Huffman Text and File Compressor

Dasari Veera Venkata Abhinav Teja
B.Tech, Computer Science and Engineering
Indian Institute of Technology Kharagpur

June 22, 2025

Project Repository

https://github.com/abhinavteja2005/HuffmanCompressor

Contents

1 Project Overview

This report outlines the design and development of a multi-interface Huffman Coding Compression Suite. The suite contains:

- A Python GUI application for text-based Huffman encoding/decoding.
- A C++ CLI-based file compressor.
- A Flask web interface that interacts with the C++ compressor.

Each component focuses on a different platform but shares the same core Huffman logic.

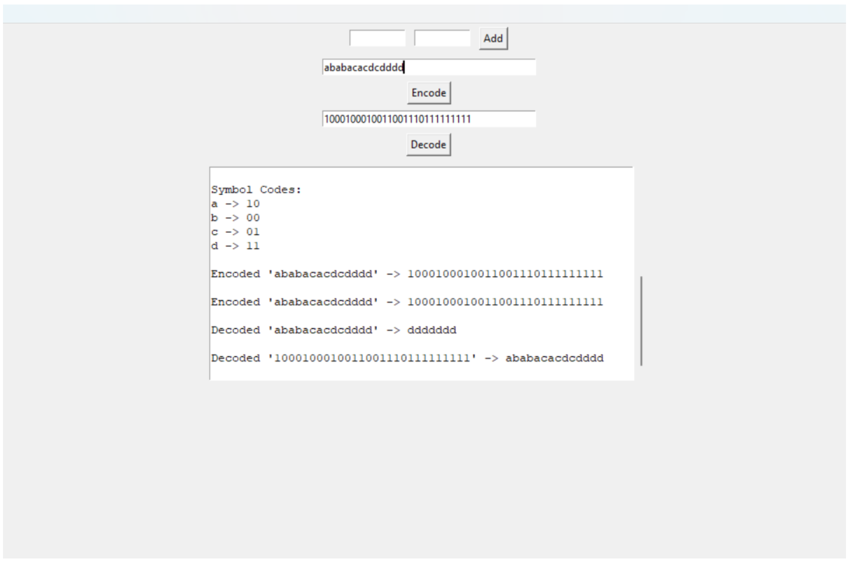
2 TextCompressorLearn (Python + Tkinter GUI)

Overview

This is a simple tool that allows users to:

- Enter characters and frequencies.
- View generated Huffman prefix codes.
- Encode a text into Huffman binary code.
- Decode a binary string back into text.

It is built using Python's `Tkinter` module.



```
ababacacdcdddd
```

```
1000100010011001110111111111
```

```
Symbol Codes:
a -> 10
b -> 00
c -> 01
d -> 11

Encoded 'ababacacdcdddd' -> 1000100010011001110111111111

Encoded 'ababacacdcdddd' -> 1000100010011001110111111111

Decoded 'ababacacdcdddd' -> ddddddd

Decoded '1000100010011001110111111111' -> ababacacdcdddd
```

Figure 1: Tkinter GUI Interface for Text Compression

3 File Compressor (C++ CLI + Web Integration)

Overview

The file compressor consists of:

- A backend CLI tool in C++ named `File Compressor`
- A Flask web frontend named `HuffmanWebApp`

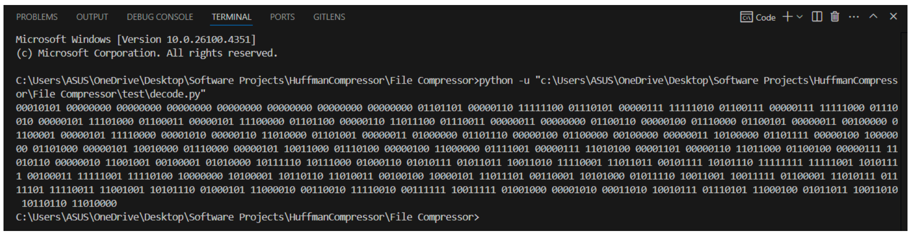
The C++ implementation handles file I/O, Huffman Tree construction, encoding, and decoding. It also writes the Huffman code table into the output binary file.

Binary Analysis Tool: decode.py

The tool includes a Python script `decode.py` for analyzing compressed files by converting bytes in a `.huff` file to binary format.

