replaced with the index found in the dictionary. The process then inserts the index and the last character of the substring into the compressed string.

Methodology

Architectural Design

System design is defined as those tasks that focus on the specifications of a detailed computer based solution; it is also called physical design. In other words system design is the construction of a technical /computer based solution for the business requirements identified during the system analysis phase. Whereas system analysis emphasizes on business problems, system design focuses on implementation of concern of the system. It addresses the Data, Process and Interface building blocks from the system designer's perspective. The system analysis facilitates the design.

Input Design

Input design deals with the specification of data; that is, the format that is used to capture the data, and the data types of different variables.

The inputs in this work are divided into two (2) categories which are:

- The compression input
- The decompression input

1. **The compression input:** The main compression form is designed such that a particular compression technique (that is either the Huffman, Run-length or Lempel-Ziv encoding), is checked from the three options available. Then an input file path is specified as in the figure 7 below:

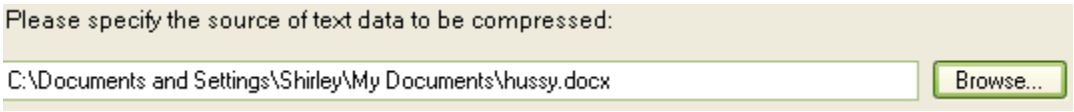


Figure 7. Sample Compression Input

The compression process can then begin, after clicking the compression button that is at the bottom left hand side of the main form. There is a progress bar on the main form that displays the progress of the compression after the 'view compression' has been clicked, as shown in the figure 8 below:

