Middle-East Journal of Scientific Research 23 (8): 1598-1603, 2015
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2015.23.08.22429

Huffman Coding for Lossless Data Compression-A Review

[1]P. Ezhilarasu, [2]N. Krishnaraj and [3]V. Suresh Babu

[1]Department of Computer Science and Engineering,
Hindusthan College of Engineering and Technology,
Coimbatore-641032, India
[2]Department on Information Technology, Valliammai Engineering College,
Kattankulathur, Chennai 603203, Tamilnadu, India
[3]Department of Electronics and Communication Engineering,
Hindusthan College of Engineering and Technology,
Coimbatore - 641032, India

Abstract: In this paper, we discuss Huffman coding data compression techniques. Two type of input taken. First, input with similar probability of unique characters is considered. Then, input with different probability of unique characters is considered. Its compression ratio, space savings and average bits also calculated. Each condition compared with other conditions.

Key words: Huffman · Compression · Encoding · Decoding

INTRODUCTION

Data compression defined as the rearrangement of data in such a way that, the size of the target data is less than that of the size of the input data. The decompression technique used to get the source data. After decompression if some data unavailable, then the compression called as lossy compression. If none of the data missed, then the compression called as lossless compression. The Huffman coding comes under lossless compression. Each compression technique looks for two important aspects. Those are complexity in terms of time and space.

Because of data reduction, only small amount of time needed for data transfer between source and destination. For instance, if the volume of the source data is 240MB and the transfer rate is 30 kbps. The time need for the transfer obtained by the given equation 1.

Time needed for transfer of data = Input data / transfer rate

(1 MB = 1024 KB and 1 KB = 8 kb) (1)

Input data = 240 MB
= 240 * 1024 KB
= 240 * 1024 * 8 kb
Transfer rate = 30 kbps
So time taken for
transfer of data = (240 * 1024 * 8) / 30
= 8 * 1024 * 8
= 65536 seconds

If the given data compressed into 40MB, then the time taken for transfer will be 10923 seconds.

If the destination allowed amount of storage is 60GB, then the target machine can store the following number of files by using the equation 2.

Total number of files can be stored = Total amount of storage / Volume of the file (2)

(1GB = 1024 MB)
= 60 GB / 240 MB
= 60 * 1024 MB / 240 MB
=256 files

Corresponding Author: P. Ezhilarasu, Department of Computer Science and Engineering,
Hindusthan College of Engineering and Technology, Coimbatore-641032, India.

Therefore, the destination system can store 256 files for uncompressed data.
For compressed file, it can store

= 60 * 1024 MB / 40 MB
= 1.5 * 1024
= 1536 files.

The compression ratio affects both the space and time complexity. It calculated by using the following equation 3.

Compression ratio = Uncompressed original data / Compressed Data (3)

= 240 MB / 40 MB
= 6:1

Space savings also calculated by using compressed and uncompressed data. It obtained by using the following equation 4.

Space savings = 1- (Compressed Data/Uncompressed original data) (4)

In compression, we have the following types

- Lossy compression.
- Lossless compression.

In this paper, we discuss Huffman lossless data compression algorithm.

Related Work: A Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression [1].. Shannon-Fano coding technique was developed independently by Shannon and Fano. Initially Shannon introduced the concept in 1948 [2] and later the message encoding was implemented by Fano in 1949 [3]. Huffman coding, an algorithm developed by David A. Huffman while he was a Ph.D. student at MIT and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes" [4]. Huffman coding produces better optimal code than Shannon-Fano coding. Mark Nelson and Jean-loup Gailly [1995] represented the basics of data compression algorithms. It includes lossless and lossy algorithms [5]. David Salomon [2000] explained many different compression algorithms altogether with their uses, limitations and common usages. He gave an overview on lossless and lossy compression [6]. Khalid Sayood [2000] delineated an introduction into the various areas of coding algorithms, both lossless and lossy, with theoretical and mathematical background information [7]. Many books [8-11] published about the data compression techniques.

Huffman Encoding: In the field of data compression, Shannon–Fano coding is a top down approach. Whereas, Huffman coding is a bottom up approach.

Algorithm:

- Get the input data.
- Read the data character by character.
- Identify unique characters and its occurrences.
- Find the probability of each unique character.
- Write the most probable characters to the left of the code table. The least probable characters placed at the right of the code table.
- Find the two least most probability (occurrence) characters. Create a tree for that, two characters.
- The right part assigned the value 1 and the left part assigned the value 0.
- Apply step six and seven recursively to the remaining list until all unique character grouped into a single group with the collective probability of one.

Basic Example: If in a message (M), whose length is 100 we have eight, unique characters (m1, m2, m3, m4, m5, m6, m7, m8) with occurrences are 20,20,15,15,10,10,5,5. The probability (P) of each unique character given as (p1, p2, p3, p4, p5, p6, p7, p8) given in the equation 5.

