Drew Kozak

Professor Ramirez

CS1501

6/29/18

Project 3 Write Up: LZW

The purpose of project 3 was to modify the LZW algorithm that came from the authors code. The project focused around allowing variable size code words of different length, or bits to be implemented, such as 9 bits and 16 bits long. The syncing of reading in the bytes for the compress and expand methods in the algorithm was the main task at hand that gave some trouble. It is really important to understand how to convert between the two so that the file can be compressed and decompressed properly and not lose any data.

The compress method utilized a while loop that allowed for appending characters using the StringBuilder data structure. The prefix length would append until it was a different length than the characters, which meant that the last character appending the codeword at hand had not been added to the symbol table yet. Incrementing the codeword in this manner created an easy process to follow that would allow the codewords to expand up to the size of the max bit width, in this case 2^bit width. The StringBuilder made appending this work in O(1) time, which was much faster than the authors code which utilized just a string data structure, that would create new strings each time a character was added to the code word. We learned reading in a file as a stream input text is much faster than reading it in as a string.

The expand method was trickier to get a grasp of. It would be extremely easy to corrupt a file if the bits were not synced and added correctly. In archived files, the file may compress properly but it was easy to lose information and thus these files were also corrupted. Corrupt jpg files were a nightmare at first, since the file would not even open if the ID bits were changed in decompression. Other common problems with the jpg were a difference in color of the picture and the picture may not look exactly the same coming out of decompression.

The worst compression ratios were the GIFs and JPG files, UNIX compression didn’t compress it at all, it stayed at a 1:1 ratio for the frosty.jpg file. LZW original was a .71:1 compression ratio, LZWmod reset was .77:1 and LZWmod no reset was .733:1. I would attribute the expanding size to the entropy associated with image files like JPG and GIFs. Mid tier compression ratios were displayed in the .txt and .tar files. Essentially the UNIX compress, and all forms of LZW were roughly the same, however UNIX was slightly ahead by a few hundredths places. High entropy exists in .tar files that have .txt files inside them which attributes to this compression, and the text files compressed very nice. The best compression were in the .bmp files, which have the greatest entropy of any of the files given. Wacky.bmp compressed by 225:1 ratio which surprised me a lot, but that can be attributed to high entropy levels.