Professor

Alberto Silitti

TA

Hamna Aslam

Project N°2

Information Theory

BS4-2

Yaroslav Lantsov, Ekaterina Baba, Alexander Karavaer, Danila Romanov, Rami Al-Naim, Oleg Souzdalev

INTRODUCTION

This project aims at understanding the role and importance of both compression and encoding algorithms through the implementation of a messenger application.

Compression algorithms are used to reduce the size of a file to a minimum in order to increase efficiency in the channel, whereas Coding algorithms are used to protect the sent information from noise that is encountered by the message in the channel.

The following algorithms were chosen for this project :

* Compression LZ77, LZW and Huffman
* Coding Repetion, SF and Hamming

The tasks were distributed as follows :

1. Algorithms implementation
   1. Alexander Karavaev
   2. Rami Al-Naim
   3. Ekaterina Baba
   4. Oleg Souzdalev
2. Server
   1. Danila Romanov
3. Interface
   1. Danila Romanov
   2. Yaroslav Lantsov
4. Report
   1. Yaroslav Lantsov
   2. Rami Al-Naim
   3. Alexander Karavaev
   4. Oleg Souzdalev

Statistics

## All time presented in seconds and sizes in bytes.

Table 1

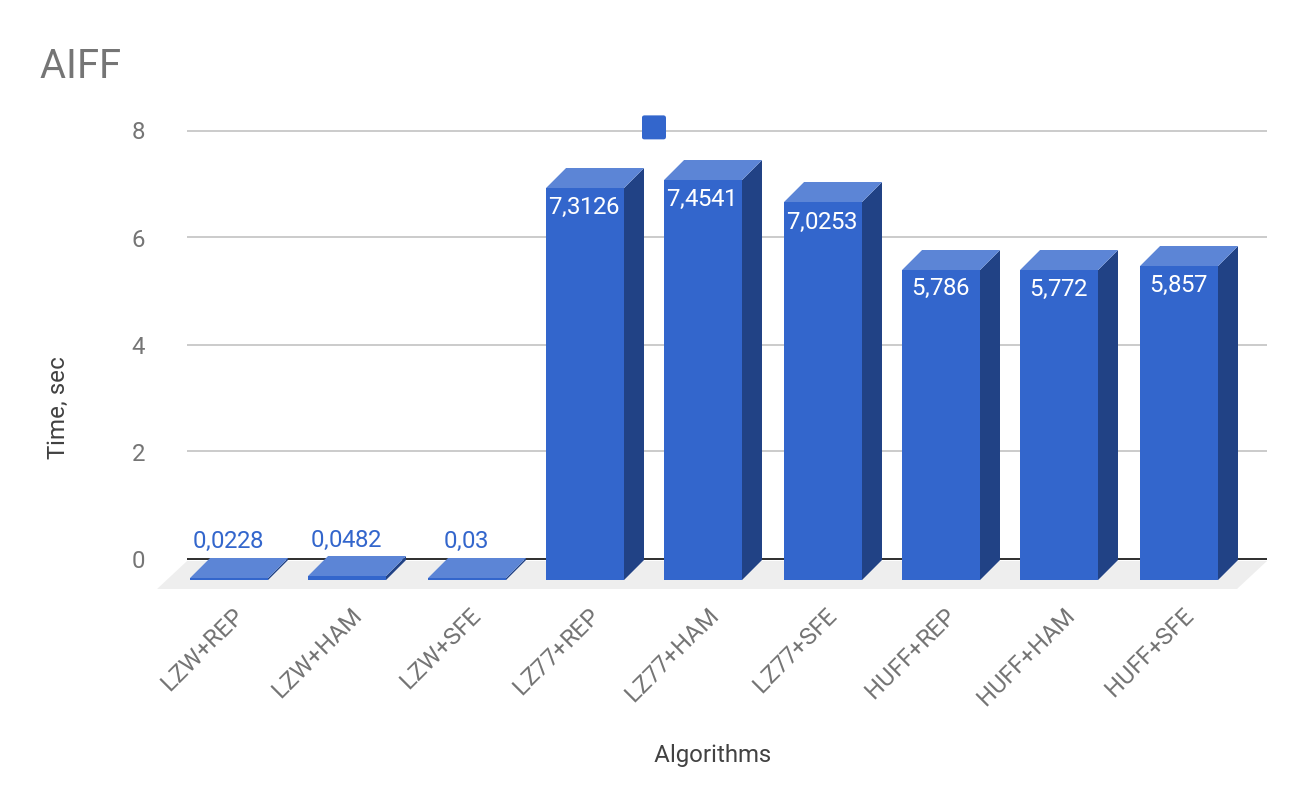
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Compression | Encoding | AIFF | BPM | GIF | RTF | TIF | WAV | TXT |
| LZW | Repetition | Time:  0.0228  Before:  77638  After:  61611 | Time:  0.067  Before:  49206  After:  108366 | Time:  0.0068  Before:  11473  After:  44475 | Time:  0.0622  Before:  48045  After:  98007 | Time:  1.6781  Before:  5094330  After:  20475609 | Time:  0.023  Before:  10300  After:  31442 | Time:  0.0237  Before:  8814  After:  14334 |
| LZW | Hamming | Time:  0.0482  Before:  77638  After:  41074 | Time:  0.0897  Before:  49206  After:  72244 | Time:  0.0327  Before:  11473  After:  29650 | Time:  0.0971  Before:  48045  After:  65338 | Time:  5.3970  Before:  5094330  After:  13650408 | Time:  0.032  Before:  10300  After:  20964 | Time:  0.0201  Before:  8814  After:  4773 |
| LZW | SFE | Time:  0.03  Before:  77638  After:  20538 | Time:  0.04  Before:  49206  After:  36123 | Time:  0.0398  Before:  11473  After:  14826 | Time:  0.0347  Before:  48045  After:  32670 | Time:  1.9534  Before:  5094330  After:  6825204 | Time:  0.0192  Before:  10300  After:  10482 | Time:  1.708  Before:  8814  After:  14625 |
| LZ77 | Repetition | Time:  7.3126  Before:  77638  After:  88491 | Time:  8.378  Before:  49206  After:  85185 | Time:  8.027  Before:  11473  After:  38109 | Time:  4.158  Before:  48045  After:  57306 | Time:  1.7883  Before:  5094330  After:  20475609 | Time:  5.443  Before:  10300  After:  32841 | Time:  1.7396  Before:  8814  After:  14625 |
| LZ77 | Hamming | Time:  7.4541  Before:  77638  After:  58994 | Time:  8.52  Before:  49206  After:  56790 | Time:  7.8041  Before:  11473  After:  25406 | Time:  4.2808  Before:  48045  After  38204: | - | Time:  5.344  Before:  10300  After:  21894 | Time:  1.742  Before:  8814  After:  9750 |
| LZ77 | SFE | Time:  .0253  Before:  77638  After:  29498 | Time:  8.032  Before:  49206  After:  28396 | Time:  7.683  Before:  11473  After:  12704 | Time:  3.8773  Before:  48045  After:  19103 | - | Time:  5.287  Before:  10300  After:  10948 | Time:  1.596  Before:  8814  After:  9750 |
| Huffman | Repetition | Time:  5.786  Before:  77638  After:  61607 | Time:  5.884  Before:  49206  After:  108051 | Time:  0.4971  Before:  11473  After:  32551 | Time:  5.126  Before:  48045  After:  91728 | -: | Time:  0.3394  Before:  10300  After:  25953 | Time:  0.1775  Before:  8814  After:  14985 |
| Huffman | Hamming | Time:  5.772  Before:  77638  After:  41072 | Time:  5.818  Before:  49206  After:  72034 | Time:  0.536  Before:  11473  After:  21702 | Time:  4.993  Before:  48045  After:  61152 | -: | Time:  0.4851  Before:  10300  After:  17302 | Time:  0.1869  Before:  8814  After:  9990 |
| Huffman | SFE | Time:  5.857  Before:  77638  After:  20405 | Time:  5.820  Before:  49206  After:  34857 | Time:  0.4694  Before:  11473  After:  10852 | Time:  5.483  Before:  48045  After:  30577 | -  : | Time:  0.427  Before:  10300  After:  8652 | Time:  0.1861  Before:  8814  After:  4996 |

