DEPARTMENT OF INFORMATION TECHNOLOGY

Mini Project Synopsis

Title of the project: - **Huffman Reimagined:Innovative Applications of Huffman**

Student Details:-

Student Name	**Roll Number**
Tushar Singh	03211503121
Aditya Gupta	35111503121

Broad area of technology:- **Multimedia Compression** ____________________

Software/ hardware proposed to be use:- ______________________________

____

Mentor/Guide Name: **Dr. Ajay Dureja** ______________________________

Mentor/Guide **Mini Project Coordinator**

Huffman Reimagined: Innovative Applications of Huffman

MINI PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY ENGINEERING

by

Tushar Singh (03211503121)

Aditya Gupta (35111503121)

Guided by

Dr. Ajay Dureja, Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY ENGINEERING BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING (AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI) DELHI – 110063

Oct 2023

ABSTRACT

This research delves into the versatile applications of Huffman coding in compressing diverse data types, including text, images, and audio. The investigation systematically evaluates the effectiveness of Huffman coding in terms of file size reduction, compression and decompression speeds, and overall efficiency for each specific data category.

The study begins by scrutinizing text compression, assessing how well Huffman coding performs in encoding textual information. It considers language specificity and structural characteristics to provide a nuanced understanding of its efficacy in handling various linguistic elements. Moving on to image compression, the research explores how Huffman coding addresses spatial redundancies and effectively manages color information. Additionally, it investigates the impact of Huffman coding on visual quality, providing insights into its role in maintaining or altering the fidelity of compressed images.

In the realm of audio data compression, the research probes into the challenges posed by sound information. It analyzes Huffman coding's effectiveness in handling aspects such as signal processing and frequency distribution, offering a comprehensive view of its performance in compressing audio data while preserving its essential characteristics.

By thoroughly examining Huffman coding across text, images, and audio, this research aims to provide nuanced insights into its applicability and limitations. The findings contribute to the advancement of data compression techniques, impacting areas such as storage, efficiency, and multimedia data processing. The research seeks to enhance our understanding of Huffman coding's role in the broader context of data management, providing valuable information for researchers and practitioners working towards optimizing compression algorithms and advancing multimedia technologies.

Introduction

In information theory, the goal is usually to transmit information in the fewest bits possible in such a way that each encoding is unambiguous. For example, to encode A, B, C, and D in the fewest bits possible, each letter could be encoded as "1". However, with this encoding, the message "1111" could mean "ABCD" or "AAAA"—it is ambiguous.

Encodings can either be fixed-length or variable-length.

A **fixed-length encoding** is where the encoding for each symbol has the same number of bits. For example:

A	00
B	01
C	10
D	11

A **variable-length encoding** is where symbols can be encoded with different numbers of bits. For example:

A	000
B	1
C	110
D	1111

David A. Huffman, the brilliant mind behind **Huffman coding**, succeeded in designing the most efficient compression method of this kind.Huffman code is a way to encode information using variable-length strings to represent symbols depending on how frequently they appear. The idea is that symbols that are used more frequently should be shorter while symbols that appear more rarely can be longer. This way, the number of bits it takes to encode a given message will be shorter, on average, than if a fixed-length code was used. In messages that include many rare symbols, the string produced by variable-length encoding may be longer than one produced by a fixed-length encoding.

The Huffman coding algorithm takes in information about the frequencies or probabilities of a particular symbol occurring. It begins to build the **prefix tree** from the bottom up, starting with the two least probable symbols in the list. It takes those symbols and forms a subtree containing them, and then removes the individual symbols from the list. The algorithm sums the probabilities of elements in a subtree and adds the subtree and its probability to the list. Next, the algorithm searches the list and selects the two symbols or subtrees with the smallest probabilities. It uses those to make a new subtree, removes the original subtrees/symbols from the list, and then adds the new subtree and its combined probability to the list. This repeats until there is one tree and all elements have been added.

❖ Huffman Coding Construction Algorithm

Step 1: Initializing the Priority Queue

- Create a leaf node for each symbol in the input and add it to a priority queue. This priority queue is implemented as a min heap, and nodes are ordered by their probabilities (frequencies).

Step 2: Constructing the Huffman Tree

- While there is more than one node in the queue (i.e., the min heap):

i. Remove the two nodes with the highest priority (lowest probability or frequency) from the queue.

ii. Create a new internal node with these two nodes as children, and set its probability equal to the sum of the probabilities of the two child nodes.

iii. Add the new internal node back to the priority queue.

Step 3: Completing the Huffman Tree

- Once the queue contains only one node, it becomes the root node, and the Huffman tree construction is complete.

