COMPRESSION COMPARISON LAB

1. INTRODUCTION

Data compression is the process of reducing the size of data by removing the redundancy for efficient storage and easier transmission. This project focuses on the implementation of two lossless compression techniques: **Delta Encoding** and **Run-Length Encoding (RLE)**. Lossless compression ensures that the original data can be reconstructed exactly from the compressed data.

2. OBJECTIVES

The main objective of this project is to:

- Demonstrate the implementation of delta encoding and run-length encoding.
- Calculate and compare the compression ratio of different data types.

3. BACKGROUND THEORY

Delta encoding is a compression technique where the differences between the consecutive values are stored rather than the individual absolute values. It is ideal for data that has values which change gradually.

Run-length encoding is a compression technique that replaces the sequences of repeated values with a single value and a count *(the number of times it has been repeated)*. For example; a sequence such as *(555555)* will be stored as *(5,6).* This technique is ideal for data that has repeating values.

The Combined Approach: In this project delta encoding is applied first to increase the repetition in the sequential data, followed by run-length encoding to compress the delta encoding values. This increases the compression efficiency for data with a predictable pattern.

4. METHODOLOGY

Input Data Types

This system supports two types of input files:

1. Text files – The characters are converted to their ASCII values before compression.
2. Numeric files – The numbers are processed directly as integers.

Compression Pipeline

The compression process follows the following steps:

1. Reading the file inputs.
2. Converting the characters to ASCII values (if the file uploaded is text).
3. Delta encoding is applied to the values.
4. Run-length encoding is applied to the delta encoding results.
5. The compression ratio is calculated.

The Compression Ratio

The compression ratio shows how small or big the compressed data is compared to the original size. In this system, the ratio is calculated in terms of the symbol counts rather than the memory bytes to clearly show the effectiveness of the algorithm.

Compression ratio = Original Size/ Compressed Size
Where;
Original size = the number of symbols before compression
Compressed size = the number of pairs multiplied by two

5. IMPLEMENTATION

The compression algorithms are implemented in C++, chosen for its high efficiency.
The key function include:

- File reading functions for both text and numeric files
- ASCII conversion for text files
- Delta encoding function
- Run-length encoding function
- Compression ratio calculation function

The program is the presented through a HTML- based web app which visualizes the same compression process as the core algorithm and produces identical results. The web app:

- Allows the user to upload a file (whether a text or numeric file).
- Performs the compression and calculates the compression ratio.
- Displays the delta encoding results, run-length results, original size, compressed size and the compression ratio.

6. RESULTS AND ANALYSIS

For the files that were used; there were three sets of data:

a) Constant data – The values were constant; they did not change. It produced the highest compression ratio due to high redundancy.
b) Linear data – The values were changing gradually. This set produced moderate compression as delta values were being repeated.i.e., the predictability was high.
c) Random data – The values were random. This set produced the lowest compression ratio since there was less predictability in the data for the algorithm to exploit.

These results were exhibited by both text and numeric files.

INPUT TYPE: TEXT FILES			
Data Set	***Original Size** (symbols)*	***Compressed Size** (pairs*2)*	***Compression Ratio** (original size/compressed size)*
Constant data	50	4	12.50

Linear data	11	4	2.75
Random data	17	32	0.63
INPUT TYPE: NUMERIC FILES			
Data Set	**Original size (symbols)**	**Compressed size (pairs*2)**	**Compression ratio (original size/ compressed size)**
Constant data	18	4	4.50
Linear data	10	4	2.50
Random data	12	20	0.60

7. DISCUSSION

The results show that data compression relies mostly on data structure rather than data size only. Both delta encoding and run-length encoding were effective where redundancy was present. To determine whether the compression was successful, the following logic was used:

- If the ratio is greater than one, compression was excellent.
- If the ratio is equal to one, there was no compression.
- If the compression is less than one, compression was poor.

For constant and linear data, the ratio was greater than one; hence compression was excellent. For random data, the ratio was less than one; hence compression was poor.

For linear data in both types of inputs, run-length encoding alone would have failed since there are no repeated values. Delta encoding creates the repeated values by storing the differences between consecutive values in the data thus allowing run-length encoding to function.

For random data, the compressed size is larger than the original size. This is because the algorithm needs to store a value and its count for every entry. Since the data has values which frequently and rarely repeat, delta encoding fails to create repeated values for run-length encoding to compress effectively. Instead of shrinking the data, run-length encoding doubles the storage requirement for each unique value leading to data expansion.

8. CONCLUSION

This project successfully implemented and compared delta encoding and run-length encoding for both text and numerical data. The results show that combining delta encoding with run-length encoding can significantly improve compression efficiency for structured data. However, for random data, compression is ineffective and may lead to data expansion.