Assignment 6: WRITEUP.pdf

Katrina VanArsdale

March 12, 2023

1 Description

This program provides an encode program and a decode program. The encode program performs LZ78 compression on a given file while the decode program performs LZ78 decompression.

2 Lessons learned and Thoughts

While working on this assignment I learned a lot about Tries and structures. I have a better understanding of how I should be allocating memory and how arrays work within structures. What we did for the Game of Life was pretty similar in terms of structures but with some differences that I had to better understand. I had a little trouble with the difference between the trie_node_create function and the trie_create function. At first, I basically had the same code for each and then called node create in trie create to make the EMPTY_CODE root but then I was having memory leaks. Then I realized that I shouldn't be allocating memory for another trie in trie_create and I could just call trie_node_create with EMPTY_CODE.

When I was trying to get encode to work I had a problem with it writing way too many lines. I realized I should've had a global variable for my buffer indexes so I implemented that and replaced the original static variables I had within functions. Then I had to use the global index in flush_pairs and flush_words because what was going on was that it was writing the whole block into the outfile. I also had a problem with the read_bytes buffer when using stdin because in this case read() isn't able to read it all at once so it gets called over and over again. But when it reads into the buffer it starts from 0 every time it's called so it was overriding the bytes already in the buffer. To fix this I added the total bytes read to the buffer when reading in a loop.

I had a problem with write_pair because when I tried to encode big files after encoding 4KB it would start encoding incorrectly. When looking at the hexdump of the outfile the outputs gradually contained more 'f's. At first, I thought it was a problem with doing '% 8' on the index of the buffer because code can be 16 bits so I changed the 8's to 16's but that caused more problems so I undid it. Then I realized that because the way write_pair works is with an or statement in the buffer so it was checking for 1s or 0s and I wasn't setting the buffer to 0 after filling it so there were leftover 1s which caused the new codes to be set to 1 when they should be 0. Once I figured that out all I had to do was use memset to set the whole buffer to 0 once it was filled.

Another issue I ran into was trying to get the statistics correct for encode and decode. Originally I just added to the total bits and syms wherever I saw either involved but I was getting incorrect stats. For encode, it was a pretty easy fix because the stats for the compressed file kept coming out as less than they should be. I realized I wasn't counting the bits from the header so I fixed that and it worked! But then with decode the uncompressed file size was sometimes more than it should be and other times it was more than it should be. Originally I was adding to total syms inside read_pair but then I took a closer look at what decode was calling and decided to add to total syms in write_word instead and that fixed it.

3 Conclusion

Overall, I have learned a lot about buffers and how to use them through trial and error and I have a better picture of how structures like Tries work. Understanding how the functions all worked together took a while but it was

especially important during the debugging phase to figure out where the problems were coming from. Throughout this class, I've gotten a lot more comfortable figuring out the problems in my code and looking through all my resources to get where I need to be. The assignment docs, google, lectures, and discord have all been incredibly helpful with my understanding of the code and pointing me in the right direction.