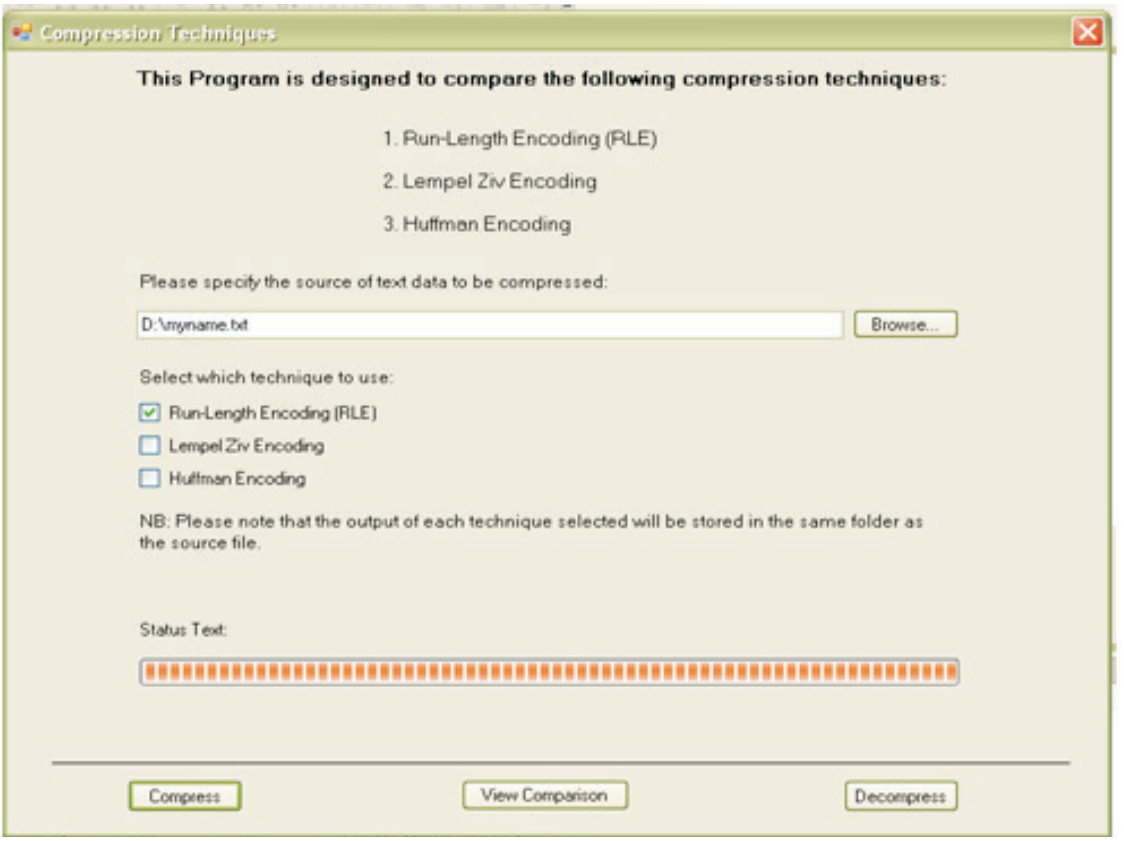


Figure 8. Main Compression Form

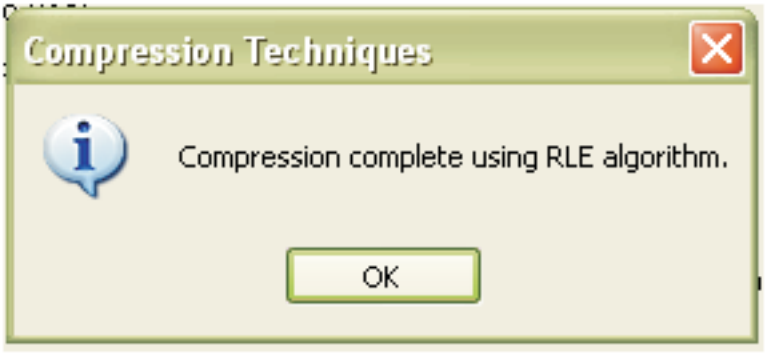


Figure 9. Compression Completion Confirmation

The figure 9 below is a window that pops up to display a message which confirms that the compression has been completed.

2. **The decompression input:** The decompression process has been designed such that when an already compressed file is located and chosen to be decompressed, the compression technique that was used to compress it is automatically checked to be used to perform the decompression. In the figure below, the run-length encoding technique has been used to compress the file that is now being decompressed.

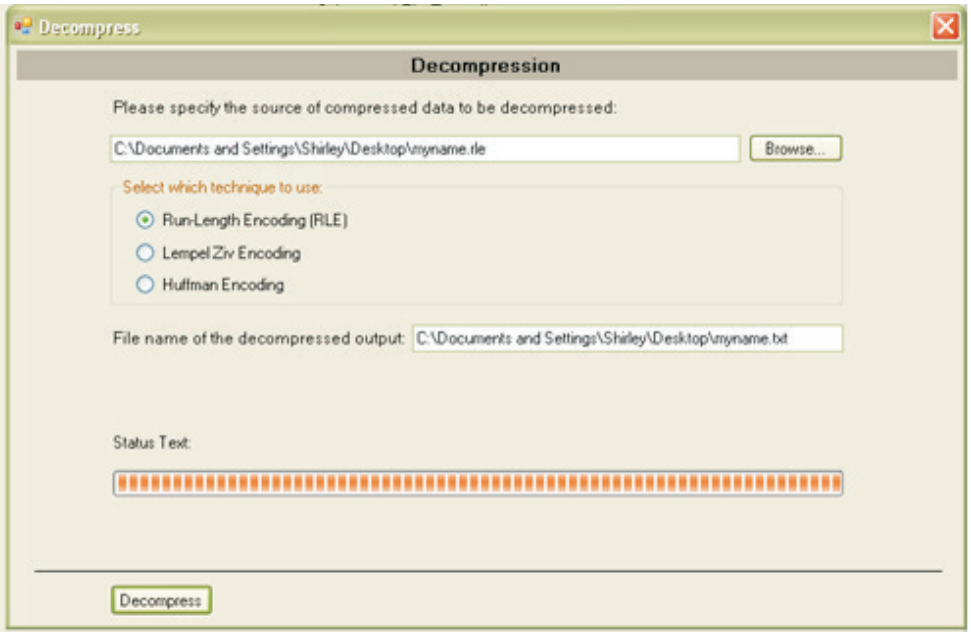


Figure 10. Decompression Form

The figure 10 below shows how a message automatically comes on to confirm that the decompression has been successfully completed.

