QUANTUM SERIES

For
B.Tech Students of Fourth Year
of All Engineering Colleges Affiliated to
Dr. A.P.J. Abdul Kalam Technical University,
Uttar Pradesh, Lucknow
(Formerly Uttar Pradesh Technical University)

Data Compression

By

Chetan Singhal

Quantum Page™
QUANTUM PAGE PVT. LTD.
Ghaziabad ■ New Delhi

PUBLISHED BY : **Apram Singh**
Quantum Publications®
(A Unit of Quantum Page Pvt. Ltd.)
Plot No. 59/2/7, Site - 4, Industrial Area,
Sahibabad, Ghaziabad-201 010

Phone : 0120 - 4160479
Email : pagequantum@gmail.com **Website: www.quantumpage.co.in**
Delhi Office : 1/6590, East Rohtas Nagar, Shahdara, Delhi-110032

Data Compression (CS/IT : Sem-8)
1st Edition : *2011-12*
2nd Edition : *2012-13*
3rd Edition : *2013-14*
4th Edition : *2014-15*
5th Edition : *2015-16*
6th Edition : *2016-17*
7th Edition : *2017-18*
8th Edition : *2018-19*
9th Edition : *2019-20*

Price: Rs. 110/- only

Printed at : Balajee Offset, Delhi.

CONTENTS

RCS 087 : DATA COMPRESSION

1 UNIT

Introduction

PART-1

Compression Techniques : Lossless Compression and Lossy Compression, Measures of Performance, Modeling and Coding.

CONCEPT OUTLINE : PART-1

- Data compression is an art of science of representing information in a compact form.
- Lossless compression involves no loss of information.
- Lossy compression involves some loss of information and data cannot be reconstructed exactly same as original.
- Modeling and coding are the two phases for the development of any data compression algorithm for a variety of data.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. What is data compression and why we need it ? Explain compression and reconstruction with the help of block diagram.

UPTU 2013-14, Marks 05

UPTU 2015-16, Marks 02

UPTU 2015-16, Marks 10

Answer

1. In computer science and information theory, data compression is the process of encoding information using fewer bits than a decoded representation would use through use of specific encoding schemes.
2. It is the art or science of representing information in a compact form. This compaction of information is done by identifying the structure that exists in the data.
3. Compressed data communication only works when both the sender and the receiver of the information understand the encoding scheme.
4. For example, any text makes sense only if the receiver understands that it is intended to be interpreted as characters representing English language.
5. Similarly, the compressed data can only be understood if the decoding method is known by the receiver.

Need of data compression :

1. Compression is needed because it helps to reduce the consumption of expensive resources such as a hard disk space or transmission bandwidth.
2. As an uncompressed text or multimedia (speech, image or video) data requires a huge amount of bits to represent them and thus require large bandwidth, this storage space and bandwidth requirement can be decreased by applying proper encoding scheme for compression.
3. The design of data compression schemes involves trade off among various factors including the degree of compression, the amount of distortion introduced and the computational resources required to compress and decompress the data.

Compression and reconstruction :

1. A compression technique or compression algorithm refers two algorithms *i.e.*, compression algorithm and reconstruction algorithm.
2. The compression algorithm takes an input X and generates a representation X_c that requires fewer bits, and the reconstruction algorithm operates on the compressed representation X_c to generate the reconstruction Y. These operations are shown in Fig. 1.1.1.

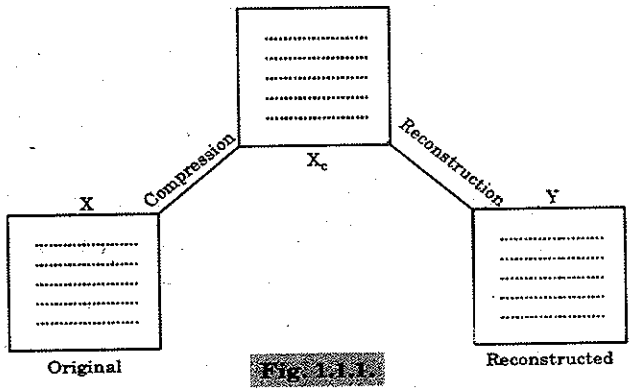


Fig. 1.1.1.

Que 1.2. What do you mean by data compression ? Explain its application areas. UPTU 2011-12, Marks 05

Answer

Data compression : Refer Q. 1.1, Page 5D, Unit-1.

Applications of data compression :

1. **Audio :**
   a. Audio data compression reduces the transmission bandwidth and storage requirements of audio data.
   b. Audio compression algorithms are implemented in software as audio codecs.
   c. Lossy audio compression algorithms provide higher compression at the cost of fidelity and are used in numerous audio applications.
   d. These algorithms rely on psychoacoustics to eliminate or reduce fidelity of less audible sounds, thereby reducing the space required to store or transmit them.
2. **Video :**
   a. Video compression uses modern coding techniques to reduce redundancy in video data.
   b. Most video compression algorithms and codecs combine spatial image compression and temporal motion compensation.
   c. Video compression is a practical implementation of source coding in information theory.
3. **Genetics :** Genetics compression algorithms are the latest generation of lossless algorithms that compress data using both conventional compression algorithms and genetic algorithms adapted to the specific datatype.
4. **Emulation :**
   a. In order to emulate CD-based consoles such as the Playstation 2, data compression is desirable to reduce huge amounts of disk space used by ISOs.
   b. For example, Final Fantasy XII (Computer Game) is normally 2.9 gigabytes. With proper compression, it is reduced to around 90% of that size.

Que 1.3. What do you understand by lossless and lossy compression ?

OR

What do you mean by lossless compression ? Compare lossless compression with lossy compression. UPTU 2011-12, Marks 05 UPTU 2015-16, Marks 02 UPTU 2015-16, Marks 10

Answer

Lossless compression :

1. In lossless compression, the redundant information contained in the data is removed.
2. Due to removal of such information, there is no loss of the data of interest. Hence it is called as lossless compression.

3. Lossless compression is also known as data compaction.
4. Lossless compression techniques, as their name implies, involve no loss of information.
5. If data have been losslessly compressed, the original data can be recovered exactly from the compressed data.
6. Lossless compression is generally used for applications that cannot tolerate any difference between the original and reconstructed data.
7. Text compression is an important area for lossless compression.
8. It is very important that the reconstruction is identical to the text original, as very small differences can result in statements with very different meanings.

Lossy compression :

1. In this type of compression, there is a loss of information in a controlled manner.
2. The lossy compression is therefore not completely reversible.
3. But the advantage of this type is higher compression ratios than the lossless compression.
4. The lossless compression is used for the digital data.
5. For many applications, the lossy compression is preferred due to its higher compression without a significant loss of important information.
6. For digital audio and video applications, we need a standard compression algorithm.
7. Lossy compression techniques involve some loss of information, and data that have been compressed using lossy techniques generally be recovered or reconstructed exactly.
8. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression.

Que 1.4. **What is data compression and why we need it ? Describe two applications where lossy compression technique is necessary for data compression.** **UPTU 2014-15, Marks 10**

Answer

Data compression and its need : Refer Q. 1.1, Page 5D, Unit-1.

Applications where lossy compression is necessary for data compression :

1. Lossy image compression can be used in digital cameras, to increase *storage capacities with minimal* degradation of picture quality.

2. In lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the audio signal.

Que 1.5. **What is data compression and why we need it ? Explain compression and reconstruction with the help of block diagram. What are the measures of performance of data compression algorithms ?** **UPTU 2012-13, Marks 10**

OR

What are the measures of performance of data compression algorithms ? **UPTU 2013-14, Marks 05**

UPTU 2015-16, Marks 10

Answer

Data compression and its need : Refer Q. 1.1, Page 5D, Unit-1.

Compression and reconstruction : Refer Q. 1.1, Page 5D, Unit-1.

Measures of performance of data compression :

1. A compression algorithm can be evaluated in a number of different ways.
2. We could measure the relative complexity of the algorithm, the memory required to implement the algorithm, how fast the algorithm performs on a given machine, the amount of compression, and how closely the reconstruction resembles the original.
3. A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression to the number of bits required to represent the data after compression. This ratio is called the compression ratio.
4. Another way of reporting compression performance is to provide the average number of bits required to represent a single sample.
5. This is generally referred to as the rate.
6. In lossy compression, the reconstruction differs from the original data.
7. Therefore, in order to determine the efficiency of a compression algorithm, we have to have some way of quantifying the difference.
8. The difference between the original and the reconstruction is often called the distortion.
9. Lossy techniques are generally used for the compression of data that originate as analog signals, such as speech and video.
10. In compression of speech and video, the final arbiter of quality is human.

11. Because human responses are difficult to model mathematically, many approximate measures of distortion are used to determine the quality of the reconstructed waveforms.
12. Other terms that are also used about differences between the reconstruction and the original are fidelity and quality.
13. When the fidelity or quality of a reconstruction is high, the difference between the reconstruction and the original is small.

Que 1.6. Explain modeling and coding with the help of suitable example. **UPTU 2013-14, Marks 05**

Answer

The development of any data compression algorithm for a variety of data can be divided into two phases : Modeling and Coding.

i. Modeling :

a. In this phase, we try to extract information about any redundancy or similarity that exist in the data and describe the redundancy of data in the form of model.

b. This model act as the basis of any data compression algorithm and the performance of any algorithms will depend on how well the model is being formed.

ii. Coding :

a. This is the second phase. It is the description of the model and a description of how the data different from the model are encoded, generally encoding is done using binary digits.

b. Example :

Consider the following sequence of number

9, 10, 11, 12, 13

By examining and exploiting the structure of data in a graph paper it seems to be a straight line, so we modeled it with the equation,

$x = n + 9 \qquad n = 0, 1, 2, ..$

PART-2

Mathematical Preliminaries for Lossless Compression : A Brief Introduction to Information Theory, Models : Physical Models, Probability Models, Markov Models, Composite Source Model.

CONCEPT OUTLINE : PART-2

- A fundamental limit to lossless data compression is called entropy rate.
- Entropy is the measure of the uncertainty associated with a random variable.
- Physical model, probability model and Markov model are the three approaches for building mathematical model.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.7. Explain entropy or entropy rate as given by Shannon.

Answer

1. In information theory, entropy is the measure of the uncertainty associated with a random variable.
2. It is usually refer to as Shannon entropy, which quantifies, in the sense of an expected value, the information contained in a message usually in units such as bit.
3. Shannon entropy is a measure of the average information content *i.e.*, the average number of binary symbol needed to code the output of the source.
4. Shannon entropy represents an absolute limit on the best possible lossless compression of any communication under certain constraints treating message to be encoded as a sequence of independent and identically distributed random variable.
5. The entropy rate of a source is a number which depends only on the statistical nature of the source. Consider an arbitrary source

$$X = \{X_1, X_2, X_3, X_4, ...\}$$

6. Following are various models :

i. Zero order model : The characters are statistically independent of each other and every letter of alphabet are equally likely to occur. Let m be the size of the alphabet. In this case, the entropy rate is given by,

$$H = \log_2 m \text{ bits/character}$$

For example, if the alphabet size is $m = 27$ then the entropy rate would be,

$$H = \log_2 27$$
$$= 4.75 \text{ bits/character}$$

ii. **First order model :** The character are statistically independent. Let m be the size of the alphabet and let P_i is the probability of the i^{th} letter in the alphabet. The entropy rate is,

$$H = -\sum_{i=1}^{m} P_i \log_2 P_i \text{ bits/character}$$

iii. **Second order model :** Let $P_{j|i}$ be the conditional probability that the present character is the j^{th} letter in the alphabet given that the previous character is the i^{th} letter. The entropy rate is :

$$H = -\sum_{i=1}^{m} P_i \sum_{j=1}^{m} P_{j|i} \log_2 P_{j|i} \text{ bits/character}$$

iv. **Third order model :** Let $P_{k|j,i}$ be the conditional probability that the present character is the k^{th} letter in the alphabet given that the previous character is the j^{th} letter and the one before that is the i^{th} letter. The entropy rate is,

$$H = -\sum_{i=1}^{m} P_i \sum_{j=1}^{m} P_{j|i} \log_2 P_{k|j,i} \text{ bits/character}$$

v. **General model :** Let B_n represents the first n characters. The entropy rate in the general case is given by,

$$H = -\lim_{n\to\infty} \frac{1}{n} \sum P(B_n) \log_2 P(B_n) \text{ bits/character}$$

Que 1.8. **Consider the following sequence :**

1, 2, 3, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10

Assuming the frequency of occurrence of each number is reflected accurately in the number of times it appears in the sequence and the sequence is independent and identically distributed, find the first order entropy of the sequence.

Answer

$$P(1) = \frac{1}{16} \qquad P(6) = \frac{1}{16}$$

$$P(2) = \frac{2}{16} = \frac{1}{8} \qquad P(7) = \frac{1}{16}$$

$$P(3) = \frac{2}{16} = \frac{1}{8} \qquad P(8) = \frac{2}{16} = \frac{1}{8}$$

$$P(4) = \frac{2}{16} = \frac{1}{8} \qquad P(9) = \frac{2}{16} = \frac{1}{8}$$

$$P(5) = \frac{2}{16} = \frac{1}{8} \qquad P(10) = \frac{1}{16}$$

First order entropy is given by,

$$H = -\sum_{i=1}^{10} P(i) \log_2 P(i)$$

$$= -\left[4\left[\frac{1}{16}\log_2\frac{1}{16}\right] + 6\left[\frac{1}{8}\log_2\frac{1}{8}\right]\right]$$

$$= 3.25 \text{ bits}$$

Que 1.9. **What do you understand by information and entropy ? Find the first order entropy over alphabet A = $\{a_1, a_2, a_3, a_4\}$ where $P(a_1) = P(a_2) = P(a_3) = P(a_4) = 1/4$.** **UPTU 2013-14, Marks 05**

Answer

Information :

1. The amount of information conveyed by a message increases as the amount of uncertainity regarding the message becomes greater.
2. The more it is known about the message a source will produce, the less the uncertainity, and less the information conveyed.
3. The entropy of communication theory is a measure of this uncertainity conveyed by a message from a source.
4. The starting point of information theory is the concept of uncertainity.
5. Let us define an event as an occurrence which can result in one of the many possible outcomes.
6. The outcome of the event is known only after it has occurred, and before its occurrence we do not know which one of the several possible outcomes will actually result.
7. We are thus uncertain with regard to the outcome before the occurrence of the event.
8. After the event has occurred, we are no longer uncertain about it.
9. If we know or can assign a probability to each one of the outcomes, then we will have some information as to which one of the outcomes is most likely to occur.

Entropy : Refer Q. 1.7, Page 11D, Unit-1.

Numerical :

First order entropy is given by,

$$H = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

$$H = -\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{4}\log_2\frac{1}{4}\right]$$

$$H = \frac{1}{4}\log_2 4 + \frac{1}{4}\log_2 4 + \frac{1}{4}\log_2 4 + \frac{1}{4}\log_2 4$$

$$H = 4\left[\frac{1}{4}\log_2 4\right]$$

$$H = 2 \text{ bits}$$

Que 1.10. **Prove that the average codeword length l of an optimal code for a source S is greater than or equal to entropy $H(S)$.**

Answer

1. According to Kraft-McMillan, if we have uniquely decodable code C with k codeword then the following inequality holds :

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1 \qquad \ldots(1.10.1)$$

2. It states that if we have a sequence of positive integers $\{l_i\}_{i=1}^{k}$ which satisfy equation (1.10.1). Consider a source S with alphabet $A = \{a_1, a_2, \ldots\ldots\ldots\ldots, a_k\}$, and probability model $\{P(a_1), P(a_2), \ldots\ldots\ldots\ldots, P(a_k)\}$, the average codeword length is given by,

$$\bar{l} = \sum_{i=1}^{k} P(a_i) l_i$$

3. Therefore, the difference between entropy of the source $H(S)$ and the average length as,

$$\begin{aligned} H(s) - \bar{l} &= -\sum_{i=1}^{k} P(a_i) \log_2 P(a_i) - \sum_{i=1}^{k} P(a_i) l_i \\ &= \sum_{i=1}^{k} P(a_i) \left(\log_2 \left[\frac{1}{P(a_i)} \right] - l_i \right) \\ &= \sum_{i=1}^{k} P(a_i) \left(\log_2 \left[\frac{1}{P(a_i)} \right] - \log_2 [2^{l_i}] \right) \\ &= \sum_{i=1}^{k} P(a_i) \log_2 \left[\frac{2^{-l_i}}{P(a_i)} \right] \leq \log_2 \left[\sum_{i=1}^{k} 2^{-l_i} \right] \end{aligned}$$

4. The last inequality is obtained using Jensen's inequality, which states that if $f(x)$ is a concave function, then $E[f(X)] \leq f(E[X])$. The log function is a concave function.
As the code is an optimal code.

