Faculty of Engineering & Technology

Electrical & Computer Engineering Department

INFORMATION AND CODING THEORY

ENEE5304

"Huffman Code"

Prepared by:

Janna Hamad #1200853.

Sawsan Omair #1200822.

Instructor: Dr. Wael Hashlamoun.

Section: 2.

Date: 28/11/2024

Introduction

Huffman coding is the data compression algorithm developed by David A. Huffman in 1952; it assigns variables length binary codes according to the frequency of the character. Popular characters receive short codes and less popular ones get their codes longer so that there will be efficient space and transmission. With a prefix-free property, the decoder yields distinct representations, making it the right choice in file compression and multimedia coding applications. On Huffman coding this project looks at the working of Huffman code, how Huffman coding is done and a comparison between Huffman coding and fixed length coding.

Introduction to Huffman Coding

Huffman coding pioneered by David A. Huffman in 1952 forms the basis of data compression techniques that are lossless in nature. Unlike the fixed link encoding system, for example ASCII which provides equal code link, Huffman coding provides variable binary code links. These codes are computed depending on the number of time each character appeared in the input data and given a short code for a frequently used character and a long code for characters that are less used.

Prefix rule

To remove the inherent ambiguity in the decoding process, no code assigned to any character is a prefix of any other code in prefix coding. This means that no code word can either be a part of another code word or be the first portion of another code word. This property is preserved in Huffman coding tree structure to guarantee that the coding is prefix-free and hence decoding process is unique and easy.

Huffman Coding Work

The generation of Huffman code starts by estimating the frequency of all characters that is entered as the input data. A binary tree is then constructed gradually by taking two symbols which are least probable and merging them together until only one tree is left. This is known as Huffman code, in the tree, each leaf node represent a character while the path from root to the node represents the binary code for that character.

The performance of Huffman coding can then be evaluated by substituting the mean bits per character with the optimal entropy of the data that is, the ideal mean number of bits per character. The quantification detail computed by Shannon's formula is used like entropy and offers the lower

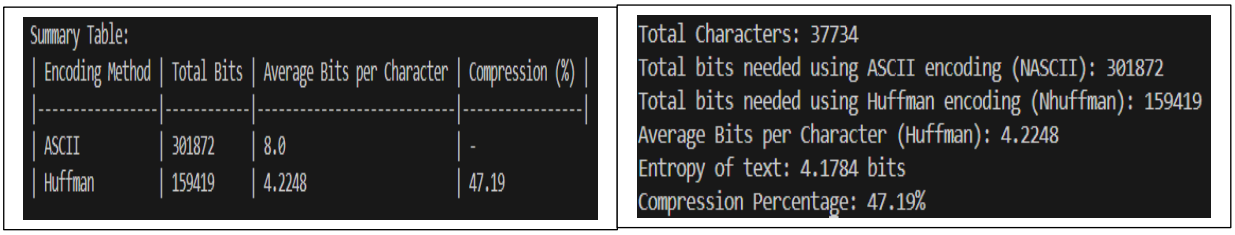
bound of lossless compression. This bound is frequently approached by Huffman coding, which is essentially a key element in applications of data compression such as file compression, data transmission, and multimedia encoding.

Results and Analysis

Evaluation of the character frequencies for **To Build A Fire** and Huffman codes indicates how this format effectively compresses data. For example, symbols like the space ‘e’ or ‘t’ are assigned shorter codes, while signs, hyphens, semicolons, periods, question and exclamation marks are assigned longer codes. This makes the average length of the code to be less than those of fixed length encoding mechanisms such as ASII that uses 8 bits for each character despite the frequency of the character. The entropy values indicate how each character is involved in the information content and the often used characters are most involved. Thus, as it means the code lengths are adjusted to match with the character frequency, the Huffman coding has been found to be efficient and could be used more and more especially for encoding text that have different distribution of characters.