Listing 1: decode.py - View Binary Content

```python
with open("test/compress.huff", "rb") as f:
    byte = f.read(1)
    while byte:
        print(f'{ord(byte):08b}', end='␣')
        byte = f.read(1)
```



```
Microsoft Windows [Version 10.0.26100.4351]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\Software Projects\HuffmanCompressor\File Compressor>python -u "c:\Users\ASUS\OneDrive\Desktop\Software Projects\HuffmanCompress
or\File Compressor\test\decode.py"
00010101 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01101101 00000110 11111100 01110101 00000111 11111010 01100111 00000111 11111000 01110
010 00000101 11101000 01100011 00000101 11100000 01101100 00000110 11011100 01110011 00000011 00000000 01100110 00000100 01110000 01100101 00000011 00100000 0
1100001 00000101 11110000 00001010 00000110 11010000 01101001 00000011 01000000 01101110 00000100 01100000 00100000 00000011 10100000 01101111 00000100 100000
00 01101000 00000101 10010000 01110000 00000101 10011000 01110100 00000100 11000000 01111001 00000111 11010100 00001101 00000110 11011000 01100100 00000111 11
010110 00000010 11001001 00100001 01010000 10111110 10111000 01000110 01010111 01011011 10011010 11110001 11011011 00101111 10101110 11111111 11111001 1010111
1 00100011 11111001 11110100 10000000 10100001 10110110 11010011 00100100 10000101 11011101 00110001 10101000 01011110 10011001 10011111 01100001 11010111 011
11101 11110011 11001001 10101110 01000101 11000010 00110010 11110010 00111111 10011111 01001000 00001010 00011010 10010111 01110101 11000100 01011011 10011010
 10110110 11010000
C:\Users\ASUS\OneDrive\Desktop\Software Projects\HuffmanCompressor\File Compressor>
```

Figure 2: Viewing Binary of .huff File using decode.py

4 HuffmanWebApp (Flask Web Interface)

Overview

This is a user-friendly web interface built using Flask to wrap the C++ Huffman compressor. It allows users to:

- Upload text files to compress.
- Upload `.huff` files to decompress.
- Download the processed files directly from the browser.

Flask API Functionality

The Flask app exposes a route `/` that supports both GET and POST requests. Upon file upload:

- Files are saved to the `uploads/` folder.
- Depending on the action (compress/decompress), a subprocess runs the C++ executable.
- The resulting file is served back to the user.

Key Flask Code: app.py

Listing 2: Flask Logic for Compression

```python
if action == "compress":
    subprocess.run([compressor_path, "-c", input_path, output_path],
        check=True)
else:
    subprocess.run([compressor_path, "-d", input_path, output_path],
        check=True)
```

