Compression Using Huffman Coding

Mamta Sharma

S.L. Bawa D.A.V. college,

Abstract

Data compression is also called as **source coding.** It is the process of encoding information using fewer bits than an uncoded representation is also making a use of specific encoding schemes. Compression is a technology for reducing the quantity of data used to represent any content without excessively reducing the quality of the picture. It also reduces the number of bits required to store and/or transmit digital media. Compression is a technique that makes storing easier for large amount of data. There are various techniques available for compression in my paper work , I have analyzed Huffman algorithm and compare it with other common compression techniques like Arithmetic, LZW and Run Length Encoding.

Keywords:

LZW, Huffman, DCT, RLE, JPEG, MPEG, Compression Formats, Quantization, Wavelets.

1. INTRODUCTION

Compression refers to reducing the quantity of data used to represent a file, image or video content without excessively reducing the quality of the original data. It also reduces the number of bits required to store and/or transmit digital media. To compress something means that you have a piece of data and you decrease its size. There are different techniques who to do that and they all have their own advantages and disadvantages. One trick is to reduce redundant information, meaning saving sometimes once instead of 6 times. Another one is to find out which parts of the data are not really important and just leave those away. Arithmetic coding is a technique for lossless data compression. It is a form of variable-length entropy encoding. Arithmetic coding encodes the entire message into a single number, a fraction n where $(0.0 \leq n < 1.0)$ but where other entropy encoding techniques separate the input message into its component symbols and replace each symbol with a code word. Run length encoding method is frequently applied to images or pixels in a scan line. It is a small compression component used in JPEG compression

Huffman coding is a loseless data compression technique. Huffman coding is based on the frequency of occurrence of a data item i.e. pixel in images. The technique is to use a lower number of bits to encode the data in to binary codes that occurs more frequently. It is used in JPEG files. JPEG 2000 is a wavelet-based image compression standard. Wavelets are functions that satisfy certain mathematical requirements and are used in representing data or other functions. LZW compression replaces strings of characters with single codes. Compression occurs when a single code is output instead of a string of characters. The comparison between various compression algorithms helps us to use a suitable technique for various applications.

2. FUNDAMENTALS FOR COMPRESSION

2.1 Types

Lossy compression means that some data is lost when it is decompressed. Lossy compression bases on the assumption that the current data files save more information than human beings can "perceive". Thus the irrelevant data can be removed.

Lossless compression means that when the data is decompressed, the result is a bit-for-bit perfect match with the original one. The name lossless means "no data is lost", the data is only saved more efficiently in its compressed state, but nothing of it is removed.

2.2 Digital data representation Digital data consist of a sequence of symbols chosen from a finite alphabet. In order for data compression to be meaningful, there is a standard representation for the uncompressed data that codes each symbol using the same number of bits. Compression is achieved when the data can be represented with an average length per symbol that is less than that of the standard representation. Therefore for compression to be meaningful, a standard representation should be defined for the data to be compressed.

2.3 Color representation An image consists of various colors of different intensities and brightness. Red, green and blue light sources forms a set of primary colors; this is an addictive system since the presence of all the primary colors, all set to their maximum intensities, results in the perception of the color white. This phenomenon of color perception is caused by the way that the human eye detects and processes light, which makes it possible to

Manuscript received May 5, 2010
Manuscript revised May 20, 2010

represent an image as a set of three intensity signals in two spatial dimensions.

2.4 Digitization In order to be processed by computers, an image that is captured by a light sensor must first be digitized. Digitization consists of three steps:1) spatial sampling,2) temporal sampling, and 3) quantization.

1) **Spatial sampling**: Spatial sampling consists of taking measurements of the underlying analog signal at a finite set of sampling points in a finite viewing area. The two dimensional sets of sampling points are transformed into a one-dimensional set through a process called raster scanning. The two main ways to perform raster scanning are progressive and interlaced.

2) **Temporal sampling**: It is performed mainly in motion estimation for video sequences. The human visual system is relatively slow in responding to temporal changes. By taking at least 16 samples per second at each grid point, an illusion of motion is maintained. This observation is the basis for motion picture technology, which typically performs temporal sampling at a rate of 24 frames/sec.

