Jinlang Wang

CS 1501

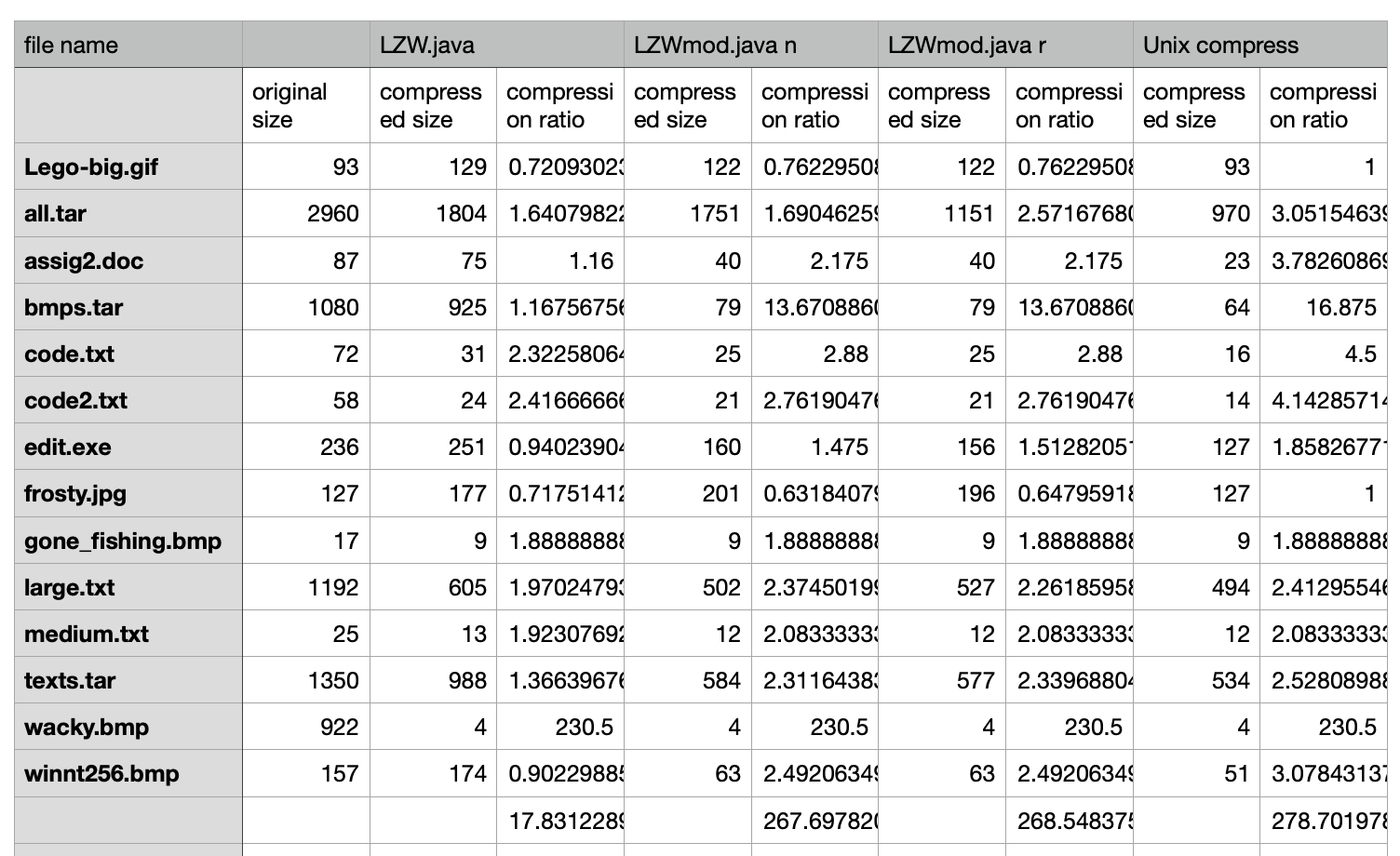
Dr.Ramirez

June 17th, 2020

Project 3 Write Up

The goal for project 3 was to modify the author’s version of LZW.java, which will provide a better compression ratio and a faster run time. In this project, I compared four different compression algorithms with 14 different files. The following is the analysis.

The following data’s sizes are measured in kilobytes. Compression ratio is calculated with this formula: original size/compressed size.



The first algorithm is Robert Sedgewick’s own LZW version. This performed very well, and had an average compression ratio of 17.83. Based on this, I improved the over author version. LZWmod had an average compression ratio of 268, which had a better average compression than the original. LZWmode n would not reset the dictionary when the dictionary is full, while LZWmod r will reset the whole dictionary when the dictionary is full. The performance is slightly better for LZW mod r when we compress large file, such as all.tar. This is because LZWmode used codewords from 9 bits to 16 bits, which allow more spaces for codewords. Thus, the dictionary never full, and will not be resetted. But we will still assume that LZWmod r would have better compression rate when we have files that have different patterns, such as a .tar file. This is because different files have different patterns. If we delete the previous codewords from the dictionary and memorize new codewords, new patterns will be compressed pretty well. This further explains why the compression ratio for .txt files are better than others. Since files such as .gif, .bmp files have unpredictable patterns, which is less likely to reuse the codewords.

It seems that with .jpg and .gif files, the UNIX compression’s compression ratios are the worst. There is no compression at all. LZW and LZWmod even expanded the frosty.jpg. This is no doubt due to the entropy with image files. These image files most likely have high data entropy which results in extremely poor compression ratios. But there are some exception. Some files, such as wacky.bmp, compressed pretty well. There might exists a convenient way to encode, which allows for long patterns. The LZW has a better compression ratio than LZWmod. I think this might because using a 12 bits codeword might be more optimal. However, the Unix compress has the same compression ratio with LZW. I think the Unix compress is more clever, and it knows when and how to set the codeword length in some special circumstance.

In those four versions, I found that .txt files compressed best. Even with LZW, .txt have an average compression ratio greater than 2. Therefore, all .txt files have better compression ratios regardless which method was used. This might because .txt files can reuse codewords betters. Since .txt files is composed with English words, which are composed with 26 letters. There might be a lot of repeated string patterns, which can be reused.