Probability of a character (P) = Occurrence of the character / Total length of the message (5)

The probability of the unique characters (m1, m2, m3, m4, m5, m6, m7, m8) calculated as (p1, p2, p3, p4, p5, p6, p7, p8) using the equation 4 and is given in the coding table as given in the Table 1.

The probability of each unique character is always between zero and one. Here the highest probability is 0.2 and the least is 0.05. Initially the list is having eight unique characters m1, m2, m3, m4, m5, m6, m7, m8. The sum of the probability is one. The grouping done by grouping two least probability characters, as marked as bold and given in table 1. The process illustrated in the Table 2-8.

Table 1: Coding table for the message(M) with unique characters(m1,m2,m3,m4,m5,m6,m7,m8)

CHARACTER	m1	m2	m3	m4	m5	m6	m7	m8
OCCURRENCE	20	20	15	15	10	10	5	5
PROBABILITY	0.2	0.2	0.15	0.15	0.1	0.1	0.05	0.05

Table 2: Grouping of m7, m8

CHARACTER	m1	m2	m3	m4	m5	m6	m7(0),m8(1)
OCCURRENCE	20	20	15	15	10	10	10
PROBABILITY	0.2	0.2	0.15	0.15	0.1	0.1	0.1

Table 3: Grouping of m6, m7, m8

CHARACTER	m1	m2	m6(0), m7(10), m8(11)	m3	m4	m5
OCCURRENCE	20	20	20	15	15	10
PROBABILITY	0.2	0.2	0.2	0.15	0.15	0.1

Table 4: Grouping of m4, m5

CHARACTER	m4(0), m5(1)	m1	m2	m6(0), m7(10), m8(11)	m3
OCCURRENCE	25	20	20	20	15
PROBABILITY	0.25	0.2	0.2	0.2	0.15

Table 5: Grouping of m3, m6, m7, m8

CHARACTER	m6(00), m7(010), m8(011), m3(1)	m4(0), m5(1)	m1	m2
OCCURRENCE	35	25	20	20
PROBABILITY	0.35	0.25	0.2	0.2

Table 6: Grouping of m1, m2

CHARACTER	m1(0), m2(1)	m6(00), m7(010), m8(011), m3(1)	m4(0), m5(1)
OCCURRENCE	40	35	25
PROBABILITY	0.4	0.35	0.25

Table 7: Grouping of m3, m4, m5, m6, m7, m8

CHARACTER	m6(000), m7(0010), m8(0011), m4(10), m5(11), m3(01)	m1(0),m2(1)
OCCURRENCE	60	40
PROBABILITY	0.6	0.4

Table 8: Grouping of m1, m2, m3, m4, m5, m6, m7, m8

CHARACTER	m6(0000), m7(00010), m8(00011), m4(010), m5(011), m3(001),m1(10), m2(11)
OCCURRENCE	100
PROBABILITY	1.0

Table 9: Huffman Encoding for given input

Message	m1	m2	m3	m4	m5	m6	m7	m8
Probability	20	20	15	15	10	10	5	5
Encoding vector	10	11	001	010	011	0000	00010	00011

The encoding vector derived from the table 8 and given in the Table 9.

The total number of bits needed

= 20 *2 + 20 *2 + 15*3 + 15*3 +10*3 +10*3 + 5*5 +5*5
=40+40+45+45+30+30+25+25
=280 bits

The size of the input as uncompressed

= 100 * 8
= 800 bits

The size of the compressed data using binary coding =100*3=300 bits.

A.

B. ADVANCED EXAMPLE

Table 3: Huffman coding character occurrence for the given input

S.no	Symbol	Occurrence	Probability
1.	Space character " "	16	0.115108
2.	e	14	0.100719
3.	i	13	0.093525
4.	a	13	0.093525
5.	r	12	0.086331
6.	n	10	0.071942
7.	d	6	0.043165
8.	t	6	0.043165
9.	o	5	0.035971
10.	h	4	0.028777
11.	m	4	0.028777
12.	s	3	0.021583
13.	u	3	0.021583
14.	g	3	0.021583
15.	U	3	0.021583
16.	E	2	0.014388
17.	C	2	0.014388
18.	.	2	0.014388
19.	l	2	0.014388
20.	v	2	0.014388
21.	b	2	0.014388
22.	c	2	0.014388
23.	D	1	0.007194
24.	P	1	0.007194
25.	G	1	0.007194
26.	S	1	0.007194
27.	B	1	0.007194
28.	z	1	0.007194
29.	p	1	0.007194
30.	f	1	0.007194
31.	y	1	0.007194
32.	,	1	0.007194

