Emre Sonrez
Timothy Blumberg
CompSci 201: Data Structures and Algorithms
Dr. Tabitha Peck
Huffman Assignment: Analysis.pdf
Due: 17/04/2014

Huffman Compression: An Analysis of the Compression Effectiveness

Introduction

The Huffman compression algorithm was first published in 1950 by David Huffman, who was then a graduate student at MIT. It relies upon analyzing which bit sequences of a file occur the most frequently, so that they can then be compressed the most in order to save the most disk space possible.

An Analysis of Compression Effectiveness

Using this compression algorithm, we were able to compress practically any file that could be stored on a computer. An interesting question can then be posed by this wide-range of possible file types; do certain types of files compress better than others? In order to address this problem most analytically, we chose to assess the differences in compression between .tiff and .txt files.

We found that .txt files actually compress better than .tiff files on average. Figure 1 shows the spectrum of compression degree as the number of characters (out of the 256 possible for the 8 bits per letter that we used) increases. Whenever the files use a similar number of the binary sequences out of the possible 256. As competent reader might notice, the low letter count TIFF files compress about as much as the average TXT files, but most TIFF files are high letter count files, so the average compression rate is much lower than the average TXT file compression rate. Table 1 outlines the basic statistics that were calculated during the analysis of the two different file formats.

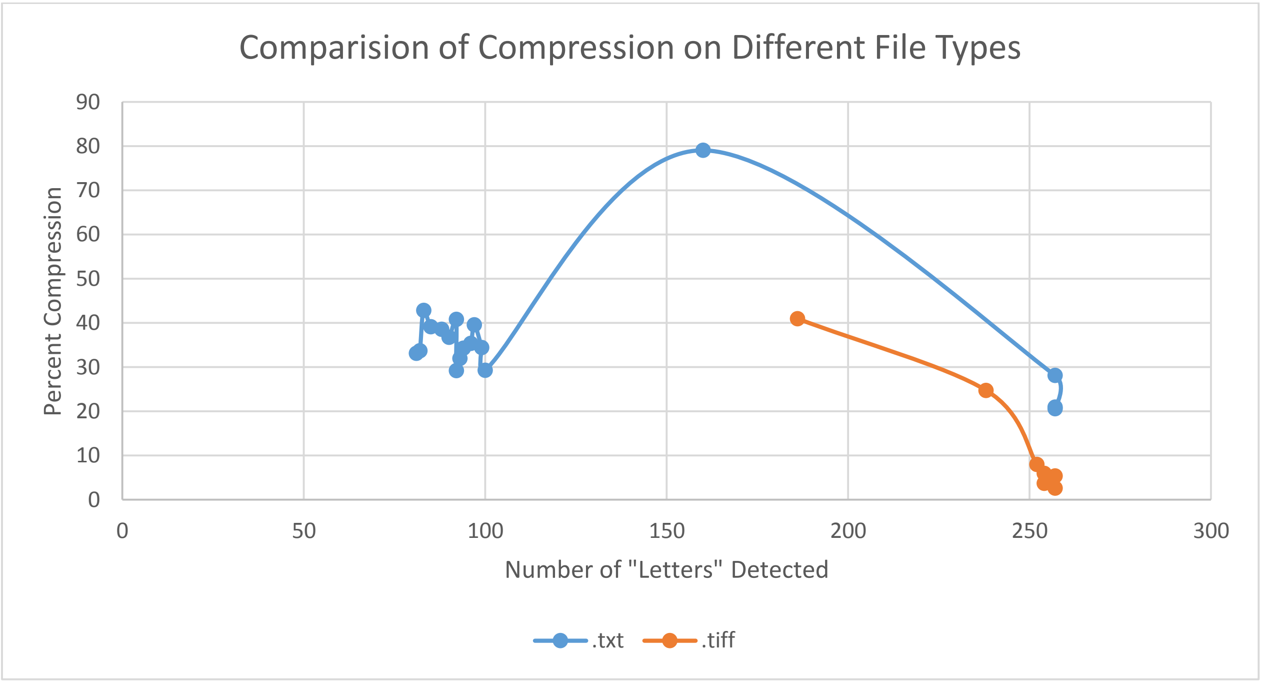

Figure 1: Comparison of Compression as a Function of Letter Count

Emre Sonrez
Timothy Blumberg
CompSci 201: Data Structures and Algorithms
Dr. Tabitha Peck
Huffman Assignment: Analysis.pdf
Due: 17/04/2014

