Sushruti Bansod

CS 1501: Algorithm Implementation

11/6/20

[SDB88@pitt.edu](mailto:SDB88@pitt.edu)

Assignment 3: Lempel-Ziv-Welch Compression

This assignment required me to understand the compression techniques presented by the LZW algorithm. The objective was to update the given LZW file and improve its implementation and performance in multiple ways. Although the actual compression algorithm was relatively straightforward, this project proved many difficulties. The statement which was made in class that the there would only be an addition of a few lines of code was a bit intimidating but pushed me to complete the tasks in the most efficient way possible. Through this discussion I will elaborate on milestones and conclude with an analysis of the compression ratios.

I chose to visualize the goals as 2 different aspects, compression, and expansion. Compression was fairly simple, a nested loop structure allowed me to implement reading the codewords ‘byte-by-byte’. Basically, appended the characters until the prefix length was different than the length of the character. This allowed me to be sure that the last characters to be appended, has not been added to the symbol table yet. Using this mindset, I incremented the codeword process. This incrementing method also made is easy to expand the codeword size when the maximum had been reached. Although this took me some time to figure out, it was fairly simple to implement.

The expansion portion proved to be more difficult. It was complicated to figure out the syncing of when to add. I tried many ways to implement it and it took me a long time to figure out something that worked. The roadblock I faced was the files getting corrupted very easily. The corruption could be easily seen in the image files. Some of the corrupted files would either not open or look completely different than the original, with different colors and missing pixels.

Since the objective was to improve efficiency, I want to analyze compression rates. The worst compression occurred with the image files. The unix compression wasn’t able to actually compress the image and the ratio was 1/1. The frosty.jpg file actually compressed well, with a improvement of 0.7/1. The other compression which was not terrible were the text and archive files. They provided around a 2.17 with LZW and LZW mod. This reduced the file by almost 50%. Similarly the text files gave a moderately good compression rate as well. Lastly, the best compression was seen with the bmp files. The best being the wacky.bpm file. The 900kb file was able to be compressed to a 4kb file size which is over 225/1 compression ratio.