Table 5: Encoding vector for each unique characters for the given input

S.no	Unique Character	Probability	EncodingVector
1.	Space character " "	0.115108	011
2.	e	0.100719	101
3.	i	0.093525	111
4.	a	0.093525	0000
5.	r	0.086331	0001
6.	n	0.071942	0100
7.	d	0.043165	1101
8.	t	0.043165	00100
9.	o	0.035971	00111
10.	h	0.028777	11000
11.	m	0.028777	11001
12.	s	0.021583	001100
13.	u	0.021583	001101
14.	g	0.021583	001010
15.	U	0.021583	001011
16.	E	0.014388	010110
17.	C	0.014388	010111
18.	.	0.014388	010100
19.	l	0.014388	010101
20.	v	0.014388	100010
21.	b	0.014388	100011
22.	c	0.014388	100000
23.	D	0.007194	1001010
24.	P	0.007194	1001011
25.	G	0.007194	1001000
26.	S	0.007194	1001001
27.	B	0.007194	1001110
28.	z	0.007194	1001111
29.	p	0.007194	1001100
30.	f	0.007194	1001101
31.	y	0.007194	1000010
32.	,	0.007194	1000011

Table 6: Size of the compressed data for the given input

S.NO	Unique Character	Occurrence	Encoding Vector	Total Length
1.	Space character " "	16	011	48
2.	e	14	101	42
3.	i	13	111	39
4.	a	13	0000	52
5.	r	12	0001	48
6.	n	10	0100	40
7.	d	6	1101	24
8.	t	6	00100	30
9.	o	5	00111	25
10.	h	4	11000	20
11.	m	4	11001	20
12.	s	3	001100	18
13.	u	3	001101	18
14.	g	3	001010	18
15.	U	3	001011	18
16.	E	2	010110	12
17.	C	2	010111	12
18.	.	2	010100	12
19.	l	2	010101	12
20.	v	2	100010	12
21.	b	2	100011	12
22.	c	2	100000	12
23.	D	1	1001010	7
24.	P	1	1001011	7
25.	G	1	1001000	7
26.	S	1	1001001	7
27.	B	1	1001110	7
28.	z	1	1001111	7
29.	p	1	1001100	7
30.	f	1	1001101	7
31.	y	1	1000010	7
32.	,	1	1000011	7

Input: "Dr. Ezhilarasu Umadevi Palani obtained his Under Graduate degree in Computer Science and Engineering from Bharathiar University, Coimbatore."

The input placed between " ". The input has 139 characters with 32 unique characters. Each unique character has some occurrences, as shown in table 3.

The probability and encoding vector of each unique character represented in the Table 5.

The size of the compressed data derived from the Table 5. It is given in the Table 6.

The given input after encoding will be

100101000010101000101101001111110001110101010000000100000011000011010110010111100100001101101110001011100000101010000010011101100111100011001000000011101001011101011110001110011000110010110100110110100001011100100000010000110100110100000010010101111011010010100001101101011111010001101011100111100110011000011010010010100010111001001100000111101010010000010101100000100110101101011001000010101110100100010100001111010000101001110011010001001111100101110011101100000000001000000100110001110000000101100101101001111000101010001001100111001000010100000110110101110011111111100110001100000001000011100011010100

The total number of bits needed is 614 bits.

The size of the input as uncompressed

= 139 * 8
= 1112 bits

The size of the compressed data using binary coding =695 bits.

Huffman Decoding: The Huffman decoding implemented by replacing the encoding vector from the starting of the input. i.e., 1001010 replaced by D and so on. The decoding process stops after step number 139(no of characters).

Input after Encoding:

100101000010101000101101001111110001110101010000000
010000001100001101011001011110010000110110110001011
1101110010110000010101000001001110110011110001100100
0000001110100101110101111000111001100011001011010011
0110110100010111001000000010000110100110100000001001
0101111011010010100001101101011111010001101011100111
1110011001100001101001001010001011100100110000011
1101010010000010101100000100110101101011001000010101
1101001011010001111010000101001110011010001001111
1001011100111011000000000001000000100110001110000000
0101100101101001111000101010001001100111001001000010
1000011011010111001111111100110001100000010000111
10001101010100

Input after Decoding:

Step1

D000101010001011010011111100011101010100000001000
0001100001101011001011110010000110110110001011101
11001011000000101010000001001110110011110001100100000
0111010010111010111100011100110001100101101001101
1010001011100100000001000011010011010000000100101011
1101101001010000110110101111101000110101110011111
00110011000011010010010100010111001001100000111101
01001000000101011000000100110101101011001000010101
11010010110100011110100001010011100110100010011111
00101110011101100000000000100000010011000111000000
0010110010110100111100010101000100110011100100100
00101000011011010111001111111100110001100000010000
1110001101010100