3) **Quantization**: After spatial and temporal sampling, the image consists of a sequence of continuous intensity values. The continuous intensity values are incompatible with digital processing, and one more step is needed before this information is processed by a digital computer. The continuous intensity values are converted to a discrete set of values in a process called quantization. Quantization can be viewed as a mapping from a continuous domain to a discrete range. A particular quantization mapping is called a quantizer.

2.5 Redundancy Redundancy exists in two forms: spatial and temporal.

The former, also called intraframe redundancy, refers to the redundancy that exists within a single frame of video or image, while the latter, also called interface redundancy, refers to the redundancy that exists between consecutive frames within a video sequence.

2.6 Vector Quantization In vector quantization, an image is segmented into same-sized blocks of pixel values. The blocks are represented by a fixed number of vectors called code words. The code words are chosen from a finite set called a codebook. the size of the codebook affects the rate as well as the distortion while a smaller codebook has the opposite effects.

2.7 Discrete Cosine Transform Discrete cosine transform (DCT) is a Fourier-related transform[41] similar to the discrete Fourier transform[42](DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality). The DCT is similar to the discrete Fourier transform, it transforms a signal or image from the spatial domain to the frequency domain.

3. COMPRESSION TECHNIQUES

3.1 SIMPLE REPETITION

If in a sequence a series on *n* successive tokens appears we can replace these with a token and a count number of occurrences. We usually need to have a special flag to denote when the repeated token appears

For Example

879000000000000000000000000000000

We can replace with

879f32

Where f is the flag for zero.

Compression savings depend on the content of the data.

Applications:

1. Suppression of zero's in a file **(**Zero Length Suppression) .

2. Silence in audio data, or pauses in conversation etc**.**

3. Bitmaps

4. Blanks in text or program source files

5. Backgrounds in images

6. Other regular image or data tokens

3.2 RLE

This encoding method [34] is frequently applied to images (or pixels in a scan line). In this instance, sequences of

image elements (X1,X2…….,Xn)are mapped to pairs (c1,ℓ1),(c2,ℓ2),……..,(cn,ℓn) where c_i represent image intensity or color and l_i the length of the ith run of pixels (Not dissimilar to zero length suppression above). The savings are dependent on the data. In the worst case (Random Noise) encoding is heavier than original file.

Applications:

It is a small compression component used in JPEG compression.

3.3 PATTERN SUBSTITUTION

This is a simple form of statistical encoding. Here we substitute a frequently repeating pattern(s) with a code. The code is shorter than pattern giving us compression.

More typically tokens are assigned according to frequency of occurrence of patterns:

- Count occurrence of tokens
- Sort in Descending order
- Assign some symbols to highest count tokens

A predefined symbol table may used i.e assign code *i* to token *i*.

3.4 ENTROPY ENCODING

Lossless compression frequently involves some form of entropy encoding and is based on information theoretic techniques. Shannon is father of information theory. Some entropy encoding techniques are discussed below.

3.5 THE SHANNON-FANO ALGORITHM

This is a basic information theoretic algorithm. A simple example will be used to illustrate the algorithm:

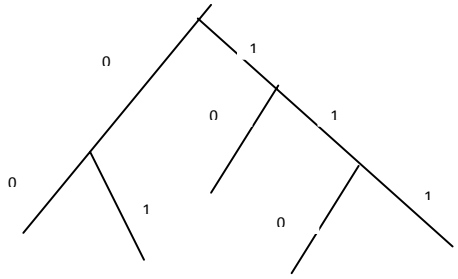
Symbol	A	B	C	D	E
Count	15	7	6	6	5

Encoding for the Shannon-Fano Algorithm

It follows a top-down approach :

1. Sort symbols according to their frequencies/probabilities, e.g., ABCDE.

2. Recursively divides into two parts, each with approx. same number of counts.



Symbol	Count	log (1/p)	Code	Subtotal (# of bits)
A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15