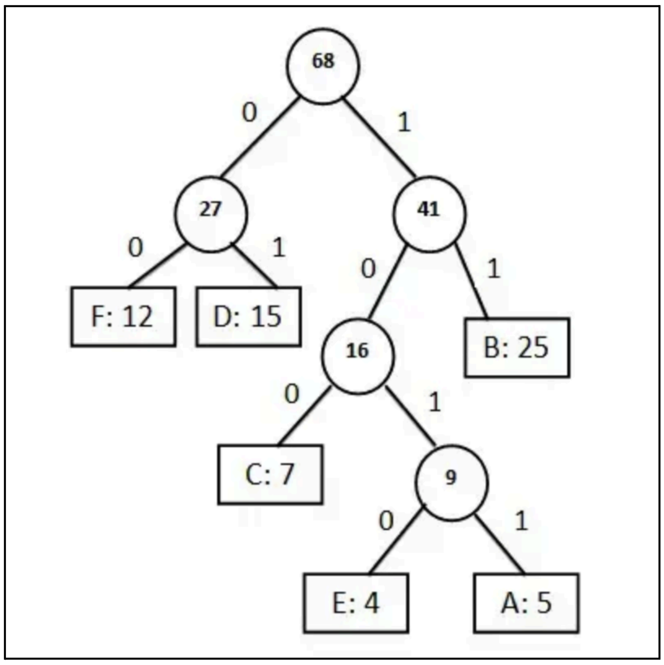
Note:

- The use of a priority queue ensures that nodes with lower probabilities are given higher priority.

- Merging nodes based on frequency creates a full binary tree, and this tree structure forms the basis for the Huffman coding.

Example:

Values	**A**	**B**	**C**	**D**	**E**	**F**
Frequency	5	25	7	15	4	12



❖ Huffman Coding Complexity

- The time complexity for encoding each unique character based on its frequency is O(nlog n).
- Extracting minimum frequency from the priority queue takes place 2*(n-1) times and its complexity is O(log n). Thus the overall complexity is O(nlog n).

❖ The Problem With Image Compression

- It is harder than text encoding.
- Each format has various ways of storing pinery pixel data.
- It is tedious to extract raw pixel data from raw binary format.

Literature Survey

1) Comparison Between (RLE and Huffman) Algorithms for Lossless Data Compression(Dr. Amin Mubarak Alamin Ibrahim and Dr. Mustafa Elgili Mustafa)

In the above mentioned paper, the authors address the growing need for effective data compression techniques in the field of multimedia, particularly in terms of storage capacity and transmission speed. The paper focuses on the comparison between Run Length Encoding (RLE) and Huffman compression algorithms, both of which are lossless compression methods. The authors conducted a thorough analysis, compressing more than 30 text files using C++ programs and calculating the compression ratios with Microsoft Excel. The study concludes that Huffman compression, in most cases, outperforms RLE compression in reducing the file size effectively. This comparison highlights the benefits of Huffman coding for lossless data compression in various applications.

2) Efficient Data Compression for IoT Devices using Huffman Coding Based Techniques(Amlan Chatterjee, Rushabh Jitendrakumar Shah, and Khondker S. Hasan)

In the above mentioned paper the authors address the challenges of managing and transmitting data generated by Internet of Things (IoT) devices in resource-constrained environments. They propose a graph compression technique based on pattern matching and Huffman coding principles to reduce the memory requirements for representing graphs, specifically using the adjacency matrix. The proposed method divides patterns into high and low occurrences, utilizing shorter identifiers for high occurrence patterns and longer ones for low occurrence patterns. The results of simulations demonstrate that this approach can achieve up to 80% compression compared to the standard adjacency matrix representation. These findings are particularly relevant for IoT applications, where efficient data storage and processing are essential. Future work in this area could involve adapting the technique to different data domains and further optimizing pattern generation for improved compression.

3) 3D Medical Image Compression Using Huffman Encoding Technique(Vaishali G. Dubey and Jaspal Singh)

In the following paper, the authors introduce a novel JPEG2000-based lossy image compression method. Their approach divides 3D medical images into non-overlapping tiles and applies 2D discrete wavelet transform (DWT) to each tile. Subsequently, hard thresholding and Huffman coding are employed for compression. This method excels in terms of shorter coding and reduced computational requirements, and it significantly outperforms traditional JPEG compression due to its utilization of 2D DWT. The paper also discusses the JPEG2000 image compression standard, emphasizing its importance in achieving higher compression ratios and superior image quality, particularly in medical imaging. The authors provide a detailed explanation of JPEG2000, including tile decomposition, DWT implementation, and Huffman encoding. In summary, their paper contributes an efficient and effective 3D medical image compression technique, with practical applications in the medical imaging field.