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1,$$

hence $H(S) - \bar{l} \leq 0$

or $H(S) \leq \bar{l}$

Que 1.11. **The joint probabilities of the transmitted and received message of a communication system is given as :**

$P(X, Y) =$

	Y_1	Y_2	Y_3	Y_4
X_1	1/4	0	1/10	0
X_2	0	1/4	0	1/20
X_3	0	0	1/10	1/20
X_4	0	1/20	0	1/10
X_5	0	0	0	1/20

Calculate $H(X)$ and $H(Y)$. **UPTU 2013-14, Marks 05**

Answer

Given : $P(X, Y) =$

	Y_1	Y_2	Y_3	Y_4
X_1	1/4	0	1/10	0
X_2	0	1/4	0	1/20
X_3	0	0	1/10	1/20
X_4	0	1/20	0	1/10
X_5	0	0	0	1/20

To find : $H(X)$ and $H(Y)$

$P(X_1) = 1/4 + 0 + 1/10 + 0 = 0.35$
$P(X_2) = 0 + 1/4 + 0 + 1/20 = 0.3$
$P(X_3) = 0 + 0 + 1/10 + 1/20 = 0.15$
$P(X_4) = 0 + 1/20 + 0 + 1/10 = 0.15$
$P(X_5) = 0 + 0 + 0 + 1/20 = 0.05$

$H(X) = P(X_1) \log_2 [1/P(X_1)] + P(X_2) \log_2 [1/P(X_2)] + P(X_3) \log_2 [1/P(X_3)] + P(X_4) \log_2 [1/P(X_4)] + P(X_5) \log_2 [1/P(X_5)]$

$= 0.35 \log_2 (1/0.35) + 0.3 \log_2 (1/0.3) + 0.15 \log_2 (1/0.15) + 0.15 \log_2 (1/0.15) + 0.05 \log_2 (1/0.05)$

$= 0.5301 + 0.5211 + 0.4105 + 0.4105 + 0.2161$

$H(X) = 2.0883$ bits/message

Similarly,

$P(Y_1) = 1/4 + 0 + 0 + 0 + 0 = 0.25$
$P(Y_2) = 0 + 1/4 + 0 + 1/20 + 0 = 0.3$
$P(Y_3) = 1/10 + 0 + 1/10 + 0 + 0 = 0.2$
$P(Y_4) = 0 + 1/20 + 1/20 + 1/10 + 1/20 = 0.25$

$H(Y) = P(Y_1) \log_2 [1/P(Y_1)] + P(Y_2) \log_2 [1/P(Y_2)] + P(Y_3) \log_2 [1/P(Y_3)] + P(Y_4) \log_2 [1/P(Y_4)]$

$= 0.25 \log_2 (1/0.25) + 0.3 \log_2 (1/0.3) + 0.2 \log_2 (1/0.2) + 0.25 \log_2 (1/0.25)$

$= 0.5000 + 0.5211 + 0.4644 + 0.5000$

$H(Y) = 1.9855$ bits/message

Que 1.12. **What do you understand by information and entropy ? Give an alphabet $A = \{a_1, a_2, a_3, a_4\}$, find the first-order entropy in the following cases :**

i. **$P(a_1) = 1/2, P(a_2) = 1/4, P(a_3) = P(a_4) = 1/8$**

ii. $P(a_1) = 0.505, P(a_2) = 1/4, P(a_3) = 1/8$ and $P(a_4) = 0.12$.

And also differentiate between static length and variable length coding schemes. Explain with the help of examples.

UPTU 2012-13, Marks 10

OR

Differentiate between static length and variable length coding scheme. Explain with the help of an example.

UPTU 2011-12, Marks 05

UPTU 2015-16, Marks 10

Answer

Information and entropy : Refer Q. 1.7, Page 11D, Unit-1.

i. First-order entropy is given by,

$$h = -\left[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{8}\log_2\frac{1}{8}\right]$$

$$= \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{1}{8}\log_2 8$$

$$= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{3}{8}$$

$$= 1.75 \text{ bits}$$

ii. $P(a_1) = 0.505, P(a_2) = \frac{1}{4}, P(a_3) = \frac{1}{8}$ and $P(a_4) = 0.12$

$$H = -\left[0.505 \log_2 0.505 + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + 0.12 \log_2 0.12\right]$$

$$= -[0.505(-0.985644) + 0.25(-2) + 0.125(-3) + 0.12(-3.05)]$$

$$= -[-0.49775 - 0.5 - 0.375 - 0.366] = 1.73875$$

Difference between static length and variable length coding schemes :

Static length codes :

1. Static length codes are also known as fixed length codes. A fixed length code is one whose codeword length is fixed.
2. The ASCII code for the letter 'a' is 1000011 and for the letter 'A' is coded as 1000001.
3. Here, it should be notice that the ASCII code uses the same number of bits to represent each symbol. Such codes are called static or fixed length codes.

Variable length codes :

1. A variable length code is one whose codeword length is not fixed.
2. For example, consider a table given below :

x_i	Code 1	Code 2	Code 3
x_1	00	0	0
x_2	01	1	10
x_3	10	00	110
x_4	11	11	111

In this table, code 1 is fixed length code and code 2 and code 3 are variable length codes.

Que 1.13. What is average information ? What are the properties used in measure of average information ? **UPTU 2011-12, Marks 05**

UPTU 2015-16, Marks 10

Answer

Average information :

1. Average information is also called as entropy.
2. If we have a set of independent events A_i, which are the set of outcomes of some experiments S, such that

$$\bigcup A_i = S$$

where S is the sample space, then the average self-information associated with the random experiment is given by,

$$H = \Sigma P(A_i)\, i(A_i) = -\Sigma P(A_i) \log_b P(A_i)$$

3. The quantity is called the entropy associated with the experiment.
4. One of the many contributions of Shannon was that he showed that if the experiment is a source that puts out symbol A_i from a set A, then the entropy is a measure of the average number of binary symbols needed to code the output of the source.
5. Shannon showed that the best that a lossless compression scheme can do is to encode the output of a source with an average number of bits equal to the entropy of the source.

Given a set of independent events A_1, A_2, ... A_n with probability $p_i = P(A_i)$, the following properties are used in the measure of average information H :

1. We want h to be a continuous function of the probabilities p_i. That is, a small change in p_i should only cause a small change in the average information.
2. If all events are equally likely, that is, $p_i = 1/n$ for all i, then H should be a monotonically increasing function of n. The more possible outcomes there are, the more information should be contained in the occurrence of any particular outcome.

3. Suppose we divide the possible outcomes into a number of groups. We indicate the occurrence of a particular event by first indicating the group it belongs to, then indicating which particular member of the group it is.
4. Thus, we get some information first by knowing which group the event belongs to and then we get additional information by learning which particular event (from the events in the group) has occurred. The information associated with indicating the outcome in a single stage.

Que 1.14. Explain different approaches for building mathematical model also define two state Markov model for binary images.

UPTU 2014-15, Marks 10

UPTU 2011-12, Marks 05

UPTU 2015-16, Marks 10

Answer

There are several approaches to building mathematical models :

Physical models :

1. In speech-related applications, knowledge about the physics of speech production can be used to construct a mathematical model for the sampled speech process. Sampled speech can then be encoded using this model.
2. Models for certain telemetry data can also be obtained through knowledge of the underlying process.
3. For example, if residential electrical meter readings at hourly intervals were to be coded, knowledge about the living habits of the populace could be used to determine when electricity usage would be high and when the usage would be low. Then instead of the actual readings, the difference (residual) between the actual readings and those predicted by the model could be coded.

Probability models :

1. The simplest statistical model for the source is to assume that each letter that is generated by the source is independent of every other letter, and each occurs with the same probability.
2. We could call this the ignorance model, as it would generally be useful only when we know nothing about the source. The next step up in complexity is to keep the independence assumption, but remove the equal probability assumption and assign a probability of occurrence to each letter in the alphabet.
3. For a source that generates letters from an alphabet $A = \{a_1, a_2, ... a_M\}$, we can have a probability model $P = \{P(a_1), P(a_2),..., P(a_M)\}$.
4. Given a probability model (and the independence assumption), we can compute the entropy of the source using equation (1.14.1).

$$H(s) = -SP(X_1) \log P(X_1) \quad ...(1.14.1)$$

5. We can also construct some very efficient codes to represent the letters in A.
6. Of course, these codes are only efficient if our mathematical assumptions are in accord with reality.

Markov models :

1. One of the most popular ways of representing dependence in the data is through the use of Markov models.
2. For models used in lossless compression, we use a specific type of Markov process called a discrete time Markov chain.
3. Let $\{x_n\}$ be a sequence of observations. This sequence is said to follow a k^{th}-order Markov model if

$$P(x_n | x_{n-1},..., x_{n-k}) = P(x_n | x_{n-1},.., x_{n-k}, ...) \quad ...(1.14.2)$$

4. In other words, knowledge of the past k symbols is equivalent to the knowledge of the entire past history of the process.
5. The values taken on by the set $\{x_{n-1}, ... , x_{n-k}\}$ are called the states of the process. If the size of the source alphabet is l, then the number of states is l^k.
6. The most commonly used Markov model is the first-order Markov model, for which

$$P(x_n | x_{n-1}) = P(x_n | x_{n-1}, x_{n-2}, x_{n-3}, ...) \quad ...(1.14.3)$$

7. Equations (1.14.2) and (1.14.3) indicate the existence of dependence between samples.
8. However, they do not describe the form of the dependence. We can develop different first-order Markov models depending on our assumption about the form of the dependence between samples.
9. If we assumed that the dependence was introduced in a linear manner, we could view the data sequence as the output of a linear filter driven by white noise.
10. The output of such a filter can be given by the difference equation,

$$x_n = \rho x_{n-1} + \epsilon_n \quad ...(1.14.4)$$

where ϵ_n is a white noise process. This model is often used when developing coding algorithms for speech and images.

11. The use of the Markov model does not require the assumption of linearity.
12. For example,
    a. Consider a binary image.
    b. The image has only two types of pixels, white pixels and black pixels.

c. We know that the appearance of a white pixel as the next observation depends, to some extent, on whether the current pixel is white or black.

d. Therefore, we can model the pixel process as a discrete time Markov chain.

e. Define two states S_w and S_b (S_w would correspond to the case where the current pixel is a white pixel, and S_b corresponds to the case where the current pixel is a black pixel).

f. We define the transition probabilities $P(w/b)$ and $P(b/w)$, and the probability of being in each state $P(S_w)$ and $P(S_b)$. The Markov model can then be represented by the state diagram shown in Fig. 1.14.1.

g. The entropy of a finite state process with states S_i is simply the average value of the entropy at each state :

$$H = \sum_{i=1}^{M} P(S_i)\, H(S_i) \qquad ...(1.14.5)$$

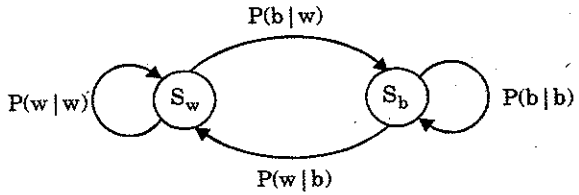


Fig. 1.14.1. A two-state Markov model for binary images.

h. For our particular example of a binary image

$$H(S_w) = -P(b/w) \log P(b/w) - P(w/w) \log P(w/w)$$

where $P(w/w) = 1 - P(b/w)$. $H(S_b)$ can be calculated in a similar manner.

Composite source model :

1. In many applications, it is not easy to use a single model to describe the source.
2. In such cases, we can define a composite source, which can be viewed as a combination or composition of several sources, with only one source being active at any given time.
3. A composite source can be represented as a number of individual sources S_i, each with its own model M_i and a switch that selects a source S_i with probability P_i (as shown in Fig. 1.14.2).

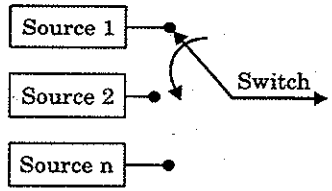


Fig. 1.14.2. A composite source.

4. This is an exceptionally rich model and can be used to describe some very complicated processes.

Que 1.15. What is zero frequency model in Markov models in text compression ?

UPTU 2013-14, Marks 05

UPTU 2015-16, Marks 05

Answer

1. As expected, Markov models are particularly useful in text compression, where the probability of the next letter is heavily influenced by the preceding letters.
2. In current text compression literature, the kth-order Markov models are more widely known as finite context models.
3. Consider the word preceding.
4. Suppose we have already processed preceding and are going to encode the next letter.
5. If we take no account of the context and treat each letter as a surprise, the probability of the letter g occurring is relatively low.
6. If we use a first-order Markov model or single-letter context (that is, we look at the probability model given n), we can see that the probability of g would increase substantially.
7. As we increase the context size (go from n to in to din and so on), the probability of the alphabet becomes more and more skewed, which results in lower entropy.
8. The longer the context, the better its predictive value.
9. If we were to store the probability model with respect to all contexts of a given length, the number of contexts would grow exponentially with the length of context.
10. Consider a context model of order four (the context is determined by the last four symbols).
11. If we take an alphabet size of 95, the possible number of contexts is 95^4 (more than 81 million).

12. Context modeling in text compression schemes tends to be an adaptive strategy in which the probabilities for different symbols in the different contexts are updated as they are encountered.
13. However, this means that we will often encounter symbols that have not been encountered before for any of the given contexts (this is known as the zero frequency problem).
14. The larger the context, the more often this will happen.
15. This problem could be resolved by sending a code to indicate that the following symbol was being encountered for the first time, followed by a prearranged code for that symbol.
16. This would significantly increase the length of the code for the symbol on its first occurrence.
17. However, if this situation did not occur too often, the overhead associated with such occurrences would be small compared to the total number of bits used to encode the output of the source.
18. Unfortunately, in context-based encoding, the zero frequency problem is encountered often enough for overhead to be a problem, especially for longer contexts.

PART-3

Coding : Uniquely Decodable Codes, Prefix Codes.

CONCEPT OUTLINE : PART-2

- A code is uniquely decodable if the mapping $C^*: A_x^* \rightarrow A_z^*$ is one to one, that is, $\forall_x$ and x' in A_x^*, $x \neq x' \Rightarrow C^*(x) \neq C^*(x')$.
- A code C is a prefix code if no codeword w_i is the prefix to another codeword w_j $(i \neq j)$.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.16. Write a short note on coding. What do you understand by uniquely decodable codes ?

Answer

1. Coding means the assignment of binary sequences to elements of an alphabet.
2. The set of binary sequences is called a code, and the individual members of the set are called codewords.
3. An alphabet is a collection of symbols called letter.

4. The ASCII code uses the same number of bits to represent each symbol.
5. Such a code is called a fixed-length code.
6. If we want to reduce the number of bits required to represent different messages, we need to use a different number of bits to represent different symbols.
7. If we use fewer bits to represent symbols that occur more often, on the average we would use fewer bits per symbol.
8. The average number of bits per symbol is often called the rate of the code.

Uniquely decodable codes :

1. A code is uniquely decodable if the mapping $C^*: A_x^* \rightarrow A_z^*$ is one to one, that is, $\forall x$ and x' in A_x^*, $x \neq x' \Rightarrow C^*(x) \neq C^*(x')$.
2. Suppose we have two binary codeword a and b where a is 'l' bit long and b is 'm' bit long, $l < m$ and if the first 'l' bit of b are identical to a then a is called the prefix of b and remaining $m - l$ bits of b are called dangling suffix.
3. For example : $a = 0110$ and $b = 011001$. Then a is prefix of b and dangling suffix is 01.
4. To decide uniquely decodability of code, construct a list of all the codeword and then examine all pairs of codewords.
5. When we find such a pair, add the dangling suffix to the list.
6. Now repeat this procedure using this larger list. Continue this procedure until one of the following two things happen :
   a. We get a dangling suffix that is a codeword – then the code is not uniquely decodable.
   b. There is no more unique dangling suffixes – then the code is uniquely decodable.
7. For example codeword = {0, 01, 011}
   a. Codeword 0 is prefix to codeword 01, here 1 is the dangling suffix so add 1 to the list {0, 01, 011, 1}
   b. Codeword 0 is also prefix of code word 011, here 11 is the dangling suffix. Add 11 to the list {0, 01, 011, 1, 11}
   c. Code used 01 is the prefix of codeword 011, here 1 is the dangling suffix but 1 is already added in the list in step 1.
   d. So the list is,
      {0, 01, 011, 1, 11}
      There is no more unique dangling suffix [case 6(b)]
      So, the codeword is uniquely decodable.

Que 1.17. Let X = {A, B, C, D}. Consider the code C defined by

X	C
A	0
B	1
C	01
D	10

Determine whether the codes are uniquely decodable or not ?

