Synopsis Report

On

Zipper- A File Compression Model

Submitted in partial fulfillment of the Requirement for the Degree of

Bachelor of Technology
In
Computer Science and Engineering

Submitted BY

Riya Chauhan
1/19/FET/BCS/006

Yash Sharma
1/19/FET/BCS/061

Under the supervision of
Rachna Behl

Department of Computer Science and Engineering
Faculty of Engineering & Technology
Manav Rachna International Institute of Research and Studies, Faridabad
November 2022

INDEX

Abstract

The basic requirement of the Text compression mechanism is that the combination of compression and decompression mechanisms be lossless otherwise data cannot be restored in the original format. The data compression by text substitution mechanism is used by the scheme of data compression mechanisms and includes changes among various factors, like degree of compression, and amount of distortion introduced; for lossy compression algorithm and the computational resources required to compress and uncompress data.
Safe Transaction of data can be considered paramount these days, Individuals with malicious intent can catch onto the data being transferred and use it against the entity for whom the information is predominant. So to prevent that from happening we have many methodologies available.
Compression is useful because it reduces the resources required to store and transmit data. Computational resources are consumed in the compression and decompression processes. Data compression is subject to a space-time complexity trade-off.
In this report, we discuss the project Zipper, a file compression model that uses the Huffman encoding compression algorithm to achieve the desired goal.

I. Introduction

Text files can be compressed to make them smaller and faster to send, and unzipping files on devices has a low overhead. The process of encoding involves changing the representation of a file so that the compressed output takes less space to transmit while retaining the ability to reconstruct the original file exactly from its compressed representation.

The size of the text file can be reduced by compressing it, which converts the text to a smaller format that takes up less space. It typically works by locating similar strings/characters within a text file and replacing them with a temporary binary representation to reduce the overall size. There are two types of file compression, namely Lossy Compression and Lossless compression.

Lossy compression shrinks a file by permanently removing certain elements, particularly redundant elements.

Lossless compression can restore all elements of a file during decompression without sacrificing data and quality.

The main goal of this work is to build a file compression model using lossless compression using the Huffman coding algorithm.

Huffman coding is a lossless data compression algorithm, a technique of compressing data to reduce the size of the data without any loss in the details present. Huffman coding was first developed by David Huffman.

Huffman coding is generally useful to compress the data in which there are frequently occurring characters.

I.1. Challenges

While there are not many drawbacks to file compression but there is one significant downside: some data can be lost during the compression of the file. Also, the available softwares are open-sourced and can be used by anyone which doesn’t ensure the confidentiality of the information. This motivates us to design a software that can compress the file without losing any data along with its privacy.

I.2. Objectives

- The objective behind the project is to create a model that enables the user to transfer files in a secure manner using the Huffman encoding algorithm. It is the most famous algorithm to compress text.
- To provide safe transmission of files between individuals and enterprises acquiring the software only

II. Literature Review

Data compression is the process of reducing data's original representation to smaller bits so that it uses less storage space and transmits more quickly over networks [2]. David Huffman, an MIT undergraduate, developed the Huffman algorithm in 1952. It is one of the first and most well-known techniques for text compression [5]. The Huffman code operates on similar principles as the Morse code. Only a small number of bits are used to encode each character, with shorter bits used for characters that appear frequently and longer bits used for characters that appear less frequently. The code is used to convert the initial message (the contents of the input data) into a collection of codewords depending on the type of map. Static approaches are used with the Huffman algorithm. A static method is one that consistently employs the same code map, however, the order in which characters appear can be altered. There are two steps in this process. The first stage is to determine the map code and the frequency of occurrence of each symbol. The final step is to turn the message into a group of code that will be broadcast. Huffman, meanwhile, employs the symbol-wise method based on symbol coding. A technique called "symbol wise" determines how frequently each character appears in each operation. It is difficult to convert text characters into symbols [3]. The sign that appears more frequently will have a shorter code than the symbol that does not appear as frequently.

