DATA COMPRESSION USING HUFFMAN CODING

B.TECH SEM V

IN

ELECTRONICS & COMMUNICATION ENGINEERING

2EC503

DIGITAL COMMUNICATION

SUBMITTED BY:

SHUBHAM SONI (18BEC110)

GUIDED BY:

DR. YOGESH TRIVEDI

CONTENTS

Introduction

In the recent years, the use of digital audio and high quality video has replaced DVDs and CD-quality digital audio. The emerging applications in the various domains like multimedia systems, mobile and wireless networks face a series of problems like channel bandwidth and limited storage. This has created a huge demand for data compression. As a result, many compression techniques have evolved and the most popular and used compression technique is Huffman coding. The lossless data compression techniques which include DEFLATE and GZIP prominently use Huffman encoding method as the primary tool for compression.

The data compression term refers to process where there is use of fewer bits than an uncoded representation using various encoding schemes. It is a technology for reducing the size of data storage keeping in mind that the quality of the data is not reduced excessively. To compress something means we need to decrease the size of the data, thus provides reduction in storage space, transmission time and the bandwidth used. A popular method is to reduce the redundant information which means coding only one bit instead of repeatation.

Compression techniques can be divided into two types: 1) Lossy and 2) Lossless compression. Huffman coding is a lossless compression technique which is used to reduce the number of bits which depends on the probability of occurrence. For image compression, the information is removed from the image upto an extent that the human eye can tolerate or cannot identify the minute differences between the compressed and original image.

Huffman coding is especially used in applications where the loss of information is not tolerated like text or image. It is also known as prefix coding or prefix elimination and is based on the frequency of occurrence of a data item. i.e pixels in images. Huffman coding is also popular in telecom networks. For eg: In an office if one person wants to make a call to another person working in the same office, he/she has to make a call by dialing a minimum number on the telephone. These codes are less in number so as to make sufficient numbers for every employee in the company. These codes are generated by Huffman algorithm, where it helps to find the prefix code and eliminating the same.

Fundamentals of Compression

Types

There are two types of compression methods:

1) Lossless

Lossless compression means while compression or after compression “no data is lost”. The result of the compression is exactly the same or bit-for-bit perfect match of the original data. The data is saved more efficiently in a more efficient way and nothing is removed.

2) Lossy

When some data is lost due to the compression it is known as lossy compression. Lossy compression is based on the assumption that the original input data files save more information than what human beings can perceive. Thus the irrelevant data can be removed.

The color representation

The primary colors of an image are red, green and blue light sources and it also consists of various colors of different intensities and brightness. White color is formed with the presence of all these three primary colors at the maximum intensities. This color perception phenomenon is caused by the way that the human eye detects and processes light which makes it possible to represent an image as the source of three primary signals at the maximum intensities in the two spatial dimensions.

Digital Data representation

A finite alphabet consists of a sequence of symbols which further consists of digital data. In an uncompressed data there is a special representation which codes each symbol using the same number of bits. When the average length per symbol is less than that of the standard representation, compression is achieved. For the compression to be meaningful a standard representation should be defined for the data to be compressed.

Digitization

For the computers to understand an image which is captured by a light sensor, it first needs to be digitized. Digitalization consists of three major steps 1) Spatial Sampling 2) Temporal Sampling 3) Quantization

1) Spatial Sampling:

Raster Scanning is a process in which the two dimensional sets of sampling points are converted to a one dimensional set. In spatial sampling, the underlying analog signals are measured at a finite set of sampling points in a finite viewing area

2) Temporal Sampling:

Human eyes can only see a motion picture if minimum 16 samples per second are taken at each grid point. The temporal sampling is mainly performed in motion estimation for video sequences. It is typically performed at a rate of 24 frames/sec and this observation is the basis of motion picture technology.

3) Quantization:

Digital Signal Processing does not support the continuous intensity values and one more step is needed after spatial and temporal sampling. Thus the continuous intensity values are converted to discrete values which is known as quantization. It can be viewed as a mapping from a continuous domain to a discrete range. A particular quantization mapping is called a quantizer.

Redundancy

It exists in two forms:

1) Temporal
2) Spatial

Temporal redundancy is also known as intraframe redundancy which refers to redundancy that exists in a single frame of video or image and the latter is also known as interframe redundancy which means the redundancy that exists between two consecutive frames in a video sequence.