Answer

All the three source sequence AAD, ACA, AABA produces the code sequence 0010. Thus, from the code sequence 0010, we cannot tell which of the three source sequences it come from. Therefore, C is not uniquely decodable.

Que 1.18. Determine whether the code {01, 1, 2, 210} is uniquely decodable ?

Answer

1. Codeword 2 is prefix of codeword 210, with dangling suffix 10 so add 10 to the list {01, 1, 2, 210, 10}.
2. Codeword 1 is prefix of code 10 with dangling suffix 0 so add 0 to the list {01, 1, 2, 210, 10, 0}.
3. Codeword 0 is the prefix to codeword 01 with dangling suffix 1 and 1 itself is a codeword {case 1}.
4. So, the above code is not uniquely decodable.

Que 1.19. Determine whether the following codes are uniquely decodable :

i. {0, 10, 110, 111}

ii. {1, 10, 110, 111} **UPTU 2013-14, Marks 05**

Answer

i. In the list {0, 10, 110, 111} no codeword is prefix of another, so it is uniquely decodable.

ii. In the list {1, 10, 110, 111}, the codeword 1 is prefix to 10 with dangling suffix 0, 1 is prefix to 110 with dangling suffix 10 and 1 is also prefix to 111 with dangling suffix 11. Now, add all the suffix to the list *i.e.* {1, 10, 110, 111, 0, 10, 11}, here we can see that the codeword 10 is already present in the list, so it is not a uniquely decodable.

Que 1.20. Determine whether the following code is uniquely decodable ?
{0, 01, 11, 111}

Answer

{0, 01, 11, 111}

1. The codeword 0 is prefix of codeword 01 with dangling suffix 1 add it to list {0, 01, 11, 111, 1}.
2. The codeword 1 is prefix to 111 with dangling suffix 11 which in itself is a codeword so it is not uniquely decodable.

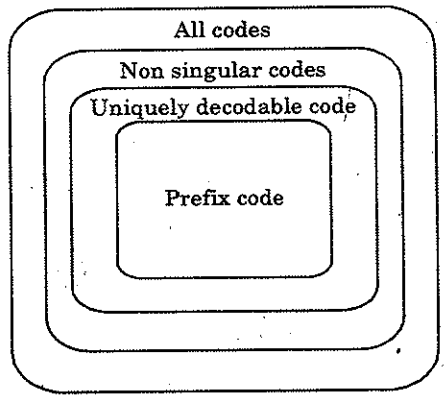
Que 1.21. What do you understand by prefix code ? Determine whether the following codes are uniquely decodable :

i. {0, 01, 110, 111}
ii. {1, 10, 110, 111}
And also explain modeling and coding with the help of suitable examples. **UPTU 2012-13, Marks 10**

Answer

1. A code C is a prefix code if no codeword w_i is the prefix to another codeword $w_j (i \neq j)$.
2. Equivalently, $w_j \neq w_i$ for any $w \in T^*$.
3. Prefix code are also known as instantaneous code.
4. Prefix codes are variable length code and are uniquely decodable that is all prefix codes are uniquely decodable but converse is not always true, that is, all uniquely decodable codes may not be prefix code.
5. For example, code {0, 01, 011, 0111} is uniquely decodable but not a prefix code.

   Code {0, 10, 110, 111} is a prefix code.



Numerical :

i. {0, 01, 110, 111, 1} :

1. The codeword 0 is prefix to codeword 01, here 1 is dangling suffix so add 1 to the list {0, 01, 110, 111, 1}.
2. The codeword 1 is prefix to 110 with dangling suffix 10 and 1 is also prefix of 111 with dangling suffix 11.
3. Now the list is : {0, 01, 110, 111, 1, 10, 11}.
4. The codeword 11 is prefix to 110 with dangling suffix 0 and 11 is also prefix to 111 with dangling suffix 1. Here 1 and 0, both are already in the list, so, it is not uniquely decodable.

ii. {1, 10, 110, 111} : Refer Q. 1.19, Page 24D, Unit-1.

Modeling and coding : Refer Q. 1.6, Page 10D, Unit-1.

Que 1.22. **Prove the following theorem. Let C be a code with M codewords with length $l_1, l_2 \ldots\ldots l_N$. If C is uniquely decodable, then**

$$K(c) = \sum_{i=1}^{N} 2^{-li} \leq 1$$

Answer

1. Consider the n^{th} power of $K(C)$. If $K(C)$ is greater than one, the $K(C)^n$ should grow exponentially with n.
2. This inequality is known as the Kraft–McMillian inequality.
3. If it does not grow exponentially then this is proof that

   $\sum_{i=1}^{N} 2^{-li} \leq 1$
4. Let n be any arbitrary integer. Then

$$\left[\sum_{i=1}^{N} 2^{-\ell_i}\right]^n = \left(\sum_{i_1=1}^{N} 2^{-\ell_{i_1}}\right)\left(\sum_{i_2=1}^{N} 2^{-\ell_{i_2}}\right)\ldots\ldots\ldots\left(\sum_{i_n=1}^{N} 2^{-\ell_{i_n}}\right) \quad \ldots (1.22.1)$$

$$= \sum_{i_1=1}^{N}\sum_{i_2=1}^{N}\ldots\sum_{i_n=1}^{N} 2^{-(\ell_{i_1}+\ell_{i_2}+\ldots\ldots+\ell_{i_n})} \quad \ldots (1.22.2)$$

5. The exponent $l_{i1} + l_{i2} + \ldots + l_{in}$ is simply the length of n codewords from the code C.
6. If $l = \max\{l_1, l_2 \ldots\ldots l_N\}$, then the largest value that the exponent can have is less than or equal to nl.
7. Therefore, equation (1.22.2) can be written as :

$$K(C)^n = \sum_{k=n}^{n\ell} A_k 2^{-k}$$

   Where A_k is the number of combinations of n codewords that have a combined length of k.
8. The number of possible distinct binary sequence of length k is 2^k.
9. If this code is uniquely decodable, then each sequence can represent one and only one sequence of codeword.
10. Therefore, the number of possible codewords whose combined length is K cannot be greater than 2^k.
11. So,

$$A_k \leq 2^k$$

$$K(C)^n = \sum_{k=n}^{n\ell} A_k 2^{-k} \leq \sum_{k=n}^{n\ell} 2^k\, 2^{-k} = n\ell - n + 1 = n(\ell - 1) + 1 \quad \ldots (1.22.3)$$

    But if $K(C)$ is greater than one, it will grow exponentially with the increase of n but $n(l-1) + 1$ can grow linearly only.
12. So, if $K(C)$ is greater than one, we can find an 'n' large enough that the equation (1.22.3) is violated.
13. Therefore, for a uniquely decodable code C, $K(C)$ is less than or equal to one.

Que 1.23. **Explain modeling and coding with the help of examples. What do you understand by prefix code ?**

UPTU 2014-15, Marks 10

Answer

Modeling and coding : Refer Q. 1.6, Page 10D, Unit-1.
Prefix code : Refer Q. 1.21, Page 24D, Unit-1.

Que 1.24. **What do you understand by uniquely decodable codes ?**

Answer

Refer Q. 1.16, Page 22D, Unit-1.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Define data compression. Also, explain lossless and lossy compression.

Ans: Refer Q. 1.1 and Q.1.3.

Q. 2. Explain with example, the two phases of development of data compression algorithms.

Ans: Refer Q. 1.6.

Q. 3. Describe the following :
i. Physical model ii. Probability model iii. Markov model

Ans: Refer Q. 1.14.

Q. 4. What is uniquely decodable code and prefix code ?

Ans: Refer Q. 1.16 and Q. 1.21.

Q. 5. What do you understand by information and entropy ? Give an alphabet $A = \{a1, a2, a3, a4\}$, find the first-order entropy in the following cases :
i. $P(a1) = 1/2, P(a2) = 1/4, P(a3) = P(a4) = 1/8$
ii. $P(a1) = 0.505, P(a2) = 1/4, P(a3) = 1/8$ and $P(a4) = 0.12$.
And also differentiate between static length and variable length coding schemes. Explain with the help of examples.

Ans: Refer Q. 1.12.

2 UNIT

Huffman Coding

PART-1

The Huffman Coding Algorithm : Minimum Variance Huffman Codes.

CONCEPT OUTLINE : PART-1

- Coding is assigned codewords to the modeled data.
- Huffman coding algorithm is a bottom-up approach and it has following properties :
  i. Unique prefix code
  ii. Optimality
- Minimum variance Huffman tree is obtained by making a small change in the conventional Huffman algorithm *i.e.*, when two or more symbols have the same probabilities as that of newly combined probability of the symbols.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. Write down Huffman coding algorithm. How this algorithm is used to design Huffman code for a source that takes letter from an alphabet set $A = \{a_1, a_2, a_3, a_4, a_5\}$.

UPTU 2013-14, Marks 05

Answer

1. This technique was developed by David Huffman.
2. The codes are generated using the Huffman technique or procedure is called Huffman codes. These codes are prefix codes and are optimum for a given model (set of probabilities).
3. Assigning codewords to the modeled data is known as coding.
4. The Huffman procedure is based on two observations regarding optimum prefix codes.
   a. In an optimum code, symbols that occur more frequently (have a higher probability of occurrence) will have shorter codewords than symbols that occur less frequently.
   b. In an optimum code, the two symbols that occur least frequently will have the same length.
5. It is easy to see that the first observation is correct.

6. If symbols that occur more often had codewords that were longer than the codewords for symbols that occurred less often, the average number of bits per symbol would be larger than if the conditions were reversed.
7. Therefore, a code that assigns longer codewords to symbols that occur more frequently cannot be optimum.
8. The Huffman procedure is obtained by adding a simple requirement to these two observations.
9. This requirement is that the codewords corresponding to the two lowest probability symbols differ only in the last bit.
10. That is, if γ and δ are the two least probable symbols in an alphabet, if the codeword for γ was m * 0, the codeword for δ would be m * 1.
11. Here, m is string of 1s and 0s, and * denotes concatenation.
12. The Huffman coding has the following properties :
    - **a. Unique prefix code :** No Huffman code is a prefix of any other Huffman code. It prevents any ambiguity in decoding.
    - **b. Optimality :** Minimum redundancy code proved optimal for a given data model *i.e.*, a given, accurate, probability distribution :
        - i. The two least frequency symbols will have the same length for their Huffman codes, differing only at the last bit.
        - ii. Symbol that occurs more frequently will have shorter Huffman codes than symbols that occur less frequently.
    - c. The average code length for an information source S is strictly less than $H(S) + 1$

$$\overline{l} < H(S) + 1$$

Where $\overline{l}$ is the average code length.

Huffman coding algorithm :

Huffman coding algorithm is a bottom up approach, steps of the algorithm are :

i. Initialization : Put all symbols on a list sorted according to their frequency count.

ii. Repeat until the list has only one symbol left :
    - a. From the list, pick two symbols with the lowest frequency counts. From a Huffman subtree that has these two symbols as child nodes and create a parent node.
    - b. Assign the sum of the children's frequency counts to the parent and insert it into the list such that the order is maintained.
    - c. Delete the children from the list.

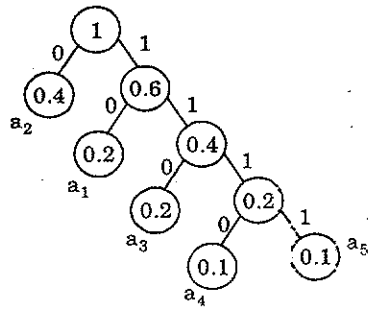
iii. Assign a codeword for each leaf, based on the path from the root.

Numerical :

Arranging symbol in order of their probabilities as :

a_2	0.4
a_1	0.2
a_3	0.2
a_4	0.1
a_5	0.1

Apply Huffman algorithm,

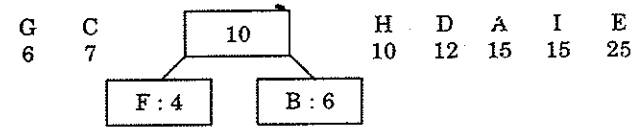


Symbol	Frequency	Codeword	Code length
a_2	0.4	0	1
a_1	0.2	10	2
a_3	0.2	110	3
a_4	0.1	1110	4
a_5	0.1	1111	4

Que 2.2. Draw the Huffman tree for the following symbols whose frequency occurrence in a message text is started along with their symbol below :
A : 15, B : 6, C : 7, D : 12, E : 25, F : 4, G : 6, H : 10, I : 15
decoding the message 1110100010111011. **UPTU 2014-15, Marks 05**

Answer

A : 15, B : 6, C : 7, D : 12, E : 25, F : 4, G : 6, H : 10, I : 15
F : 4, B : 6, G : 6, C : 7, H : 10, D : 12, A : 15, I : 15, E : 25



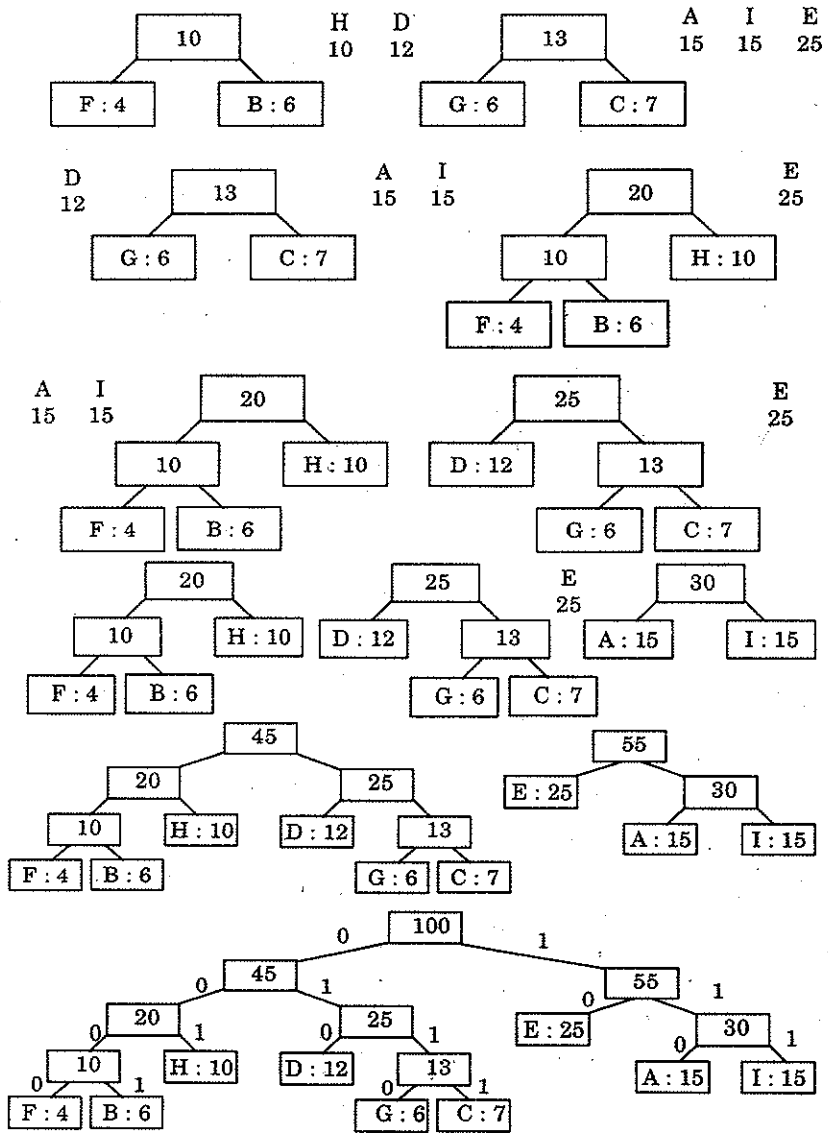


Table 2.2.1.

Symbol	Codeword
A	110
B	0001
C	0111
D	010
E	10
F	0000
G	0110
H	001
I	111

Decoding : 1110100010111011

111 = I

010 = D

001 = H

0111 = C

011 = Not found

Since there is no codeword for 011, hence the given message cannot be decoded.

Que 2.3. What is the limitation of Huffman coding ? Explain by an example. **UPTU 2011-12, Marks 05**

Answer

1. In Huffman coding method, the coding rate is the average number of bits used to represent a symbol from a source and, for a given probability model, the entropy is the lowest rate at which the source can be coded. We can tighten this bound somewhat.
2. The Huffman algorithm will generate a code whose rate is within p_{max} +0.086 of the entropy, where p_{max} is the probability of the most frequently occurring symbol.
3. In applications where the alphabet size is large, p_{max} is generally quite small, and the amount of deviation from the entropy, especially in terms of a percentage of the rate, is quite small.

4. However, in cases where the alphabet is small and the probability of occurrence of the different letters is skewed, the values of p_{max} can be quite large and the Huffman code can become rather inefficient when compared to the entropy.
5. One way to avoid this problem is to block more than one symbol together and generate an extended Huffman code.