Character	Frequency	Probability	Entropy (bits)	Huffman Code	Length (bits)
	7049	0.1868	0.4521	111	3
!	3	0.0001	0.0011	1000000101011	13
"	2	0.0001	0.0008	1000000101001	13
'	20	0.0005	0.0058	10110000111	11
,	436	0.0116	0.0744	000110	6
-	89	0.0024	0.0206	10000000	8
.	414	0.0110	0.0714	000100	6
:	2	0.0001	0.0008	1000000101010	13
;	26	0.0007	0.0072	1000000110	10
?	1	0.0000	0.0004	1000000101000	13
a	2264	0.0600	0.2435	1001	4
b	484	0.0128	0.0806	000111	6
c	779	0.0206	0.1156	110110	6
d	1515	0.0401	0.1862	11010	5
e	3887	0.1030	0.3378	010	3
f	794	0.0210	0.1172	00000	5
g	620	0.0164	0.0974	100001	6
h	2278	0.0604	0.2445	1010	4
i	1983	0.0526	0.2234	0110	4
j	20	0.0005	0.0058	1000000100	10
k	304	0.0081	0.0560	1000001	7
l	1127	0.0299	0.1513	10001	5
m	678	0.0180	0.1042	101101	6
n	2077	0.0550	0.2303	0111	4
o	1971	0.0522	0.2225	0011	4
p	421	0.0112	0.0724	000101	6
q	17	0.0005	0.0050	10110000110	11
r	1481	0.0392	0.1833	10111	5
s	1795	0.0476	0.2090	0010	4
t	2937	0.0778	0.2867	1100	4
u	800	0.0212	0.1179	00001	5
v	179	0.0047	0.0366	10110001	8
w	788	0.0209	0.1166	110111	6
x	34	0.0009	0.0091	1011000010	10
y	356	0.0094	0.0635	1011001	7
z	61	0.0016	0.0150	101100000	9
â	14	0.0004	0.0042	10000001011	11
”	14	0.0004	0.0042	10000001110	11
€	14	0.0004	0.0042	10000001111	11

Huffman coding is identified as a compression method that performs better than ASCII encoding. For the same set (37,734) characters the ASCII encoding therefore uses a fixed 8 bits per character meaning that the dataset takes up 301872 bits without compression. On the other hand, Huffman encoding vastly decreases the data size to 159,419 bits, that is, the compression ratio is 47,19%. Concerning efficiency, Huffman encoding has an average of bits per character equal to 4.2248 which is however close to the entropy of the text, that is 4.1784. Huffman coding works by assigning shorter bit codes to frequently used symbols and longer codes to rarer ones, effectively eliminating redundancy. This makes Huffman code one of the best compression methods from the standpoint of efficiency yet it is amazingly close to the most efficient code possible and therefore practical in data compression.



Summary Table:

Encoding Method	Total Bits	Average Bits per Character	Compression (%)
ASCII	301872	8.0	-
Huffman	159419	4.2248	47.19

Total Characters: 37734
Total bits needed using ASCII encoding (NASCII): 301872
Total bits needed using Huffman encoding (Nhuffman): 159419
Average Bits per Character (Huffman): 4.2248
Entropy of text: 4.1784 bits
Compression Percentage: 47.19%

The Huffman coding results of selected symbols as shown in this figure are the symbols : [‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘m’, ‘z’, space, and ‘.’]. This table connects a symbol with its probability, huffman code and the length of the codeword in bits.

Sub-table for selected symbols:

Symbol	Probability	Codeword	Length of Codeword
a	0.0600	1001	4
b	0.0128	000111	6
c	0.0206	110110	6
d	0.0401	11010	5
e	0.1030	010	3
f	0.0210	00000	5
m	0.0180	101101	6
z	0.0016	101100000	9
	0.1868	111	3
.	0.0110	000100	6

Conclusion

In conclusion, Huffman coding is a relevant and reliable lossless data compression algorithm that efficiently addresses the issue of data redundancy using a character frequency. It gets high compression ratios more than fixed length such as ASCII, and approaches shannons entropy. Its application in the areas such as text and multimedia encoding confirms its continued relevance in the general area of data compression and coding theory.