Table 1: Comparison of Compression Statistics Between File Formats

	TXT Files	TIFF Files
Total % Compression:	43.241	18.137
Compress speed (bits / sec)	158037	384205

Therefore, the TXT files compress far better than the TIFF files because they contain fewer of the possible binary sequences, so the total file can be compressed more. An interesting finding that is shown in Table 1 is that the TIFF files actually compress faster by well over a factor of two than the TXT files that we tested. This faster compression is most likely due to the fewer number of compression actions that need to be taken to correctly compress the TIFF files as compared to the TXT files which are being more completely compressed.

Analyzing Multiple Compression

The second area of exploration that we undertook was how thoroughly could files be compressed before it was no longer useful? To explore this topic, we compressed files more than once and then compared how the compression changed as the number of compressions increased. We ran a modified version of the compression analysis utility `HuffMark` that would compress the files several times and measure how things changed as the number of compressions increased. Table 2 contains the hard numbers for these iterative tests and this data is visualized in Figure 2.

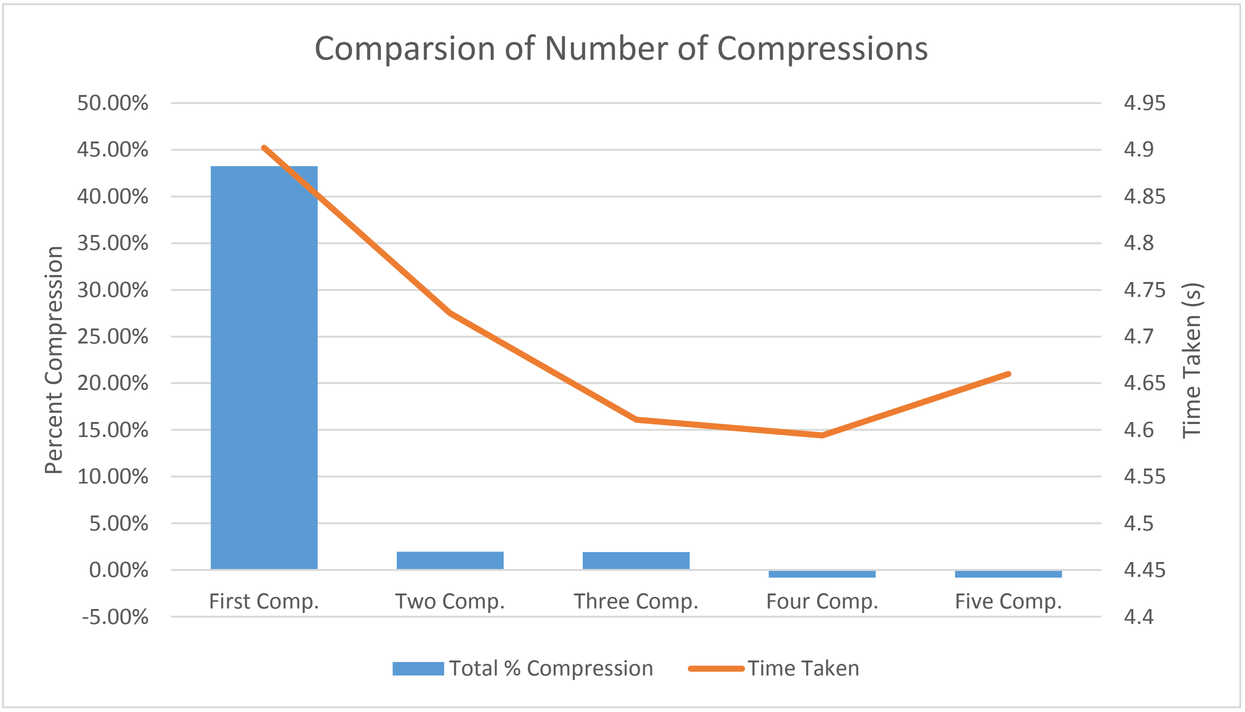

Figure 2: How the Number of Compressions Affects Compression Statistics

Emre Sonrez
Timothy Blumberg
CompSci 201: Data Structures and Algorithms
Dr. Tabitha Peck
Huffman Assignment: Analysis.pdf
Due: 17/04/2014