Que 2.4. Explain minimum variance Huffman codes.

Answer

1. By performing the sorting procedure in a slightly different manner, we could have found a different Huffman code.
2. In the first re-sort, we could place a'_4 higher in the list, as shown in Table. 2.4.1.
3. Now combine a_1 and a_3 into a'_1, which has a probability of 0.4.
4. Sorting the alphabet a_2, a'_4, a'_1 and putting a'_1 as far up the list as possible, we get Table 2.4.2.
5. Finally, by combining a_2 and a'_4 and re-sorting, we get Table 2.4.3.
6. If we go through the unbundling procedure, we get the codewords in Table 2.4.4.
7. The procedure is summarized in Fig. 2.4.1.
8. The average length of the code is

   $l = 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 = 2.2$ bits/symbol.
9. The two codes are identical in terms of their redundancy.
10. However, the variance of the length of the codewords is significantly different.
11. This can be clearly seen from Fig. 2.4.2.

Table 2.4.1 : Reduced four-letter alphabet

Letter	Probability	Codeword
a_2	0.4	$c(a_2)$
a'_1	0.2	α_1
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$

Table 2.4.2 : Reduced three-letter alphabet

Letter	Probability	Codeword
a'_1	0.4	α_2
a_2	0.4	$c(a_2)$
a'_4	0.2	α_1

Table 2.4.3 : Reduced two-letter alphabet

Letter	Probability	Codeword
a'_2	0.6	a_3
a'_1	0.4	a_2

Table 2.4.4 : Minimum variance Huffman code

Letter	Probability	Codeword
a_1	0.2	10
a_2	0.4	00
a_3	0.2	11
a_4	0.1	010
a_5	0.1	011

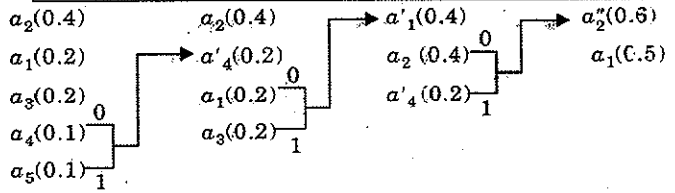


Fig. 2.4.1. The minimum variance Huffman encoding procedure.

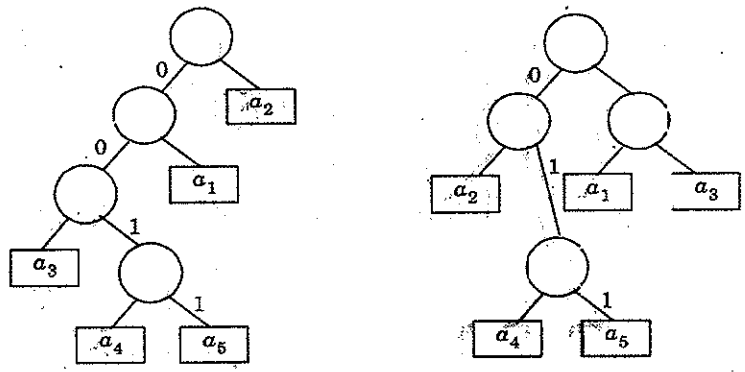


Fig. 2.4.2. Two Huffman codes corresponding to the same probabilities.

Que 2.5. What do you mean by optimality of Huffman codes ?

Answer

1. Optimality of Huffman codes can be proven rather simply by first writing down the necessary conditions that an optimal code has to satisfy and then showing that satisfying these conditions necessarily leads to designing a Huffman code.
2. The necessary conditions for an optimal variable-length binary code are as follows :
   - **a. Condition 1 :** Given any two letters a_j and a_k, if $P[a_j] \geq P[a_k]$, then $l_j \leq l_k$, where l_j is the number of bits in the codeword for a_j.
   - **b. Condition 2 :** The two least probable letters have codewords with the same maximum length l_m.
   - **c. Condition 3 :** In the tree corresponding to the optimum code, there must be two branches stemming from each intermediate node.

     If there were any intermediate node with only one branch coming from that node, we could remove it without affecting the decipherability of the code while reducing its average length.
   - **d. Condition 4 :** Suppose we change an intermediate node into a leaf node by combining all the leaves descending from it into a composite word of a reducing alphabet. Then, if the original tree was optimal for the original alphabet, the reduced tree is optimal for the reduced alphabet.

Que 2.6. Write a short note on non-binary Huffman code.

Answer

1. The binary Huffman coding procedure can be easily extended to the non-binary case where the code elements come from an m-ary alphabet, m is not equal to two.
2. We obtained the Huffman algorithm based on the observations that in an optimum binary prefix code and the requirement that the two symbols with the lowest probability differ only in the last position.
   - a. Symbols that occur more frequently (have a higher probability of occurrence) will have shorter codewords than symbols that occur less frequently, and
   - b. The two symbols that occur least frequently will have the same length.
3. We can obtain a non-binary Huffman code in almost exactly the same way.

4. The obvious thing to do would be to modify the second observation to read : "The m symbol that occur least frequently will have the same length," and also modify the additional requirement to read "The m symbols with the lowest probability differ only in the last position."

Que 2.7. Consider the source alphabet A, B, C, D, E having probabilities P(A) = 0.16, P(B) = 0.51, P(C) = 0.09, P(D) = 0.13, P(E) = 0.11. Design the Huffman code.

Answer

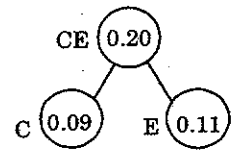
Step 1 : Sort all symbols according to their probabilities :

B	A	D	E	C
0.51	0.16	0.13	0.11	0.09

Step 2 : Build the tree.

a. Pick two symbols with the lowest frequency count and form a tree with root as the sum of the two and making lowest frequency symbol as a left child and the other as right child of root.

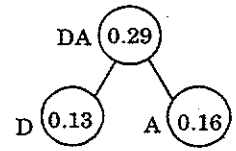
b. Here two smallest nodes are C and E with probabilities 0.09 and 0.11 respectively.



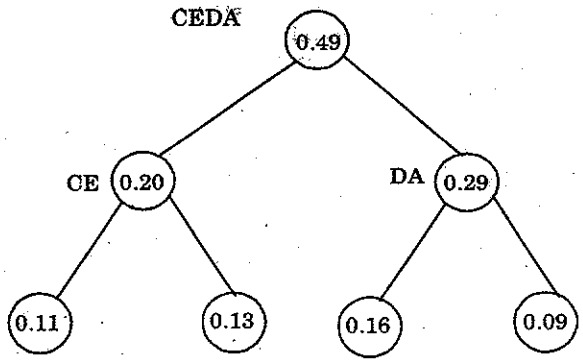
c. Again arrange it into the original list and delete the children from the original list.

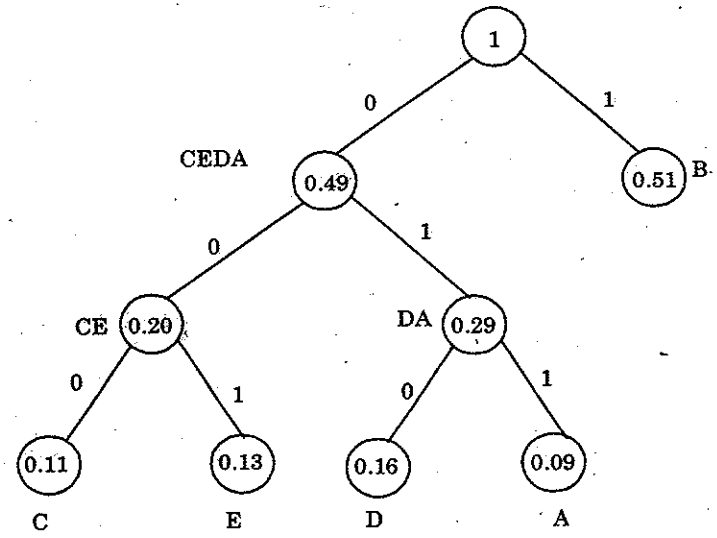
B	CE	A	D
0.51	0.20	0.16	0.13

d. Repeating step 2, we get,



B	DA	CE
0.51	0.29	0.20

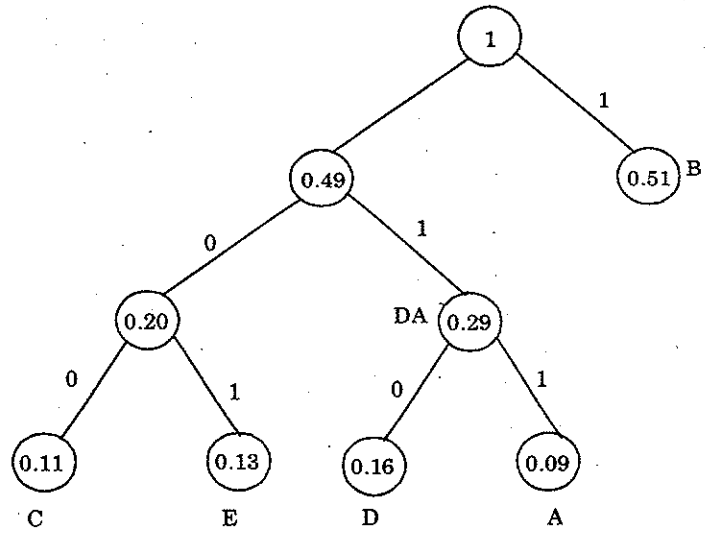




B
0.51

CEDA
0.47

Step 3 : Label left branches of the tree with 0 and right branches of the tree with 1.



Step 4 : Create Huffman code by traversing from the root to leaf.

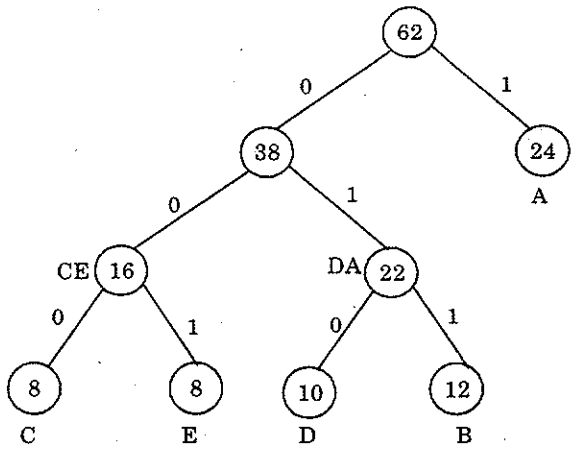
Symbol	Codeword
B	1
A	011
D	010
E	001
C	000

Que 2.8. **Considering the following source alphabet and their frequencies. Create the Huffman tree and calculate the average length of the code :**

Symbol	Frequency
A	24
B	12
C	10
D	8
E	8

Answer

Applying Huffman algorithm, tree formed is :



Symbol	Frequency	Code	Code length
A	24	0	1
B	12	011	3
C	10	010	3
D	8	001	3
E	8	000	3

Average length = 24 × 1 + 12 × 3 + 10 × 3 + 8 × 3 + 8 × 3 = 138 bits/symbol

PART-2

Adaptive Huffman Coding : Update Procedure, Encoding Procedure, Decoding Procedure.

CONCEPT OUTLINE : PART-2

- **Adaptive Huffman coding :** In adaptive Huffman coding procedure, neither transmitter nor receiver knows anything about statistic of the source sequence at the start of transmission.
- **Update procedure :** The update procedure requires that the nodes be in a fixed order. This ordering is preserved by numbering the nodes.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.9. Explain adaptive Huffman coding.

Answer

1. Huffman coding requires knowledge of the probabilities of the source sequence.
2. If this knowledge is not available, Huffman coding becomes a two-pass procedure : the statistics are collected in the first pass, and the source is encoded in the second pass.
3. In order to convert this algorithm into a one-pass procedure, Faller and Gallaher independently developed adaptive algorithms to construct the Huffman code based on the statistics of the first symbols already encountered.
4. Theoretically, if we wanted to encode the $(k + 1)$th symbol using the statistics of the first k symbols, we could recompute the code using the Huffman coding procedure each time a symbol is transmitted.
5. However, this would not be a very practical approach due to the large amount of computation involved—hence, the adaptive Huffman coding procedure.
6. In order to describe how the adaptive Huffman code works, we add two other parameters to the binary tree : the weight of each leaf, which is written as a number inside the node, and a node number.
7. The weight of each external node is simply the number of times the symbol corresponding to the leaf has been encountered.
8. The weight of each internal node is the sum of the weights of its offspring.
9. The node number y_i is a unique number assigned to each internal and external node.
10. If we have an alphabet of size n, then the $2n - 1$ internal and external node can be numbered as $y_1, \ldots, y_{2n-1}$ such that if x_j is the weight of node y_j, we have $x_1 \leq x_2 \leq \ldots \leq x_{2n-1}$.
11. Furthermore, the node y_{2j-1} and y_{2j} are offspring of the same parent node, or siblings, for $1 \leq j < n$, and the node number for the parent node is greater than y_{2j-1} and y_{2j}.
12. These last two characteristics are called the sibling property, and any tree that possesses this property is a Huffman tree.
13. In the adaptive Huffman coding procedure, neither transmitter nor receiver knows anything about the statistics of the source sequence at the start of transmission.
14. The tree at both the transmitter and the receiver consists of a single node that corresponds to all symbols not yet transmitted (NYT) and has a weight of zero.
15. As transmission progresses, nodes corresponding to symbols transmitted will be added to the tree, and the tree is reconfigured using an update procedure.

16. Before the beginning of transmission, a fixed code for each symbol is agreed upon between transmitter and receiver.
17. A simple (short) code is as follows :
    a. If the source has an alphabet $(a_1, a_2, \ldots, a_m)$ of size m, then pick e and r such that $m = 2^e + r$ and $0 \le r < 2^e$.
    b. The letter a_k is encoded as the $(e + 1)$-bit binary representation of $k - 1$, if $1 \le k \le 2r$; else, a_k is encoded as the e-bit binary representation of $k - r - 1$.
    c. For example, suppose $m = 25$, then $e = 4$, and $r = 10$.
    d. The symbol a_1 is encoded as 00000, the symbol a_2 is encoded as 00001, and the symbol a_{22} is encoded as 1011.
18. When a symbol is encountered for the first time, the code for the NYT node is transmitted, followed by the fixed code for the symbol.
19. A node for the symbol is then created, and the symbol is taken out of the NYT list.

Que 2.10. Differentiate between conventional Huffman coding and adaptive Huffman coding. UPTU 2014-15, Marks 05

Answer

1. Adaptive Huffman coding was first conceived by Faller and Gallagher and then further refined by Knuth (so it is often called FGK algorithm).
2. Adaptive Huffman coding overcomes the following drawback of conventional Huffman coding :
    i. Huffman coding assigns an output code to each symbol, with the output codes being as short as 1 bit, or considerably longer than the input symbols, strictly depending on their probabilities. The optimal number of bits to be used for each symbol is log base 2 of (1/P). Thus if probability of a character is 1/256, the optimal number of bits per character is log base 2 of 256 or 8. The problem with this scheme lies in the fact that Huffman codes have to be an integral number of bits. For example if the probability of character is 1/3, the optimum value of bits to code the character is around 1.6. The Huffman coding schemes has to assign either 1 or 2 bits to the code and either choice lead to a longer compressed message than its theoretical value. This non optimal coding becomes a noticeable problem when the probability of character becomes very high.
    ii. Huffman coding suffers from the fact that the uncompresser need same knowledge of the probabilities of the symbols in the compressed file, providing this information need more bit to encode the file. If this information is not available compressing the file requires two passes :

First pass :

To find out the frequency of each symbol and construct the Huffman tree.

Second pass :

1. To compress the file.
2. In adaptive Huffman coding procedure, neither transmitter nor receiver knows anything about the statistics of the source sequence at the start of the transmission.
3. The tree at both the transmitter and the receiver consists of a single node that corresponds to all symbols NYT (Not Yet Transmitted) and has a weight of zero.
4. As transmission progresses, node corresponding to symbols transmitted will be added to the tree and tree is reconfigured using an update procedure.

Encoder	**Decoder**
Initial_code( );	Initial_code( );
while not EOF	while not EOF
{	{
get (c);	decode (c);
encode (c);	output (c);
update_tree (c);	update_tree (c);
}	}

5. Initial_code assign symbols with same initially agreed upon codes, without any prior knowledge of the frequency counts update_tree construct an adaptive Huffman tree, it basically does two things :
    i. Increment the frequency count of the symbol.
    ii. Update the configuration of the tree.