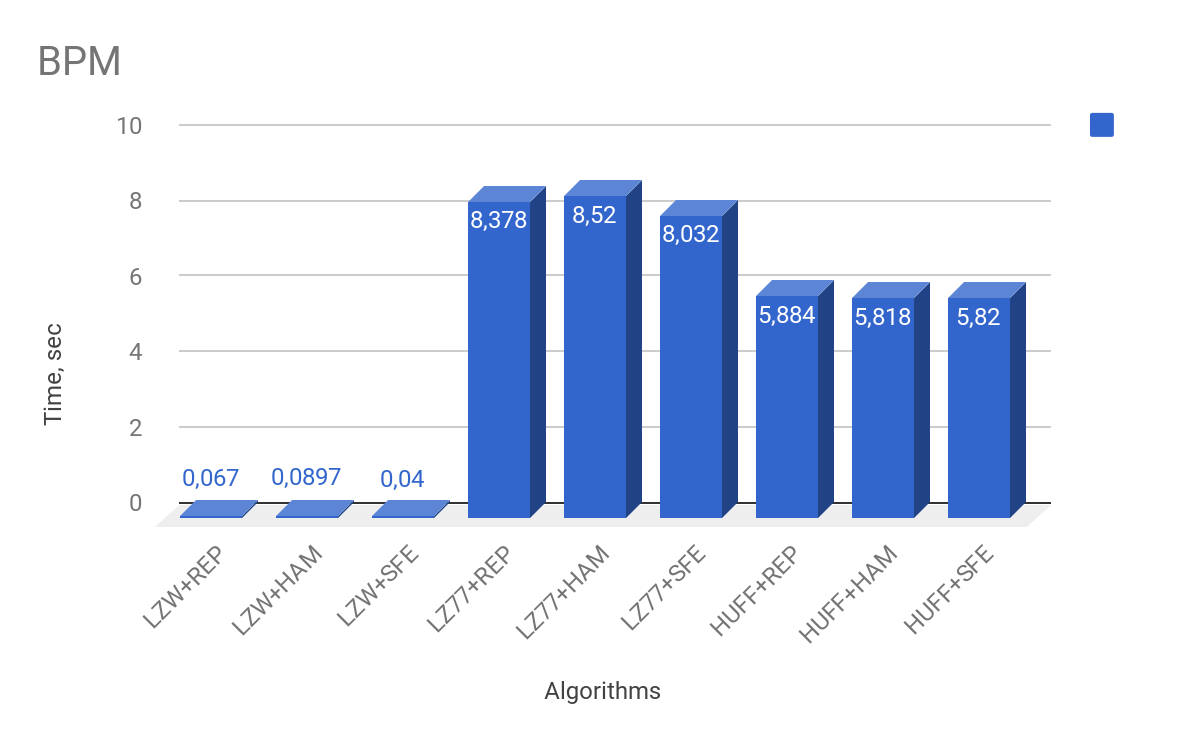
## **Table 2. File sizes after compression comparison**