TOTAL (# of bits): 89

3.6 HUFFMAN CODING

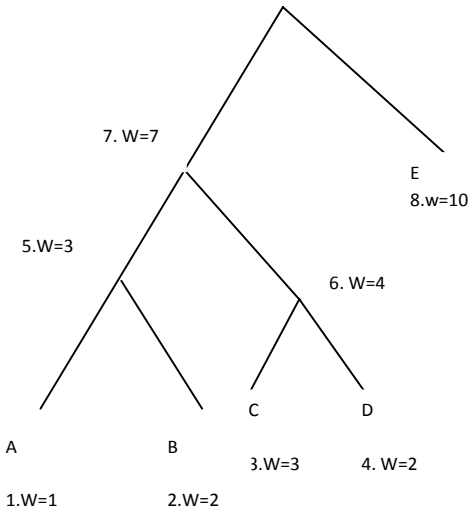
Huffman coding[24] is based on the frequency of occurrence of a data item (pixel in images). The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a Code Book which may be constructed for each image or a set of images. In all cases the code book plus encoded data must be transmitted to enable decoding.

3.7 ADAPTIVE HUFFMAN CODING

The key is to have both encoder and decoder to use exactly the same *initialization* and update model routines. Update model does two things: (a) increment the count, (b) update the Huffman tree.[25] During the updates, the Huffman tree will be maintained its *sibling property*, i.e.

the nodes (internal and leaf) are arranged in order of increasing weights (see figure).

When swapping is necessary, the farthest node with weight W is swapped with the node whose weight has just been increased to W+1.**Note:** If the node with weight W has a subtree beneath it, then the subtree will go with it. The Huffman tree could look very different after node swapping.



3.8 ARITHMETIC CODING

Huffman coding and the like use an integer number (k) of bits for each symbol, hence k is never less than 1. Sometimes, e.g., when sending a 1-bit image, compression becomes impossible. [15]

3.9 LZW

LZW compression replaces strings of characters with single codes. It does not do any analysis of the incoming text. Instead, it just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters.

The code that the LZW algorithm outputs can be of any arbitrary length, but it must have more bits in it than a single character. The first 256 codes (when using eight bit characters) are by default assigned to the standard character set. The remaining codes are assigned to strings as the algorithm proceeds. The sample program runs as shown with 12 bit codes. This means codes 0-255 refer to individual bytes, while codes 256-4095 refer to substrings.

LZW compression works best for files containing lots of repetitive data. This is often the case with text and monochrome images. Files that are compressed but that do not contain any repetitive information at all can even grow bigger!

Advantages and Disadvantages:

LZW compression is fast.

Royalties have to be paid to use LZW compression algorithms within applications (see below).

Applications:

LZW compression can be used in a variety of file formats[30]:

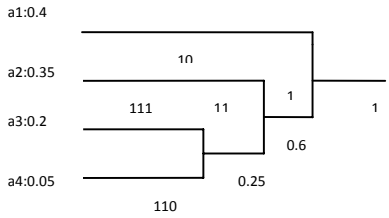
- TIFF files
- GIF files

4. HUFFMAN ALGORITHM

Huffman coding is an entropy encoding algorithm used for lossless data compression in computer science and information theory. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix-free code (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters 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: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to do this in linear time if input probabilities (also known as *weights*) are sorted.

For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding [40]e.g., ASCII coding.



BASIC TECHNIQUE

Assume you have a source generating 4 different symbols $\{a_1,a_2,a_3,a_4\}$ with probability {0.4;0.35;0.2;0.05}. Generate a binary tree from left to right taking the two less probable symbols, putting them together to form another equivalent symbol having a probability that equals the sum of the two symbols. Keep on doing it until you have just one symbol. Then read the tree backwards, from right to left, assigning different bits to different branches.

The final huffman code is:

SYMBOL	CODE
a1	0
a2	10
a3	111
a4	110

The technique works by creating a binary tree of nodes. These can be stored in a regular array, the size of which depends on the number of symbols(N). A node can be either a leaf node or an internal node. Initially, all nodes are leaf nodes, which contain the symbol itself, the weight (frequency of appearance) of the symbol and optionally, a link to a parent node which makes it easy to read the code (in reverse) starting from a leaf node. Internal nodes contain symbol weight, links to two child nodes and the optional link to a parent node. As a common convention, bit '0' represents following the left child and bit '1' represents following the right child. A finished tree has N leaf nodes and N−1 internal nodes.