Que 2.11. Explain the update procedure of adaptive Huffman coding algorithm with the help of a flow chart. UPTU 2011-12, Marks 05

Answer

1. The update procedure requires that the nodes be in a fixed order. This ordering is preserved by numbering the nodes.
2. The largest node number is given to the root of the tree, and the smallest number is assigned to the NYT node.
3. The number from the NYT node to the root of the tree are assigned in increasing order from the left to right and from lower level to the upper level. The set of nodes with the same weight makes up a block.

4. Flowchart of the updating procedure is shown in Fig. 2.11.1.
5. The function of the update procedure is to preserve the sibling property.
6. The tree at the transmitter is updated after each symbol is encoded, and the tree at the received is updated after each symbol is decoded.
7. After a symbol has been encoded and decoded, the external node corresponding to the symbol is that it has the largest node number, it is exchanged with the largest node number in the block, as long as the largest node number is not the parent of the node being updated.
8. The weight of the external node is then incremented.

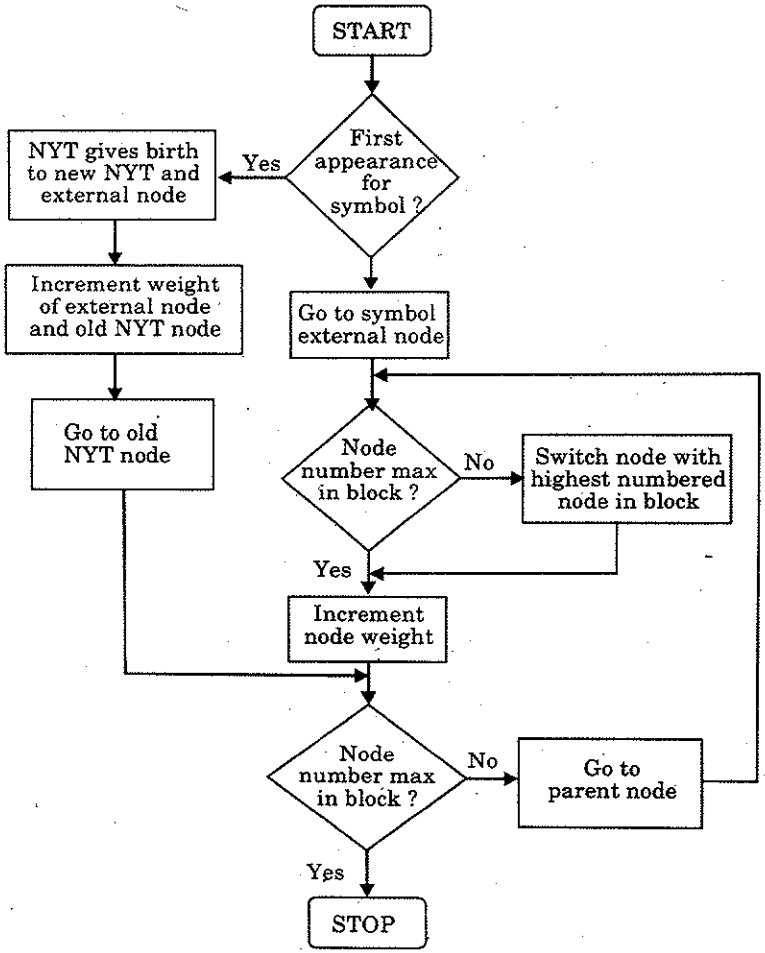


Fig. 2.11.1.

9. Once we have incremented the node's weight, we have adopted the Huffman tree at that level.
10. Now, turn to the next level by examining the parent node of the node whose weight was increment to see if it has the largest number in its block.
11. If it does not, it is exchanged with the node with the largest number in the block.
12. Again, an exception to this is when the node with the node having largest number is the parent of the node in the consideration.
13 Once an exchange has taken place, the weight of the parent node is incremented.
14. We then proceed to a new parent node and the process is repeated. The process is continued until the root of the tree is reached.
15. If the symbol to be encoded or decoded has occurred for the first time, a new external node is assigned to the symbol and a new NYT node is appended to the tree.
16. Both the new external node, we incremented it's weight by one and then go on to update all the other nodes until we reach the root of the tree.

Que 2.12. With the help of flowchart/steps, explain the encoding and decoding procedure.

Answer

Encoding procedure :

1. The flowchart shown in Fig. 2.12.1 shows the encoding procedure for adaptive Huffman coding.
2. Initially the tree at both the encoder and decoder have a single node *i.e.*, NYT (not yet transmitted).
3. Thus the codeword for the very first symbol is a predefined agreed fixed code.
4. Whenever the symbol is encountered for the first time, we send the code for NYT node, followed by the previously agreed code.
5. The code for NYT node is obtained by traversing the Huffman tree from the root to the NYT node.
6. If the symbol encoded is already being encountered *i.e.*, it has a node corresponding to the symbol in the tree, then the code for the symbol is generated by traversing the tree from the root to the external node corresponding to symbol.

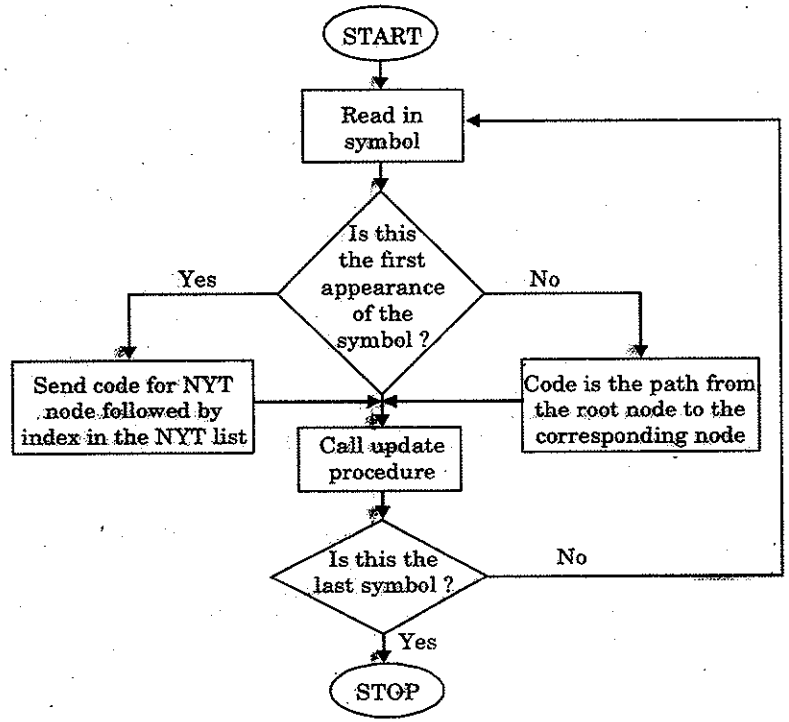


Fig. 2.12.1. Flowchart of encoding procedure

Decoding procedure :

1. The flowchart for the decoding procedure is shown in Fig. 2.12.2.
2. As we read in the received binary string, we traverse the tree in a manner identical to that used in the encoding procedure.
3. Once a leaf is encountered, the symbol corresponding to that leaf is decoded.
4. If the leaf is the NYT node, then we check the next e bits to see if the resulting number is less than r.
5. If it is less than r, we read in another bit to complete the code for the symbol.
6. The index for the symbol is obtained by adding one to the decimal number corresponding to the e- or $e+1$-bit binary string.
7. Once the symbol has been decoded, the tree is updated and the next received bit is used to start another traversal down the tree.

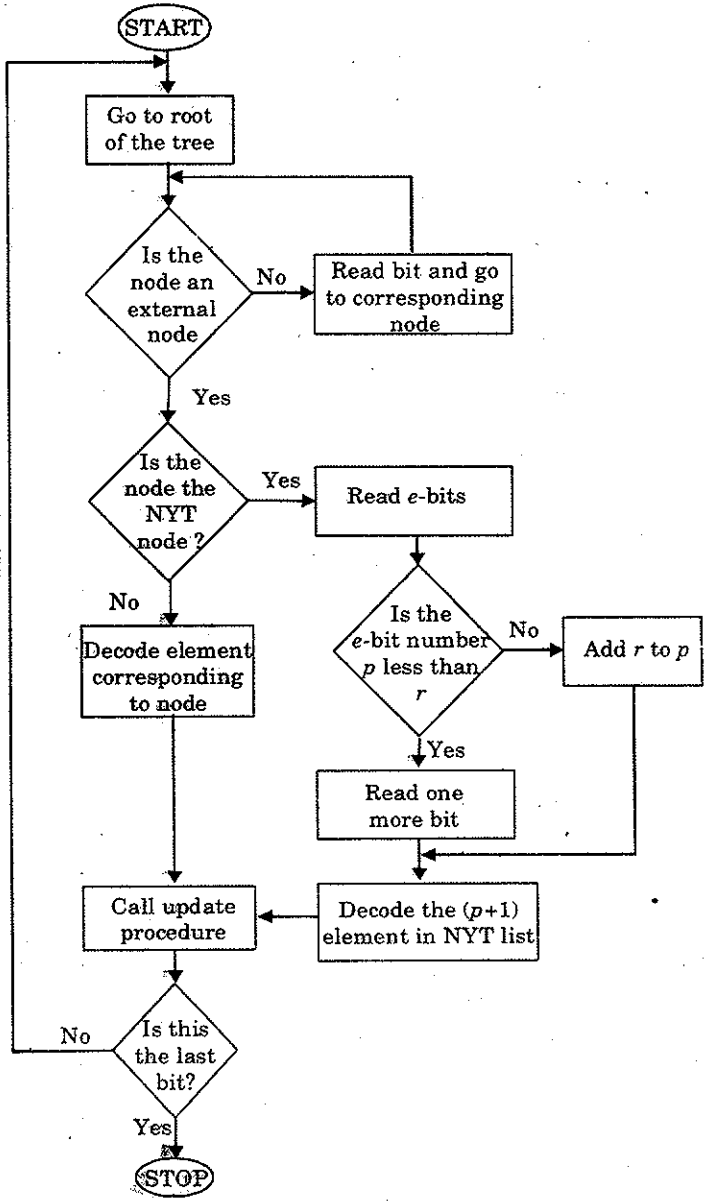


Fig. 2.12.2. Flowchart of the decoding procedure

PART-3

Golomb Codes, Rice Codes, Tunstall Codes, Application of Huffman Coding : Lossless Image Compression, Text Compression, Audio Compression.

CONCEPT OUTLINE : PART-3

- Golomb coding is a data compression scheme based on entropy encoding and optimal for geometric distribution.
- Rice code is a sequence of non negative integers and is divided into blocks of K integer piece. Each block is then coded.
- Tunstall code is an important exception. All codewords are of equal length. Each codeword represents a different number of letters.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.13. Explain Golomb code with the help of example.

Answer

1. The Golomb-Rice codes belong to a family of code designed to encode integers with the assumption that the larger an integer, the lower its probability of occurrence.
2. The simplest code for this situation is the unary code.
3. The unary code for a positive integer n is simply n 1s followed by a 0.
4. Thus, the code for 4 is 11110, and the code for 7 is 11111110.
5. The unary code is the same as the Huffman code for the semi-infinite alphabet {1, 2, 3, . . . } with probability model, or :

$$P[k] = \frac{1}{2^k}$$

6. Because the Huffman code is optimal, the unary code is also optimal for this probability model.
7. Although the unary code is optimal in very restricted conditions, we can see that it is certainly very simple to implement.
8. One step higher in complexity are a number of coding schemes that split the integer into two parts, representing one part with a unary code and the other part with a different code.

9. An example of such a code is the Golomb code.
10. The Golomb code begins "Secret Agent 00111 is back at the Casino again, playing a game of chance, while the fate of mankind hangs in the balance."
11. Agent 00111 requires a code to represent runs of success in a roulette game, and Golomb provides it.
12. The Golomb code is actually a family of codes parameterized by an integer $m > 0$.
13. In the Golomb code with parameter m, we represent an integer $n > 0$ using two numbers q and r, where

$$q = \left\lfloor \frac{n}{m} \right\rfloor$$

and

$$r = n - qm.$$

14. $\lfloor x \rfloor$ is the integer part of x.
15. In other words, q is the quotient and r is the remainder when n is divided by m.
16. The quotient q can take on values 0, 1, 2, . . . and is represented by the unary code of q.
17. The remainder r can take on the values 0, 1, 2, ..., $m - 1$.
18. If m is a power of two, we use the $\log_2 m$-bit binary representation of r.
19. If m is not a power of two, we could still use $\lceil \log_2 m \rceil$ bits, where $\lceil x \rceil$ is the smallest integer greater than or equal to x.
20. We can reduce the number of bits required if we use the $\lfloor \log_2 m \rfloor$-bit binary representation of r for the first $2^{\lceil \log_2 m \rceil} - m$ values, and the $\lceil \log_2 m \rceil$-bit binary representation of $r + 2^{\lceil \log_2 m \rceil} - m$ for the rest of the values.

Que 2.14. Design a Golomb code for m = 5.

UPTU 2013-14, Marks 05

Answer

1. $\lceil \log_2 5 \rceil = 3$ and $\lfloor \log_2 5 \rfloor = 2$
2. By $2^{\lceil \log_2 m \rceil} - m$ *i.e.*, $2^3 - 5 = 3$ values of r (*i.e.*, $r = 0, 1, 2$) will be represented by the 2 bit binary representation of r and next two values (*i.e.*, $r = 3, 4$) will be represented by the 3 bit $\lceil \log_2 5 \rceil$ representation of $r + 3$ ($r + 2^{\lceil \log_2 m \rceil} - m = r + 23 - 5 = r + 3$).
3. Golomb code for $m = 5$:

n	q	r	Codeword
0	0	0	000
1	0	1	001
2	0	2	010
3	0	3	0110
4	0	4	0111
5	1	0	1000
6	1	1	1001
7	1	2	1010
8	1	3	10110
9	1	4	10111
10	2	0	11000
11	2	1	11001
12	2	2	11010
13	2	3	110110
14	2	4	110111
15	3	0	111000

Que 2.15. How Rice code can be viewed ? Explain the implementation of the Rice code in the recommendation for lossless compression from the Consultative Committee on Space Data Standard. Explain adaptive Huffman coding. How is it different from conventional Huffman coding ? **UPTU 2012-13, Marks 10**

Answer

Rice code :

1. The Rice code was originally developed by Robert F. Rice.
2. The Rice code can be viewed as an adaptive Golomb code.
3. In the Rice code, a sequence of non negative integers (which might have been obtained from the preprocessing of other data) is divided into blocks of J integers apiece.
4. Each block is then coded using one of several options, most of which are a form of Golomb codes.
5. Each block is encoded with each of these options, and the option resulting in the least number of coded bit is selected.
6. The particular option used is indicated by an identifier attached to the code for each block.

CCSDS recommendation for lossless compression :

1. The algorithm consists of a preprocessor (the modeling step) and a binary coder (coding step).
2. The preprocessor removes correlation from the input and generates a sequence of non negative integers.
3. This sequence has the property that smaller values are more probable than larger values.
4. The binary coder generates a bitstream to represent the integer sequence.
5. The binary coder is our main focus at this point.
6. The preprocessor functions as follows : Given a sequence $\{y_i\}$, for each y_i we generate a prediction $\hat{y}_i$.
7. A simple way to generate a prediction would be to take the previous value of the sequence to be a prediction of the current value of the sequence.
$$\hat{y}_i = y_i - 1$$
8. We then generate a sequence whose element are the difference between y_i and its predicted value $\hat{y}_i$.
$$d_i = y_y - \hat{y}_i$$
9. The d_i value will have a small magnitude when our prediction is good and a large value when it is not.
10. Assuming an accurate modeling of the data, the former situation is more likely than the latter.
11. Let y_{max} and y_{min} be the largest and smallest values that the sequence $\{y_i\}$ takes on.
12. It is reasonable to assume that the values of $\hat{y}$ will be confined to the range $[y_{min}, y_{max}]$.
13. Define
$$T_i = \min \{y_{max} - \hat{y}, \hat{y} - y_{min}\} \quad ...(2.15.1)$$
14. The sequence $\{d_i\}$ can be converted into a sequence of non negative integer $\{x_i\}$ using the following mapping :
$$x_i = \begin{cases} 2d_i & 0 \le d_i \le T_i \\ 2|d_i| - 1 & -T_i \le di < 0 \\ T_i + |d_i| & \text{otherwise} \end{cases} \quad ...(2.15.2)$$
15. The value of x_i will be small whenever the magnitude of d_i is small.
16. Therefore, the value of x_i will be small with higher probability.
17. The sequence $\{x_i\}$ is divided into segments with each segment being further divided into blocks of size J.
18. It is recommended by CCSDS that J have a value of 16.