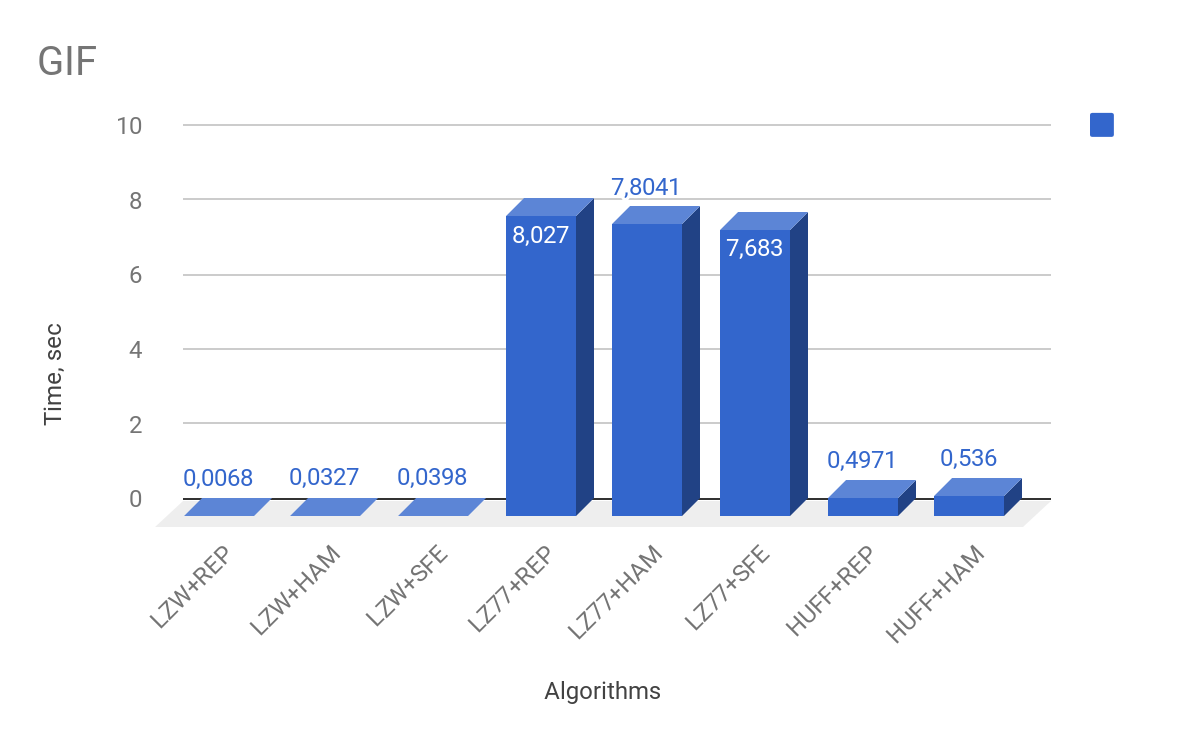
# **NOTE** files used for this table are the same as for the TABLE 1

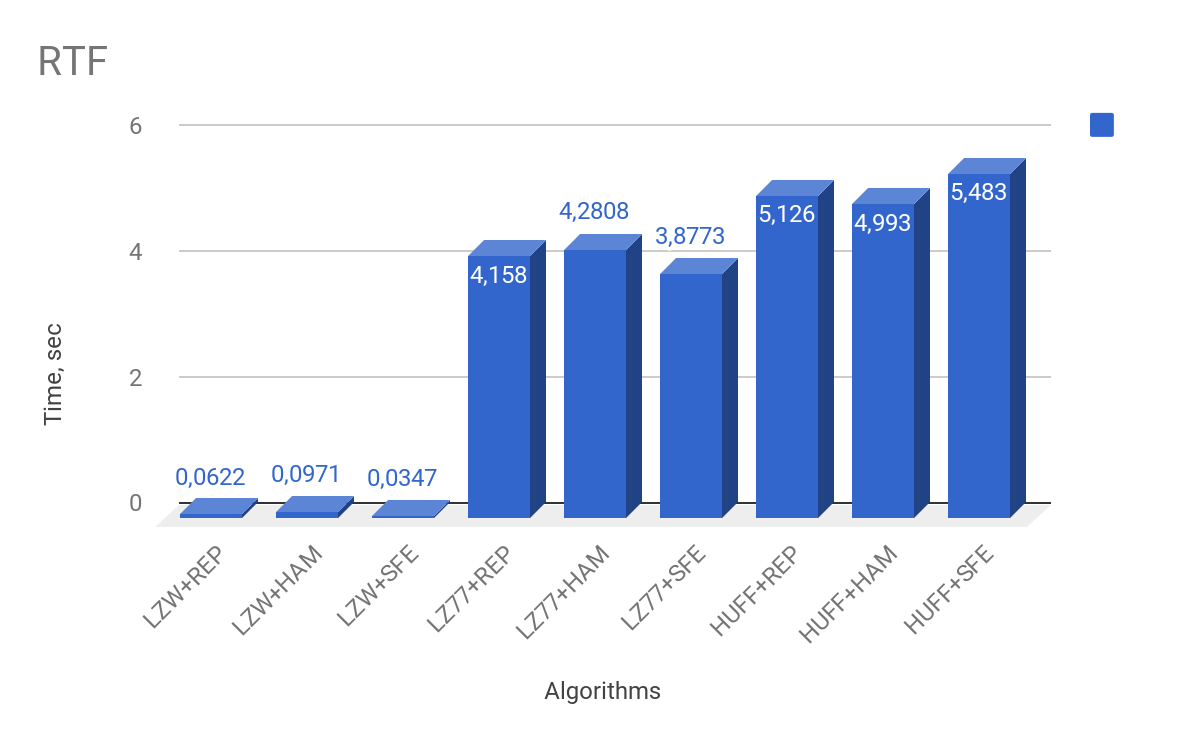
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | AIFF | BPM | GIF | RTF | TIF | WAV | TXT |
| LZW | 20537 | 36122 | 14825 | 32669 | 6825204 | 10482 | 4778 |
| LZ77 | 29497 | 28395 | 12703 | 19102 | - | 10947 | 4875 |
| Huffman | 20536 | 36017 | 10851 | 30576 | - | 8651 | 4995 |