A linear-time* method to create a Huffman tree is to use two queues, the first one containing the initial weights (along with pointers to the associated leaves), and combined weights (along with pointers to the trees) being put in the back of the second queue. This assures that the lowest weight is always kept at the front of one of the two queues.

Creating the tree:

1. Start with as many leaves as there are symbols.
2. Enqueue all leaf nodes into the first queue (by probability in increasing order so that the least likely item is in the head of the queue).
3. While there is more than one node in the queues:
    1. Dequeue the two nodes with the lowest weight.
    2. Create a new internal node, with the two just-removed nodes as children (either node can be either child) and the sum of their weights as the new weight.
    3. Enqueue the new node into the rear of the second queue.
4. The remaining node is the root node; the tree has now been generated.

This is implimented using the following function:

```
short create_tree()

{

   void find_lowest_freqs(void);

   short only_one_up_ptr_left(void);

   double maxfreq = 0 ;

   struct chardata *new_node = NULL;

   fprintf(fpp,"Creating tree from frequencies...");



   while (maxfreq < 0.99999 )

   {

      find_lowest_freqs();

    if ((new_node = (struct chardata *)malloc(sizeof
(struct chardata)) )

         == NULL

        )
```

```
    {
      printf(fpp,"Insufficient memory, malloc() failed in create_tree().")
      ;
      return FALSE;
    }
    new_node->up    = NULL;
    new_node->left  = ptr2;
    new_node->right = ptr1;
    new_node->charnum = -1;
    ptr1->up = new_node;
    ptr2->up = new_node;
    new_node->frequency = ptr1->frequency + ptr2->frequency;
    maxfreq = new_node->frequency;
#ifdef VERBOSE
    fprintf(fpp,"Newly created freq == %f\n", maxfreq);
#endif
  }
  root = new_node;
  if (only_one_up_ptr_left())
  {
    fprintf(fpp,"Done creating tree.");
#ifdef verbose
    fprintf(fpp,"Win: apparently only one remaining up-pointer.");
#endif
  }
  else
  {
    fprintf(fpp,"Lose: apparently more than one remaining up-pointer.");
    return FALSE;
  }
  return TRUE;
}
```

It is generally beneficial to minimize the variance of codeword length. For example, a communication buffer receiving Huffman-encoded data may need to be larger to deal with especially long symbols if the tree is especially unbalanced. To minimize variance, simply break ties between queues by choosing the item in the first queue. This modification will retain the mathematical optimality of the Huffman coding while both minimizing variance and minimizing the length of the longest character code.

This method is linear time assuming that you already have the leaf nodes sorted by initial weight. If not, sorting them will take $O(n \log n)$ time.

4.1 MAIN PROPERTIES

1. Unique Prefix Property: no code is a prefix to any other code (all symbols are at the leaf nodes) -> great for decoder, unambiguous.

2. If prior statistics are available and accurate, then Huffman coding is very good

3. The frequencies used can be generic ones for the application domain that are based on average experience, or they can be the actual frequencies found in the text being compressed.

4. Huffman coding is optimal when the probability of each input symbol is a negative power of two.

5. The worst case for Huffman coding can happen when the probability of a symbol 6

ceeds $2^{-1} = 0.5$, making the upper limit of inefficiency unbounded. These situations often respond well to a form of blocking called run-length encoding.

4.2 APPLICATIONS

1. Arithmetic coding can be viewed as a generalization of Huffman coding; indeed, in practice arithmetic coding is often preceded by Huffman coding, as it is easier to find an arithmetic code for a binary input than for a nonbinary input.

2. Huffman coding is in wide use because of its simplicity, high speed and lack of encumbrance by patents.

3. Huffman coding today is often used as a "back-end" to some other compression method. DEFLATE (PKZIP's algorithm) and multimedia codecs such as JPEG and MP3 have a front-end model and quantization followed by Huffman coding.

