Data Compression using Java(SRS)

1. Introduction

Data compression is the process of reducing the size of a data file such that it uses less space than the original file and can be transferred easily. The file is encoded using encoding information that uses fewer bits than the original representation. There are many data compression algorithms which aim to compress data of different format.

There are different algorithms even for a single data format that uses different approach to compress the data. In all of data compression algorithms, we first need to study and analyze the characteristics of the data in order to deduce some patterns that can help us to compress the data into a smaller size and compact representation. The main aim of the compression techniques is to reduce the size of the data file by removing redundancy in stored data, thus increasing data density and making it easier to transfer data files. In this paper, we will study about different image compression techniques and text compression techniques used in real world. The methods such as Huffman coding, Arithmetic coding and LZW coding are considered.

Purpose

The purpose of developing a Data Compression system using Java is to address the fundamental need for efficient storage and transmission of digital information. Data compression involves reducing the size of data files while retaining essential information, thereby optimizing storage space and reducing bandwidth requirements for data transmission over network.

Scope

Implementation of Compression Algorithms: The project will involve the implementation of various compression algorithms in Java, such as Huffman coding, Lempel-Ziv-Welch (LZW), Run-Length Encoding (RLE), and others. These algorithms will be designed to effectively reduce the size of data files while preserving essential information.

1. Overall Description

Product Prespective

The Data Compression application is designed to compress and decompress data efficiently using various algorithms implemented in Java. It targets users who need to reduce storage space or optimize data transmission over networks.

Software Requirement

* Operating System: Windows 7 & above.
* System Type: 32/64- bit operating system
* Programming Language used: Java version 8.0 & above

Hardware Requirement

* RAM: 2 GB
* Processor: Intel core processor & above

Functional Requirement

R1: Compression Algorithm Implementation:

* Implement various compression algorithms including Huffman coding, Lempel-Ziv-Welch (LZW), and Run-Length Encoding (RLE) using Java libraries.
* Ensure each algorithm effectively reduces the size of data files while preserving essential information.

R2: Compression Process**:**

* Design the compression process to analyze input data and apply the selected compression algorithm.
* Ensure efficient processing of data to achieve high compression ratios and minimize processing time

R3: Decompression Process:

* Develop functionality to decompress compressed data files using the appropriate decompression algorithms.
* Ensure the decompression process accurately restores the original data without loss of information.

R4: Libraries Implementation

* Utilize existing Java libraries for data compression algorithms to streamline development and ensure compatibility.
* Integrate external libraries seamlessly into the project environment for efficient algorithm implementation.

Non- Functional Requirements

* Portability

Portability considerations should prioritize the adoption of cross-platform Java libraries and frameworks to mitigate platform-specific dependencies. It should be capable of running on different hardware architectures, guaranteeing flexibility in deployment scenarios. Additionally, portability testing should be conducted rigorously to validate the software's performance and functionality across different platforms.

* Compression Speed

The software should strive to achieve high compression speeds, minimizing the time required to compress data files while maintaining efficient resource utilization. Optimization techniques, such as algorithmic improvements and parallel processing, should be implemented to enhance compression speed without compromising compression ratios.

* Compression Ratio

The software should aim to achieve high compression ratios, effectively reducing the size of data files while preserving essential information. It should prioritize the development and implementation of efficient compression algorithms, such as Huffman coding and Lempel-Ziv-Welch, to maximize compression ratios without significant loss of data quality. Performance testing should be conducted to evaluate and optimize compression ratios under different data types and sizes