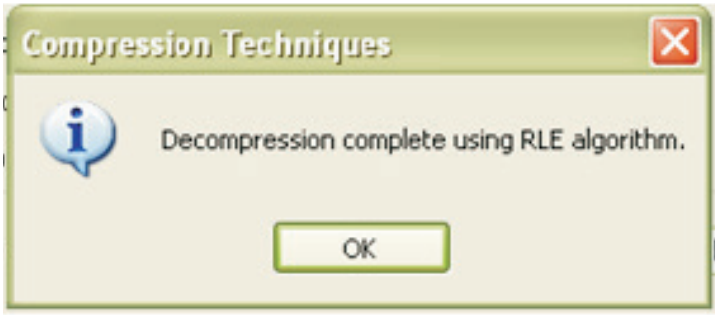


Figure 11. Decompression Completion Confirmation

Output Design

The output design displays the results obtained after series of computational processes. The result generated from a system gives the reliability of that system.

In this work, the output file is usually a compressed version of the source file. The output file is stored where the source file is stored, but with an extension that clearly shows the compression technique that has been used to compress it

'.hff': for files that have been compressed using the Huffman encoding.

'.rle': for files that have been compressed using the Runlength encoding.

'.lzw': for files that have been compressed using the Lempel-Ziv encoding.

The output file (which will be the compressed version of the input file or the decompressed version of an already compressed file) using any of the three (3) techniques is the output that is produced as a result. For the compressed output, the output file cannot be opened in any form using any application, it can only be stored or transferred in its compressed form and then it has to be decompressed when it needs to be opened and used. The decompressed file gives back the initial input file with its original size, since the compression and decompression algorithms are inverses of one another. The figure 12 below shows how the output file is automatically saved in the same source with the input file.

(a) (b)

Fig. 12. Input Text File (a), Compressed File (b)

Program Tester Design

A detailed study and comparison of these lossless data compression techniques (that is the Run-length, Huffman and Lempel-Ziv encoding techniques), was done using some criteria. The program design processes wee given in figures 13 and 14.

