



Cairo University Faculty of Engineering

Multimedia Project

Semester Spring 2018

Team 13

**Introduction:**

This is a project for Cairo University Faculty of Engineering Computer Department for Semester 2017/2018 second year, Multimedia.

The main purpose of the project is writing a program that uses the best lossless compression technique to compress Arabic text (set of Arabic book reviews).

**History:**

First, we tried **simple implantation algorithm** which converts each char from 16 bit to 7 bit. It gave us good results (compression ratio higher than 2 & close range), then we tried **Huffman algorithm.** The results were better than the first (2-3.2 compression ratio) but had very wide range.

We implemented **Extended Huffman algorithm,** but it takes too much time for compression and decompression, so we skipped it. We tried to implement and test **arithmetic algorithm,** but it produced bad results in most text.

We read about **Deflate**, **LZ77** & **LZW** and decided to use LZW to achieve higher efficiency. We tried to apply Huffman on the output, but the result wasn’t good. We discovered a very interesting way to compress binary sequences - we will talk about it in the next section - that gave us a very good compression ratio (3.80-3.97). We then tried to compress the binary sequence by Run-Length encoding, but it resulted in bigger files than the originals! We also read and searched about **LZMA algorithm** and **integer arithmetic algorithm** but they were too complex and time-consuming for us.

**The final compression algorithm:**

**Main algorithm:** We decide to use LZW for the project but we improved it for better results, we store a dictionary including all letters that appear in all files (113 characters).

**Improved algorithm:** We considered it a waste to store each number in 32 bits when - most of the time - the number didn’t exceed 24 bits. So we decided to limit the dictionary so each number has less than 32 bits, we noted that different size limits gave us different results for each file. We tried 14,16,18,20,22,24,26 bits. For most files, the best result was using 16 bits or 20 bits. This presents a problem, that some files work best with 16 bits while others work best with 20 bits.

After that, we discovered there’s no need to store each number with its size limit. We implemented an algorithm that stores numbers by different size limits depending on the position of the number.

For example: For the first number we know it will be less than 114 which is less than bits, then store it in only 7 bits and so on until size of dictionary exceeds 128. Now, the maximum number can store it in 8 bit and so on. We apply the same algorithm for decode.

Each algorithm was implemented in an independent class.

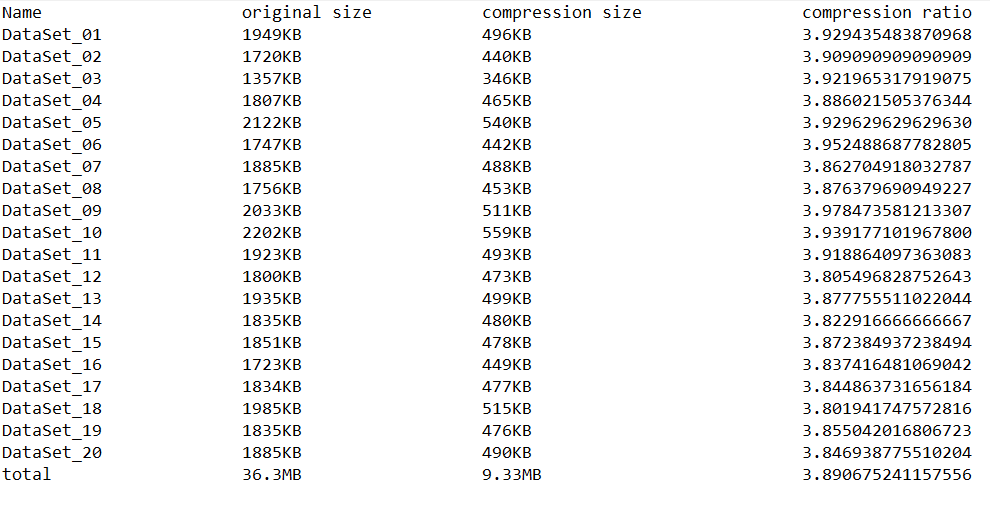
**Content of the compressed files:**

Only need the dictionary with all letters for all data set.

**Result:**

For the pervious compression technique:

* Simple implantation algorithm: ratio about 2.2
* Huffman: ratio about 3
* Extend Huffman : unknown
* Arithmetic :0.5-1
* Run length encoding : about 0.5
* LZW with limit : 3-3.5
* Improved LZW: about 3.9



**Taskboard:**

|  |  |
| --- | --- |
| **Name** | **Task** |
| Omar Mohammed | Read & write on the file |
| Interface |
| Search |
| Testing |
| Mohammed Emad | Huffman |
| Extend Huffman |
| Run length encode |
| Document |
| Waleed Mohammed | Simple implantation algorithm |
| Arithmetic |
| Document |
| Yahia Ali | LZW |
| Integration |
| Search |
| Testing |

Note: Integration includes renaming variables & files, adding important comments, and editing for optimization & validation.

# Glossary

|  |  |
| --- | --- |
| LZMA | Lempel–Ziv–Markov chain algorithm |
| LZW | Lempel–Ziv–Welch algorithm |
| LZ77 | Lempel–Ziv algorithm at 1977 |