19. The coded block is transmitted along with an identifier that indicates which particular option was used.
20. Each block is then coded using one of the following options :

a. Fundamental sequence :

i. This is unary code.

ii. A number n is represented by a sequence of n 0s followed by a 1 (or a sequence of n 1s followed by a 0.)

b. Split sample options :

i. These options consist of a set of codes indexed by a parameter m.

ii. The code for a k-bit number n using the mth split sample option consists of the m least significant bits of k followed by a unary code representing the $k - m$ most significant bits.

iii. For example, suppose we wanted to encode the 8-bit number 23 using the third split sample option.

iv. The 8-bit representation of 23 is 00010111.

v. The three least significant bits are 111.

vi. The remaining bits (00010) correspond to the number 2, which has a unary code 001.

vii. Therefore, the code for 23 using the third split sample option is 111011.

viii. Notice that different values of m will be preferable for different values of x_i, with higher values of m used for high-entropy sequences.

c. Second extension option :

i. The second extension option is useful for sequence with low entropy when, in general, many of the values of x_i will be zero.

ii. In the second extension option, the sequence is divided into consecutive pairs of samples.

iii. Each pair is used to obtain an index γ using the following transformation :

$$\gamma = \frac{1}{2}(x_i + x_{i+1})(x_i + x_{i+1} + 1) \qquad ...(2.15.3)$$

and the value of γ is encoded using a unary code.

iv. The value of γ is an index to a lookup table with each value of γ corresponding to a pair of values x_i, x_{i+1}.

d. Zero block option :

i. The zero block option is used when one or more of the block of z_i are zero generally when we have long sequences of y_i that have the same value.

ii. In this case, the number of zero blocks is transmitted using the code shown in Table 2.15.1.

iii. The ROS code is used when the last five or more block in a segment are all zero.

21. The Rice code has been used in several space applications, and variations of the Rice code have been proposed for a number of different applications.

Table 2.15.1. Code used for zero block option

Number of All-Zero Blocks	Codeword
1	1
2	01
3	001
4	0001
5	00001
6	000001
⋮	⋮
63	$\overbrace{000..01}^{63\,0s}$
ROS	00001

Adaptive Huffman coding : Refer Q. 2.9, Page 40D, Unit-2.

Different from conventional Huffman coding : Refer Q. 2.10, Page 42D, Unit-2.

Que 2.16. Explain Golomb codes and Tunstall codes.

UPTU 2011-12, Marks 05

Answer

Golomb code : Refer Q. 2.13, Page 48D, Unit-2.

Tunstall codes :

1. Most of the variable-length codes that we look encode letters from the source alphabet using codewords with varying numbers of bits : codewords with fewer bits for letters that occur more frequently and codewords with more bits for letters that occur less frequently.
2. The Tunstall code is an important exception.
3. In the Tunstall code, all codewords are of equal length.
4. However, each codeword represents a different number of letters.
5. An example of a 2-bit Tunstall code for an alphabet $A = \{A, B\}$ is shown in Table 2.16.1.
6. The main advantage of a Tunstall codes is that errors in codewords do not propagate, unlike other variable-length codes, such as Huffman codes, in which an error in one codeword will cause a series of errors to occur.

Table 2.16.1 : A 2-bit Tunstall code

Sequence	Codeword
AAA	00
AAB	01
AB	10
B	11

Que 2.17. **For an alphabet $A = \{a_1, a_2, a_3, a_4, a_5\}$ with probabilities $P(a_1) = 0.15$, $P(a_2) = 0.04$, $P(a_3) = 0.26$, $P(a_4) = 0.05$ and $P(a_5) = 0.50$.**

i. Calculate the entropy of this source.

ii. Find a Huffman code for this source.

iii. Find the average length of the code in (ii) and its redundancy. And also design a 3-bit Tunstall code.

For an alphabet $A = \{a_1, a_2, a_3\}$ with probabilities $P(a_1) = 0.7$, $P(a_2) = 0.2$, $P(a_3) = 0.1$. Design a 3-bit Tunstall code. **UPTU 2012-13, Marks 10**

Answer

i. Entropy of the source is given by,

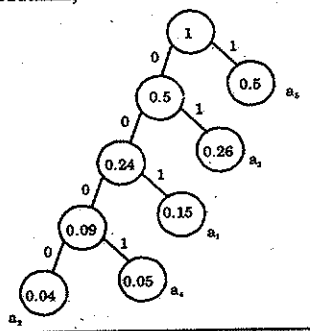
$$H = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

$$H = -[0.15 \log_2 0.15 + 0.04 \log_2 0.04 + 0.26 \log_2 0.26 + 0.05 \log_2 0.05 + 0.5 \log_2 0.5]$$

$$= [0.412 + 0.186 + 0.507 + 0.217 + 0.502]$$

$$= 1.824$$

ii. Apply Huffman algorithm,



Symbol	Codeword	Code length
a_5	1	1
a_3	01	2
a_1	001	3
a_2	0000	4
a_4	0001	4

iii. Average length of code

$$= 0.5 \times 1 + 0.26 \times 2 + 0.15 \times 3 + 0.05 \times 4 + 0.04 \times 4$$

$$= 0.5 + 0.52 + 0.45 + 0.2 + 0.16$$

$$= 1.83$$

Redundancy = Entropy – Average length of code

$$= 1.82 - 1.83$$

$$= -0.01 \text{ bits/symbol}$$

3-bit Tunstall code : For 3-bit Tunstall code, the maximum number of codewords in the codebook is $2^3 = 8$.

Symbol	Initial probability
a_1	0.7
a_2	0.2
a_3	0.1

Taking the symbol with highest probability (*i.e.*, a_1) and concatenate it with every other symbol and remove a_1 from the list. So, the entries in the codebook will be :

Sequence	Probability
a_2	0.2
a_3	0.1
a_1a_1	0.49
a_1a_2	0.14
a_1a_3	0.7

As number of codewords is less than maximum value, again apply the same procedure with a_1a_1 (maximum probability)

Sequence	Probability
a_2	0.2
a_3	0.1
a_1a_2	0.14
a_1a_3	0.7
$a_1a_1a_2$	0.098
$a_1a_1a_3$	0.049
$a_1a_1a_1$	0.343

We have to stop here as if we apply one more iteration, the number of codeword increases the maximum limit. Thus, the sequence will become :

Sequence	Codeword
a_2	000
a_3	001
a_1a_2	010
a_1a_3	011
$a_1a_1a_2$	100
$a_1a_1a_3$	101
$a_1a_1a_1$	110

Que 2.18. **For an alphabet $A = \{a_1, a_2, a_3\}$ with probabilities $P(a_1) = 0.7$, $P(a_2) = 0.2$, $P(a_3) = 0.1$**

Design a 3-bit Tunstall code. **UPTU 2013-14, Marks 05**

Answer

Refer Q. 2.16, Page 53D, Unit-2.

Que 2.19. Write short notes on the following :

i. Rice code

ii. Non-binary Huffman code **UPTU 2013-14, Marks 05**

Answer

i. Refer Q. 2.15, Page 50D, Unit-2.
ii. Refer Q. 2.6, Page 36D, Unit-2.

Que 2.20. Write short notes on the following :

i. Golomb codes

ii. Rice codes **UPTU 2014-15, Marks 05**

Answer

i. Refer Q. 2.13, Page 48D, Unit-2.
ii. Refer Q. 2.15, Page 50D, Unit-2.

Que 2.21. What is redundancy of code ? How it can be defined and calculated ?

OR

Explain redundancy code with the help of one example.

UPTU 2013-14, 2014-15, Marks 05

Answer

1. Sequence of symbols which are dependent upon one another is termed as redundant.
2. The main concept in detecting or correcting errors is redundancy.
3. Redundancy is achieved through various coding schemes.
4. The sender adds redundant bits through a process that creates a relationship between redundant bits and the actual bits.
5. The receiver checks the relationship between the two sets of bits to detect or correct errors.
6. The redundancy E is defined as,

$$E = 1 - \frac{H(y/x)}{H(x)}$$

where E is the redundancy of the message.

$H(y/x)$ is the entropy of the message where intersymbol influence is taken into amount.
$H(x)$ is the entropy assuming all symbols are independent.

7. The redundancy of a sequence of symbol can be measured by noting the amount by which the entropy has been reduced.

Que 2.22. What are the various application of Huffman coding and also give various steps required in encoding procedure ?

UPTU 2013-14, Marks 10

Answer

Applications of Huffman coding :

1. **Lossless image compression :**
   a. A simple application of Huffman coding to image compression would be to generate a Huffman code for the set of values that any pixel may take.
   b. For monochrome images, this set usually consists of integers from 0 to 255.
   c. The original (uncompressed) image representation uses 8 bits/pixel.
   d. The image consists of 256 rows of 256 pixels, so the uncompressed representation uses 65,536 bytes.
   e. The compression ratio is simply the ratio of the number of bytes in the uncompressed representation to the number of bytes in the compressed representation.
   f. The number of bytes in the compressed representation includes the number of bytes needed to store the Huffman code.
2. **Text compression :**
   a. Text compression seems natural for Huffman coding.
   b. In text, we have a discrete alphabet that, in a given class, has relatively stationary probabilities.
   c. For example, the probability model for a particular novel will not differ significantly from the probability model for another novel.
   d. Huffman coding is very much used for text compression.
   e. It is known that for two documents that are substantially different, the two set of probabilities are very much alike.
   f. Experiment shows that on an average file size dropped from 70000 bytes to about 43000 bytes with Huffman coding.
3. **Audio compression :**
   a. Huffman coding shows significant amount of reduction in size of an audio file.

b. Uncompressed CD quality audio data requires enormous amount of bandwidth for transmission.

c. Applying Huffman coding reduces its size thus a lower bandwidth is required for transmission of audio data over a communication link.

Que 2.23. Write down the applications of Huffman coding in text compression and audio compression.

OR

Explain lossless image compression with an example.

UPTU 2011-12, Marks 05

Answer

Refer Q. 2.22, Page 57D, Unit-2.

Que 2.24. Explain minimum variance Huffman code and encoding procedure taking a suitable example. What are the various applications of Huffman coding ? **UPTU 2012-13, Marks 10**

Answer

Minimum variance Huffman code : Refer Q. 2.4, Page 34D, Unit-2.

Encoding procedure :

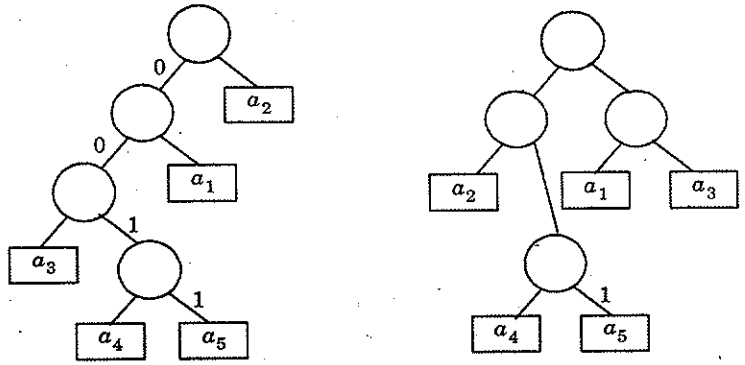
1. When more than two "symbols" in a Huffman tree have the same probability, different merge orders produce different Huffman codes.

Table 2.24.1.

Symbol	Step 1	Step 2	Step 3	Step 4	Codeword
a_2	0.4 →	0.4 →	0.4 →	0.6 0	00
a_1	0.2 →	0.2 →	0.4 ⌐0	0.4 1	10
a_3	0.2 →	0.2 ⌐0	0.2 ⌙1		11
a_4	0.1 ⌐0	0.2 ⌙1			010
a_5	0.1 ⌙1				011

The average codeword length is still 2.2 bits/ symbol. But variances are different.

2. Two code trees with same symbol probabilities :

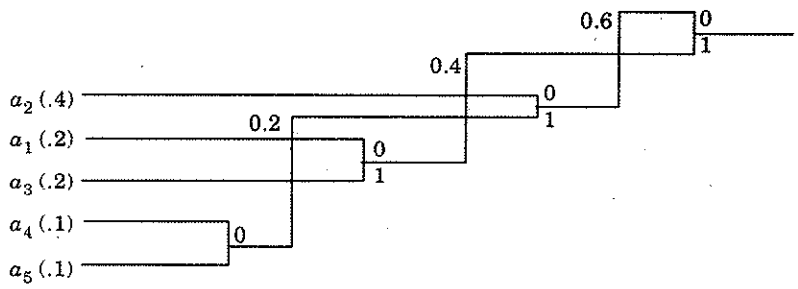


We prefer a code with smaller length variance.

Fig. 2.24.1.

3. Huffman's well-known coding method constructs a minimum redundancy code which minimizes the expected value of the word length.
4. We characterize the minimum redundancy code with the minimum variance of the word length.
5. An algorithm is given to construct such a code. It is shown that the code is in a certain unique sense.
6. Furthermore, the code is also shown to have a strong property in that it minimizes a general class of functions of the minimum redundancy codes as long as the functions are non-decreasing with respect to the path lengths from the root to the internal nodes of the corresponding decoding trees.
7. To build Huffman code with minimum variance you need to break ties with one of the following methods (of course probability of node is the most important) :
   a. Select node that represents shortest tree.
   b. Select node that was created earliest (consider leafs as created at start).
8. According to where in the list the combined source is placed, we obtain different Huffman codes with the same average length (same compression performance).
9. In some applications we do not want the codeword lengths to vary significantly from one symbol to another (example: fixed-rate channels).
10. To obtain a minimum variance Huffman code, we always put the combined symbol as high in the list as possible.

11. For example,



a_2 (.4)	00
a_1 (.2)	10
a_3 (.2)	11
a_4 (.1)	010
a_5 (.1)	011

Average length = 2.2 bits
Entropy = 2.122 bits

Applications of Huffman coding : Refer Q. 2.22, Page 57D, Unit-2.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Define Huffman coding and write its properties.
Ans. Refer Q. 2.1.

Q. 2. Explain minimum variance Huffman codes.
Ans. Refer Q. 2.4.

Q. 3. Draw the flowchart of encoding and decoding procedure.
Ans. Refer Q. 2.12.

Q. 4. Write a short note on :
i. Golomb code
ii. Rice code
iii. Tunstall code
Ans.
i. Refer Q. 2.13.
ii. Refer Q. 2.15.

iii. Refer Q. 2.16.

Q. 5. Design Colomb code for $m = 5$ where values of m are 0, 1, ..., 10.
Ans. Refer Q. 2.14.

Q. 6. Draw the Huffman tree for the following symbols whose frequency occurrence in a message text is started along with their symbol below :
A : 15, B : 6, C : 7, D : 12, E : 25, F : 4, G : 6, H : 10, I : 15
Decoding the message 1110100010111011.
Ans. Refer Q. 2.2.

3 UNIT

Arithmetic Coding

PART-1

Coding a Sequence, Generating a Binary Code, Comparison of Binary and Huffman Coding, Applications : Bi-Level Image Compression-The JBIG Standard, JBIG2, Image Compression.

CONCEPT OUTLINE : PART-1

- Arithmetic coding is useful when dealing with sources with small alphabets, such as binary sources.
- JBIG recommends arithmetic coding as the coding scheme for coding binary images.
- JBIG2 is an advance version of JBIG algorithm, it uses the same arithmetic coding scheme as JBIG.
- By using arithmetic coding, we get rates closer to the entropy than by using Huffman code.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.1. Define arithmetic coding in brief. Write about its application.

Answer

1. Method of generating variable-length codes called arithmetic coding.
2. Arithmetic coding is especially useful when dealihg with sources with small alphabets, such as binary sources, and alphabets with highly skewed probabilities.
3. It is also a very useful approach when, for various reason, the modeling and coding aspects of lossless compression are to be kept separate.
4. In arithmetic coding, a unique identifier or tag is generated for the sequence to be encoded.
5. This tag corresponds to a binary fraction, which becomes the binary code for the sequence.
6. The generation of the tag and the binary code are the same process.
7. However, the arithmetic coding approach is easier to understand if we conceptually divide the approach into two phases.
8. In the first phase, a unique identifier or tag is generated for a given sequence of symbols.

9. This tag is then given a unique binary code.
10. A unique arithmetic code can be generated for a sequence of length m without the need for generating codewords for all sequences of length m.

Applications of arithmetic coding :

1. Arithmetic coding is used in a variety of lossless and lossy compression applications.
2. It is a part of many international standards. In the area of multimedia, there are few principle organizations that develop standards.
3. Arithmetic coding is used in image compression, audio compression, and video compression standards.

Que 3.2. What do you mean by binary code ? How a sequence is coded ?

Answer

Binary code :

1. A binary code is a way of representing text or computer instructions by the use of the binary number system 0 and 1.
2. This is accomplished by assigning a bit string to each particular symbol or instruction.
3. For example, a binary string of eight binary digits (bits) can represent any of 256 symbols, letters or instructions.
4. In computing, binary codes are used for many methods of encoding data, such as character strings into bit strings.
5. These methods may be fixed-width or variable-width. In a fixed-width binary code, each letter, digit, or other interpreted as a binary number, is usually displayed in code tables in octal, decimal or hexadecimal notation.
6. There are many character sets and character encoding for them.
7. A bit string, interpreted as a binary number, can be translated into a decimal number.