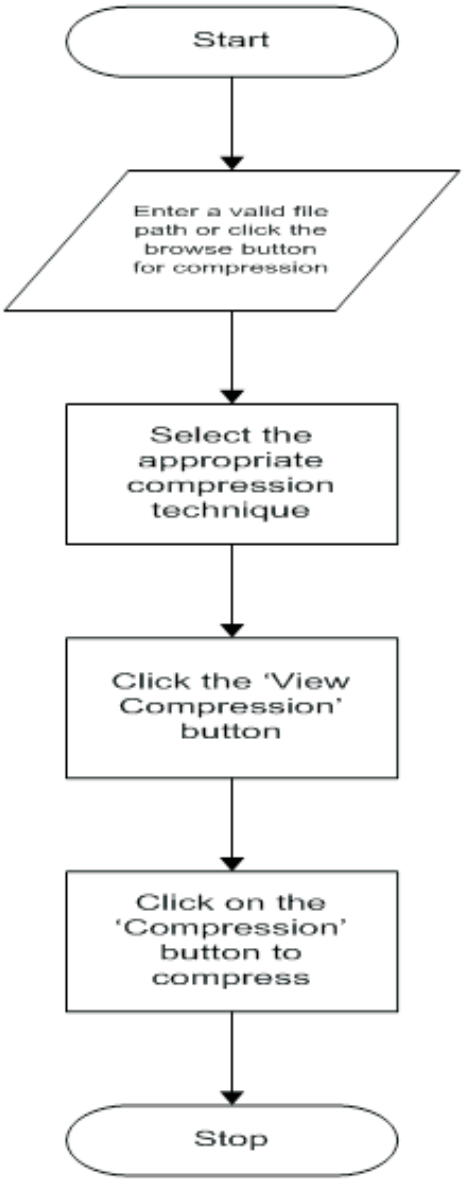


Figure 13. Compression Process

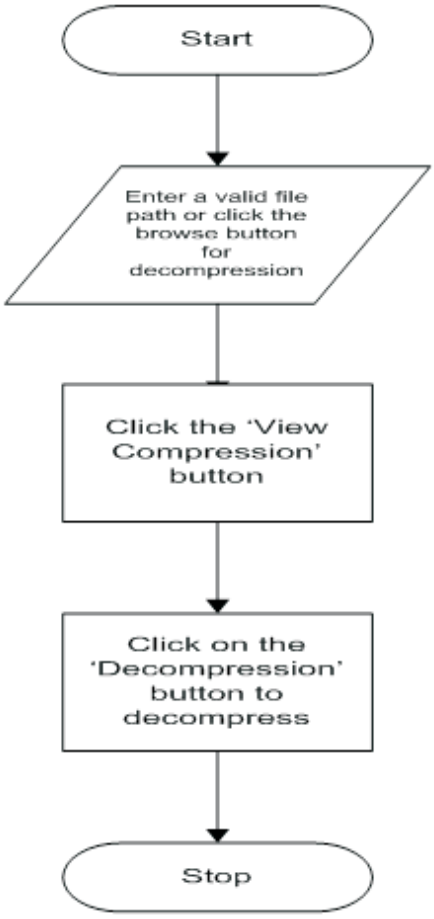


Figure 14. Decompression Process

Testing and Results

The program was developed using C#. Testing of the system was done as a correctness testing. The data that has been used for the testing of the three (3) compression techniques that are being compared are:

- A 65 kilobytes textual document on notepad.
- A 193 kilobytes PDF document.
- A 12.7 kilobytes Microsoft word document.
- A 108 kilobytes Web document
- A 1.16 Megabytes image document

Results and Discussion

After running the tester using the above data sets, the results obtained were displayed in figures 20, 21, Tables 1 and 2 of Appendix A

The results of the compression displayed show the degree (ratio) of compression of the different techniques on the sample documents that have been used, and the technique(s) that are most appropriate for each kind of data.

In the Notepad compression displayed in figure 15, it implies that the compression rate of the Lempel-Ziv-Welch is much more higher and because it compresses 65KB to 383Bytes. The next is the Run-length encoding which compresses the text document to 1.99 KB. The least in this format is the Huffman which compresses the text document 29.8 KB.

In Figure 16, a PDF document that is 193 Kilobytes is compressed using the Lempel-Ziv-Welch technique to 173 Kilobytes. The Run-length and Huffman techniques have been proven to be ineffective in compressing PDF document

Figure 17 shows that the Run-length technique is much more effective in compressing Microsoft Office documents (though not so good), the Huffman and Lempel-Ziv-Welch techniques do not compress files in this format at all. The Run-length encoding technique has been found to have a poor compression strength on web documents compared to the other two (2) methods.

The web document that Figure 18 illustrates shows that the Lempel-Ziv is the best for compressing web documents which reduces the 108 KB document to 25.3 KB. Next is the Huffman encoding technique which also compresses the web document to 69.3 KB and followed by the Run-length which only reduces it by 8 KB less than its original size.

The last is Figure 19 which is used to try a sample Corel Draw image document. The three techniques could not compress the image at all, this confirms the fact that the Lossless encoding algorithms are used only for text data and not other forms like audio, video and images.

Figures 20 and 21 are bar charts that give clearer pictorial representations that will help to make the comparisons easier to understand. The three (3) techniques have both their compression and decompression bar charts as shown in figures 20 and 21 of Appendix A.