4.3 ADVANTAGES

- Algorithm is easy to implement
- Produce a lossless compression of images

4.4 DISADVANTAGES

- Efficiency depends on the accuracy of the statistical model used and type of image.
- Algorithm varies with different formats, but few get any better than 8:1 compression.
- Compression of image files that contain long runs of identical pixels by Huffman is not as efficient when compared to RLE.
- The Huffman encoding process is usually done in two passes. During the first pass, a statistical model is built, and then in the second pass the image data is encoded based on the generated model. From here we can see that Huffman encoding is a relatively slow process as time is required to build the statistical model in order to archive an efficient compression rate.
- Another disadvantage of Huffman is that, all codes of the encoded data are of different sizes (not of fixed length). Therefore it is very difficult for the decoder to know that it has reached the last bit of a code, and the only way for it to know is by following the paths of the up-side down tree and coming to an end of it (one of the branch). Thus, if the encoded data is corrupted with additional bits added or bits missing, then whatever that is decoded will be wrong values, and the final image displayed will be garbage.
- It is required to send Huffman table at the beginning of the compressed file ,otherwise the decompressor will not be able to decode it. This causes overhead.

4.5 ADAPTIVE HUFFMAN CODING

Adaptive huffman coding (Dynamic Huffman coding) is an adaptive coding technique based on Huffman coding, building the code as the symbols are being transmitted, having no initial knowledge of source distribution, that allows one-pass encoding and adaptation to changing conditions in data.

The benefit of one-pass procedure is that the source can be encoded in real time, though it becomes more sensitive to transmission errors, since just a single loss ruins the whole code.

CONCLUSIONs

I have studied various techniques for compression and compare them on the basis of their use in different applications and their advantages and disadvantages. I have concluded that arithmetic coding is very efficient for more frequently occurring sequences of pixels with fewer bits and reduces the file size dramatically.RLE is simple to implement and fast o execute. LZW algorithm is better to use for TIFF, GIF and Textual Files. It is easy to implement, fast and lossless algorithm whereas Huffman algorithm is used in JPEG compression. It produces optimal and compact code but relatively slow. Huffman algorithm is based on statistical model which adds to overhead.

The above discussed algorithms use lossless compression technique.JPEG technique which is used mostly for image compression is a lossy compression technique.JPEG 2000 is advancement in JPEG standard which uses wavelets.

Future Scope and Work

With the advancements in compression technology, it is now very easy and efficient to compress video files. Various video compression techniques are available. The most common video compression standard is MPEG (Moving Picture Experts Group)[31]. It is a working group of ISO/IEC charged with the development of video and audio encoding standards. MPEG's official designation is ISO/IEC JTC1/SC29 WG11.Many advances are being made for improving the video quality Advancements in MPEG standard are MPEG-1(MP3)[33],MPEG-2,MPEG-3,MPEG-4(MPEG-4 Part 2 or Advanced Simple Profile)

and MPEG-4 Part 10 (or Advanced Video Coding or H.264). MPEG-7. A formal system for describing multimedia content. MPEG-21 describes this standard as a multimedia framework. MPEG standard is very efficient in the use of DVDs.Various H.261 standards could be used in future for advancement in video conferencing technology

Other applied fields that are making use of wavelets in the coming future, include astronomy, acoustics, nuclear engineering, sub-band coding, signal and image processing, neurophysiology, music, magnetic resonance imaging, speech discrimination, optics, fractals, turbulence, earthquake-prediction, radar, human vision, and pure mathematics applications such as solving partial differential equations.

Comparison of common algorithms with Huffman Coding

ALGORITHM	HUFFMAN CODING	LZW	ARITHMETIC CODING	RUN LENTH CODING
ADVANTAGES	1.Easy to implement 2. Lossless technique 3. Produces optimal and compact code	1.Easy to implement 2. Fast compression. 3. Lossless technique. 3. Dictionary based technique.	1. Efficiently represents more frequently occurring sequences of pixels values with fewer bits. 2. Reduce file size dramatically. 3. Lossless compression.	1. Simple to implement 2. Fast to execute. 3. Lossless compression.
DISADVANTAGES	1. Relatively slow. 2. Depends upon statistical model of data. 3. Decoding is difficult due to different code lengths. 4. Overhead due to Huffman tree.	1. Management of string table is difficult. 2. Amount of storage needed is indeterminate. 3. Royalties have to be paid to use LZW compression.	1. It is a paid algorithm (protected by patent). 2. Statistical technique.	1. Compression ratio is low as compared to other algorithms.
APPL ICATION	Used in JPEG.	Used in TIFF and GIF files.	Used mostly for frequently occurring sequences of pixels.	Used mostly for TIFF, BMP and PCX files.

