Huffman Analysis

Parker Pearson (jpp20), Cannon Palms (lcp27)

CS 201 - Data Structures & Algorithms

April 21, 2015

Which compresses more, binary or text files?

Huffman compression works by assigning prefixes to sequences of eight bits based on their relative frequencies within a given file. A sequence of eight bits could range from anywhere between 0-255. However, when working with text files, particularly *English* text files (things get tricky when extended ASCII codes must be considered), the eight-bit sequences generally span the values 32-126. This is because this range represents upper- and lowercase letters, numbers, and most common symbols. This narrowed range generally results in shorter prefix codes which means smaller compression files. When dealing with binary files such as videos and images, no such narrowed range exists. Therefore, text files typically compress more.

How much additional compression do you get by compressing an already compressed file? When does this become ineffective?

Compressing an already compressed file is minimally effective for files with many different characters/bytes. For example, one compression of Calgary Corpus with the TreeHuffProcessor compressed the file 43.714%. Yet, compressing the already compressed file gave a net compression of only 45.129 percent. Similarly, compressing Waterloo Corpus with the TreeHuffProcessor once gave a total compression of 18.215% while compressing it twice gave a total of 19.365%. In some cases, double compression of a file resulted in a file *larger* than the singly-compressed file. Additionally, this slight increase in saved space took over twice as long. A single compression of Waterloo Corpus took 12.690 seconds while a double compression took 31.771 seconds. The amount of bytes saved by each consecutive compression rapidly decreases, and additional compressions will eventually cause the file size to increase.

However, for files consisting largely of a single character (for text files) or a single byte (for binary files), additional compression of compressed files is much more effective. For a text file consisting of the character ‘a’ repeated over a million times, a single compression compressed the file by 87.500%, while double compression compressed the file by 98.437%. This process can be repeated with files of this variety for a number of compressions greater than two for even greater efficiency; the only constraint is an increase in time.

Repeated compression becomes ineffective when the number of Huffman prefix codes required for an additional compression exceeds that of the previous compression, keeping in mind that the header created factors into this number. For files of a single character, this

upper bound on increased compression efficiency occurs after $ceil(log_8(n))$ compresses, where n is the number of times the character is repeated. For other files, this upper bound is reached much sooner.

Can a file be designed that should compress a lot? When is it no longer worthwhile to keep compressing that file?

Files such as the one mentioned above that consist largely of one single character or a few characters see high compression rates with Huffman compression. The condition that renders additional compressions ineffective is also included above.

Tabular HuffMark data for TreeHuffProcessor and SimpleHuffProcessor implementations of HuffMark has been reproduced below.

Calgary Corpus - TreeHuffProcessor

File	Original Size (bytes)	Compressed Size (bytes)	Compression Percentage (%)	Compression Time (s)	Uncompression Time (s)
bib	111261	72883	34.494	0.104	0.129
book1	768771	438498	42.961	0.595	0.887
book2	610856	368443	39.684	0.477	0.672
geo	102400	72920	28.789	0.085	0.120
news	377109	246539	34.624	0.328	0.432
obj1	21504	16414	23.670	0.023	0.024
obj2	246814	194459	21.212	0.230	0.282
paper1	53161	33478	37.025	0.045	0.059
paper2	82199	47751	41.908	0.059	0.091
paper3	46526	27401	41.106	0.036	0.057
paper4	13286	7980	39.937	0.010	0.018
paper5	11954	7566	36.707	0.009	0.014
paper6	38105	24162	36.591	0.030	0.044
pic	513216	106781	79.194	0.141	0.583
progc	39611	26051	34.233	0.030	0.048
progl	71646	43113	39.825	0.055	0.090
progp	49379	30347	38.543	0.038	0.055
trans	93695	65365	30.236	0.076	0.103
TOTAL	**3251493**	**1830151**	**43.714**	**2.384**	**3.713**

Waterloo Corpus - TreeHuffProcessor

File	Original Size (bytes)	Compressed Size (bytes)	Compression Percentage (%)	Compression Time (s)	Uncompression Time (s)
clegg.tif	2149096	2033923	5.359	2.582	2.568
frymire.tif	3706306	2187824	40.970	2.826	4.213
lena.tif	786568	765474	2.682	0.929	0.889
monarch.tif	1179784	1109298	5.974	1.337	1.343
peppers.tif	786568	756295	3.849	0.925	0.885
sail.tif	1179784	1084822	8.049	1.313	1.336
serrano.tif	1498414	1126948	24.791	1.375	1.728
tulips.tif	1179784	1135186	3.78	1.403	1.365
TOTAL	**12466304**	**101997700**	**18.181**	**12.690**	**14.327**

Calgary Corpus - SimpleHuffProcessor

File	Original Size (bytes)	Compressed Size (bytes)	Compression Percentage (%)	Compression Time (s)	Uncompression Time (s)
bib	111261	73794	33.675	0.111	0.145
book1	768771	439408	42.843	0.608	0.736
book2	610856	369334	39.538	0.496	0.736
geo	102400	73591	28.134	0.101	0.128
news	377109	247427	34.388	0.317	0.442
obj1	21504	17084	20.554	0.022	0.028
obj2	246814	195130	20.940	.243	.263
paper1	53161	34370	35.347	0.041	0.065
paper2	82199	48648	40.817	0.065	0.091
paper3	46526	28308	39.157	0.037	.051
paper4	13286	8893	33.065	0.011	0.015
paper5	11954	8464	29.195	0.010	0.013
paper6	38105	25056	34.245	0.030	0.04
pic	513216	107585	79.037	0.137	0.568
progc	39611	26947	31.971	0.034	0.044
progl	71646	44016	38.565	0.052	0.083
progp	49379	31247	36.720	0.038	0.058
trans	93695	66251	29.291	0.087	0.110
TOTAL	**3251493**	**1845553**	**43.240**	**2.326**	**3.608**

Waterloo Corpus - SimpleHuffProcessor

File	Original Size (bytes)	Compressed Size (bytes)	Compression Percentage (%)	Compression Time (s)	Uncompression Time (s)
clegg.tif	2149096	2034594	5.328	2.594	2.413
frymire.tif	3706306	2188592	40.950	2.991	4.397
lena.tif	786568	766145	2.596	0.996	1.061
monarch.tif	1179784	1109973	5.917	1.451	1.522
peppers.tif	786568	756967	3.763	0.932	1.019
sail.tif	1179784	1085500	7.992	1.397	1.578
serrano.tif	1498414	1127644	24.744	1.454	1.942
tulips.tif	1179784	1135860	3.723	1.409	1.579
TOTAL	12466304	10205275	18.137	13.244	15.511