Coding a sequence :

1. In order to distinguish a sequence of symbols from another sequence of symbols, we need to tag it with a unique identifier.
2. One possible set of tags for representing sequences of symbols are the numbers in the unit interval [0, 1).
3. Because the number of numbers in the unit interval is infinite, it should be possible to assign a unique tag to each distinct sequence of symbols.
4. In order to do this, we need a function that will map sequences of symbols into the unit interval.

5. A function that maps random variables and sequences of random variables into the unit interval is the cumulative distribution function (*cdf*) of the random variable associated with the source.
6. A random variable maps the outcomes, or sets of outcomes, of an experiment to values on the real number line.
7. For example, in a coin-tossing experiment, the random variable could map a head to zero and a tail to one (or it could map a head to 2367.5 and a tail to –192).
8. To use this technique, we need to map the source symbols or letters to numbers.
9. For convenience, we will use the mapping :

$$X(a_i) = i \qquad a_i \in A$$

where $A = \{a_1, a_2, ... a_m\}$ is the alphabet for a discrete source and X is a random variable.

10. This mapping means that given a probability model P for the source, we also have a probability density function for the random variable.

$$P(X = i) = P(a_i)$$

and the cumulative density function can be defined as,

$$F_X(i) = \sum_{k=1}^{i} P(X = k)$$

Que 3.3. How a tag is generated in arithmetic coding ?

Answer

1. The procedure for generating the tag works by reducing the size of the interval in which the tag resides as more and more elements of the sequence are received.
2. Start by first dividing the unit interval into sub-intervals of the form $[F_x(i-1), F_x(i)]$, $i = 1, ..., m$.
3. Because the minimum value of the *cdf* is zero and the maximum value is one, this exactly partitions the unit interval.
4. The sub-interval $[F_x(i-1), F_x(i)]$ with the symbol a_i.
5. The appearance of the first symbol in the sequence restricts the interval containing the tag to one of these sub-intervals.
6. Suppose the first symbol was a_k.
7. Then the interval containing the tag value will be the sub-interval $[F_x(k-1), f_x(k)]$.
8. This sub-interval is now portioned in exactly the same proportions as the original interval.
9. The j^{th} interval corresponding to the symbol a_j is given by,

$[F_x(k-1) + F_x(j-1)/F_x(k) - F_x(k-1)), F_x(k-1) + F_x(j)/F_x(k) - F_x(k-1)))$.

10. Thus, the second symbol in the sequence is a_j, then the interval contains the tag value becomes $[F_x(k-1) + F_x(j-1)/F_x(k) - F_x(k-1) + F_x(j)/(F_x(k) - F_x(k-1)))$.
11. Each succeeding symbol causes the tag to be restricted to a sub-interval that's further partitioned in the same proportions.

Que 3.4. **Find the real valued tag for the sequence $a_1a_2a_3a_2a_3a_1$ over letter $\{a_1a_2a_3\}$ with probabilities {0.2, 0.3, 0.5}.**

Answer

Given : $P(a_1) = 0.2, P(a_2) = 0.3, P(a_3) = 0.5$

To find : Real valued tag for the sequence $a_1a_2a_3a_2a_3a_1$.

$\therefore$ seq = 123231

As we know, $F_x(i) = \sum_{k=1}^{i} P(x = k)$

$\therefore$ $F_x(1) = 0.2$

$F_x(2) = 0.2 + 0.3 = 0.5$

$F_x(3) = 0.5 + 0.5 = 1$

For any sequence, lower and upper limit is given by :

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_x(x_n - 1)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_x(x_n)$$

Initialize $l^{(0)} = 0$ and $u^{(0)} = 1$

Ist element of sequence is 1 so,

$$l^1 = l^{(1-1)} + (u^0 - l^0) F_x(0)$$

$$l^1 = 0 + (1 - 0) \times 0$$

$$l^1 = 0$$

$$u^1 = l^0 + (u^0 - l^0) F_x(1)$$

$$= 0 + (1 - 0) \times 0.2$$

$$= 0.2$$

So, the tag is contained in the interval [0, 0.2].

Second element of sequence is 2 so,

$$l^2 = l^1 + (u^1 - l^1) F_x(1)$$

$$= 0.04$$

$$u^2 = l^1 + (u^1 - l^1) F_x(2)$$

$$= 0 + (0.2 - 0) \times 0.5$$

$$= 0.1$$

So, the tag is contained in the interval [0.04, 0.1].

Third element of sequence is 3 so,

$$l^3 = l^2 + (u^2 - l^2) F_x(2)$$

$$= 0.04 + (0.1 - 0.04) \times 0.5$$

$$= 0.07$$

$$u^3 = l^2 + (u^2 - l^2) F_x(3)$$

$$= 0.04 + (0.1 - 0.04) \times 1$$

$$= 0.1$$

So, the tag is contained in the interval [0.07, 0.1].

Fourth element of sequence is 2 so,

$$l^4 = l^3 + (u^3 - l^3) F_x(1)$$

$$l^4 = 0.07 + (0.1 - 0.07) \times 0.2$$

$$= 0.076$$

$$u^4 = l^3 + (u^3 - l^3) F_x(2)$$

$$= 0.07 + (0.1 - 0.07) \times 0.5$$

$$= 0.085$$

So, the tag is contained in the interval [0.076, 0.085].

Fifth element of sequence is 3 so,

$$l^5 = l^4 + (u^4 - l^4) F_x(2)$$

$$= 0.076 + (0.085 - 0.076) \times 0.5$$

$$= 0.0805$$

$$u^5 = l^4 + (u^4 - l^4) F_x(3)$$

$$= 0.076 + (0.085 - 0.076) \times 1$$

$$= 0.085$$

Now, the interval for the tag is [0.0805, 0.085].

Sixth element of sequence is 1 so,

$$l^6 = l^5 + (u^5 - l^5) F_x(0)$$

$$l^6 = 0.0805$$

$$u^6 = l^5 + (u^5 - l^5) F_x(1)$$

$$= 0.0805 + (0.085 - 0.0805) \times 0.2$$

$$= 0.0814$$

Tag = Mid of interval = $\bar{T}_x(x)$

$$\bar{T}_x(x) = \frac{u(n) + l(n)}{2}$$

$$\bar{T}_x(123231) = \text{Tag}(123231) = \frac{0.0805 + 0.0814}{2} = 0.08095$$

$\therefore$ Real valued tag for the given sequence is 0.08095.

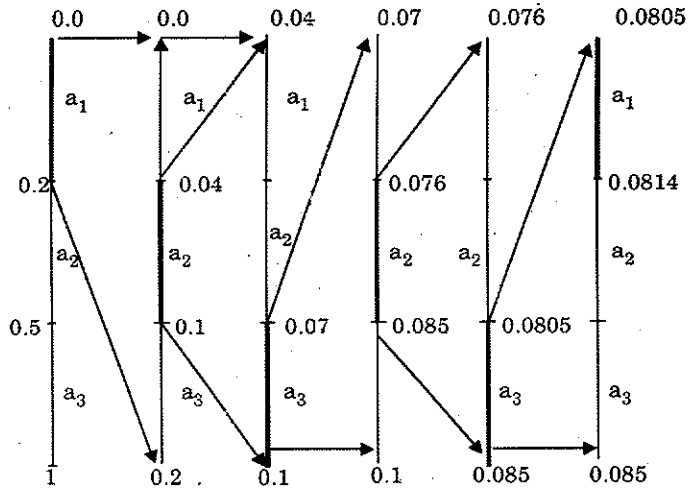


Fig. 3.4.1.

Que 3.5. **What are the advantages and problems of arithmetic coding ?**

Answer

Advantages of arithmetic coding :

1. Arithmetic coding is naturally suitable for adaptation strategies.
2. Arithmetic coding is close to optimal compression performance for sources with very low entropies.
3. It is easy to implement a system with multiple arithmetic codes.

Problems of arithmetic coding :

1. Unlike Huffman coding, the decoder requires knowledge of the number of symbols to be decoded.
2. Transmission of coded data cannot begin until the whole sequence is encoded.
3. Infinite precision is required.

Que 3.6. **Compare Huffman coding and arithmetic coding. How a tag is generated in arithmetic coding ?** **UPTU 2011-12, Marks 05**

Answer

1. Huffman algorithm requires building the entire code for all possible sequences of length m. If the original alphal ize of codebook is k^m. And for arithmetic coding, we do not need to build entire codebook instead we simply obtain the code for the tag corresponding to a given sequence.
2. By using arithmetic coding, we get rates closer to the entropy than by using Huffman code.
3. In Huffman code, we are guaranteed to obtain rate within 0.086 + P_{max} of the entropy. When the alphabet size is relatively large and probability is not too skewed, Huffman provide better result than arithmetic code.
4. Arithmetic code comes with extra complexity than Huffman code.
5. It is easy to implement multiple arithmetic code in a system than Huffman code.
6. It is much easier to adapt arithmetic codes to changing input statistics.

Generating a tag in arithmetic coding : Refer Q. 3.3, Page, Unit-3.

Que 3.7. **What do you mean by binary code ? Compare binary code with Huffman code.** **UPTU 2011-12, Marks 05**

Answer

Binary code : Refer Q. 3.2, Page 64D, Unit-3.

Comparison of binary code with Huffman code :

Binary code : In binary code, we can generate codewords for group or sequence of symbols.

Huffman code : In Huffman code, we need separate codeword for each symbol in a sequence.

Que 3.8. **Explain the JBIG standard of bi-level image compression.** **UPTU 2014-15, Marks 05**

Answer

1. The arithmetic coding approach is particularly amenable to the use of multiple coders.
2. All coders use the same computational machinery, with each coder using a different set of probabilities.
3. The JBIG algorithm makes full use of this feature of arithmetic coding.
4. Instead of checking to see if most of the pixels in the neighbourhood are white or black, the JBIG encoder uses the pattern of pixels in the neighbourhood, or context, to decide which set of probabilities to use in encoding a particular pixel.
5. If the neighbourhood consists of 10 pixels, with each pixel capable of taking on two different values, the number of possible patterns is 1024.
6. The JBIG coder uses 1024 to 4096 coders, depending on whether a low- or high-resolution layer is being encoded.

7. For the low-resolution layer, the JBIG encoder uses one of the two different neighbourhoods as shown in Fig. 3.8.1.

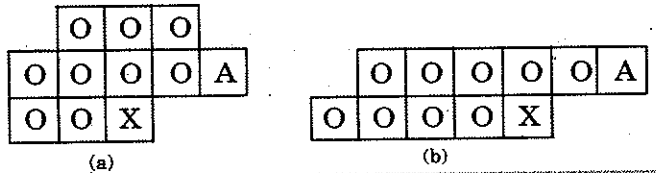


Fig. 3.8.1. (a) Three line and (b) Two line neighbourhoods.

8. The pixel to be coded is marked X, while the pixels to be used for templates are marked O or A.
9. The A and O pixels are previously encoded pixels and are available to both encoder and decoder.
10. The A pixel can be thought of as a floating member of the neighbourhood. Its placement is dependent on the input being encoded.
11. In the JBIG standard, the 1024 arithmetic coders are a variation of the arithmetic coder known as the QM coder.
12. The QM coder is a modification of an adaptive binary arithmetic coder called the Q coder which in turn is an extension of another binary adaptive arithmetic coder called the skew coder.

Que 3.9. Define progressive transmission. Compare MH, MR, MMR and JBIG.

Answer

1. In some applications, we may not always need to view an image at full resolution.
2. For example, if we are looking at the layout of a page, we may not need to know what each word or letter on the page is.
3. The JBIG standard allows for the generation of progressively lower-resolution images.
4. If the user is interested in some gross patterns in the image (for example, if they were interested in seeing if there were any figures on a particular page) they could request a lower-resolution image, which could be transmitted using fewer bits.
5. Once the lower-resolution image was available, the user could decide whether a higher-resolution image was necessary.
6. The JBIG specification recommends generating one lower-resolution pixel for each 2 × 2 block in the higher-resolution image.
7. The number of lower-resolution images (called layers) is not specified by JBIG.

8. A straightforward method for generating lower-resolution images is to replace every 2 × 2 block of pixels with the average value of the four pixels, thus reducing the resolution by two in both the horizontal and vertical directions. This approach works well as long as three of the four pixels are either black or white.
9. However, when we have two pixels of each kind, we run into trouble; consistently replacing the four pixels with either a white or black pixel causes a severe loss of detail, and randomly replacing with a black or white pixel introduces a considerable amount of noise into the image.
10. Instead of simply taking the average of every 2 × 2 block, the JBIG specification provides a table-based method for resolution reduction. The table is indexed by the neighbouring pixels as shown in Fig. 3.9.1, in which the circles represent the lower-resolution layer pixels and the squares represent the higher-resolution layer pixels.
11. Each pixel contributes a bit to the index. The table is formed by computing the expression :

$$4e + 2(b + d + f + h) + (a + c + g + i) - 3(B + C) - A$$

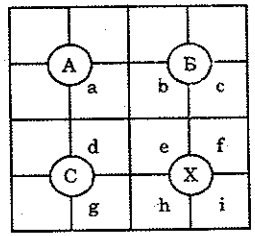


Fig. 3.9.1. Pixels used to determine the value of lower-level pixel.

Comparison of MH, MR, MMR and JBIG :

1. The JBIG algorithm performs better than the MMR algorithm, which performs better than the MR algorithm, which in turn performs better than the MH algorithm.
2. The level of complexity also follows the same trend, although we could argue that MMR is actually less complex than MR.
3. A comparison of the schemes for some facsimile sources is shown in Table 3.9.1.
4. The modified READ algorithm was used with $K = 4$, while the JBIG algorithm was used with an adaptive three line template and adaptive arithmetic coder to obtain the results.
5. As we go from the one-dimensional MH coder to the two-dimensional MMR coder, we get a factor of two reduction in file size for the sparse text sources.
6. We get even more reduction when we use an adaptive coder and an adaptive model, as is true for the JBIG coder.

7. When we come to the dense text, the advantage of the two-dimensional MMR over the one-dimensional MH is not as significant as the amount of two-dimensional correlation becomes substantially less.

Table 3.9.1. Comparison of binary image coding schemes

Source Description	Original Size (pixels)	MH (bytes)	MR (bytes)	MMR (bytes)	JBIG (bytes)
Letter	4352 × 3072	20,605	14,290	8,531	6,682
Sparse text	4352 × 3072	26,155	16,676	9,956	7,696
Dense text	4352 × 3072	135,705	105,684	92,100	70,703

Que 3.10. Write a short note on JBIG2.

Answer

1. The JBIG2 standard was approved in February of 2000.
2. Besides facsimile transmission, the standard is also intended for document storage, archiving, wireless transmission, print spooling, and coding of images on the web.
3. The standard provides specifications only for the decoder, leaving the encoder design open.
4. This means that the encoder design can be constantly refined, subject only to compatibility with the decoder specifications.
5. This situation also allows for lossy compression, because the encoder can incorporate lossy transformations to the data that enhance the level of compression.
6. The compression algorithm in JBIG provides excellent compression of a generic bi-level image.
7. The compression algorithm proposed for JBIG2 uses the same arithmetic coding scheme as JBIG.
8. However, it takes advantage of the fact that a significant number of bi-level images contain structure that can be used to enhance the compression performance.
9. A large percentage of bi-level images consist of text on some background, while another significant percentage of bi-level images are or contain halftone images.
10. The JBIG2 approach allows the encoder to select the compression technique that would provide the best performance for the type of data.

11. To do so, the encoder divides the page to be compressed into three types of regions called symbol regions, halftone regions, and generic regions.
12. The symbol regions are those containing text data, the halftone regions are those containing halftone images, and the generic regions are all the regions that do not fit into either category.
13. The partitioning information has to be supplied to the decoder.
14. The decoder requires that all information provided to it be organized in segments that are made up of a segment header, a data header, and segment data.
15. The page information segment contains information about the page including the size and resolution.
16. The decoder uses this information to set up the page buffer.
17. It then decodes the various regions using the appropriate decoding procedure and places the different regions in the appropriate location.

Que 3.11. What do you mean by binary code ? Compare binary code with Huffman code. What are adaptive compression schemes ? What is the basic difference between adaptive and statistical compression scheme ? Discuss with the model of adaptive compression. **UPTU 2012-13, Marks 10**

Answer

Binary Code : Refer Q. 3.2, Page 64D, Unit-3.

Comparison of binary code with Huffman code : Refer Q. 3.7, Page 69D, Unit-3.