The data that was collected during our experiments indicates that on average with TXT files, the 3rd compression offers the last amount of compression. The first compression is always the most significant, but the next two iterations usually only slightly decrease the amount that a file can be compressed, although they do actually compress the file slightly more. After three compressions, the "compressed" file size was slightly larger than the writing file size. At this point, the compression was no longer serving any function, but testing was continued until 10 compression trials had been completed. The dataset here was truncated to include only the most pertinent information. The compression algorithm was then extremely useful in the first compression, but lacked any serious utility in any of the following trials. It is also of note that once the compressions began to fail to reduce the file size, it never got better. The rate at which the compression trial was increasing the file size did seem to eventually slow down at around -%18.

Table 2: Comparison of Compression Iteration Effectiveness

	First Comp.	Two Comp.	Three Comp.	Four Comp.	Five Comp.
Total % Compression	43.24%	1.96%	1.93%	-0.82%	-0.82%
Time Taken	4.902	4.725	4.611	4.594	4.660
Compress spd. (bits/s)	376479	390582	400300	397071	391634

We also then constructed and ran a similar battery of tests on a test file that was designed to be highly compressible. We created a file `a.txt` (in the test directory), that consisted of 433 KB of only the letter a. This file design easily and drastically compresses because there is only one letter, so algorithm can save quite a bit of space on each character. We also compressed this file 10 times, and its compression effectiveness per trial stayed positive until the 5th compression trial. This file not only compressed far more initially, but also in each compression run thereafter. The compression trials did start to become ineffective at the 5th compression trial though.

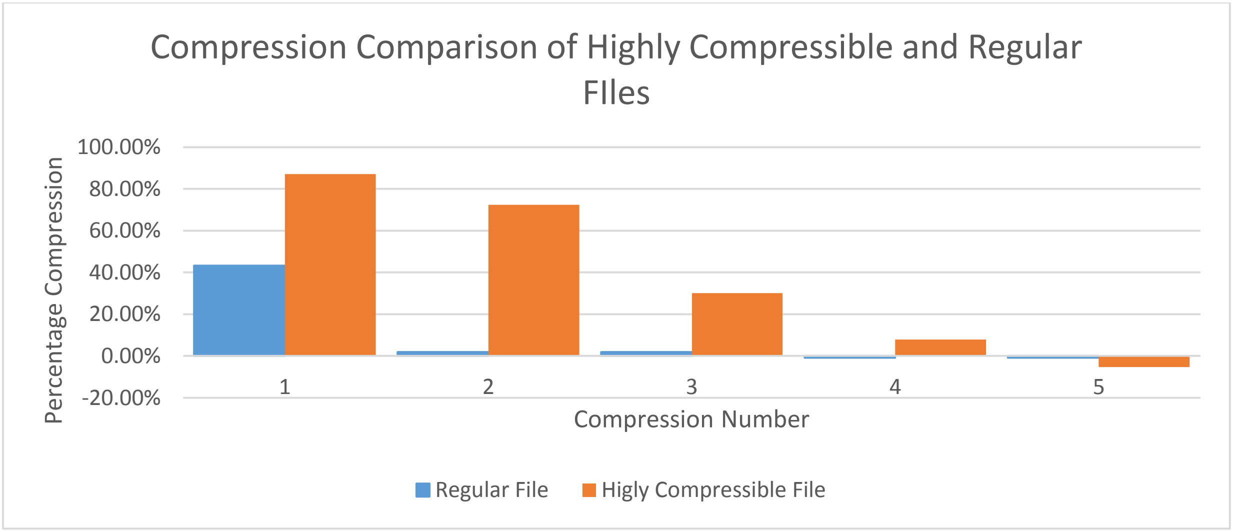

Figure 3: Performance Comparison of Multiple Compression Trials on Differently Styled Files