Out of the four different implementations of LZW Compression(12-bit codewords, resizable codewords(9-16), resizable codewords with reset, and the Unix compress), I discovered that the compression algorithms which performed the best with the highest compression ratios were dictionary reset and Unix compression.

This can be observed in the table below where it can be seen that for almost every file, the compression ratios of the Dictionary reset and Unix compression types have a higher compression ratio than the other two types. Furthermore, it does make sense that the dictionary reset would perform well because the dictionary reset compression algorithm makes use of flushIfFull. flushIfFull is a flag used in this algorithm that completely empties out and resets the codebook once the codebook has been filled to its max capacity. By resetting the codebook when full, more room is then created for different codewords with different patterns to fill up the codebook which allows for better compression allowing for new, additional codewords can now become compressed. In addition to the flushIfFull flag, this algorithm also allows for variable size codewords(9-16). This means that the codeword size always starts at 9 bits. However, if during the process of compressing a file, the algorithm finds that a larger codebook is needed, it will increment the codeword size to be able to account for a larger frequency in the codeblocks until reaching the max size of 16. This flexibility improves compression efficiency by making the algorithm adaptable to increased frequencies in the codeblock. The Unix compression works well for similar reasons as it is the most efficient compression algorithm. Both of these algorithms work well because they are able to adapt to differing code block frequencies.

The compression algorithm involving the 9-16 sized codebooks with no reset does not work efficiently because with that compression algorithm, once the codebook is filled, it is not able to be emptied out and reset. Furthermore, this means that any new codeblocks the algorithm comes across will not be able to be properly compressed as there won’t be any more room in the codebook to create codewords for those codeblocks. Similarly, the 12 bit codeword size compression algorithm does not perform well because the size of each codeword and thus the size of the codebook is always static. It can never be increased if more room is needed to allow for new codeblocks to be encoded. This makes the algorithm very rigid and unable to adapt to an increase in code block frequencies or to different patterns in the data.

As can be observed by the table, for the (9,16) algorithm without dictionary reset and 12 bit fixed codeword size algorithms, the all.tar file performed the worst for the followed by frosty.jpg and Lego-big.gif. These files most likely performed very poorly because they had large frequency codeblocks, resulting in not enough codewords being available to properly compress the files. And because the codebook fills up faster, the algorithm may continue the compression even after the codebook has already filled up which may result in the compressed file being larger than the original file in some cases. Similarly, for the (9,16) algorithm with the dictionary reset and Unix compression algorithms, the all.tar file and Lego-big.gif files performed poorly. They most likely performed poorly because of the same reason where there weren’t many patterns in the data, resulting in smaller sized codeblocks to be encoded.

For all algorithm types, the compression of wacky.bmp was significantly more efficient than the compression of any of the other files. I think this may be the case because the file may have large amounts of patterns which would have led to better compression because larger codeblocks could be compressed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File name | Original Size | Test type | Compressed size | Compression ratio |
| wacky.bmp | 900 | (9-16)-Do  nothing | 3.951 | 227.790433 |
| wacky.bmp | 900 | (9-16)-Reset | 4 | 225 |
| wacky.bmp | 900 | 12-bit | 5 | 180 |
| wacky.bmp | 900 | Unix | 4 | 225 |
| winnt256.bmp | 153 | (9-16)-Do  nothing | 62.931 | 2.43123421 |
| winnt256.bmp | 153 | (9-16)-Reset | 62 | 2.46774194 |
| winnt256.bmp | 153 | 12-bit | 157 | 0.97452229 |
| winnt256.bmp | 153 | Unix | 62 | 2.46774194 |
| gone\_fishing.bmp | 16.9 | (9-16)-Do  nothing | 8.962 | 1.88573979 |
| gone\_fishing.bmp | 16.9 | (9-16)-Reset | 9 | 1.87777778 |
| gone\_fishing.bmp | 16.9 | 12-bit | 11 | 1.53636364 |
| gone\_fishing.bmp | 16.9 | Unix | 9 | 1.87777778 |
| frosty.jpg | 123 | (9-16)-Do  nothing | 163.789 | 0.75096618 |
| frosty.jpg | 123 | (9-16)-Reset | 40 | 3.075 |
| frosty.jpg | 123 | 12-bit | 174 | 0.70689655 |
| frosty.jpg | 123 | Unix | 40 | 3.075 |
| Lego-big.gif | 91.1 | (9-16)-Do  nothing | 122.493 | 0.74371597 |
| Lego-big.gif | 91.1 | (9-16)-Reset | 120 | 0.75916667 |
| Lego-big.gif | 91.1 | 12-bit | 126 | 0.72301587 |
| Lego-big.gif | 91.1 | Unix | 120 | 0.75916667 |
| medium.txt | 24 | (9-16)-Do  nothing | 12.439 | 1.92941555 |
| medium.txt | 24 | (9-16)-Reset | 13 | 1.84615385 |
| medium.txt | 24 | 12-bit | 13 | 1.84615385 |
| medium.txt | 24 | Unix | 13 | 1.84615385 |
| code.txt | 67.8 | (9-16)-Do  nothing | 24.29 | 2.79127213 |
| code.txt | 67.8 | (9-16)-Reset | 24 | 2.825 |
| code.txt | 67.8 | 12-bit | 30 | 2.26 |
| code.txt | 67.8 | Unix | 78 | 0.86923077 |
| code2.txt | 53.8 | (9-16)-Do  nothing | 20.318 | 2.64789842 |
| code2.txt | 53.8 | (9-16)-Reset | 20.318 | 2.64789842 |
| code2.txt | 53.8 | 12-bit | 23 | 2.33913043 |
| code2.txt | 53.8 | Unix | 20 | 2.69 |
| all.tar | 2.89 | (9-16)-Reset | 1178.21 | 0.00245287 |
| all.tar | 2.89 | (9-16)-Do  nothing | 1792.78 | 0.00161202 |
| all.tar | 2.89 | 12-bit | 2,007 | 0.00143996 |
| all.tar | 2.89 | Unix | 1170 | 0.00247009 |
| large.txt | 1174 | (9-16)-Reset | 523.714 | 2.24168153 |
| large.txt | 1174 | (9-16)-Do  nothing | 497.971 | 2.35756701 |
| large.txt | 1174 | 12-bit | 580 | 2.02413793 |
| large.txt | 1174 | Unix | 510 | 2.30196078 |
| bmps.tar | 1,050 | (9-16)-Reset | 80.913 | 12.9769011 |
| bmps.tar | 1,050 | (9-16)-Do  nothing | 84 | 12.5 |
| bmps.tar | 1,050 | 12-bit | 900 | 1.16666667 |
| bmps.tar | 1,050 | Unix | 80 | 13.125 |
| texts.tar | 1,310 | (9-16)-Reset | 590.584 | 2.2181434 |
| texts.tar | 1,310 | (9-16)-Do  nothing | 597.847 | 2.19119608 |
| texts.tar | 1,310 | 12-bit | 986 | 1.32860041 |
| texts.tar | 1,310 | Unix | 581 | 2.25473322 |