Huffman Coding- Final Project

Submitted to Dr. Md. Al Hasan

Table of Contents:

1. Overview
2. Step by step execution in Local Machine – Xcode Platform supporting C++.
3. Step by step execution in Linux machine
4. Contribution

1. Overview:

Huffman coding is a widely used method for lossless data compression, particularly for compressing text files. It was developed by David A. Huffman in 1952. The core idea behind Huffman coding is to assign variable-length codes to input characters based on their frequencies in the data. Characters that occur more frequently are assigned shorter codes, while characters that occur less frequently are assigned longer codes.

**Compression with Huffman Coding:**

1. Character Frequency Calculation: The compression process starts with calculating the frequency of each character in the input text.

2. Building the Huffman Tree: Using the character frequencies, a Huffman tree is constructed. The tree is built in a greedy manner, where the least frequent characters are assigned codes at the deepest levels of the tree.

3. Assigning Huffman Codes: Traversing the Huffman tree from root to leaf, Huffman codes are assigned to each character. These codes are binary sequences, with shorter codes assigned to more frequent characters.

4. Encoding: Once the Huffman codes are determined, the input text is encoded by replacing each character with its corresponding Huffman code. The encoded text is typically a binary sequence, which represents the original text in a more compact form.

5. Output: The compressed data, along with the Huffman tree or code table, is stored in a compressed file format. This file format usually includes metadata to facilitate decompression, such as the original text size and a marker to identify the data as Huffman compressed.

**Decompression with Huffman Coding:**

1. Reading Huffman Codes: The decompression process starts by reading the Huffman codes and the Huffman tree or code table from the compressed file.

2. Reconstructing the Huffman Tree: Using the information from the compressed file, the Huffman tree is reconstructed.

3. Decoding: The encoded data from the compressed file is then decoded using the Huffman tree. Starting from the root of the tree, each bit of the encoded data is used to traverse the tree until a leaf node (character) is reached. The character at the leaf node is then output.

4.Output: As the encoded data is decoded, the original text is reconstructed character by character.

**Advantages of Huffman Coding:**

1. Lossless Compression: Huffman coding is a lossless compression technique, meaning that the original data can be perfectly reconstructed from the compressed data.

2. Efficient Compression: Huffman coding produces optimal compression results for text data, often achieving significant reductions in file size.

3. Fast Decompression: Decompression with Huffman coding is typically fast, especially when compared to more complex compression algorithms.

1. Step by step execution in Local Machine:

Our main goal is to implement Huffman Coding using C++ as our preferred programming language. A detailed explanation of execution in local machine for compression and decompression is shown below.

2.1 Compression:

The compression folder has, the main.cpp, test.txt, output.hzip

A screenshot of a computer

Description automatically generated

**Step1:**

 We used the Xcode, to run our C++ code. This can be easily installed in the Mac OS.

**Step2:  Components**

This program implements Huffman coding to compress text files efficiently. It takes a text file as input, generates Huffman codes for each character, and produces a compressed binary file. It consists of the following components:

1**. Source Code Files:**

   - main.cpp: Contains the main function and command-line argument parsing.

   - huffman.cpp: Contains functions for Huffman compression.

2**. Makefile:**

   -  Defines rules for compiling the source code files and generating the executable.

# Compiler

CXX = g++

# Compiler flags

CXXFLAGS = -std=c++11 -Wall -Wextra

# Directories

SRC\_DIR = src

BIN\_DIR = bin

# Source files

MAIN\_SRC = $(SRC\_DIR)/main.cpp

HUFFMAN\_SRC = $(SRC\_DIR)/huffman.cpp

# Executable

MAIN\_EXE = $(BIN\_DIR)/compress

all: $(MAIN\_EXE)

$(MAIN\_EXE): $(MAIN\_SRC) $(HUFFMAN\_SRC)