Huffman Algorithm Compression Technique

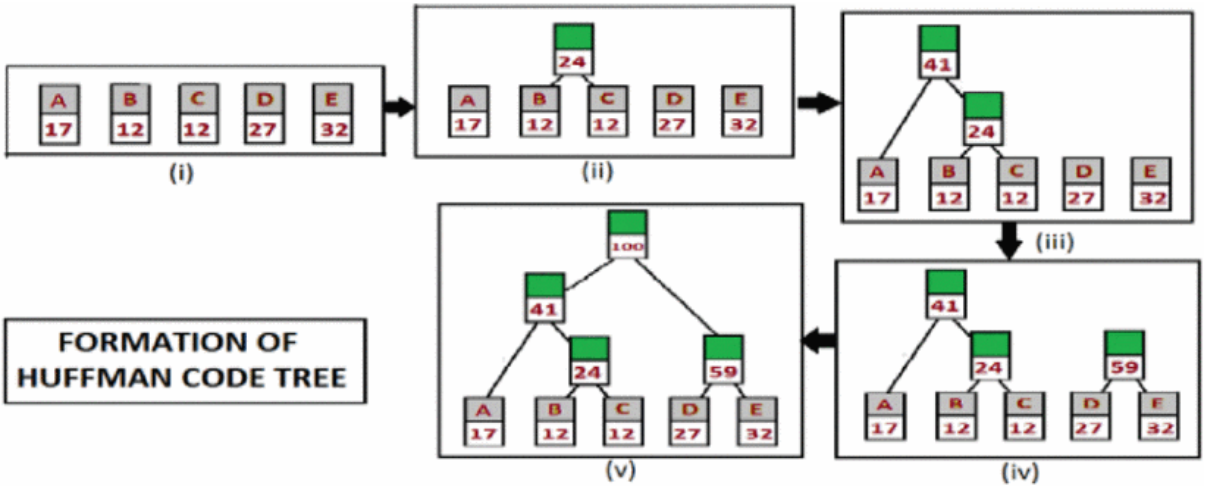
The main principle in Huffman coding is that it is based on the frequency or occurrence of pixels in images. If the data occurs more frequently a lower number of bits are used to encode the data. For each image, a codebook is constructed and in all cases the codebook and encoded data. The steps to generate an encoded sequence using Huffman Algorithm are as follows:

- The first step is to achieve the total frequency of occurrence of each character in a message or frequency of pixels in an image. This can be done with the help of two methods, one is to create fixed look-up table which can be directly given on the encoder and decoder side and other is the method in which a running estimate is maintained for each letter frequency.

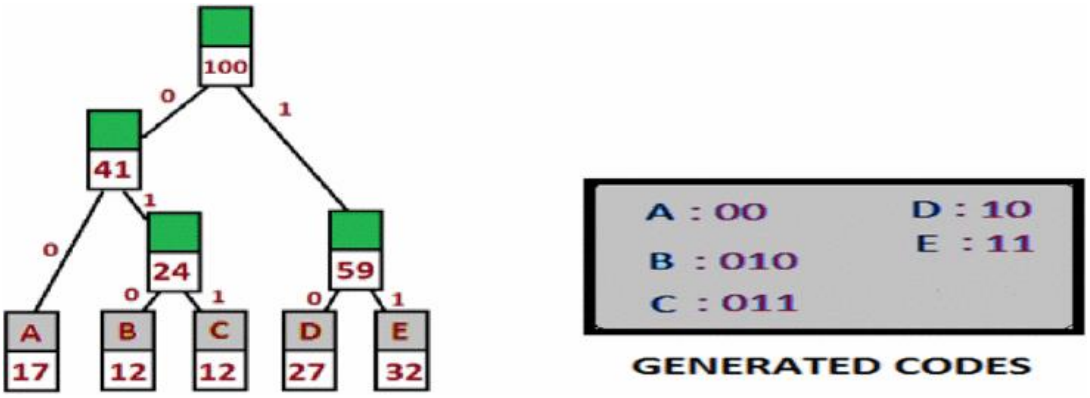
- The previous method explained above is known as static Huffman encoding and the later method explained is known as adaptive or dynamic Huffman encoding. Thus, these processes differ in acquiring the frequency table. Consider a text compression example for easy understanding where a message having characters A, B, C, D, and E with their frequencies 17, 12, 12, 27, and 32 respectively.

