Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Double Huffman Coding

Pankaj Kumar
B.Tech , Computer Science and Engineering
S.R.M.S. College of Engineering and technology Bareilly (U.P) India

Ankur Kumar Varshney
Assistant Professor, Computer Science & Engineering
Institute of Technology and Management, Aligarh, (U.P) India

Abstract: Lossless compression of a sequence of symbols is important in Information theory as well as today's IT field. Huffman coding is lossless and is most widely used. However, Huffman coding has some limitations depending on the stream of symbols appearing in a file. In fact, Huffman coding generates a code with very few bits for a symbol that has a very high probability of occurrence and a larger number of bits for a symbol with a low probability of occurrence [1]. In this paper, we present a novel technique that work on Huffman Coding and after getting codeword for the Symbol we compress it on the Basis of its Binary no. 0 and 1. It will give better result than Huffman Coding.

Keyword - Huffman coding, data compression, lossless coding, Entropy

1. INTRODUCTION

Lossless compression refers to compression methods for which the original uncompressed dataset can be recovered exactly from the compressed stream. In computer science, lossless compression is very important for the compression of text, digitized medical data, etc... In general, even when lossy compression is allowed, the overall compression scheme may be a combination of lossy compression process followed by a lossless compression [2]. The Huffman encoding algorithm provides an optimal way to compress dataset without any loss of information [3]. The semi static Huffman compression involves two passes over a dataset during encoding: one to gather symbol frequency information and another to perform the actual coding once a code has been constructed [4]. The Huffman coding gives more emphasize to those symbols in a dataset which has high probability of occurrence in comparison with those with lower probability of occurrences [5]. Although this technique gives a good compression for those dataset in which the frequency of occurrence of symbols are high, but for lower frequency data Huffman coding encodes with relatively higher number of bits. Instead of using this straightforward Huffman coding, in this paper we use Huffman coding in a novel way, we create a string of the symbol after concatenation in the alphabetic or numeric order, and compress that using 0 and 1 probability.

2. DOUBLE HUFFMAN CODING ALGORITHM

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type.

Huffman codes are part of several data formats as ZIP, GZIP and JPEG. Normally the coding is preceded by procedures adapted to the particular contents.

2.1. Double Huffman Coding Algorithm

The Huffman algorithm works from leaves to the root in the opposite direction.

1. Create a leaf node for each symbol and add it to frequency of occurrence.
2. While there is more than one node in the queue:
  i. Remove the two nodes of lowest probability or frequency from the queue.
  ii. Assign 0 and 1 respectively to any code already assigned to these nodes.
  iii. Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
  iv. Add the new node to the queue.
3. The remaining node is the root node and the tree is complete.
4. Compute the Entropy, Average Length and Redundancy.
5. After Getting the Codeword of each Symbol/Alphabet, concatenate it.
6. Find the Probability of 0 and 1.
7. Repeat Step form 1-4.

2.2. Entropy

Named after Boltzmann's H-theorem, Shannon denoted the entropy *H* of a discrete random variable *X* with possible values {x_1, ..., x_n} and probability mass function *p(X)* as,

$H(X) = \mathrm{E}[I(X)) = \mathrm{E}(-\ln p(X))$.-Here E is the operator, and *I* is the information content of *X*. *I*(*X*) is itself a random variable. The entropy can explicitly be written as

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = \sum_{i=1}^{n} p(x_i) \log_b \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i),$$

Where *b* is the base of the logarithm used. Common values of *b* are 2.

2.3. Average Length

The Average length of the Huffman coding is calculated by the multiplication of probability with its codeword length. The average code length (average number of bits per symbol) of the source is defined as

$$\text{Avg. Length} = \sum_{i=1}^{n} \text{Pi*li}$$

3. BUILDING DOUBLE HUFFMAN CODE

Example:"HUFFMAN"

Here in That Example we have 6 different alphabets [A, F, H, M, N, U].

P (A) =1/7
P (F) =2/7
P (H) =1/7
P (M) =1/7
P (N) =1/7
P (U) =1/7

Apply the Algorithm Discussed in 2.1 Sections with Step 4.

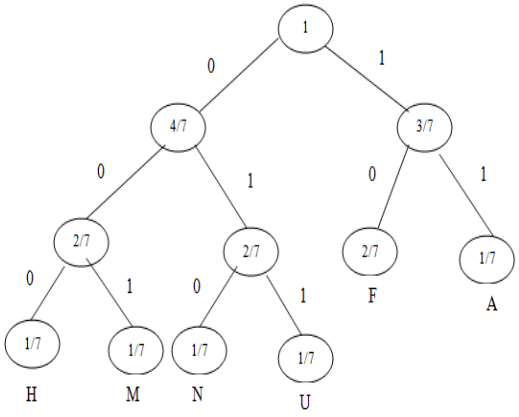

Figure 1 Huffman Tree

After Apply the algorithm Codeword for the Alphabets are:-

A=11
F=10
H=000
M=001
N=010
U=011

$$\text{Entropy} = -\sum_{i=1}^{6} P(x_i) \log_2 (x_i)$$

$=-1/7 \log_2 (1/7) -2/7 \log_2 (2/7) -1/7 \log_2 (1/7) -1/7 \log_2 (1/7) -1/7 \log_2 (1/7) -1/7 \log_2 (1/7)$