$(CXX) $(CXXFLAGS) $(MAIN\_SRC) $(HUFFMAN\_SRC) -o $(MAIN\_EXE)

clean:

rm -f $(BIN\_DIR)/\*

.PHONY: all clean

3. **Compression**

To compress the text file, we used the following arguments in the Scheme location:

(when you try to execute our source code you have to modify the scheme location according to your file path)

-f  /Users/sripadma/Desktop/Sri/Algoproject/Compression/test.txt -o //Users/sripadma/Desktop/Sri/Algoproject/Compression/output.hzip -s

**Step3:**

Compression Execution

A screenshot of a computer

Description automatically generated

Contents of Output.hzip:

A white background with black and white clouds

Description automatically generated

* 1. Decompression:

The Decompression folder has, the main.cpp, test.txt, output.hzip

A screenshot of a computer

Description automatically generated

**Step 1:**

**Huffman Decompression Program**

This program implements Huffman decoding to decompress text files that have been compressed using Huffman coding. It takes a compressed binary file as input and produces the original uncompressed text file.

**Step 2:** **Components**

The program consists of the following components:

**1. Source Code Files**:

   - main.cpp: Contains the main function and command-line argument parsing.

   - huffman.cpp: Contains functions for Huffman decompression.

**2. Makefile:**

Defines rules for compiling the source code files and generating the executable.

# Compiler

CXX = g++

# Compiler flags

CXXFLAGS = -std=c++11 -Wall -Wextra

# Directories

SRC\_DIR = src

BIN\_DIR = bin

# Source files

MAIN\_SRC = $(SRC\_DIR)/main.cpp

# Executable

MAIN\_EXE = $(BIN\_DIR)/decompress

all: $(MAIN\_EXE)

$(MAIN\_EXE): $(MAIN\_SRC)

$(CXX) $(CXXFLAGS) $(MAIN\_SRC) -o $(MAIN\_EXE)

]

clean:

rm -f $(BIN\_DIR)/\*

.PHONY: all clean

**2. Decompression:**

To decompress a Huffman compressed file, we used the following arguments in the Scheme location: (when you try to execute our source code you have to modify the scheme location according to your file path)

-f /Users/sripadma/Desktop/Sri/Algoproject/Decompression/input.hzip -o //Users/sripadma/Desktop/Sri/Algoproject/Decompression/output.txt -s

**Step 3: Output**

Decompression Execution

A computer screen shot of a computer

Description automatically generated

Decompression Output to original file

A close up of a computer screen

Description automatically generated

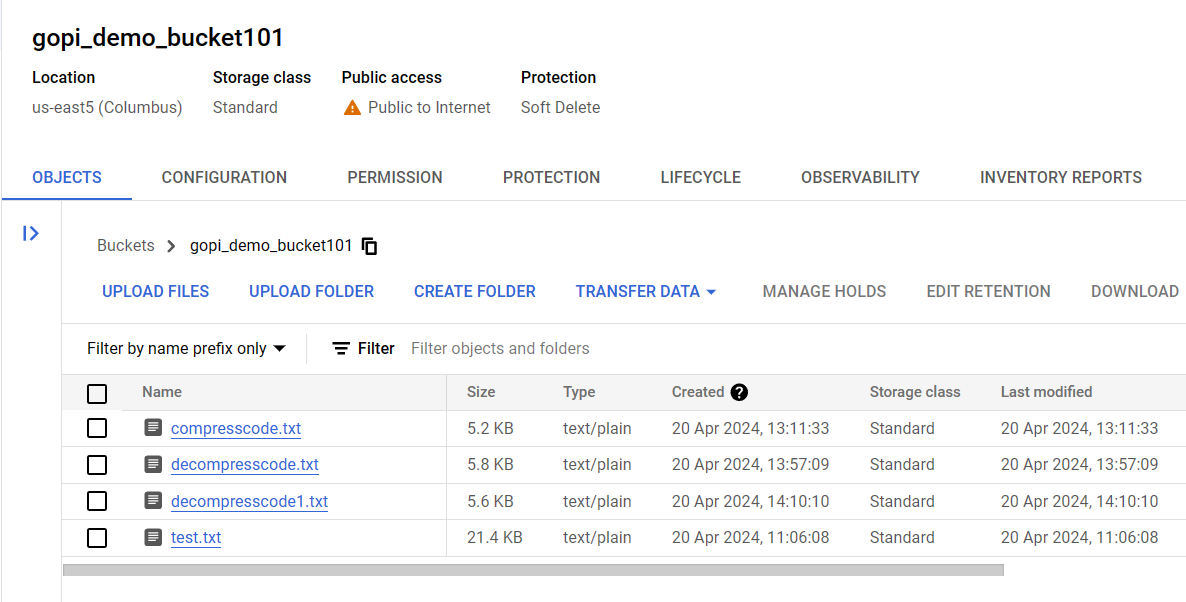
1. Step by step execution in Linux Machine