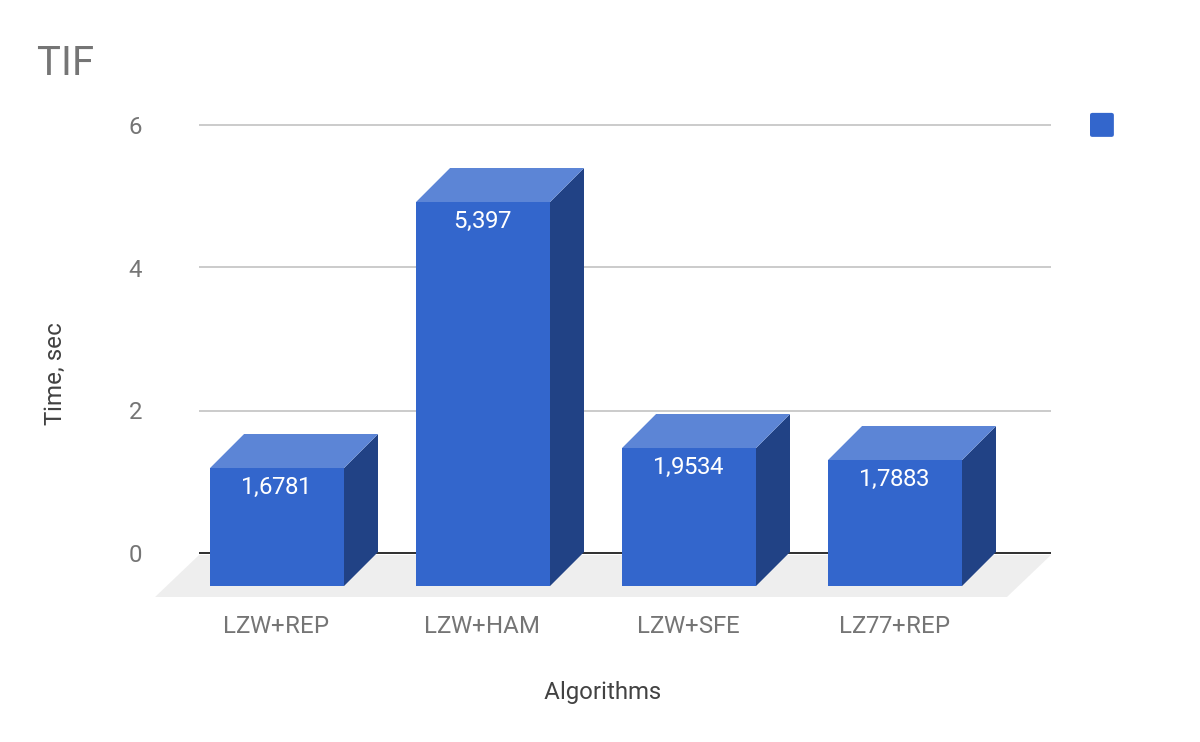
# The following 7 diagrams show time of compression with encoding for each pair of algorithms