Step2:

Dr010100010110100111111000111010101000000010000000
110000110101100101111001000011011011000101110111001
011000000101010000001001110110011110001100100000011
1010010111010111100011100110001100101101001101101
0001011100100000001000011010011010000000100101011
11011010010100001101101011111010001101011100111110
01100110000110100100101000101110010011000001111010
10010000010101100000010011010110101100100001010111
01001011010001111010000101001110011010001001111100
10111001110110000000000010000001001100011100000000
101100101101001111000101010001001100111001001000
010100001101101011100111111110011000110000001000
01110001101010100

Step3:

Dr.01011010011111100011101010100000001000000110000
110101100101111001000011011011000101110111001011
0000010101000000100111011001111000110010000000111
0100101110101111000111001100011001011010011011010
0010111001000000010000110100110100000001001010111
1011010010100001101101011111010001101011100111110
01100110000110100100101000101110010011000001111010
10010000010101100000010011010110101100100001010111
01001011010001111010000101001110011010001001111100
1011100111011000000000001000000100110001110000000
1011001011010011110001010100010011001110010010000
1010000110110101110011111111001100011000000100001
11000110101010 0

Step4:

Dr.E1001111110001110101010000000100000011000011010
1100101111001000011011011000101110111001011000001
0101000000100111011001111000110010000000111010010
1110101111000111001100011001011010011011010001011
10010000000100001101001101000000010010101111011010
01010000110110101111101000110101110011111001100110
00011010010010100010111001001100000111101010010000
01010110000001001101011010110010000101011101001011
0100011110100001010011100110100010011111001011100
11101100000000000100000010011000111000000010110010
1101001111000101010001001100111001001000010100001
1011010111001111111100110001100000010000111000110
1010100

Step 5:

Dr.Ez110001110101010000000010000001100001101011001011110010000110110110001011101110010110000010101000001001110110011110001100100000011101001011101011110001110011000110010110100110110100010111001000000100001101001101000000100101011110110100101000011011010111110100011010111001111100110011000011010010010100010111001001100000111101010010000010101100000100110101101011001000010101110100101101000111101000010100111001101000100111110010111001110110000000000100000001001100011100000000101100101101001111000101010001001100111001001000010100001101101011100111111110011000110000000100001110001101010100

Step 6-139:

Dr.Ezhilarasu Umadevi Palani obtained his Under Graduate degree in Computer Science and Engineering from Bharathiar University, Coimbatore.

RESULTS AND DISCUSSION

The compression ratio, space savings and average bits calculated for the two examples are Basic Example (8 unique characters).

Compression ratio = 800/280
= 20:7
=2.85:1
Space savings =1-(280/800)
= 1-(7/20)
= 1-0.35
= 0.65
= 65%
Average bits = 280/100
= 2.8 bits per character

Advanced Example (32 unique characters)

Compression ratio = 1112/614
= 556:307
=1.81:1
Space savings =1-(614/1112)
= 1-(307/556)
= 1-0.552
= 0.448
= 44.8%
Average bits = 614/139
=4.42 bits per character

CONCLUSION

The obtained results depicts that the input with the similar probability gives better compression ratio, space savings and average bits than the input with the different probability. The Huffman code gives better compression ratio, space savings and average bits as compared with the uncompressed data.

REFERENCES

1. http://en.wikipedia.org/wiki/Huffman_coding
2. Shannon, C.E., 1948. “A mathematical theory of communication”, Bell System Technical Journal, 27: 379-423, 623-656.
3. Fano, R.M., 1949. “The transmission of information”, Technical Report 65, Research Laboratory of Electronics, M.I.T., Cambridge, Mass.
4. Huffman, D.A., 1952. "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp: 1098-1102. Huffman's original article.
5. Mark Nelson and Jean-loup Gailly, 1995. “The Data Compression Book”, M and T Books, New York, United States of America, 2nd edition, pp: 541.
6. David Salomon, 2000. “Data Compression: The Complete Reference”, Springer, New York, Berlin, Heidelberg, United States of America, Germany, 2nd edition, pp: 823.
7. Khalid Sayood, 2000. “Introduction to Data Compression”, Morgan Kaufmann Publishers, Burlington, United States of America, 2nd edition, pp: 600.
8. David Salomon and Giovanni Motta, 2000. “Handbook of Data Compression”, Springer London.
9. Storer, J.A., 1988. “Data Compression”, Computer Science Press, Rockville, MD.
10. Blelloch, E., 2002. “Introduction to Data Compression”, Computer Science Department, Carnegie Mellon.
11. Lynch, J. Thomas, 1985. “Data Compression: Techniques and Applications”, Lifetime Learning Publications, Belmont, CA.