Character	A	B	C	D	E
Frequency	17	12	12	27	32

- In the second step, a Huffman code tree is formed as shown below:



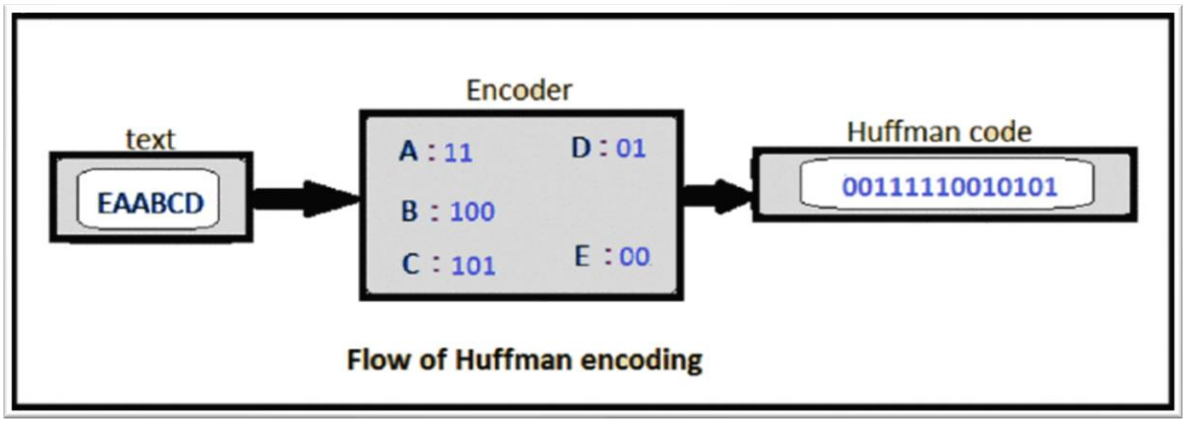
The method starts with the leaf nodes which are characters or symbols used in the message and the method starts with replacing the nodes having minimum frequency with a new parental node. The sum of the frequencies of the two leaf nodes are assigned to the parental node. In the example which we considered the leaf nodes B and C have the minimum frequency i.e 12, so they are replaced by a parental node having the sum of the frequencies i.e $12 + 12 = 24$. In the next step again the nodes having minimum frequency are replaced by the sum and a tree is formed until a single parental node is obtained. Sorting operation can be included at every step so that better accuracy is obtained.



After the parental nodes have been assigned to each pair of nodes, in the 3rd step bit '0' or '2' are assigned to the branches of the tree and the left branches are usually assigned with bit '0' while the right branches are assigned with bit '1' which depends on the on-paper structure)

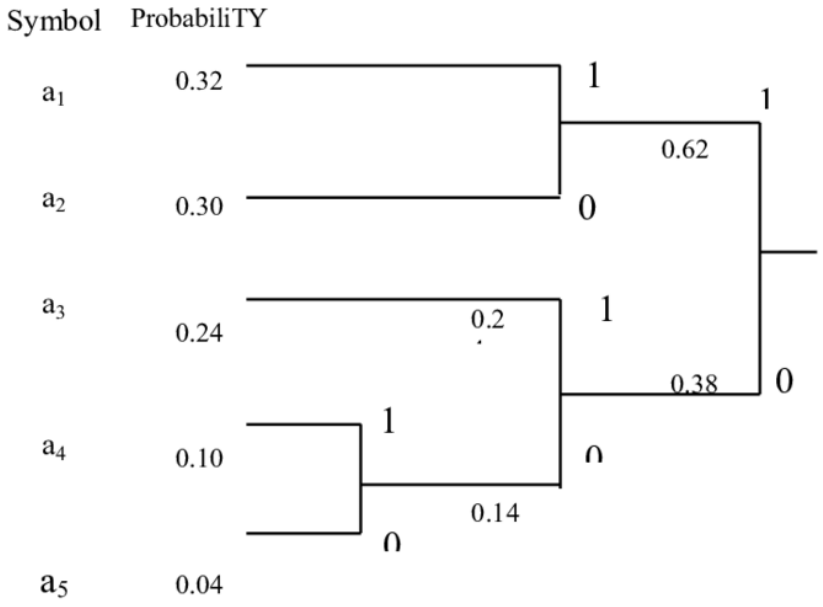
The final step is to trace down the generated Huffman codes for every symbol, starting from the single parental node down to every character the zeroes and ones are amended.