Interface Snapshots

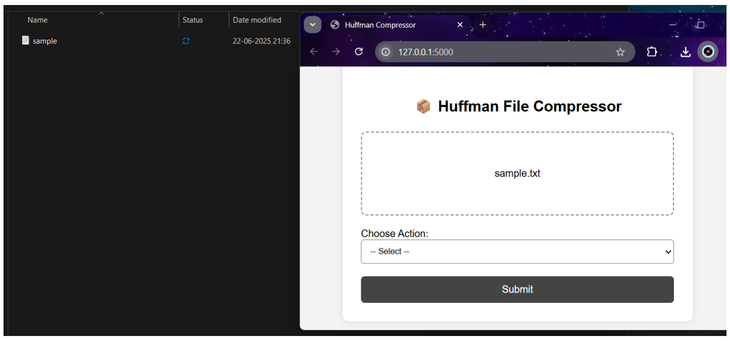


Figure 3: Initial Web Interface

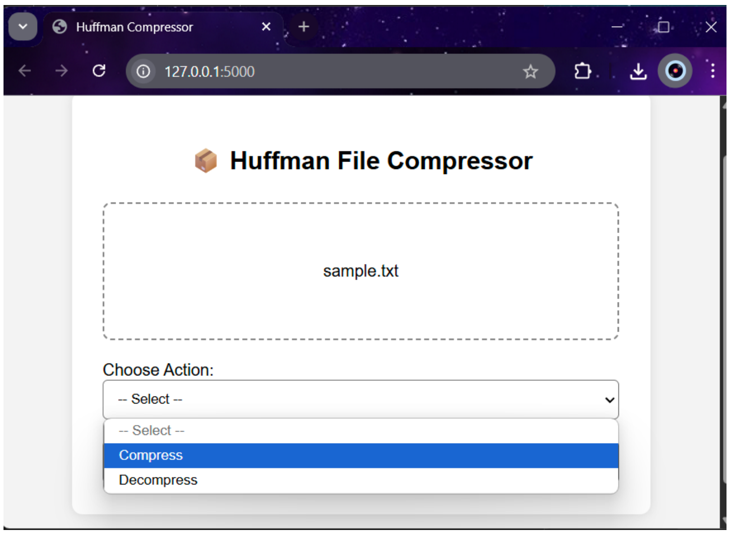


Figure 4: File Upload for Compression

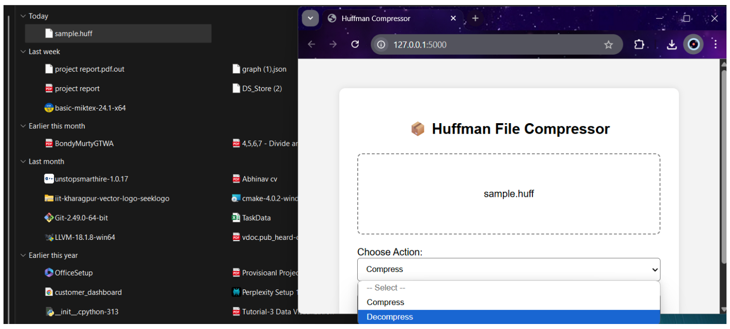


Figure 5: File Upload for Decompression

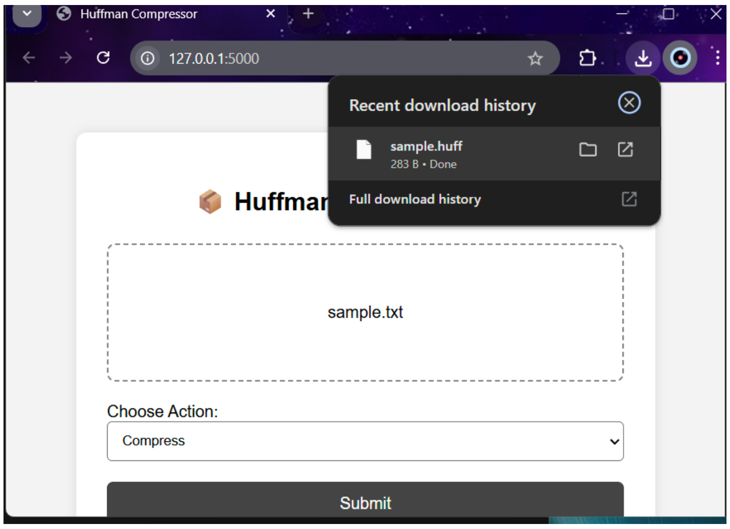


Figure 6: Compression Completed Successfully

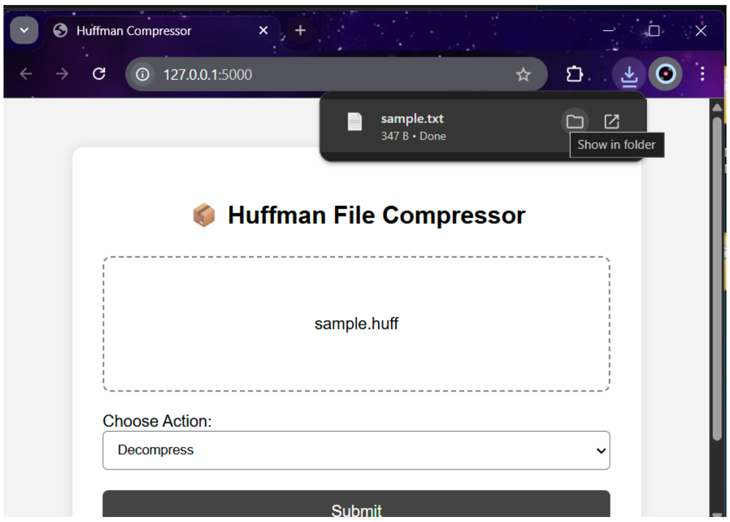


Figure 7: Decompression Completed Successfully

5 Conclusion

This suite provides a full pipeline for Huffman Coding implementation across CLI, GUI, and Web. It highlights both the theory and the software engineering practices involved in making a practical compression system.