4) An Improved Lossless Image Compression Algorithm Based on Huffman Coding(Xiaoxiao Liu,Ping An,Yilei Chen,Xipeng Huang)

In this paper the authors address the growing challenges of storing and transmitting the increasing volume of image data in modern life. They emphasize the importance of lossless image compression, particularly for applications demanding high-fidelity preservation, such as medical imaging and remote sensing. The authors review existing techniques like Run-Length Encoding, entropy coding, and dictionary-based coding, highlighting their limitations and the emergence of hybrid approaches like Deflate, CALIC, JPEG-LS, and JPEG 2000. The paper explores recent trends, including deep learning models, and introduces a novel lossless compression algorithm that combines linear prediction, integer wavelet transform (IWT), and Huffman coding. The proposed method incorporates a new prediction template and optimized IWT coefficient processing, significantly reducing redundancy in image data. Experimental results confirm its superiority over state-of-the-art algorithms, with improved compression ratios ranging from 6.22% to 72.36%, especially for complex and high-resolution images. The paper concludes by suggesting potential future work to adapt the algorithm to diverse image types and enhance its compression speed.

5) Compressing Huffman Models on Large Alphabet(Gonzalo Navarro and Alberto Ordóñez)

In the following paper authors tackle the challenge of efficiently compressing Huffman models for large alphabets. Traditional Huffman coding has a storage overhead proportional to the alphabet size and the tree depth, which becomes significant when the alphabet size approaches the text length. The paper presents a novel encoding approach that reduces the model's storage size while maintaining efficient encoding and decoding. By sorting symbols in the canonical Huffman tree alphabetically, the method reduces redundancy in the representation, resulting in a significant reduction in space usage. The proposed technique showcases a 15% reduction in storage size for real-life sequences with large alphabets, making it valuable for applications with limited memory, such as text databases and mobile devices. Overall, this paper offers an effective solution for optimizing Huffman model compression on large alphabets, minimizing storage and memory requirements while maintaining reasonable compression and decompression speeds.

6) Comparison of Image Compression Techniques Using Huffman Coding, DWT, and Fractal Algorithm(R. Praisline Jasmi, Mr. B. Perumal, and Dr. M. Pallikonda Rajasekaran)

The paper titled "Comparison of Image Compression Techniques Using Huffman Coding, DWT, and Fractal Algorithm" by R. Praisline Jasmi, Mr. B. Perumal, and Dr. M. Pallikonda Rajasekaran discusses various image compression methods and their performance. Image compression is crucial in multimedia applications, and this paper explores the utilization of Huffman coding, Discrete Wavelet Transform (DWT), and the Fractal algorithm. Huffman coding aims to reduce redundant data in input images, offering efficient lossless compression. DWT enhances compressed image quality, and the Fractal algorithm provides better compression ratios. The authors evaluate these methods by analyzing parameters such as Peak Signal to Noise Ratio (PSNR), Mean Square Error (MSE), Compression Ratio (CR), and Bits per Pixel (BPP) using 512x512-pixel input images. The results reveal that the Fractal algorithm outperforms others in terms of CR and PSNR. The paper provides a comprehensive overview of these image compression techniques and their comparative performance.

7) A Fast and Improved Image Compression Technique Using Huffman Coding(Rachit Patel, Virendra Kumar, Vaibhav Tyagi, and Vishal Asthana)

In the following paper the author focuses on image compression using Huffman coding, a straightforward and efficient compression technique. Image compression is critical for preserving image quality while reducing memory space. The paper's goal is to analyze Huffman coding, which eliminates redundant data by assessing various image quality parameters such as Peak Signal to Noise Ratio (PSNR), Mean Square Error (MSE), Bits Per Pixel (BPP), and Compression Ratio (CR) across various input image sizes. The paper introduces a new approach of splitting the input image into equal rows and columns, and then summing all the individually compressed images to provide better results while maintaining data security. The advantages of image compression techniques in image analysis and security applications are emphasized. The paper provides an organized structure with sections detailing Huffman coding, image quality parameters, mathematical analysis, and the calculated parameters for different image sizes. The results are presented in tables for easy reference, making the paper a valuable resource for those interested in image compression using Huffman coding.