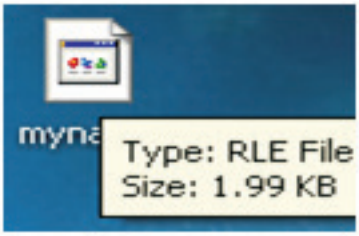
Conclusion

Computers can be used with ease and maximum accuracy and the speed of performing computation cannot be equated to the manual way. It can be concluded that if this application is fully implemented, it will help in bringing solution to compression of data by using appropriate techniques to compress the kinds of data that suit them most. This could help in effectively handling data in terms of data transfer and storage, by reducing the consumption of expensive resources, such as storage space and transmission bandwidth. Therefore the Lempel-Ziv technique has been proven to be most effective, and works better for Notepad and web documents. A data Compression and decompression tool was developed to carry out the comparison analysis. This system is strongly recommended in storage and transfer of large volume of data

Appendix A

The figures 15 to 19 below show the results of the compression of the three (3) documents using the three techniques respectively. The first image (a) of each figure is the original uncompressed sample data.

(a)



(b)

(c)

(d)

Figure 15 Comparison using a sample notepad text document

(a.)

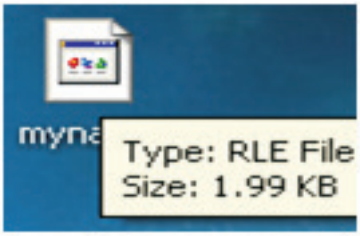
(b.)

(c.)

(d.)

Figure 16. Comparison using a sample PDF document

(a)



(b) (c) (d)

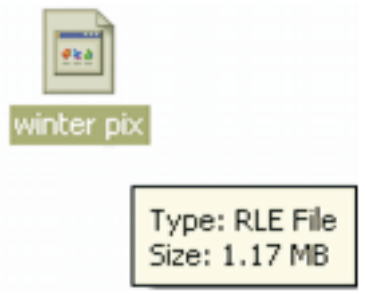
Figure 17: Comparison using a sample Microsoft word document

(a)

(b) (c) (d)

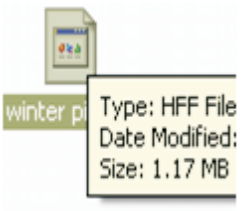
Figure 18. Comparison using a sample web document

(a)



(b)

(c)



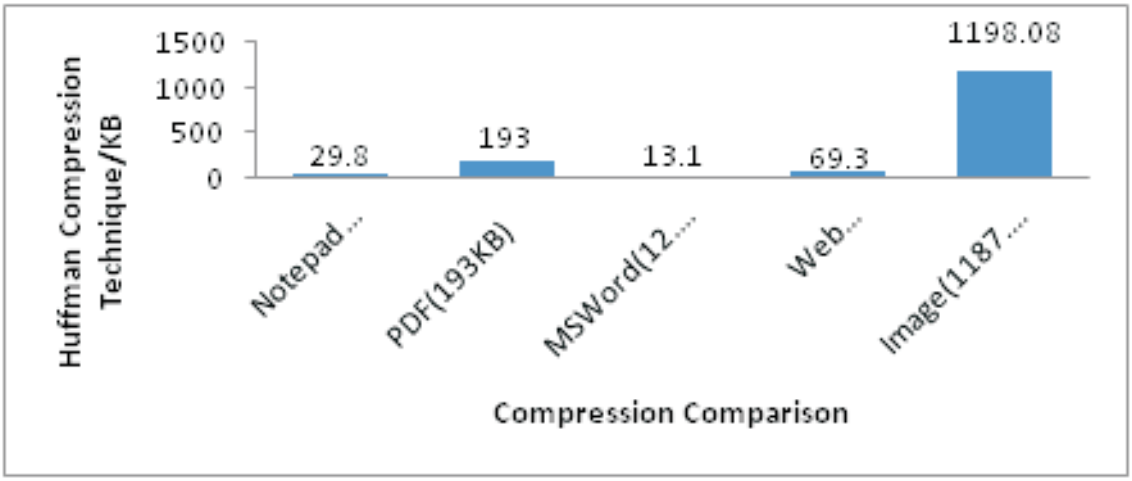
(d)