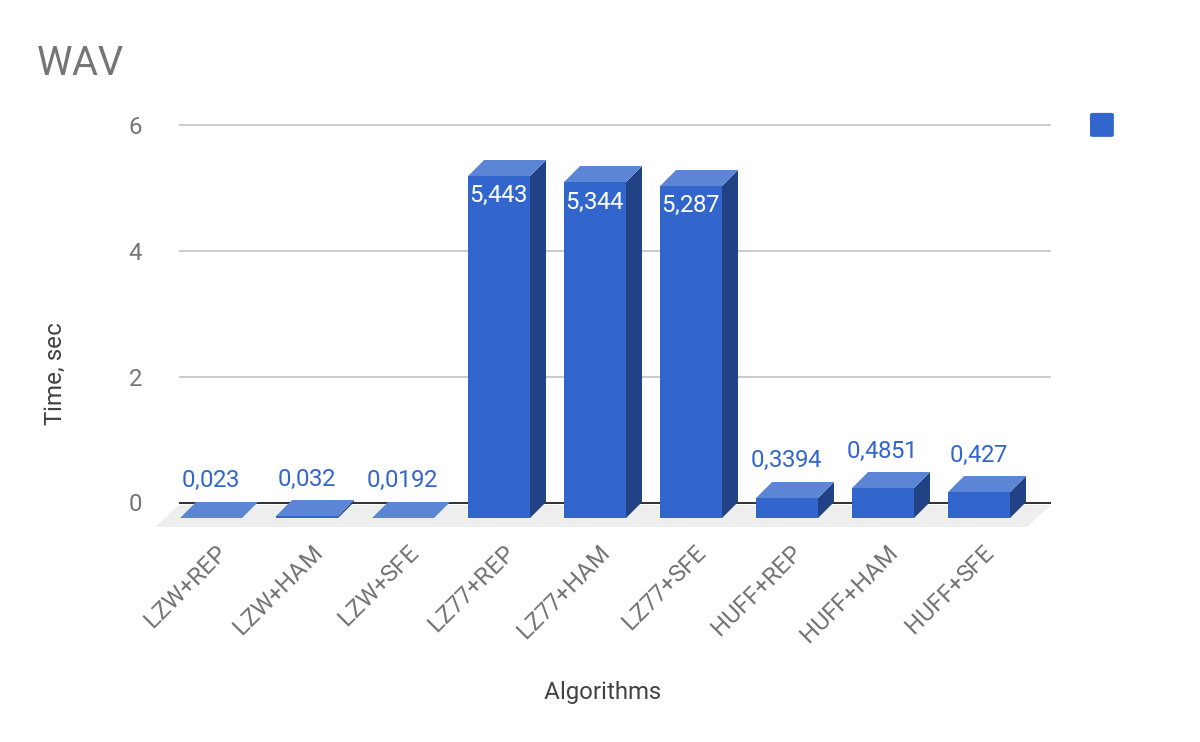




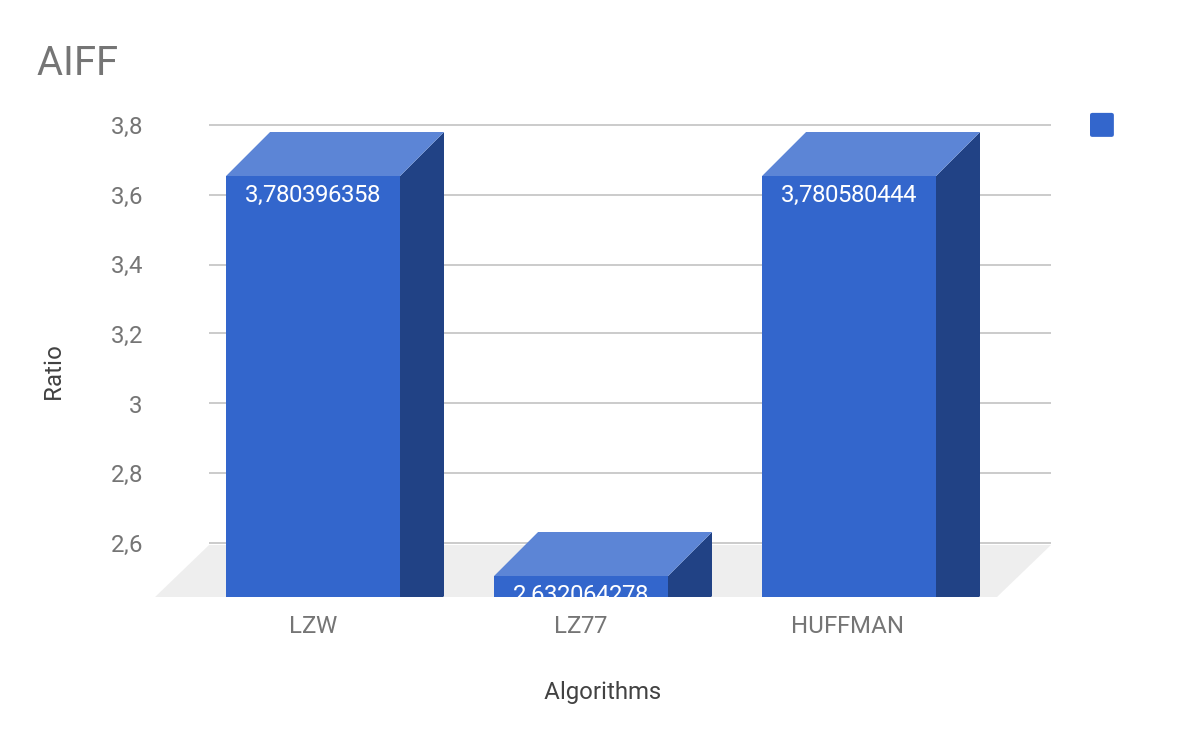


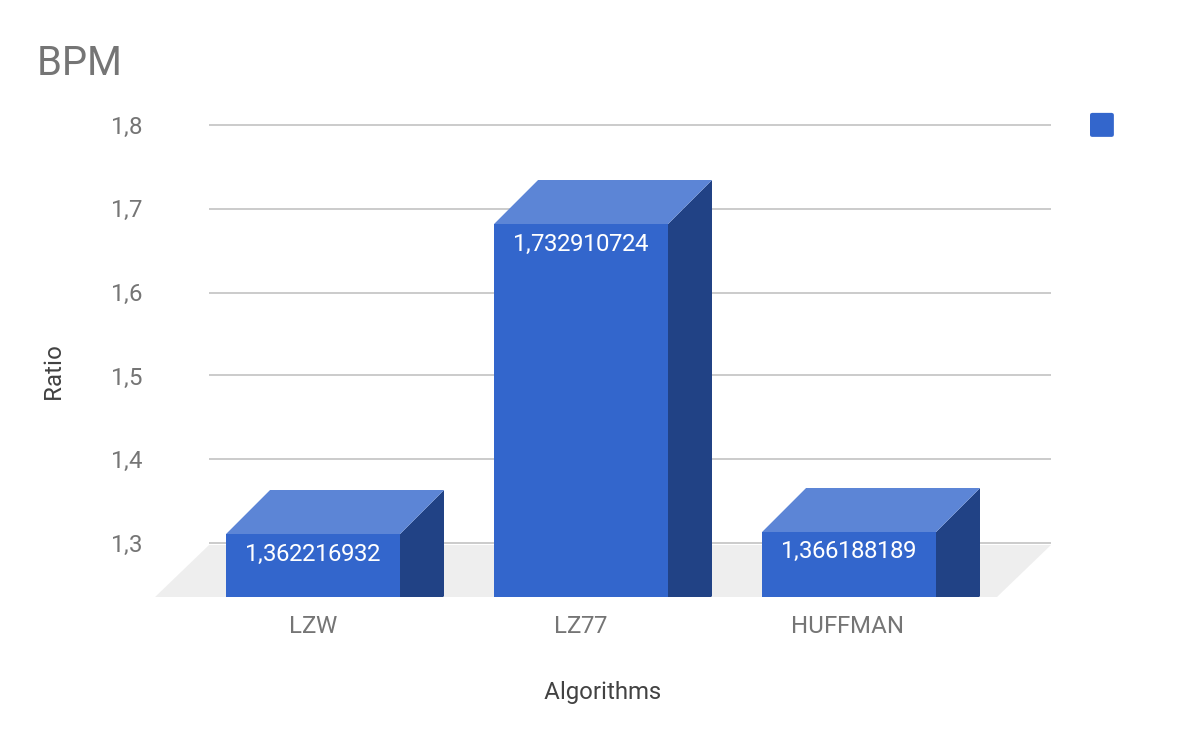


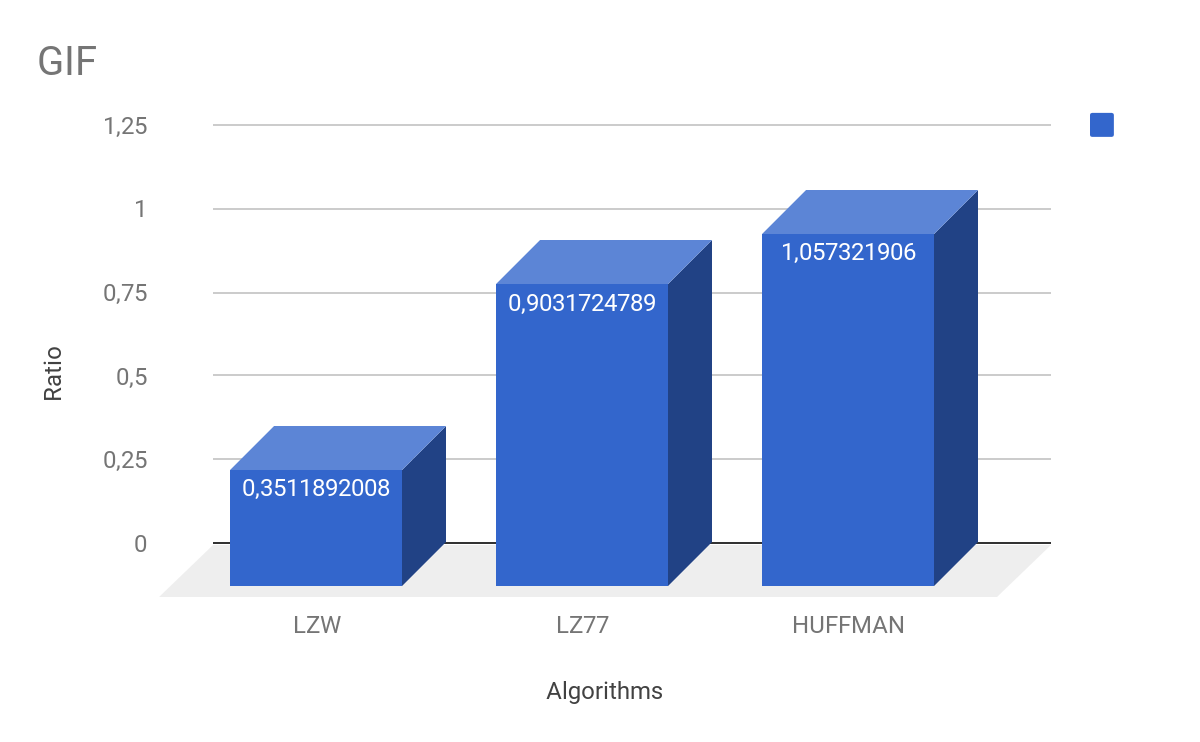
# Last 4 algorithms were removed from diagram above due to too long execution time ( + 10 minutes). Explanation is given in Conclusions paragraph