Objective

The primary objective of this project is to conduct a comprehensive exploration into the applications of Huffman coding, specifically focusing on its efficacy in compressing diverse data types, including text, images, and videos. The key goals of our investigation include a detailed assessment of Huffman coding's performance across these distinct data formats. We aim to meticulously analyze factors such as compression efficiency, resulting data ratios, and the preservation of data quality.

Through this project, we seek to deepen our understanding of the practical implications and limitations of Huffman coding as a versatile compression method. Our efforts are geared towards making significant contributions to the fields of data compression, storage efficiency, and multimedia data processing. The ultimate aim is to provide valuable insights that contribute to the optimization of compression algorithms and enhance the utilization of Huffman coding in real-world applications.

Applications of Huffman Coding

It is used broadly to encode music,images,and certain communication protocols.Typically,a variation of algorithm is used for improved efficiency.The method described is generally part of general compression algorithms such as Flat Zip for images or FLAC for music.

❖ **Text Compression**

- Frequency Analysis: Analyze the text to identify the frequency of each character or symbol to determine which ones are more common and which are less common.
- Huffman Tree Construction: Build a binary tree, called a Huffman tree, where common characters are placed closer to the root, and less common characters are deeper within the tree structure.
- Huffman Codes: Assign binary codes to characters based on their position in the Huffman tree. Common characters get shorter codes, while less common ones receive longer codes.
- Encoding: Replace characters in the text with their corresponding Huffman codes. This results in a compressed version of the text, where common characters are represented with fewer bits.
- Decoding: Use the same Huffman tree to reverse the process and convert the Huffman codes back into the original characters, effectively decompressing the text.

❖ **Image Compression**

Image compression using Huffman coding is similar to text compression but applied to image data:

- Frequency Analysis: Analyze the image data to identify which pixel values occur frequently and which occur less frequently.
- Huffman Tree Construction: Build a Huffman tree, with pixel values mapped to nodes, where common pixel values are closer to the tree's root, and less common values are deeper within the tree.
- Huffman Codes: Generate binary codes for each pixel value based on its position in the Huffman tree. Common pixel values get shorter codes, and less common values get longer codes.
- Encoding: Replace pixel values in the image with their corresponding Huffman codes. This results in a compressed image where common pixel values are represented with fewer bits.
- Decoding: Utilize the Huffman tree to decode the Huffman codes and recover the original pixel values, reconstructing the image.

- ❖ **Audio Compression**

Audio compression with Huffman coding is suitable for coding audio signals such as speech or music:

- Frequency Analysis: Examine the audio data to identify common audio samples or frequency components and less common ones.
- Huffman Tree Construction: Create a Huffman tree where common audio components are placed closer to the root, and less common components are deeper within the tree.
- Huffman Codes: Generate binary codes for each audio component or frequency component based on their position in the Huffman tree. Common components receive shorter codes, and less common components receive longer codes.
- Encoding: Replace audio components with their corresponding Huffman codes, creating a compressed audio stream. Common components are represented more efficiently.
- Decoding: Use the Huffman tree to decode the Huffman codes and restore the original audio samples or frequency components, reconstructing the audio while minimizing data size.

References

I) Patel, Rachit, Virendra Kumar, Vaibhav Tyagi, and Vishal Asthana. "A Fast and Improved Image Compression Technique Using Huffman Coding." In *IEEE WISPNET 2016 Conference*, 2016.

II) Jasmi, R. Praline, Mr. B. Perumal, and Dr. M. Pallikonda Rajasekaran. "Comparison of Image Compression Techniques using Huffman coding, DWT, and Fractal Algorithm." In *2015 International Conference on Computer Communication and Informatics (ICCCI-2015)*, January 8-10, 2015

III) Navarro, Gonzalo, and Alberto Ordonez. "Compressing Huffman Models On Large Alphabets." In *2013 Data Compression Conference*

IV) Liu, Xiaoxia, Ping An, Yilei Chen, and Xinpeng Huang. "An Improved Lossless Image Compression Algorithm Based on Huffman Coding." Journal Name

V) Chatterjee, Amlan, Rushabh Jitendrakumar Shah, and Khondker S. Hasan. "Efficient Data Compression for IoT Devices using Huffman Coding Based Techniques." In *2018 IEEE International Conference on Big Data*.

VI) Dubey, Vaishali G., and Jaspal Singh. "3D Medical Image Compression Using Huffman Encoding Technique." *International Journal of Scientific and Research Publications* 2, no. 9 (2012).

VII) Mubarak, Amin, and Mustafa Elgin Mustafa. "Comparison Between (RLE and Huffman) Algorithms for Lossless Data Compression." *International Journal of Innovative Technology and Research* 3, no. 1 (December-January 2015).