The below diagram shows the block diagram of Huffman coding algorithm where at every step the encoded sequence is generated.



For a collection of symbols with a uniform probability distribution and variety of members that may be a power of two, Huffman coding is equal to easy binary block encoding.

Eg. Consider alphabets having probability ={0.32,0.30,0.24,0.1,0.04} for symbols {a1,a2,a3,..,a5}. Here we can find the efficiency and variance.



Symbol	Codeword	Length
a1	11	2
a2	10	2
a3	01	2
a4	001	3
a5	000	3

Entropy:

$$H = \Sigma p_i log_2(\frac{1}{p_i})$$

$$= 0.32\ log_2\left(\frac{1}{0.32}\right) + 0.30log_2\left(\frac{1}{0.30}\right) + 0.24log_2\left(\frac{1}{0.24}\right) + 0.10log_2\left(\frac{1}{0.10}\right) + 0.04log_2\left(\frac{1}{0.04}\right)$$

= 0.72

Average codeword length

$$L = \Sigma p_i n_i$$

$$= 2 * 0.32 + 2 * 0.30 + 2 * 0.24 + 2 * 0.10 + 2 * 0.04$$

= 2

Adaptive Huffman Coding

When the encoder does not have any prior knowledge of the data as in multimedia systems where the future data is unknown for eg: live audio or videos, Adaptive Huffman coding is used. Adaptive Huffman coding or Dynamic Huffman coding is an adaptive coding technique where the symbols which are being transmitted, have no initial information of supply distribution, that allows one-pass cryptography and adaptation to dynamic conditions in information. In Adaptive Huffman coding, statistics are gathered and updated dynamically as the datastream arrives.

Mathematical Evaluation

The main aim in Huffman coding is to generate minimum codes for every character depending on the occurrence of the character and compressing the whole file.

When an encoder uses fixed length encoding method to encode a message, the total number of bits which are needed for the encoding of the message is given by the product of frequency of the letter and the fixed bit length. For eg: The total number of bits needed to encode a message is 3 for three bit encoding.

The following table shows the total number of bits needed for the encoding of the example discussed previously in the report using three bit encoding.

Symbol/char	Freq of Occurrence	Bit pattern	Total bits
A	17	000	17*3 = 51
B	12	001	12*3 = 36
C	12	010	12*3 = 36
D	27	011	27*3 = 81
E	32	100	32*3 = 96
			TOTAL = 300 BITS

Average length is defined as the number of bits needed to encoded the complete message which is given by $L = \Sigma f_i n_i$ where f is the frequency and n is the bit length. The total bits for encoding using three bit encoding is 300 bits.

Symbol/char	Frequency of occurrence	Huffman code	Code length	Total bits
A	17	11	2	17*2= 34
B	12	100	3	12*3= 36
C	12	101	3	12*3= 36
D	27	01	2	27*2= 54
E	32	00	2	32*2= 64
				Total= 224 bits

From the above table, one can see that using Huffman coding on the same text compression example the total number of bits has reduced to 224 bits.

Text Compression using Huffman coding in MATLAB

Matlab Code:

```
%%encoding and decoding of a paragraph

clear all;

text=char('Digital Communication is a topic which deals
with physical transfer of data and it indeed is a very
interesting topic! ');
t=double(text); %converting text data into ascii value
N=length(t);
frequency=zeros(1,128);

for k=0:127
     count=0;
     for i=1:N
     if(t(i)==k)
          count=count+1;
     end
end
frequency(k+1)=count; %count of frequency of each
character
end
sym=find(frequency)-1;  % it will get info about used
character in the line

sym_count=frequency(sym+1); % frequency of used symbols
sym_prob=sym_count/N;  %  probability of symbol

%% creating huffman code for symbol
% creating the dictionary corresponding to each used
symbol
[dict,avglen]=huffmandict(sym,sym_prob);
entropy= avglen;
entropy
enco=huffmanenco(t,dict) %encoding the data
 b=huffmandeco(enco,dict) %decoding the data
text=char(b)
L= length(text);
P=length(enco);
CompressionRatio= L*7/P
```