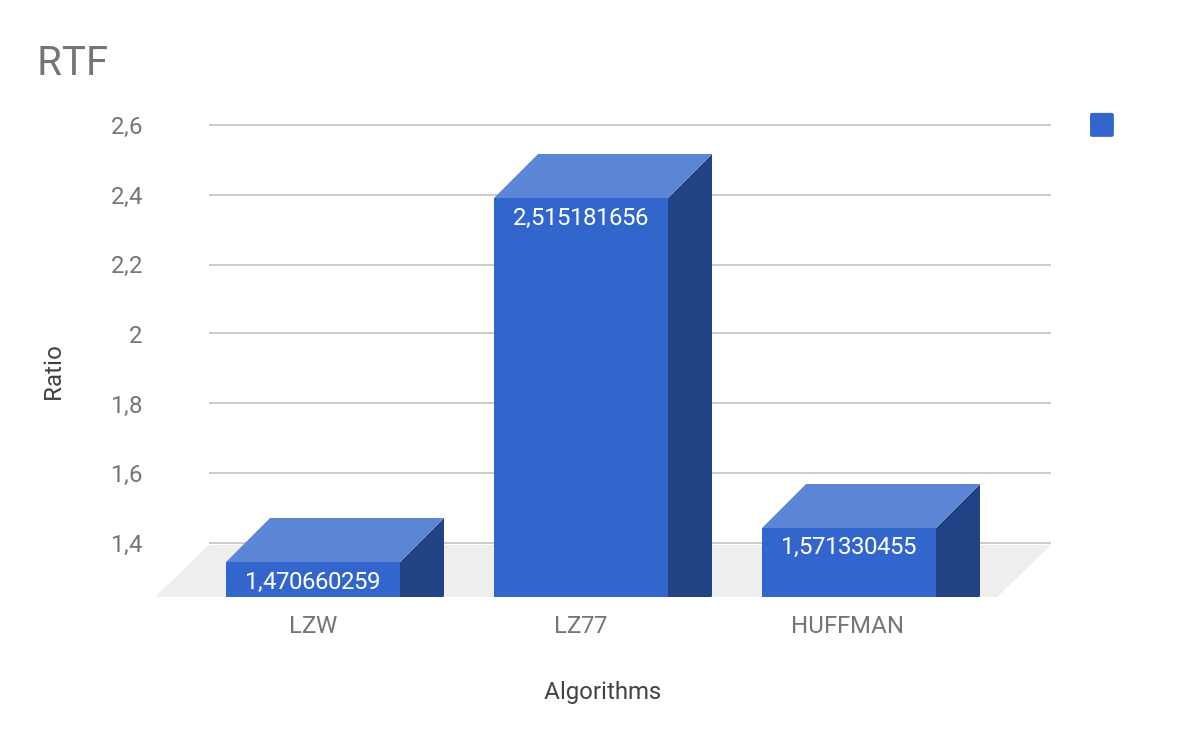


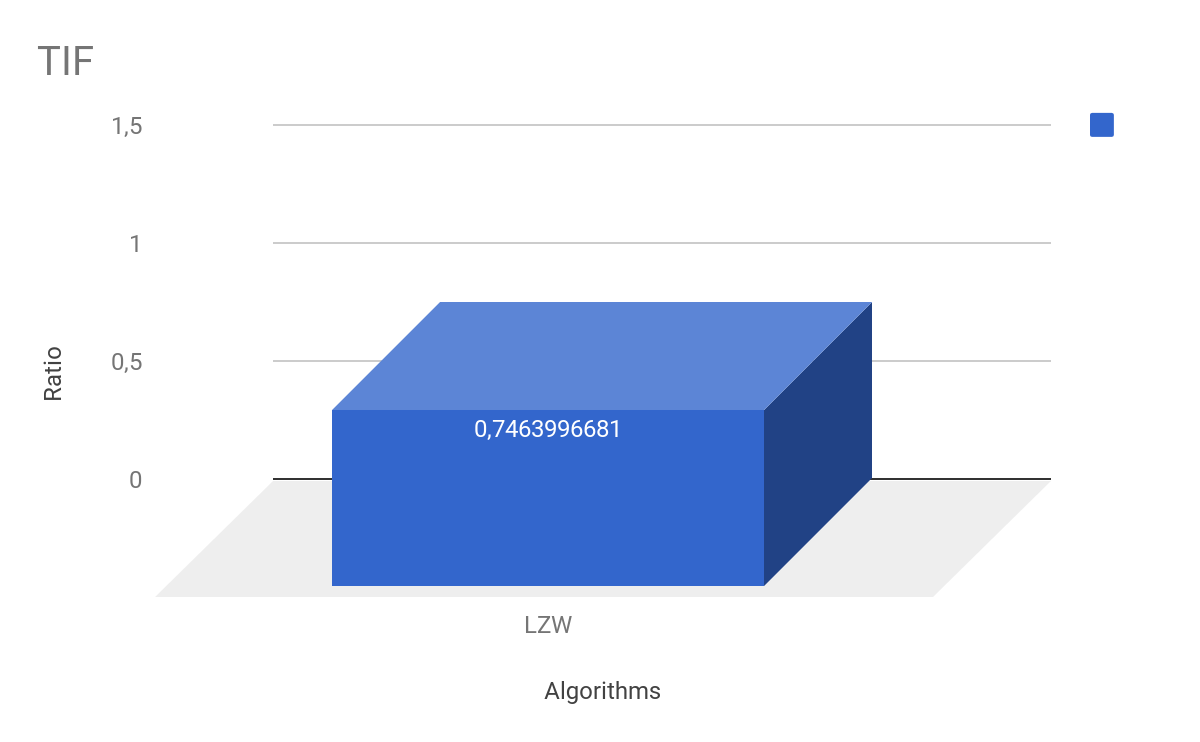
# The following 7 diagrams shows ratio of each compressing algorithm. Ratio = Initial size of file ( bytes ) / Compressed size of file ( bytes )



