

Lossless File Compressor Decompressor



|  |  |
| --- | --- |
| Muhammad Zaid Naeem (Leader) | SP24-BCS-098(A) |
| Muhammad Huzaifa Saboor | **SP24-BCS-083(A)** |

# Project Title:

**File Compression & Decompression System (Lossless Compression Module)**

Introduction:

The **File Compression & Decompression System** is a lossless compression tool designed to efficiently reduce file sizes while preserving data integrity. The system utilizes two widely used compression algorithms: **Huffman Encoding** and **LZW (Lempel-Ziv-Welch)**. It allows users to compress and decompress .TXT files, view original and decompressed contents, and manage compression history with advanced search and sorting functionalities. This project aims to demonstrate practical file compression techniques while providing a user-friendly interface through a menu-driven console application.

# Features:

1. **View Original .TXT File Content**
   * Allows the user to open and display the contents of the original text file before compression.
2. **Compress File (Huffman Algorithm)**
   * Compresses the file using Huffman Encoding, which builds an optimal tree to minimize the file size based on character frequencies.
3. **Decompress Huffman Compressed File**
   * Create new decompressed files that were compressed using Huffman Encoding to restore the original content.
4. **View Decompressed Huffman File Content**
   * Displays the contents of the file after decompression to verify the accuracy of the Huffman decompression process.
5. **Compress File (LZW Algorithm)**
   * Compresses the file using LZW (Lempel-Ziv-Welch) algorithm, which replaces repeated strings with codes, achieving high compression rates on repetitive data.
6. **Decompress LZW Compressed File**
   * Create new decompressed files compressed with LZW algorithm back into their original form.
7. **View Decompressed LZW File Content**
   * Displays the contents of the LZW decompressed file for verification.
8. **View Compression History (Unsorted)**
   * Maintains and displays a history of all compression operations, showing details like file names, algorithms used, compression ratios of size decreasing and increasing.
9. **Sort History by Compression Ratio**
   * Sort the history based on the file compression and decompression ratio by different type of sorting algorithms including quick sort, insertion sort and selection sort.
10. **Search History by ID (Binary Search)**

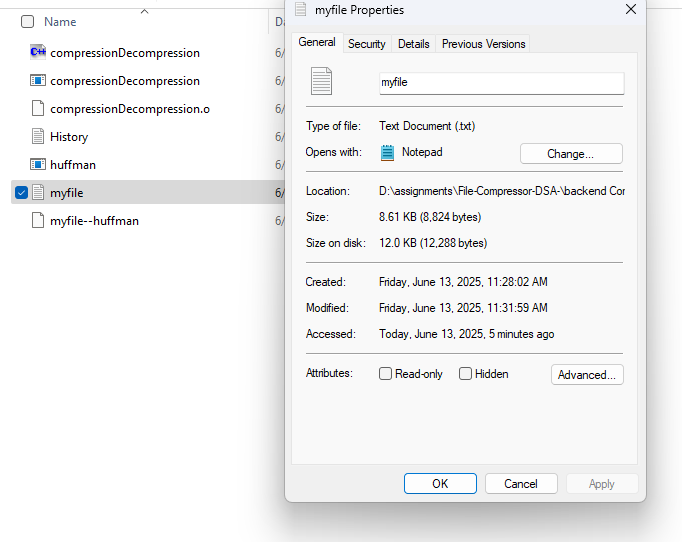
Provides a fast search option to find specific history records using binary search on unique IDs.

