Sameera Boppana

Dr. Ramirez

CS 1501

26 June 2020

LZW Comparison

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | All.tar | Asig2.doc | Bmps.tar | Code.txt | Code2.txt | Edit.exe | Frosty.jpg | Texts.tar |
| File Size | 3,031,040 | 87,040 | 1,105,920 | 72,351 | 57,701 | 236,328 | 126,748 | 1,382,400 |
| LZW | 1,846,854  Ratio: 1.6412 | 74,574  Ratio:  1.1672 | 925,079  Ratio:  1.1955 | 30,980  Ratio:  2.3354 | 24,138  Ratio:  2.3905 | 250,742  Ratio:  0.9425 | 177,453  Ratio:  0.7142 | 1,012,179  Ratio:  1.3658 |
| LZWmod w/o reset | 1,792,782 – ratio: 1.6909 | 40,041  Ratio: 2.1738 | 80,914  Ratio:  13.6678 | 24,545  Ratio:  2.9477 | 20517  Ratio:  2.8124 | 156,410  Ratio:  1.5110 | 163,790  Ratio:  0.7738 | 597,848  Ratio:  2.3123 |
| LZWmod w/ reset | 1,178,220 – ratio: 2.5726 | 40,041  Ratio:  2.1738 | 80,914  Ratio:  13.6678 | 24,545  Ratio:  2.9477 | 20517  Ratio:  2.8124 | 152,231  Ratio:  1.5524 | 171,171  Ratio:  0.7405 | 590,490  Ratio:  2.3411 |
| Unix compress | 1,179,467 – ratio: 2.5698 | 40,040  Ratio:  2.1738 | 80,913  Ratio:  13.6678 | 24,545  Ratio:  2.9477 | 20516  Ratio:  2.8124 | 151,111  Ratio:  1.5639 | 127,341  Ratio:  0.9953 | 589,697  Ratio:  2.3443 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Wacky.bmp | Winnt256.bmp | Gone\_fishing.bmp | Medium.txt | Large.txt | Lego-big.gif |
| File Size | 921,654 | 157,044 | 17,336 | 25,407 | 1,220,703 | 93,371 |
| LZW | 4,302  Ratio:  214.239 | 159,050  Ratio:  0.9874 | 9,278  Ratio:  1.8685 | 13,197  Ratio:  1.9252 | 605,184  Ratio:  2.0171 | 128,973  Ratio:  0.7240 |
| LZWmod w/o reset | 3,952  Ratio:  233.2120 | 62,932  Ratio:  2.4955 | 8,964  Ratio:  1.9340 | 12,532  Ratio:  2.0274 | 501,778  Ratio:  2.4328 | 122,494  Ratio:  0.7622 |
| LZWmod w/ reset | 3,952  Ratio:  233.2120 | 62,932  Ratio:  2.4955 | 8,964  Ratio:  1.9340 | 12,532  Ratio:  2.0274 | 527,539  Ratio:  2.3140 | 122,494  Ratio:  0.7622 |
| Unix compress | 3,952  Ratio:  233.2120 | 62,931  Ratio:  2.4955 | 8,964  Ratio:  1.9340 | 12,531  Ratio:  2.0274 | 522,673  Ratio:  2.3355 | 93,204  Ratio:  1.0018 |

In general, the original LZW program had the least amount of compression and therefore, the lowest compression ratio. The compression ratio is a measurement of how much the file compresses. All of the files had differences when it came to the original LZW program and the modified LZW. This is due to the way that the actual compression is done. Because LZW.java reads in the entire file as a string, which causes inefficiency during compression. Furthermore, the fixed codeword size of 12 is relatively small, especially for large files, such as large.txt or any of the tar files. Because of this, there are only 4096 possible codewords, and when all codewords are used, no new patterns are added. This causes most of the file to not be able to be compressed. This partly explains why the LZWmod.java is able to get better compression. This is program has the ability to increment the codeword size as soon as all possible codewords are used, causing for more compression. However, depending on the size of the file, the reset feature may or may not actually increase the compression ratio. For example, large.txt, we see that the compression ratio for LZW.java is 2.0171. Then the same file, with LZWmod.java and no reset, has a ratio of 2.4328 and a ratio of 2.3140 with the reset. When no reset is implemented, there is a higher compression ratio because the reset will only give better compression only once new patterns have been found. Therefore, if there are not many new patterns, running it without the reset is better. But for all.tar, we see that the reset has a ratio of 2.5726 while without the reset has 1.6909. Because there are many different types of files, there are going to be very different patterns, so the reset feature will allow there to be more compression. In most cases, the with and without the reset produced the same compression ratio (gone\_fishing.bmp has a ratio of 1.9340) and this is due to the fact that the file did not produce enough codewords to use up the max number of codewords.

For all four algorithms, the best compression is seen with wacky.bmp. This is because the file has a lot of white space so there will be a lot of repeat codes, allowing for compression to be even greater. In terms of worst compression, both frosty.jpg and Lego-big.gif gave really bad compression ratios for all four algorithms. In fact, in both cases, all algorithms caused an increase in bytes in the compressed version. This might be because there is a very low compression ratio because there is a low chance that there will be many repeat patterns. Because they are images, there codewords depict certain portions of the image so unless there is that exact same pattern read in with the exact same codeword length, they will not be translated to the same codeword. Therefore, there probably are not that many repeated codewords, causing the compression, so even if the dictionary is reset, there will not be any compression.