Figure 19. Comparison using a sample image document

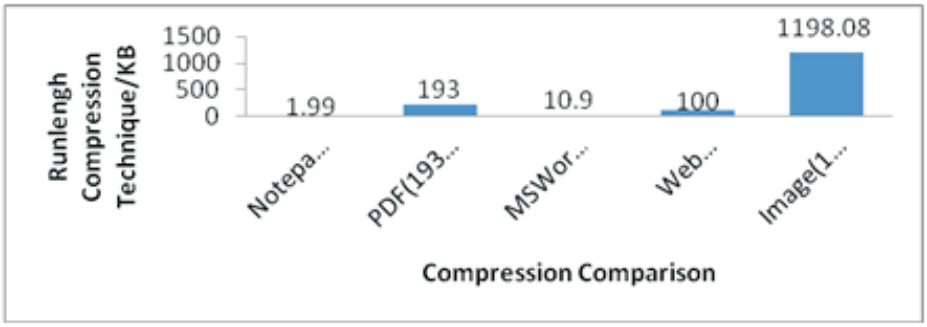
Below is a comparison table to show the different compression ratios of the three techniques.

Table 1. **Compression Comparison Table**

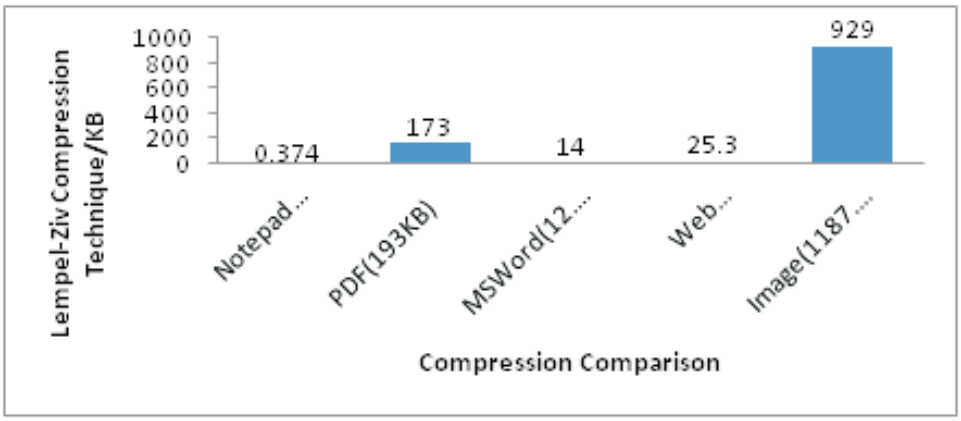
Compression Technique Document and Size	Huffman Compression Technique	Runlength Compresssion Techniques	Lempel-Ziv Compression Technique
Notepad Text (65KB)	29.8KB	1.99KB	383Bytes
PDF (193KB)	193KB	193KB	173KB
MSWord (12.7KB)	13.1 KB	10.9 KB	14.0KB
Web Document (108KB)	69.3KB	100KB	25.3KB
Images (1.16MB)	1.17MB	1.17MB	929KB



(a)

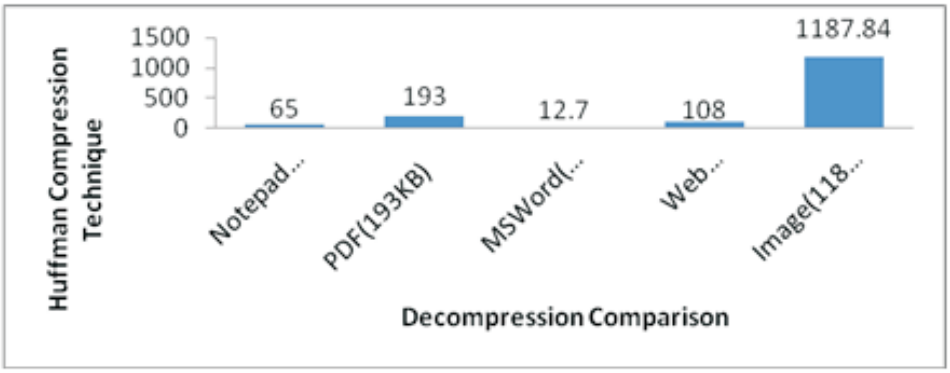


(b)