The Linux machine (CentOS) is hosted in the Google Cloud Platform.

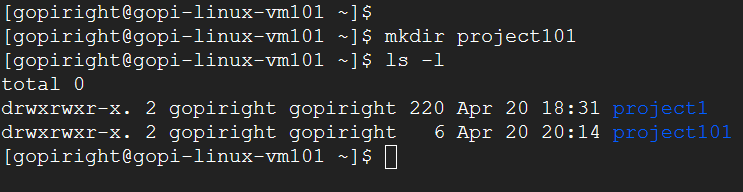
A screenshot of a computer

Description automatically generated

Large text file and source code files are located in the Google Cloud Storage



**Step 1:** Creating a new directory named: project101 in the linux machine.



**Step 2:** Moving the cloud storage files into linux machine.

A screen shot of a computer

Description automatically generated

Large text file, compression source code and decompression source code are located in the linux machine.

A screen shot of a computer

Description automatically generated

**Compression Activities in the Linux machine**

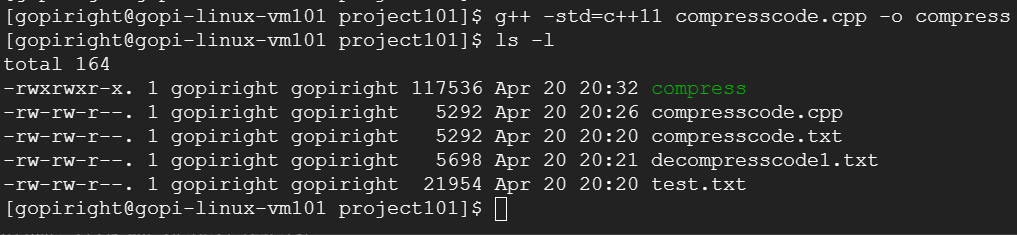
**Step 1:** creating the compresscode.cpp file and copying the compresscode.txt file content to compresscode.cpp



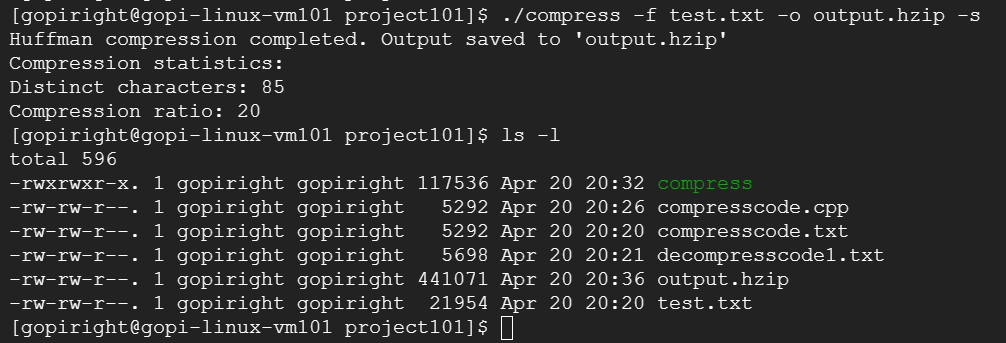
A computer screen shot of a black screen

Description automatically generated

**Step 2:** Compiling the compress source code (compresscode.cpp) using the g++ compiler in the linux machine.



**Step 3:** Executing the command to compress the large text file using the complied program.



Observe the output.hzip file generated after the Huffman compression completed successfully.

