Assignment 6 – Huffman Coding

Nithin Duvvuru

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

- Describe the goal of compression. (As a hint, why is it easy to compress the string "aaaaaaaaaaaaaaa")

  Compression is used to reduce data size for storage and efficiency. It uses redundancies in data to represent it compactly. For example, the string "aaaaaaaaaaaaaaa" compresses easily due to its repetitive nature, so storing character counts instead of each individual character will save space and time.

- What is the difference between lossy and lossless compression? What type of compression is huffman coding? What about JPEG? Can you lossily compress a text file?

  Lossless compression retains all original data, ensuring exact reconstruction after decompression. Lossy compression sacrifices data quality to achieve higher compression ratios, losing some information. Huffman coding is lossless, while JPEG uses lossy compression for images. Text files can be lossily compressed, it may result in significant loss of textual data.

- Can your program accept any file as an input? Will compressing a file using Huffman coding make it smaller in every case??

  Yes, the program can accept any file as input. Compressing a file using Huffman coding doesn't guarantee a smaller output in every case since Huffman coding depends on data redundancy. If there's no redundancy in the data, compression may not reduce the file size.

- How big are the patterns found by Huffman Coding? What kind of files does this lend itself to?

  The size of the patterns found depends on the frequency distribution of symbols. Huffman Coding generates shorter codes for frequently occurring symbols and longer codes for

occasional ones. This technique is particularly effective for compressing text files and other data with a non-uniform symbol distribution.

- Take a picture on your phone. What resolution is the picture? How much space does it take up in your storage (in bytes)?

  Photo is 3024 x 4032, takes 1.8 MB.

- If each pixel takes 3 bytes, how many bytes would you expect the picture you took to take up? Why do you think that the image you took is smaller?

  Expected amount of bytes would be 36.6 MB, but since iPhones use HEIC (High Efficiency Image File Format) which uses lossless compression the actual storage space is much less.

- What is the compression ratio of the picture you just took? To get this, divide the actual size of the image by the expected size from the question above. You should not get a number above 1.

  5 percent, 0.04918

- Do you expect to be able to compress the image you just took a second time with your Huffman program? Why or why not?

  Compressing an image that has already been compressed will most likely not result in file size reduction because the image data has already been optimized for compression during the initial compression process, which leaves no room for further compression.

- Are there multiple ways to achieve the same smallest file size? Explain why this might be.

  There are multiple ways to achieve the same smallest file size, especially in lossless compression algorithms like Huffman coding. This is because Huffman coding relies on the frequency of symbols in the data to assign variable length codes. Different arrangements of these codes can produce the same smallest compressed file size.

- When traversing the code tree, is it possible for a internal node to have a symbol?

  No, internal nodes do not have symbols associated with them.

- Why do we bother creating a histogram instead of randomly assigning a tree.

  Creating a histogram allows us to analyze the frequency of each symbol in the data which allows us to prioritize the frequently occurring symbols.

- Relate this Huffman coding to Morse code. Why does Morse code not work as a way of writing text files as binary? What if we created more symbols for things like spaces and newlines?

  Morse code is not suitable for encoding text files as binary directly because it lacks a way to represent spaces, newlines, and other characters found in text files. However, if we extend Morse code to include symbols for spaces, newlines, and other characters that its missing, it could potentially be used for text file compression, although it would likely be less efficient than Huffman coding because its encodes using a fixed length for each character.

- Using the example binary, calculate the compression ratio of a large text of your choosing

  30,335 bytes / 15,000 bytes = Compression Ratio of 2.02

Testing

List what you will do to test your code. Make sure this is comprehensive. [1] Be sure to test inputs with delays and a wide range of files/characters.

To test I will use all the given tests to make sure my program works appropriately.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{ lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[2]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[1][2][3].

Run the program and provide an input file as the argument. Can use -i to read input file given as argument, and -o to write to output file given as argument. -h prints a help message.

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

The program uses node structures to construct Huffman trees, where each node represents a symbol and its frequency. A priority queue manages the nodes during tree construction. The generation of a code table enables the mapping of symbols to Huffman codes. During compression, input data is encoded using Huffman codes and written to an output file. While decompression involves decoding the compressed data using the Huffman tree structure. These organized components ensure effective compression and decompression processes.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

[1] This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

[2] This is my footnote.

The code will start by parsing command line arguments, and validating inputs (-i, -o). It will support compression and decompression modes. Key features include histogram creation, Huffman tree construction, code table generation, and file processing. Each function will handle specific tasks: fill_histogram() counts symbol frequencies, create_tree() constructs the Huffman tree, fill_code_table() generates the code table, and huff_compress_file() performs compression. For decompression, dehuff_decompress_file() reads and reconstructs the tree.

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

```
Function: main
Outputs: exit status and prints errors to stderr
Purpose: Calls necessary functions to find optimal path in graph.

HUFF:

Function: huff_compress_file(outbuf, fin, filesize, num_leaves, code_tree, code_table)
Inputs: outbuf for writing compressed data, fin for reading input file, filesize for file size,
    num_leaves for the number of leaf nodes, code_tree for Huffman tree, code_table for Huffman
    codes.
Purpose: Compresses the input file using Huffman coding and writes the compressed data to the
    output buffer.

DEHUFF:

Function: dehuff_decompress_file(fout, inbuf)
Inputs: fout for writing decompressed data, inbuf for reading compressed data.
Purpose: Decompresses the input file using the Huffman tree and writes the decompressed data to the
     output file.
```

Results

Follow the instructions on the pdf to do this. In overleaf, you can drag an image straight into your source code to upload it. You can also look at `https://www.overleaf.com/learn/latex/Inserting_Images`

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.