After encoding characters using just a regular binary tree, David Huffman discovered that the best prefix code may be established by employing a greedy method. When choosing the two trees in a Huffman tree with the lowest frequency, the greedy algorithm is applied. The overall cost is minimized by employing a greedy approach. Cost is used to join two trees at the root of frequency, where the frequency is equivalent to the number of combined fruit trees. As a result, the overall cost of creating the Huffman tree equals the whole cost of the merger [4]. Therefore, the Huffman algorithm is one example of a compression algorithm that uses the greedy algorithm. Our goal is to calculate the total cost incurred to establish the text.

III. Detailed description of Project

Huffman coding is the lossless data compression algorithm so that no data is lost whilst using this method. Huffman algorithm works by assigning variable length codes to each character. The most frequent character gets the smallest code and the least frequent character gets the largest code.

Variable length codes assigned to input characters are named prefix codes. And according to the prefix rule, no binary code should be a prefix of another code. This is how Huffman coding ensures there would be no ambiguity when decoding the generated bit stream.

Huffman coding follows a greedy approach since it deals with generating minimum-length prefix-free binary codes.

A Greedy Algorithm solves a problem by selecting the best distance at a particular time. The greedy algorithm follows the problem-solving metaheuristic of making the optimum choice. By calculating each step, the optimal solution is met.

The software reads three arguments from the user and if failed or not found, a message is displayed that “Failed to detect files”, else the program performs the encoding of the file. The text file is read by the compiler and the Huffman tree is generated using this generated code the text file is converted into a binary format which helps in reducing its size and retaining confidential information. Similarly, in decoding the ‘.huf’ file, the program converts the binary file to a text file using the saved codes for each character.

Steps to build Huffman Tree:

1. The input to the algorithm is the array of characters in the text file.
2. The frequency of occurrences of each character in the file is calculated.
3. A struct array is created where each element includes the character along with its frequencies. They are stored in a priority queue (min-heap), where the elements are compared using their frequencies.
4. To build the Huffman tree, two elements with minimum frequency are extracted from the min-heap.
5. The two nodes are added to the tree as left and right children to a new root node which contains a frequency equal to the sum of two frequencies. A lower frequency character is added to the left child node and the higher frequency character to the right child node.
6. The root node is then again added back to the priority queue.
7. Repeat step 4 until there is only one element left in the priority queue.

8. Finally, the tree's left and right edges are numbered 0 and 1, respectively. The entire tree is traversed for each leaf node, and the corresponding 1 and 0 are appended to their code until a leaf node is encountered.
9. Once we have the unique codes for each unique character in the text, we can replace the text characters with their codes. These codes will be stored in bit-by-bit form, which will take up less space than text.

Flow Chart

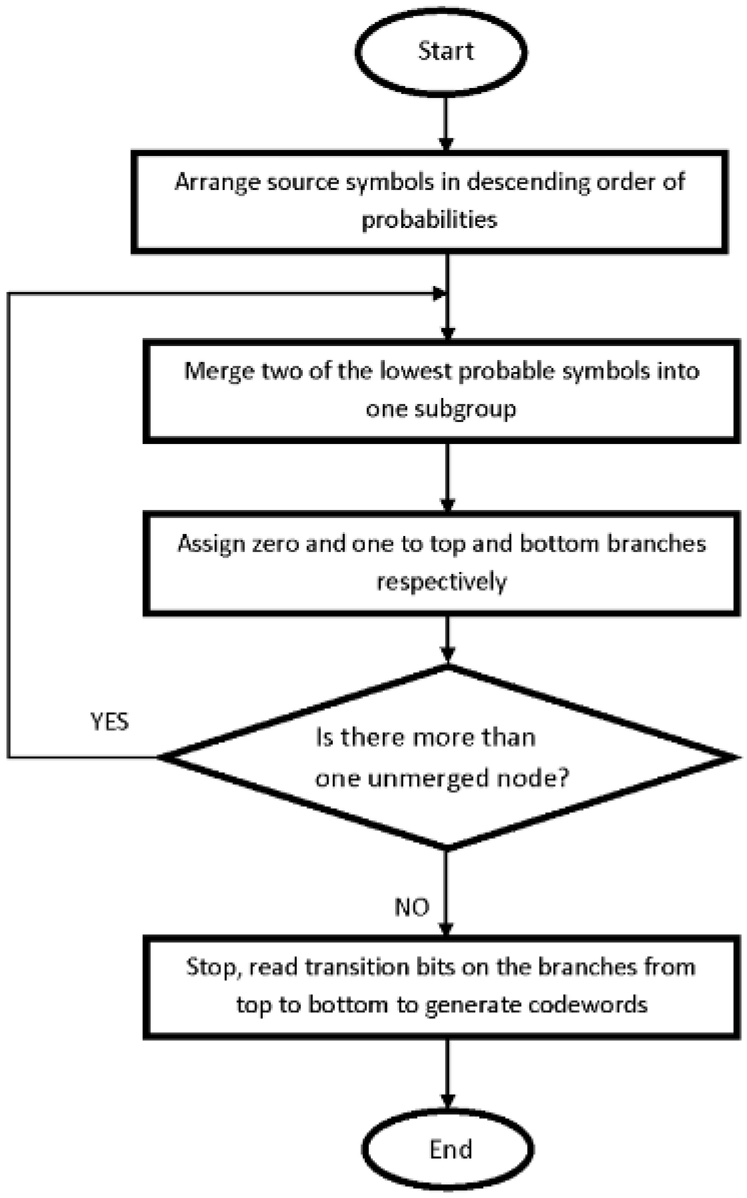
Start
Arrange source symbols in descending order of probabilities
Merge two of the lowest probable symbols into one subgroup
Assign zero and one to top and bottom branches respectively
Is there more than one unmerged node?
YES
NO
Stop, read transition bits on the branches from top to bottom to generate codewords
End