**Step 4:** Verifying the compressed file (output.hzip) content

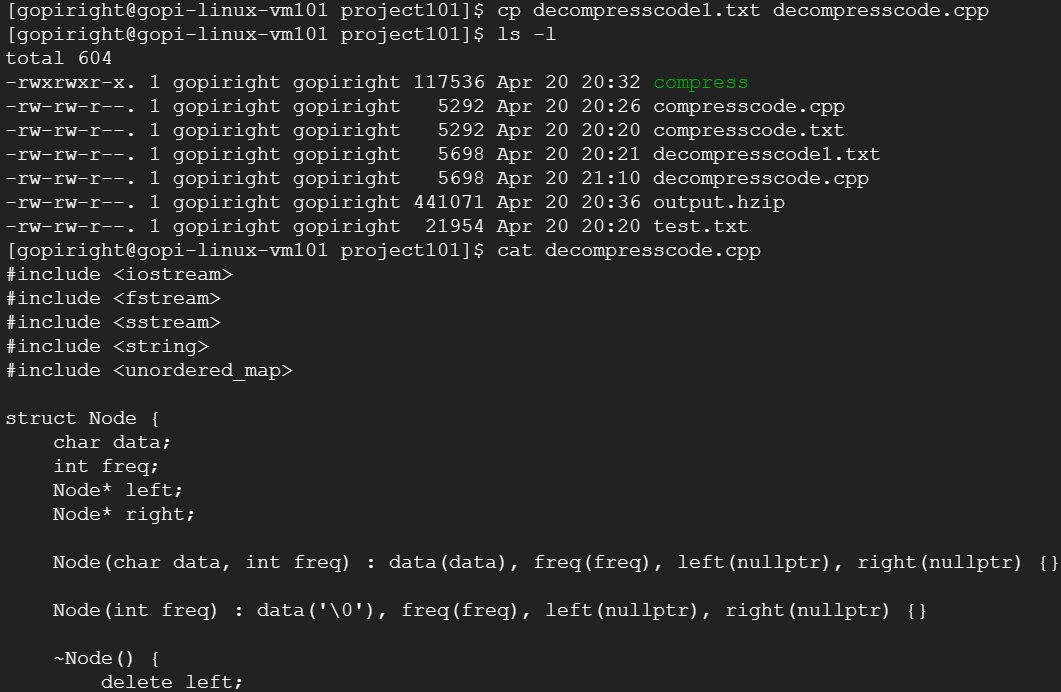
A screen shot of a computer

Description automatically generated

**Decompression Activities in the Linux machine**

**Step 1:** creating the decompresscode.cpp file and copying the decompresscode1.txt file content to decompresscode.cpp





**Step 2:** Compiling the decompress source code (decompresscode.cpp) using the g++ compiler in the linux machine.

A screen shot of a computer

Description automatically generated

**Step 3:** Executing the command to decompress the compress file (output.hzip) using the complied program.

A black screen with white text

Description automatically generated

Decompressed output.hzip file output is stored into the file: outputfile.txt



Observe the outputfile.txt file generated after the Huffman decompression completed successfully.

A screenshot of a computer

Description automatically generated

**Step 4:** Verifying the decompressed file (outputfile.txt) content

A screen shot of a computer screen

Description automatically generated

1. Contribution:

Sri Padmavathi Manoharan - Executed the compression part.

Venkata Parupudi - Executed the decompression part.