Huffman Code

Begin with a text file with the following frequencies

letter	A	B	C	D	E	F	G
frequency	2	4	6	10	13	13	16

Huffman Code

A	B	C	D	E	F	G
2	4	6	10	13	13	16

Huffman Code

Begin with a text file with the following frequencies

letter	A	B	C	D	E	F	G
frequency	2	4	6	10	13	13	16
code length	5	5	4	3	2	2	2

- $$\text{average bits per letter} = (5 \cdot 2 + 5 \cdot 4 + 4 \cdot 6 + 3 \cdot 10 \\ + 2 \cdot 13 + 2 \cdot 13 + 2 \cdot 16)/64 = \frac{168}{64} = 2.625$$

Huffman Code

Begin with a text file with the following frequencies

letter	A	B	C	D	E	F	G
frequency	2	4	6	10	13	13	16
code length	5	5	4	3	2	2	2

- $$\text{average bits per letter} = (5 \cdot 2 + 5 \cdot 4 + 4 \cdot 6 + 3 \cdot 10 + 2 \cdot 13 + 2 \cdot 13 + 2 \cdot 16)/64 = \frac{168}{64} = 2.625$$

- $$\text{Entropy} = \frac{2}{64}\log_2(32) + \frac{4}{64}\log_2(16) + \frac{6}{64}\log_2(\frac{64}{6}) + \frac{10}{64}\log_2(\frac{64}{10}) + 2 \times \frac{13}{64}\log_2(\frac{64}{13}) + \frac{1}{4}\log_2(4) \approx 2.579$$

Tree from heights

Note that given probabilities $p_A, p_B, \ldots, p_Z$, if we set

$$h_\alpha = \left\lceil log_2(\frac{1}{p_\alpha}) \right\rceil$$

then since we know from Theorem 4 that $\sum_{\alpha=A}^{Z} h_\alpha \leq 1$ then by Theorem 1 these values must correspond to heights of a (possibly incomplete) binary tree.

By the same proof as in theorem 4, this code will also have an expected code length less than or equal to $H + 1$.

Tree from heights

Begin with a text file with the following frequencies

letter	A	B	C	D	E	F	G
frequency	2	4	6	10	13	13	16

The goal is to encode each letter in such a way that minimizes the average number of bits used to store the file.

Tree from Heights

α	A	B	C	D	E	F	G
p_α	$\frac{2}{64}$	$\frac{4}{64}$	$\frac{6}{64}$	$\frac{10}{64}$	$\frac{13}{64}$	$\frac{13}{64}$	$\frac{16}{64}$
$\lceil log_2(\frac{1}{p_\alpha})\rceil$	5	4	4	3	3	3	2

Tree from heights

Begin with a text file with the following frequencies

letter	A	B	C	D	E	F	G
frequency	2	4	6	10	13	13	16
code length	4	4	3	3	3	2	2

- $$\begin{aligned}\text{average bits per letter} &= (4 \cdot 2 + 4 \cdot 4 + 3 \cdot 6 + 3 \cdot 10 \\ &+ 3 \cdot 13 + 2 \cdot 13 + 2 \cdot 16)/64 = \frac{169}{64} \approx 2.641\end{aligned}$$

- $$\begin{aligned}\text{Entropy} &= \frac{2}{64}\log_2(32) + \frac{4}{64}\log_2(16) + \frac{6}{64}\log_2(\frac{64}{6}) \\ &+ \frac{10}{64}\log_2(\frac{64}{10}) + 2 \times \frac{13}{64}\log_2(\frac{64}{13}) + \frac{1}{4}\log_2(4) \approx 2.579\end{aligned}$$

Experiment:
Random text consisting of taken from NYTimes consisting of 96,558 alphabetic characters (punctuation and spacing stripped from file).

A	B	C	D	E	F	G
7964	1466	3172	3897	11547	2023	1918
H	I	J	K	L	M	N
4626	7411	292	647	3955	2417	7007
O	P	Q	R	S	T	U
7423	1966	108	6113	6547	8947	2715
V	W	X	Y	Z		
1047	1565	139	1532	114		

Experiment:
Random text consisting of taken from NYTimes consisting of 96,558 alphabetic characters (punctuation and spacing stripped from file).

A	B	C	D	E	F	G
7964	1466	3172	3897	11547	2023	1918
H	I	J	K	L	M	N
4626	7411	292	647	3955	2417	7007
O	P	Q	R	S	T	U
7423	1966	108	6113	6547	8947	2715
V	W	X	Y	Z		
1047	1565	139	1532	114		

Calculate entropy of this file to be approximately 4.1727.

Using this text file with 96,558 characters and entropy 4.1727. Using three UNIX file compression programs `zip`, `compress` and `gzip`. I wanted to see how close to the theoretical minimum that I could get.

- `compress`:
  file length $=$ 45,122 bytes or 360,976 bits. The average number of bits per character is approximately 3.7384.
- `gzip`:
  file length $=$ 39,584 bytes or 316,672 bits. The average number of bits per character is approximately 3.2796.
- `zip`:
  file length $=$ 39,706 bytes or 317,648 bits. The average number of bits per character is approximately 3.2897.
- Wait!? How is it possible? You got better than the theoretical minimum? Oops! Read the instructions, and notice that they are encoding 32 bits at a time (not 8 bits).

Using this text file with $4 \times 96{,}558$ characters and entropy 4.1727. Using three UNIX file compression programs `zip`, `compress` and `gzip`. I wanted to see how close to the theoretical minimum that I could get.

- `compress`:
  file length $= 62{,}159$ bytes or $497{,}272$ bits. The average number of bits per character is approximately 5.15.
- `gzip`:
  file length $= 57{,}404$ bytes or $459{,}232$ bits. The average number of bits per character is approximately 4.76.
- `zip`:
  file length $= 57{,}526$ bytes or $317{,}648$ bits. The average number of bits per character is approximately 4.77.
- Thats better. These values are close (but larger than) the theoretical minimum.