Fig IV.1. Flowchart of Creation of Huffman Tree

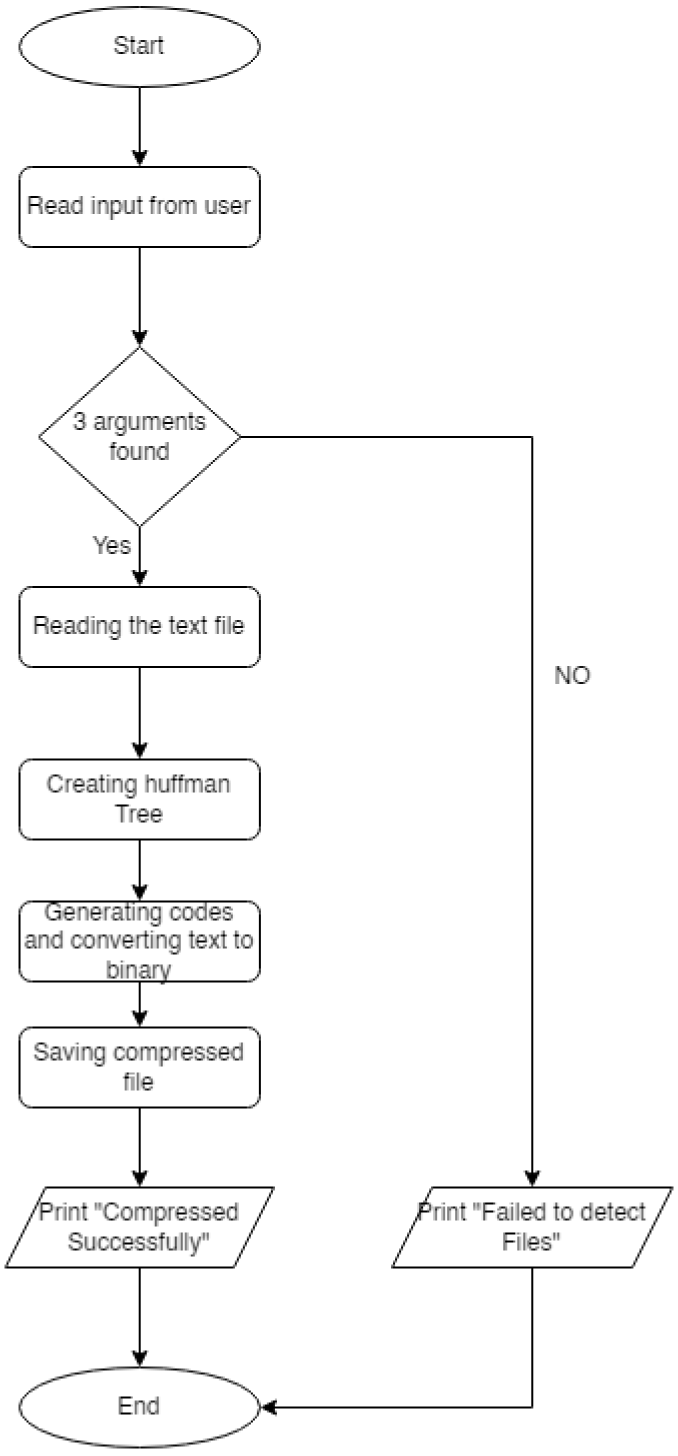

Fig.IV.2. Flowchart of Encoding of File

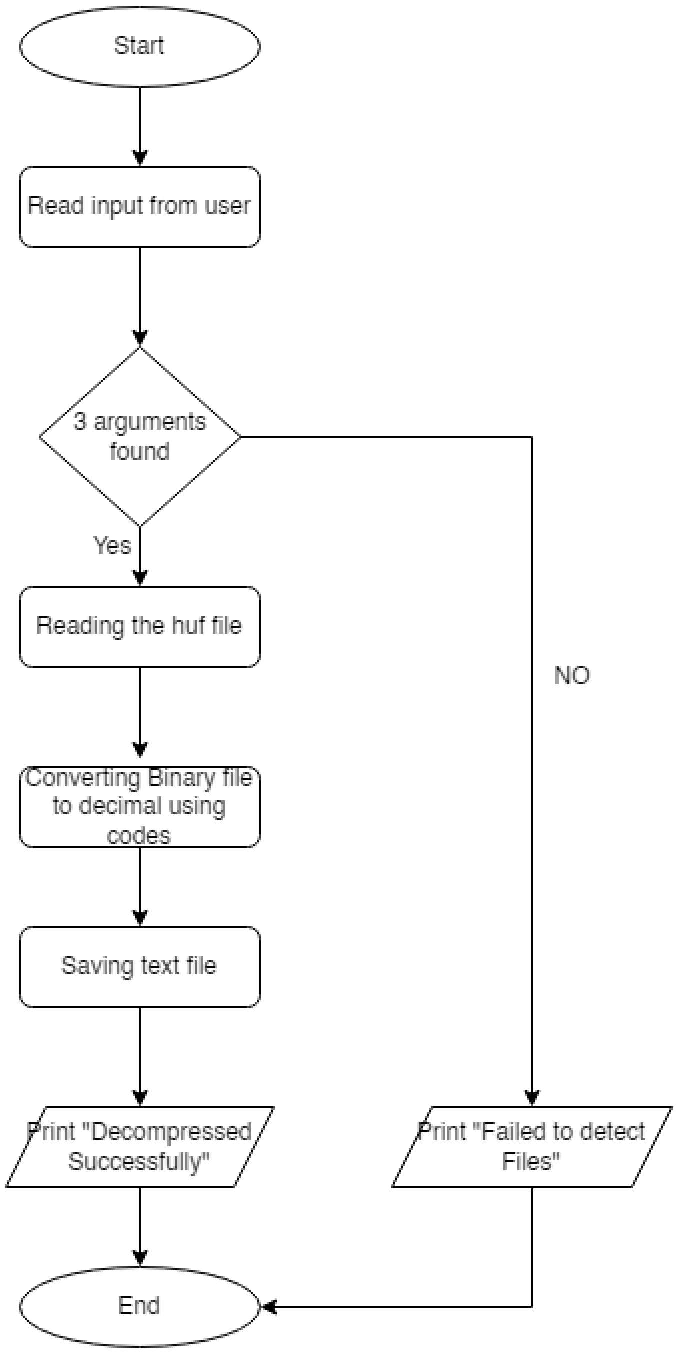
Start
Read input from user
3 arguments found
Yes
NO
Reading the huf file
Converting Binary file to decimal using codes
Saving text file
Print "Decompressed Successfully"
Print "Failed to detect Files"
End

Fig.IV.3. Flowchart of Decoding of File

IV. Platform Used

Visual Studio Code: Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

Visual Studio Code is a source-code editor that can e used with a variety of programming languages, including C#, Java, JavaScript, Go, Node.js, Python, C++, C, Rust, and Fortran.