Results:

In the below output, the average length of the encoded text and the encoded prints in the matrix are printed**.**

```
>> dcom_sessional

averagelength =

    4.1092


enco =

  Columns 1 through 21

     1     1     0     0     1     0     1     0     1     1     0     1     0

  Columns 22 through 42

     1     0     0     1     0     1     1     0     0     0     0     0     0

  Columns 43 through 63

     1     0     1     0     1     0     0     1     0     0     1     0     0

  Columns 64 through 84

     0     1     0     0     0     0     1     1     0     0     1     1     1

text =

    'Digital Communication is a topic which deals with physical transfer of data and it indeed is a very interesting topic! '


CompressionRatio =

    1.7035
```

Thus from the above results we can say that the total number of bits before compression or huffman encoding is 1.7 times more than after the line is encoded.

Image Compression using Huffman coding in MATLAB

Matlab Code:

```
% read the image
Image =
imread('C:\Users\shubh\Documents\DCOM\samplephoto.jpg');

% calculate the frequency of each pixel
[frequency,pixelValue] = imhist(Image());

% sum all the frequencies
```

```
tf = sum(frequency) ;

% calculate the frequency of each pixel
probability = frequency ./ tf ;

% create a dictionary
dict = huffmandict(pixelValue,probability);

% get the image pixels in 1D array
imageOneD = Image(:) ;

% encoding
testVal = imageOneD ;
encodedVal = huffmanenco(testVal,dict);

% decoding
decodedVal = huffmandeco(encodedVal,dict);

% display the length
kb = 8 * 1024 ;
disp(numel(de2bi(testVal))/kb) ;
disp(numel(encodedVal)/kb) ;
disp(numel(de2bi(decodedVal))/kb) ;

% get the original image from 1D Array
[rows, columns, numberOfColorChannels] = size(Image);
oi = reshape(testVal,[rows, columns,
numberOfColorChannels]) ;
imwrite(oi,'C:\Users\shubh\Documents\DCOM\original.jpg');

% get the decoded image from 1D Array
decodedVal = uint8(encodedVal);
ci = reshape(decodedVal,[rows, columns,
numberOfColorChannels]) ;
imwrite(ci,'C:\Users\shubh\Documents\DCOM\decoded.jpg');
```

Results:

```
>> dcom_sessional2
   2.2148e+03

   2.1225e+03

   2.2148e+03
```

From the above outcome, one can see that the bit length has decreased after the Huffman encoding of the photo and once it has been decoded the same number

of bits are again achieved and the quality of the image is not degarded.

Figure 1 Original Photo

Figure 2 Decoded Photo

Advantages of Huffman Coding

- Due to its property of lossless compression of images, it is preferred over other techniques.
- It generates shorter binary codes for encoding symbols/characters that appear more frequently in the input string.
- The algorithm of Huffman coding is easy to implement.

Disadvantages of Huffman Coding

- Algorithmic programs vary with completely different formats, but few get any higher than 8:1 compression.
- Compression of image files that contain long runs of identical pixels by Huffman isn't as economical when compared to RLE method.
- The Huffman encryption method is sometimes worn out two passes. Throughout the primary pass, an applied mathematics model is made, then within the second pass the image information is encoded and supports the generated model. From here we are able to see that Huffman encoding could be a comparatively slow method as time is required to create the applied mathematics model so as to archive associate in Nursing economical compression rate.
- It is needed to send the Huffman table at the beginning of the compressed file, otherwise the decompressor won't be ready to rewrite it. This causes overhead.

Conclusion

In this assignment, the need of the compression was discussed in the introduction. Further the basics of compression like digitalization, digital data representation, color data representation and the types of compressions were discussed. Thereafter the compression technique and the procedure to follow are seen. The definition and concept of adaptive huffman coding is also learnt in the report and Two MATLAB codes on compression of text and image are done and the interpretation of the outputs obtained are mentioned. Finally, the advantages and disadvantages of Huffman coding are considered.

References

1.Nidhi Dhawale, “Implementation of Huffman algorithm and study for optimization” IEEE 2010

2. Mamta Sharma, “Compression using Huffman Coding” IEEE 2014

3. www.mathworks.com