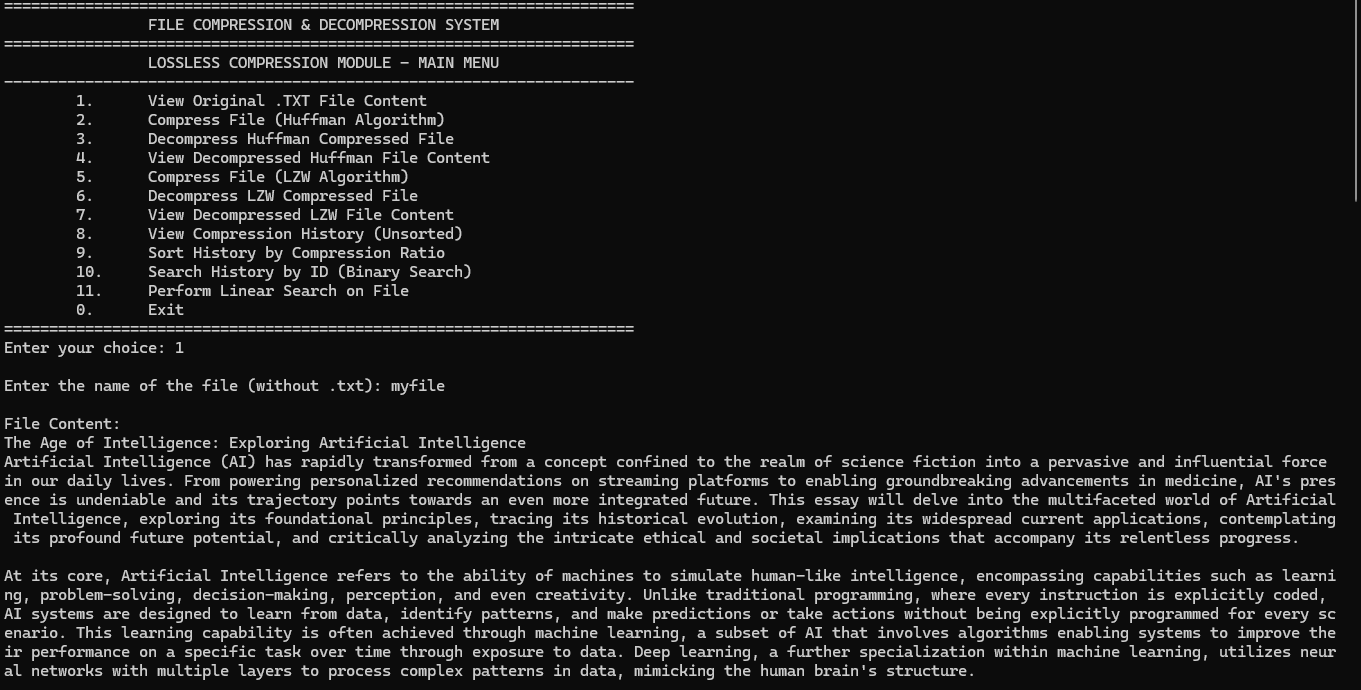
1. **Perform Linear Search on File**

Enables users to perform a linear search on files based on the operation of compression, decompression and file name.

1. **Error Handling** 
   1. Comprehensive error handling to manage any unexpected behavior like decompressing file of Huffman compression by LZW or vice versa.

Output





A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A close-up of a screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black and white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Conclusions and Findings

Throughout the development of the **File Compression & Decompression System**, we gained valuable insights into the practical implementation of lossless compression algorithms. The project successfully demonstrated how algorithms like **Huffman Encoding** and **LZW** can significantly reduce file sizes while maintaining data integrity.  
We also integrated additional functionalities such as file viewing, searching (linear & binary search), and sorting based on compression ratios, which further enriched the system’s usability.

The project provided a solid understanding of:

* How frequency-based and dictionary-based compression algorithms work.
* Efficient file handling and binary data manipulation.
* The importance of maintaining compression history for analysis.
* Implementing searching and sorting algorithms within real-world systems.

# Challenges Faced

During the course of this project, we encountered several challenges, including:

1. **Understanding and Selecting Compression Algorithms:**
   * It was initially difficult to deeply understand how different compression algorithms work and which algorithms would provide the best compression ratio for various file types.
   * Grasping the core concepts behind Huffman Trees (priority queues, binary trees) and LZW dictionaries took significant effort.
2. **Implementing Search Functionality:**
   * Applying both **linear search** and **binary search** on compression history data structures was tricky, especially ensuring correct indexing and data integrity.
3. **Sorting Challenges:**
   * While sorting the compression history by compression ratio, one major issue was that sorting modified the original ArrayList (or data structure), which led to unexpected changes elsewhere in the system.
   * We had to carefully manage temporary copies of the list to avoid altering the original dataset.
4. **File Handling and Binary Data Storage:**
   * Reading, writing, and decoding binary data accurately posed difficulties, especially in maintaining synchronization between compressed and decompressed files.
   * Ensuring the system could handle both small and large files efficiently without data loss.

**5. Code Complexity**

* + As the project progressed, the codebase became increasingly complex due to the integration of multiple algorithms, file handling mechanisms, searching, sorting, and history management.
  + Maintaining clean, organized, and bug-free code while managing multiple interdependent modules required careful planning, debugging, and frequent code reviews.

1. **Handling Equal Frequencies in Huffman Encoding**
   * One major challenge in building the Huffman tree was handling cases where multiple characters had the same frequency. In such cases, the structure of the tree — and thus the generated codes — could vary depending on the order of insertion into the min-heap.
   * To ensure deterministic and consistent encoding, we modified our min-heap implementation to include the ASCII value of characters as a secondary key. This way, when two characters had equal frequency, the one with the lower ASCII value was given higher priority, resulting in a stable and predictable Huffman tree.

# Potential Future Enhancements

While the current system is functional, several improvements can be made in the future:

1. **Support for Multiple File Formats:**

Extend compression capabilities to handle different file types such as images, audio, or video.

1. **Graphical User Interface (GUI):**

Develop a user-friendly GUI to improve interaction and accessibility for non-technical users.

1. **Cloud Integration:**

Enable users to upload and compress files directly from/to cloud storage services.

# Contributions of Each Group Member

The development of the **File Compression & Decompression System** was a collaborative effort between two dedicated team members. Both members contributed actively by participating in regular online meetings, discussing problems, sharing ideas, and jointly solving technical challenges throughout the project.

Throughout the project:

* Both members collaboratively studied and understood the concepts of lossless compression algorithms, including Huffman and LZW.
* We worked together on the design and implementation of compression and decompression modules.
* **Google Meet sessions** were conducted to discuss challenges, divide tasks, and brainstorm solutions.
* Both actively participated in debugging, testing, and improving the code to ensure smooth functioning.
* The search and sort functionalities, including compression history management, were developed with joint efforts.
* Report writing was completed through mutual coordination and equal contribution.

By effectively sharing tasks, constantly communicating, and supporting each other at every stage, both team members ensured the successful and timely completion of the project.