|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Runtime to Compress (seconds)** | | | | |
| **File** | **LZW** | **Mod -n** | **Mod -r** | **Compress.exe** |
| All.tar | 2809 | 9.83598 | 29.72625 | 0.16 |
| Assig2.doc | 1.08 | 0.41533 | 0.41129 | 0.04 |
| Bmps.tar | 235 | 22.80613 | 22.42693 | 0.06 |
| Code.txt | 0.46 | 0.28421 | 0.29220 | 0.04 |
| Code2.txt | 0.34 | 0.26319 | 0.25317 | 0.04 |
| Edit.exe | 8.66 | 1.25493 | 1.27494 | 0.07 |
| Frosty.jpg | 3.57 | 1.40098 | 1.54408 | ---------- |
| Gone\_fishing.bmp | 0.15 | 0.19814 | 0.22515 | 0.04 |
| Large.txt | 202 | 3.28959 | 4.48817 | 0.1 |
| Lego-big.gif | 2.02 | 1.32394 | 1.15982 | ---------- |
| Medium.txt | 0.19 | 0.19314 | 0.19314 | 0.04 |
| Texts.tar | 405 | 3.76766 | 5.03869 | 0.13 |
| Wacky.bmp | 1.03 | 19.53921 | 20.86679 | 0.04 |
| Winnt256.bmp | 3.5 | 1.00871 | 0.92066 | 0.05 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Runtime to Decompress (seconds)** | | | | |
| **File** | **LZW** | **Mod -n** | **Mod -r** | **Compress.exe** |
| All.tar | 0.32 | 0.35025 | 0.47834 | 0.09 |
| Assig2.doc | 0.11 | 0.10909 | 0.10711 | 0.04 |
| Bmps.tar | 0.18 | 0.15409 | 0.15311 | 0.05 |
| Code.txt | 0.1 | 0.12014 | 0.11709 | 0.04 |
| Code2.txt | 0.11 | 0.10709 | 0.10409 | 0.04 |
| Edit.exe | 0.12 | 0.14411 | 0.14411 | 0.05 |
| Frosty.jpg | 0.12 | 0.13810 | 0.14016 | ----------- |
| Gone\_fishing.bmp | 0.1 | 0.10009 | 0.09807 | 0.03 |
| Large.txt | 0.19 | 0.23016 | 0.29220 | 0.06 |
| Lego-big.gif | 0.11 | 0.15311 | 0.13109 | ---------- |
| Medium.txt | 0.1 | 0.10407 | 0.10809 | 0.06 |
| Texts.tar | 0.22 | 0.22018 | 0.31522 | 0.08 |
| Wacky.bmp | 0.13 | 0.13110 | 0.12809 | 0.05 |
| Winnt256.bmp | 0.11 | 0.13810 | 0.12108 | 0.05 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Compression Ratio (original size / new size)** | | | | |
| **File** | **LZW** | **Mod -n** | **Mod -r** | **Compress.exe** |
| All.tar | 1.639344 | 1.666667 | 2.564103 | 2.564103 |
| Assig2.doc | 1.162791 | 2.173913 | 2.173913 | 2.173913 |
| Bmps.tar | 1.190476 | 12.5 | 12.5 | 14.28571 |
| Code.txt | 2.325581 | 2.857143 | 2.857143 | 2.941176 |
| Code2.txt | 2.380952 | 2.702703 | 2.702703 | 2.777778 |
| Edit.exe | 0.943396 | 1.515152 | 1.538462 | 1.5625 |
| Frosty.jpg | 0.714286 | 0.769231 | 0.740741 | --------- |
| Gone\_fishing.bmp | 1.851852 | 1.923077 | 1.923077 | 1.923077 |
| Large.txt | 2 | 2.380952 | 2.272727 | 2.325581 |
| Lego-big.gif | 0.724638 | 0.763359 | 0.763359 | --------- |
| Medium.txt | 1.923077 | 1.960784 | 1.960784 | 2 |
| Texts.tar | 1.369863 | 9.090909 | 2.325581 | 2.439024 |
| Wacky.bmp | 250 | 250 | 250 | 250 |
| Winnt256.bmp | 99.0099 | 2.5 | 2.5 | 2.5 |

* Note: Compress.exe refused to work on Frosty and Lego-big. I believe this is because their formats mean they were already more or less maximally compressed.
* Note II: I originally calculated the Compression ratio in the inverse way, and recorded them as percentages. As such, the data in each entry is now (prev / 100)^-1

Analysis portion: As expected, Compress.exe was more that obviously the leader. It was faster, more efficient, and made better on-average decisions than my program could have. I attribute this to the fact that it is the de-facto compression algorithm of linux, and has thus received the benefit of scrutiny from the sharpest minds on the internet and around the globe. I of course don’t know the exact implementation, but I expect it would utilize the same LZW algorithm I used in a slightly more elegant way, as well as sensing other parameters about the input files to decide whether or not to use the table dumping or not (e.g. whether to use -n or -r). Indeed, in some cases, Frosty and Lego-big, it elected not to run at all!

With regards to differences in -r and -n: Performance differences were nearly negligible in the average case. However, two files demonstrate how the two methods handle information differently. -r works especially well when you have new information that repeats for a little while, and then stops repeating (e.g. videos). This was shown, unsurprisingly, to be the case in the All.tar file. In that file, lots of other files were just jammed up against each other. Because files tend to have lots of repeats, but groups of files less so, -r was the obvious choice.

-n, on the other hand, came into its own when it was handed Texts.tar to work on. Rather than a whole bunch of different files, this was just one block of very similar files (all texts). We picked up the patterns in the file very early on, and munch our way through like an all you can eat buffet. The compression gained was demonstrably much better than -r for that reason, that the file repeated itself predictably throughout and changed little.

As for the author’s LZW… Well, it tried its best: It was much, much slower. It used piping to get its input, and piping to the output (which I really didn’t appreciate) for printing to the file. And perhaps worst of all, it wasn’t even trading time for efficacy, it was both slow and ineffective. Admittedly, it was never meant to be used in a real setting where fast or useful compression is needed. Its purpose is solely to instruct the student as to how LZW works, and in that way it fulfilled its purpose. However, I’d still never use it to compress anything larger than a small travel booklet.