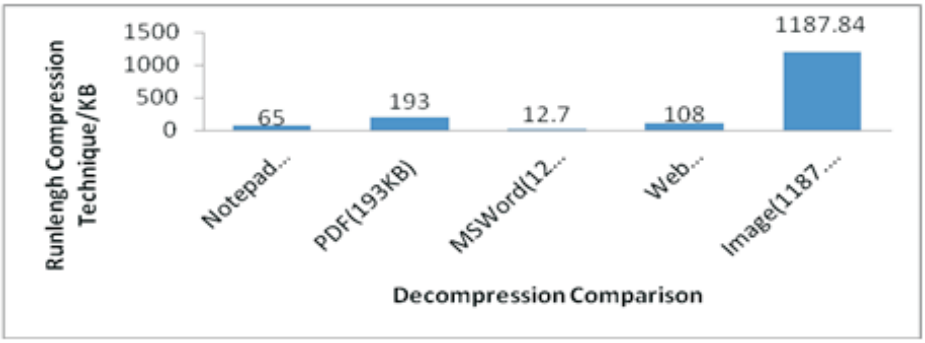


(c)

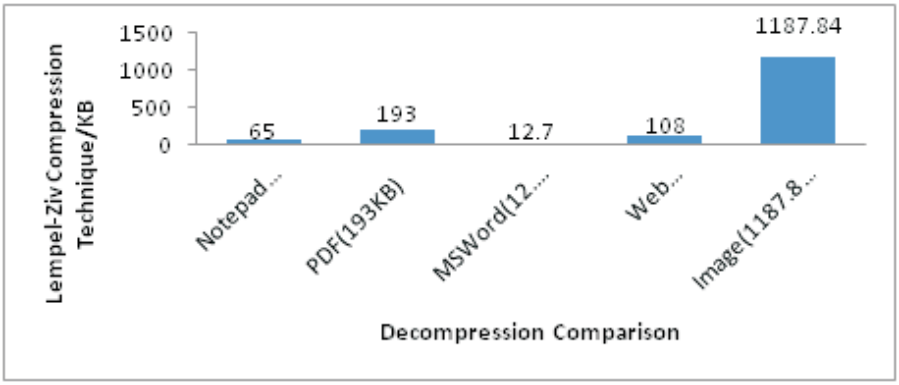
Figure 20. Compression Comparison Graphs



(a)



(b)



(c.)

Figure 21. Decompression Comparison Graphs

Table 2. Decompression Comparison Table

Decompression Technique Document and Size	Huffman Compression Technique	Runlength Compression Technique	Lempel-Ziv Compression Technique
Notepad Text (65KB)	65KB	65KB	65KB
PDF (193KB)	193KB	193KB	193KB
Microsoft Word (12.7KB)	12.7KB	12.7KB	12.7KB
Web Document (108KB)	108KB	108KB	108KB
Images (1.16MB)	1.16MB	1.16MB	1.16MB

References

Data Compression English Wikipedia, 2011, Web. 07 March 2011. <http://en.wikipedia.org/wiki/Data_compression>

Belloch G. E., 2010, Introduction to Data Compression. Mosby Publishing Inc PP 96-107.

Data Compression Foundations of Computer Science. Web. 07 March 2011. <ttp://www.csie.kuas.edu.tw/course/CS/old/english/ch-15.pp>

Paulus A.J.V. 2002, Weighting Techniques in Data Compression: Theory and Algorithms.

Salomon D. 2004, Data Compression: The Complete Reference, (3rd Edition) Northridge, California.

Dipperstain M. 1998, RunLength Encoding (RLE): Discussion and Implementation.

McNaughton J. 2001, Data Compression Theory and Algorithms. **4**: 263-286.

Sydeepta M. 1999, A Review of Data Compression Techniques. Williams & Wilkins Publishers.