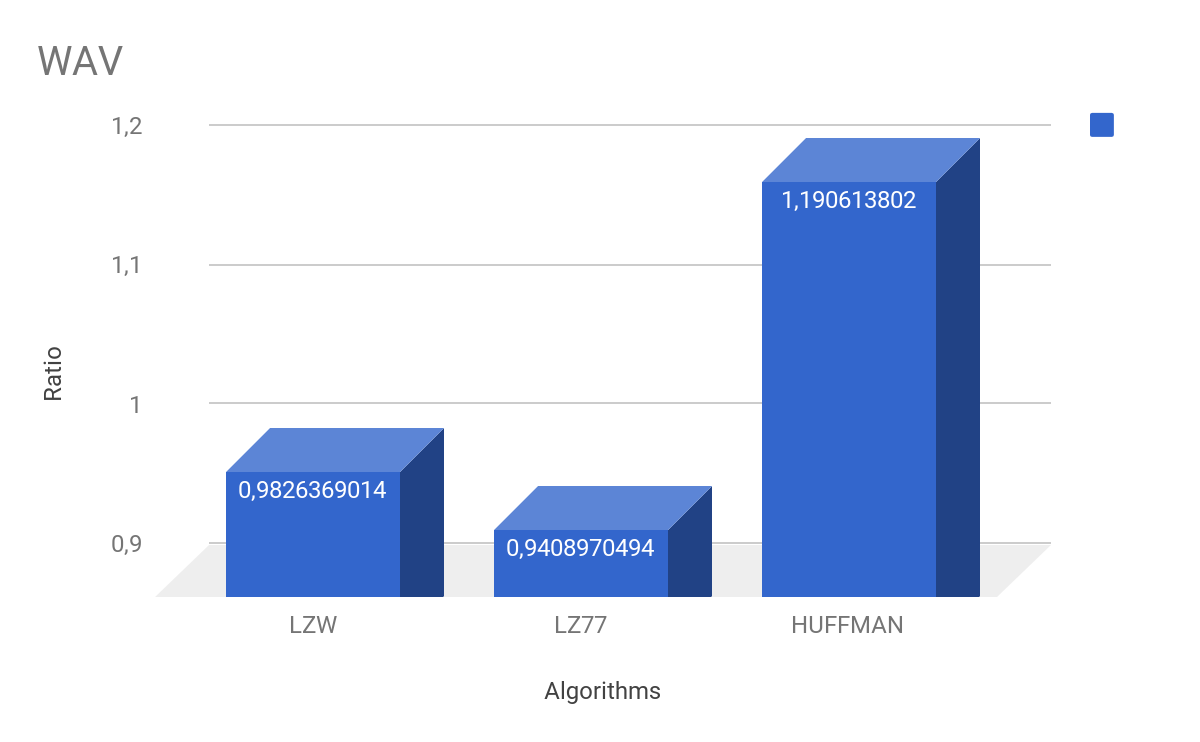


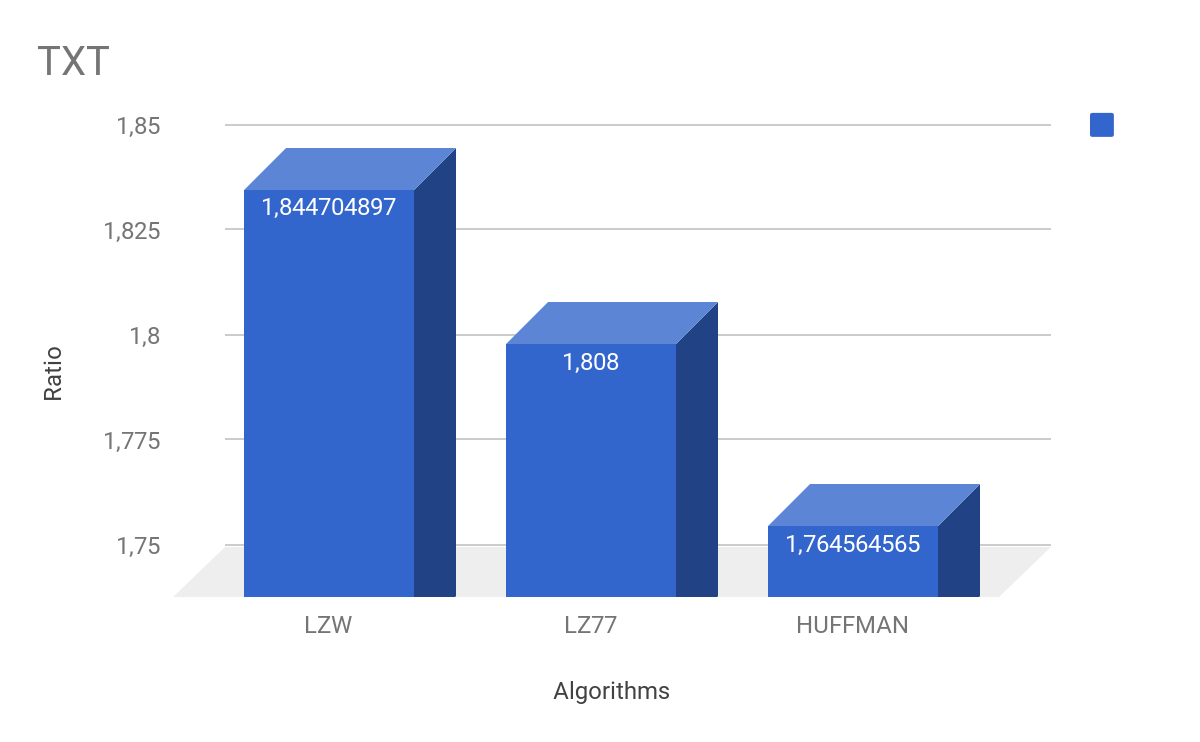






# LZ77 and Huffman were removed from diagram above due to too long execution time ( + 10 minutes) . Explanation in conclusion





Conclusions

As it can be seen from the tables and diagrams Huffman and LZ77 compression algorithms show lower performances compared to the LZW algorithm. In order to compress a message, the Huffman algorithm has to:

* calculate the frequencies of each symbol
* build a Huffman tree
* recursively go through it in order to assign codeword for each symbol
* create a new message by substituting each symbol with it’s codeword.

Hence it takes a lot of time and computing power to compress large files. However, for small files the algorithm shows good compression ratio.

For the LZ77 algorithm, size of the window is set to 1024 symbols and for each symbol the algorithm can take 1024 iterations. Because of that it takes much more time for this algorithm to compute the resulting code. Obviously it leads to a good compression rate with small files and takes long time to compress large files.

Finally, the LZW algorithm creates codewords on the fly,and is working with a pre-defined dictionary. The best compression time is achieved also because LWZ doesn’t have a window, so there is much more less computing in it.

All encoding algorithms showed nearly equal performance since they work with bits and as there is linear dependence between length of the message and time spent on the encoding.