References

[1] Darrel Hankersson, Greg A. Harris, and Peter D. Johnson Jr.Introduction to Information Theory and Data Compression. CRC Press, 1997.
[2] Gilbert Held and Thomas R. Marshall. Data and Image Compression: Tools and Techniques
[3] http://en.wikipedia.org/wiki/Discrete_Fourier_transform
[4] Terry Welch, "A Technique for High-Performance Data Compression", Computer, June 1984
[5] http://www.The LZW compression algorithm.htm
[6] Dzung Tien Hoang and Jeffery Scott Vitter .Fast and Efficient Algorithms for video Compression and Rate Control,June 20,1998.
[7] http://www.Howstuffworks How File Compression Works.htm.
[8] http://www .H_261 - Wikipedia, the free encyclopedia.htm
[9] http://en.wikipedia.org/wiki/Entropy_encoding
[10] http://www .JPEG - Wikipedia, the free encyclopedia.htm
[11] http://www.rz.go.dlr.de:8081/info/faqs/graphics/jpeg1.html
[12] http://www.amara.com/IEEEwave/IEEEwavelet.html
[13] http://cm.bell-labs.com/who/jelena/Wavelet/w_applets.html
[14] http://en.wikipedia.org/wiki/Lossless_JPEG
[15] http://en.wikipedia.org/wiki/Arithmetic_coding
[16] http://fly.cc.fer.hr/~eddie/Sem_DPCM.htm
[17] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, May 1977
[18] Rudy Rucker, "Mind Tools", Houghton Mifflin Company, 1987
[19] http://www.dogma.net/markn/index.html
[20] Huffman's original article: D.A. Huffman, "[3]" (PDF), Proceedings of the I.R.E., sept 1952, pp 1098-1102
[21] Background story: Profile: David A. Huffman, Scientific American, Sept. 1991, pp. 54-58
[22] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 16.3, pp.385–392.
[23] http://www.Huffman coding - Wikipedia, the free encyclopedia.htm
[24] http://en.wikipedia.org/wiki/Huffman_coding
[25] http://en.wikipedia.org/wiki/Adaptive_Huffman_coding
[26] Compressed Image File Formats: JPEG, PNG, GIF,XBM, BMP, John Miano, August 1999
[27] Introduction to Data Compression, Khalid Sayood, Ed Fox (Editor), March 2000

[28] Managing Gigabytes: Compressing and Indexing Documents and Images, Ian H H. Witten, Alistair Moffat, Timothy C. Bell , May 1999
[29] Digital Image Processing, Rafael C. Gonzalez, Richard E. Woods, November 2001
[30] http://en.wikipedia.org/wiki/Comparison_of_graphics_file_ formats
[31] Dzung Tien Hoang and Jeffery Scott Vitter .Fast and Efficient Algorithms for Video Compression and Rate Control,June 20,1998.
[32] V.Bhaskaran and K.Konstantinides.Image and video compression standards.Kluwer Academic Publishers,Boston,MA,1995.
[33] http://mp3.about.com
[34] http://en.wikipedia.org/wiki/Information_theory
[35] http://en.wikipedia.org/wiki/Entropy_encoding.
[36] http://en.wikipedia.org/wiki/Lossless_data_compression.
[37] http://en.wikipedia.org/wiki/Variable-length_code.
[38] http://www.pha.jhu.edu/~sundar/intermediate/history.html
[39] http://myhome.hanafos.com/~soonjp/vchx.html
[40] http://en.wikipedia.org/wiki/Block_code
[41] http://www/wavlets/An Introduction to Wavelets Historical Perspective.htm

Mamta Sharma, B.Sc(CS), M.Sc(maths), M.C.A, M.Phill(CS), M.Tech(in process) .