= 2.516 bits/Sec

$$\text{Average Length} = \sum_{i=1}^{6} P(x_i) * l_i$$

= (1/7)*2+ (2/7)*2+ (1/7)*3+ (1/7)*3+ (1/7)*3+ (1/7)*3

= 2.5713 bits/sec

Redundancy=Entropy ~ Average Length

=2.516-2.5713

=0.0553 bits/sec

Concatenate the Alphabet with Codeword, so the Codeword String is "1110000001010011"

This String will contain only two binary numbers 0 and 1. So the Probability for these numbers is:-

P (0) =9/16

P (1) =7/16

Repeat the Procedure from Step 1-4 Discussed in section 2.1

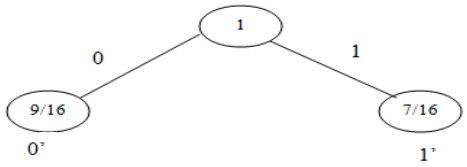

Figure 2 Binary Huffman Tree

$$\text{Entropy} = -\sum_{i=1}^{2} P(x_i) \log_2 (x_i)$$

$=-9/16 \log_2 (9/16) - 7/16 \log_2 (7/16)$

= 0.9886 bits/Sec

$$\text{Average Length} = \sum_{i=1}^{2} P(x_i) * l_i$$

= (9/16)*1+ (7/16)*1

=1 bits/sec\

Redundancy=Entropy ~ Average Length

=0.9886-1

=0.0114 bits/sec

After applying Huffman Coding

P-Probability, C-Codeword, CL-Codeword Length, E-Entropy, AL-Average Length, R-Redundancy

HUFFMAN					
P	**C**	**CL**	**E**	**AL**	**R**
P(A)=1/7	11	2	2.5160 bits/Sec	2.5713 bits/sec	0.0553 bits/sec
P (F)=2/7	10	2			
P(H)=1/7	000	3			
P(M)=1/7	001	3			
P(N)=1/7	010	3			
P(U)=1/7	011	3			

Table 1.1 Huffman Coding

After Applying Proposed Double Huffman Coding

HUFFMAN=0000111000111010					
P	**C**	**CL**	**E**	**AL**	**R**
P (0)= 9/16	0	1	0.9886 bits/Sec	1 bits/sec	0.0114 bits/sec
P (1)= 7/16	1	1			

Table 1.2 Double Huffman Coding

4. RESULT ANALYSIS

We perform the Huffman Algorithm and after that the new proposed double Huffman algorithm. The previous Huffman Algorithm will work on the Symbol or Alphabets which gives the better result but after applying the new proposed double Huffman algorithm on previous Huffman algorithm we got the best result.

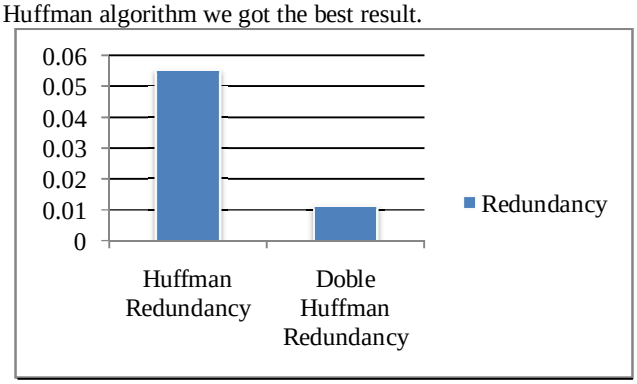

Figure 3 Redundancy Graph

Here we make a comparison of Huffman algorithm and double Huffman algorithm. In the Huffman algorithm we got 0.0553 bits/sec redundancy while in Double Huffman algorithm we got 0.0114 bits/sec redundancy which is smaller than the normal Huffman algorithm.

If we find out the efficiency of the new result than, it is more than 75% better than the normal Huffman algorithm. For this example the Efficiency

= ((0.0553-0.0114)/0.0553)*100

= 79.38%

5. CONCLUSION

Huffman Coding is a widely used technology in image, file, audio, etc. By using Huffman Coding we compress the data and find out the redundancy. But using Double Huffman Coding we reduce the redundancy of the data and the efficiency of the system is increase also. Because we know that space allocation is costly so using this double Huffman coding we save the space and increase the performance of the System.

REFERENCES

[1] M Nelson and J-L Gaily, "*The Data Compression Book*", 2nd edn., MIS Press, 1995.

[2] V Bhaskaran and K Konstantinides. "*Image and Video Compression Standards: Algorithms Architectures*". Kluwer Academic Publishers, 1997.

[3] R M Capocelli, R Giancarlo and I J Taneja, "*Bounds on the redundancy of Huffman codes*", IEEE Transactions on Information Theory, IT-32, pp 854-857, 1986.

[4] A Turpin and A Moffat, "*Housekeeping for Prefix Coding*", IEEE Transactions on Communications, 48, pp.622- 628, 2000.

[5] D. Huffman, "*A method for the construction of minimum redundancy codes*", Proc. Of the Institute of Radio Engineers, Vol 40, pp 1098-1101, 1952.

AUTHOR

Pankaj Kumar: was born in Uttar Pradesh India in November, 1990. He has completed B.Tech in Computer Science and Engineering from S.R.M.S. CET, Bareilly Uttar Pradesh, India in 2011. His interest areas are Data Compression, Network Security, Cryptography and Algorithm Analysis.

Ankur Kumar Varshney: was born in India in August, 1986. He is pursuing M.Tech in Computer Engineering from B.S.A College of Engineering and Technology, Mathura, India. He is currently working as an Assistant Professor in ITM Aligarh His interest areas are Advanced Digital Signal Processing, Data Compression.