V. Methodology

Huffman Coding:

Huffman coding is a lossless data compression algorithm. The idea is to assign variable-length codes to input characters, lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code.

The variable-length codes assigned to input characters are Prefix Codes, which means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding ensures there is no ambiguity when decoding the generated bitstream.

Let there be four characters a, b, c, and d, and their corresponding variable length codes are 00, 01, 0, and 1. This coding leads to ambiguity because the code assigned to c is the prefix of codes assigned to a and b. If the compressed bit stream is 0001, the de-compressed output may be "cccd" or "ccb" or "acd" or "ab".

VI. Outcomes

The software will take a text file as a sample and encode it using the Huffman algorithm resulting in the encryption and compression of the file. ZIP files encode information into fewer bits by removing redundant data. This “lossless data compression” ensures all the original data is intact. Zipping your files can help to reduce the amount of space they use on your computer's hard drive. It allows for keeping information confidential too only users with the software would be able to decode and access the data.

VII. Conclusion

Zipper- The file compression model lets us pack more data into a given amount of storage space. Zipper as we have named the model is secure, has data confidentiality, and is not open source. The motive is to provide safe transmission of files between individuals or enterprises having the software. Like other compression models out there, it helps in saving space on hard drives and other media, compression can dramatically improve the speed of file downloads and ensures faster transmission of files over a network or the Internet.

VIII. Application Areas

1. Mostly used for sharing large files over applications with limited space.
2. Allows users to share information confidentially over the internet.
3. Encoding and decoding ensure the lossless conversion of data.
4. Being able to send large numbers of files over email is imperative.

References

[1] A. Malik, N. Goyat and V. Saroha, "Greedy Algorithm: Huffman Algorithm," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 7, pp. 296-303, 2013.

[2] A. S. Sidhu and M. Garg, "Research Paper on Text Data Compression Algorithm using Hybrid Approach," IJCSMC, vol. 3, no. 12, pp. 1-10, 2014.

[3] H. Al-Bahadili and S. M. Hussain, "A Bit-level Text Compression Scheme Based on the ACW Algorithm," International Journal of Automation and Computing, pp. 123-131, 2010.

[4] I. Akman, H. Bayindir, S. Ozleme, Z. Akin and a. S. Misra, "Lossless Text Compression Technique Using Syllable Based Morphology," International Arab Journal of Information Technology, vol. 8, no. 1, pp. 66-74, 2011.

[5] M. Schindler, "Practical Huffman coding," 1998. [Online]. Available: http://www.compressconsult.com/huffman/.

[6] R.S. Brar and B. Singh, "A survey on different compression techniques and bit reduction Algorithm for compression of text data" International Journal of Advanced Research In Computer Science and Software Engineering (IJARCSSE) Volume 3, Issue 3, March 2013

[7] S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms" International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2, Issue 2, March 2013

[8] S. Shanmugasundaram and R. Lourdusamy, "A Comparative Study of Text Compression Algorithms" International Journal of Wisdom Based Computing, Vol.1 (3), Dec 2011

[9] S. Kapoor and A. Chopra, "A Review of Lempel Ziv Compression Techniques" IJCST Vol.4, Issue 2, April-June 2013

[10] S.R. Kodituwakku and U. S. Amarasinghe, “Comparison of Lossless Data Compression Algorithms for Text Data “Indian Journal of Computer Science & Engineering Vol 1 No 4

[11] R. Kaur and M. Goyal, “An Algorithm for Lossless Text Data Compression” International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 7, July - 2013

[12] H. Altarawneh and M. Altarawneh, "Data Compression Techniques on Text Files: A Comparison Study” International Journal of Computer Applications, Vol 26– No.5, and July 2011

[13] U. Khurana and A. Koul, “Text Compression and Superfast Searching” Thapar Institute Of Engineering and Technology, Patiala, Punjab, India-147004