Lempel-Ziv Compression

Hsiang Yun Lu

March 13, 2023

1 Functionality of LZ78 Compression

The LZ78 is a dictionary-based compression algorithm with an explicit dictionary compared to the LZ77 approach. The two elements in the compressed output are an index referring to the matching dictionary entry and the first non-matching symbol. When the symbol is not yet found in the dictionary, the word is also added to the dictionary. By constructing the dictionary using a Trie, the LZ78 algorithm can capture and hold patterns indefinitely. While the compression process of LZ77 is time-consuming due to many comparisons done to find a matched pattern, LZ78 is faster than LZ77 because it reduces the number of string comparisons in each encoding step.

2 Endianness

While implementing this assignment, I figured out the concept of endianness and how to check it on my computer. Big-endian means reading the biggest byte as the first byte, while little-endian stores the little end first, which means the first byte (or the lowest memory address) read is the littlest. Though the endianness-related functions and header file were given, it still took me some time to understand the concept before implementing them in the I/O module.

3 Bitwise Operation for Reading And Writing Pairs

To me, the most complicated part of this program should be the `read_pair` and `write_pair` functions in the `io.c`. They took me the longest time to debug. During the implementation, I gained more experience managing bit masks and all bitwise operations. Things got more complicated when we had to consider reading or writing a pair cross more than one byte. Drawings of the different situations of reading/writing bits did help me a lot in building the structure of the functions. I also realized the importance of counters. I set some useless counters at first, which messed up my design concept.

4 Buffer

Along with the bitwise operation, using a buffer made the I/O more efficient and added some challenges to implementation. In the beginning, I messed up the `read_pair` and `write_pair` functions by getting the incorrect calculation of global counters and buffer indices, so my buffer didn't flush out completely. I also learned different ways of writing or copying data between buffers, like the `memcpy()` and `memset()`.

5 I/O System Call

Before this assignment, I was not so familiar with the input-output system calls in `C`. I got a "Permission denied" error when first trying to use `open()` to get a file descriptor for outfile. I learned that `O_CREAT` flag might not be used alone and the permission mode setting for creating a file. The usage of `read()` and `write()` looks simple, but I got stuck in writing `read_bytes()` and `write_bytes()` for a while because I misinterpreted the return value and the calculation of the bytes to be read in/ written out. It was a good thing that I learned again to closely read any manual first.

6 Reference

- Huffman and Lempel-Ziv-Welch Compression
- LZ77 and LZ78
- Concept of Endianness