Adaptive compression :

1. So far we have seen two algorithms for compression of message or text which used static data and input text.
2. It generated the code based on statistical data which contains code for each symbol calculated before and then accordingly codes the current text.
3. These were the models of statistical compression which retains its disadvantage of compression based on previous data statistics.
4. In adaptive compression model, the static data is self modified so it does not needs itself to be scanned only once.
5. Examples of adaptive compression methods :
   a. Adaptive arithmetic coding
   b. Method LZW

c. Method PPMC

Difference between adaptive and statistical compression scale :

1. With respect to compression of statistical model, the adaptive model has an advantage of better compression ratio.
2. This model also corresponds to the probability (or frequency) of characters in the input message stream.
3. Here in this method, the static data is updated according to input symbol and then coded as per the regulations of compression model.
4. A similar procedure is followed at receiving end to decode the data.

Model of adaptive compression :

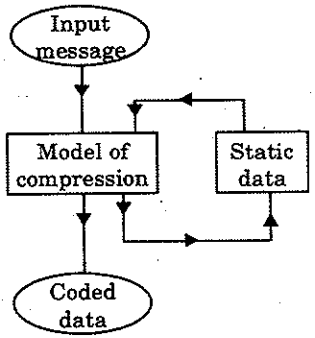


Fig. 3.11.1. Adaptive compression model

PART-2

Dictionary Techniques : Introduction, Static Dictionary, Digram Coding, Adaptive Dictionary, The LZ77 Approach, The LZ78 Approach, Applications : File Compression, UNIX Compress, Image Compression : The Graphic Interchange Format (GIF) and Compression Over Modems : V.42 bis.

CONCEPT OUTLINE : PART-2

- Static dictionary is static in nature *i.e.*, the entry in the dictionary are predefined.
- Adaptive dictionary are adaptive in nature *i.e.*, the entries in the dictionary are constructed as the symbols are received for encoding.
- LZ77 and LZ78 algorithms are used in adaptive dictionary.
- One of the more common forms of static dictionary coding is static diagram coding.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.12. Discuss the role of dictionary technique. Explain the concept of static dictionary.

Answer

Role of dictionary technique :

1. In many applications, the output of source consists of recurring patterns *i.e.*, certain pattern recur more frequently and some pattern do not occur, or if they do, occur with great rarity.
2. In such cases, encoding of the source can be done by a list or dictionary of frequently occurring patterns.
3. When the pattern in the dictionary appears in the source output, they are encoded with reference to the dictionary.
4. If the pattern does not appear in the dictionary, then it can be encoded using some other, less efficient, method.
5. In effect, we are splitting the input into two classes : frequently occurring patterns and infrequently occurring patterns.
6. For this technique to be effective, the class of frequency occurring patterns, and hence the size of the dictionary, must be much smaller than the number of all possible patterns.

Static dictionary :

1. Static dictionary is static in nature *i.e.*, the entry in the dictionary are predefined.
2. Static dictionary is mainly useful when prior knowledge of the output of the source is available.
3. They form application specific dictionary because the content of dictionary will vary from application to application.
4. Two application of different domain will have different recurring patterns of word thus making different application specific dictionary.
5. For example, if we want to compress the library record of a university, a static dictionary approach may be useful because we know in advance that words such as book title, ISBN number which are going to appear in almost all record and they can be included in the static dictionary.
6. There could be a number of other situation in which an application-specific or data-specific static dictionary based coding scheme would be the most efficient.

7. It should be noted that these schemes would work well only for the applications and data they were designed for.
8. If these schemes were to be used with different applications, they may cause an expansion of the data instead of compression.
9. A static dictionary technique that is less specific to a single application is digram coding.

Que 3.13. Explain the digram coding.

Answer

1. One of the more common forms of static dictionary coding is static digram coding.
2. In this, dictionary consists of all letters of the source alphabet followed by as many pairs of letters, called digrams.
3. The digram encoder reads a two character input and searches the dictionary to see if this input exists in the dictionary.
4. If it does, corresponding index is encoded and transmitted.
5. If it does not, the first character of pair is encoded.
6. Second character in the pair becomes the first character of next digram.
7. The encoder reads another character to complete digram, and search procedure is repeated.
8. **Example :** We have five letter alphabet A = {a, b, c, d, r}. Based on knowledge about the source, we build the dictionary as shown in Table 3.13.1.

Table 3.13.1.

Code	Entry	Code	Entry
000	a	100	r
001	b	101	ab
010	c	110	ac
011	d	111	ad

9. Suppose we want to encode the sequence *abracadabra*.
10. Encoder reads the first two characters *ab* and check to see if they exist in dictionary.
11. It does and encoded with codeword 101.
12. Then encoder reads the next two character *ra* and checks its existence.
13. If does not exists then encoder encodes *r* with codeword 100 and then reads one more character *c* to make two character pattern *ac*.
14. Continuing in this fashion, output string is 101100110111101100000.

Que 3.14. Where we use the dictionary techniques of encoding ? Also, explain various types of dictionary techniques.

UPTU 2013-14, Marks 05

Answer

Use of dictionary techniques of encoding : Refer Q. 3.12, Page 75D, Unit-3.

Types of dictionary techniques :

1. **Static dictionary :** Refer Q. 3.12, Page 75D, Unit-3.
2. **Adaptive dictionary :** Adaptive dictionary are adaptive in nature *i.e.*, the entries in the dictionary are constructed as the symbols are received for encoding. LZ algorithms used adaptive approach for building the dictionary. There is no need to transmit or store dictionary.

Que 3.15. Give differences between static and adaptive dictionary coding scheme.

UPTU 2011-12, Marks 05

Answer

S.No.	Static dictionary	Adaptive dictionary
1.	This technique is static in nature.	This technique is adjustable in nature.
2.	Prior knowledge of the output source should be available.	There is no need of prior knowledge.
3.	In static dictionary compression, the dictionary of characters and all possible frequently appearing groups of characters is prepared.	In adaptive dictionary compression, the dictionary is prepared on sequence of characters appearing in the text being compressed.
4.	Static dictionary based compression is digram coding.	LZ family of algorithm for example, LZ77, LZ78, LZW compression techniques builds the adaptive dictionary.

Que 3.16. Give LZ77 approach for adaptive dictionary based encoding.

UPTU 2014-15, Marks 05

OR

How LZ77 algorithm works ? What are the problems with LZ77 ? Explain with an example. **UPTU 2013-14, Marks 05**

Answer

LZ77 approach :

1. In the LZ77 approach, the dictionary is simply a portion of the previously encoded sequence.
2. The encoder examines the input sequence through a sliding window as shown in Fig. 3.16.1.
3. The window consists of two parts, a search buffer that contains a portion of the recently encoded sequence, and a look-ahead buffer that contains the next portion of the sequence to be encoded.
4. In Fig. 3.16.1, the search buffer contains eight symbols, while the look-ahead buffer contains seven symbols.
5. In practice, the sizes of the buffers are significantly larger; however, for the purpose of explanation, we will keep the buffer sizes small.
6. To encode the sequence in the look-ahead buffer, the encoder moves a search pointer back through the search buffer until it encounters a match to the first symbol in the look-ahead buffer.

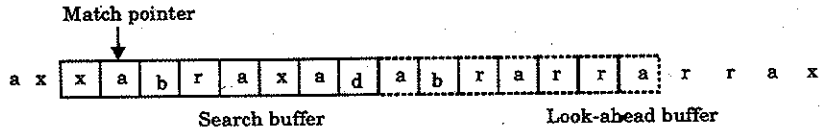


Fig. 3.16.1 Encoding using the LZ77 approach

7. The distance of the pointer from the look-ahead buffer is called the offset.
8. The encoder then examines the symbols following the symbol at the pointer location to see if they match consecutive symbols in the look-ahead buffer.
9. The number of consecutive symbols in the search buffer that match consecutive symbols in the look-ahead buffer, starting with the first symbol, is called the length of the match.
10. The encoder searches the search buffer for the longest match.
11. Once the longest match has been found, the encoder encodes it with a triple (o, l, c), where o is the offset, l is the length of the match, and c is the codeword corresponding to the symbol in the look-ahead buffer that follows the match.
12. For example, the pointer is pointing to the beginning of the longest match as shown in Fig. 3.16.2.
13. The offset o in this case is 7, the length of the match l is 4, and the symbol in the look-ahead buffer following the match is p.

14. The reason for sending the third element in the triple is to take care of the situation where no match for the symbol in the look-ahead buffer can be found in the search buffer.
15. In this case, the offset and match-length values are set to 0, and the third element of the triple is the code for the symbol itself.
16. If the size of the search buffer is S, the size of the window (search and look-ahead buffers) is W, and the size of the source alphabet is A, then the number of bits needed to code the triple using fixed-length codes is $[\log_2 S] + [\log_2 W] + [\log_2 A]$.
17. Notice that the second term is $[\log_2 W]$, not $[\log_2 S]$.
18. The reason for this is that the length of the match can actually exceed the length of the search buffer.
19. We will look at three different possibilities that may be encountered during the coding process :
    i. There is no match for the next character to be encoded in the window.
    ii. There is a match.
    iii. The matched string extends inside the look-ahead buffer.

Working of LZ77 :

1. **Encoding :**
    a. Suppose the sequence to be encoded is :
    ... cabracadabrarrarrad...
    b. Suppose the length of the window is 13, the size of the look-ahead buffer is six, and the current condition is as follows :

cabraca	*dabrar*

    with *dabrar* in the look-ahead buffer.
    c. We look back in the already encoded portion of the window to find a match for d. As we can see, there is no match, so we transmit the triple $(0, 0, C(d))$.
    d. The first two elements of the triple shows that there is no match to d in the search buffer, while $C(d)$ is the code for the character d.
    e. The contents of the buffer are

abracad	*abrarr*

    with *abrarr* in the look-ahead buffer.
    f. Looking back from the current location, we find a match to a at an offset of two. The length of this match is one.
    g. Looking further back, we have another match for a at an offset of four; again the length of the match is one.

h. Looking back even further in the window, we have a third match for *a* at an offset of seven.

i. However, this time the length of the match is four (see Fig. 3.16.2). So, we encode the string *abra* with the triple (7, 4, *C*(*r*)), and move the window forward by five characters. The window now contains the following characters :

adabrar	*rarrad*

j. Now the look-ahead buffer contains the string *rarrad*.

k. Looking back in the window, we find a match for *r* at an offset of one and a match length of one, and a second match at an offset of three with a match length of what at first appears to be three.

l. We can use a match length of five instead of three.

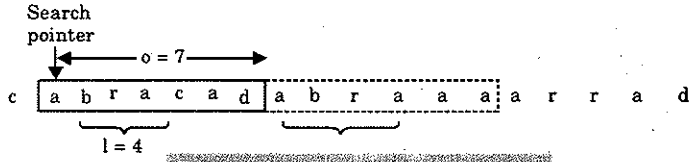


Fig. 3.16.2. The encoding process.

2. Decoding process :

a. Let us assume that we have decoded the sequence *cabraca* and we receive the triples (0, 0, *C*(*d*)), (7, 4, *C*(*r*)), and (3, 5, *C*(*d*)).

b. The first triple is easy to decode; there was no match within the previously decoded string, and the next symbol is *d*.

c. The decoded string is now *cabracad*.

d. The first element of the next triple tells the decoder to move the copy pointer back seven characters, and copy four characters from that point. The decoding process works as shown in Fig. 3.16.3.

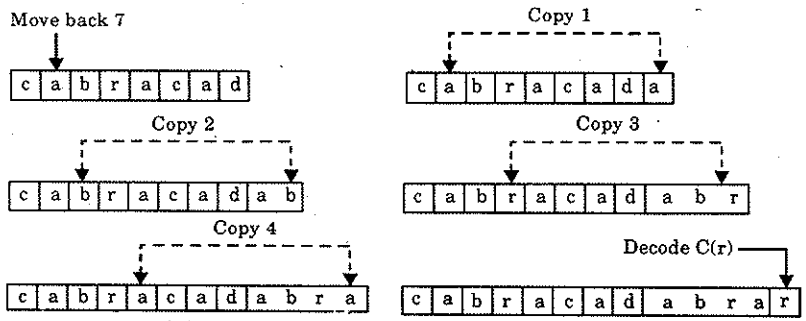


Fig. 3.16.3. Decoding of the triple (7, 4, C (r)).

Problems in LZ77 are :

1. In order to work efficiently, patterns should occur close together. This assumption is removed in LZ78.
2. The coding of < *o*, *l*, *c*> is not efficient.
3. Performance of LZ77 increases with the size of buffer. But large buffers require efficient search strategies.
4. Using a triplet to encode a single character is inefficient.

Que 3.17. Explain the LZ78 approach.

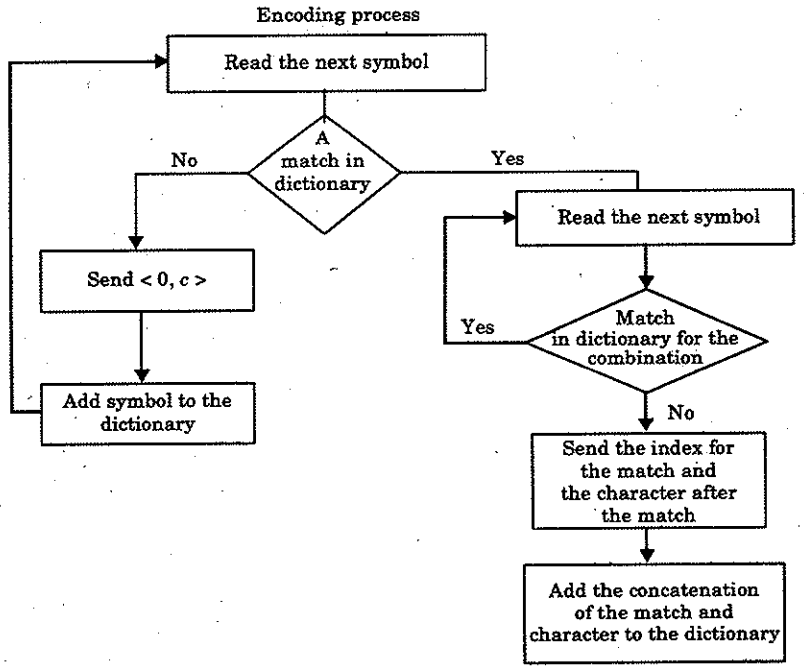
Answer

LZ78 algorithm :

1. While LZ77 algorithm work on past data, the LZ78 algorithm attempts to work on future data.
2. It does this by forward scanning the input buffer and matching it against the dictionary it maintain.
3. It will scan into the buffer until it cannot find a match in the dictionary.
4. At this point, it will output the location of the word in the dictionary if one is available, the match length and a character that cause a match failure.
5. The resulting word is added to the dictionary.
6. LZ78 compression algorithm drops the reliance on size of search buffer, thus overcoming the drawback of LZ77 algorithm.
7. In LZ78, encoder makes use of an explicit dictionary.
8. Phrases in the dictionary are built one at a time, adding a new symbol to an existing phrase when a match occurs.
9. The dictionary has to be built at both the encoder and decoder in an identical manner. Input are coded as <*i*, *c*> where

   i = index in the dictionary to the longest matching phrase.

   c = codeword of symbol following the match – index value of 0 is used in case of no match.



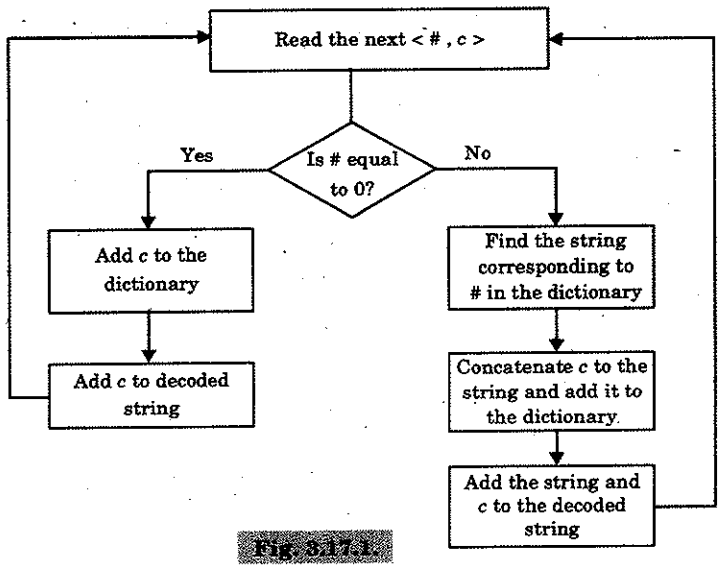


Fig. 3.17.1.

LZ78 compression algorithm have following weaknesses :

1. Dictionary grows without bound.
2. Long phrases appear late.
3. Inclusion of first non-matching symbol may prevent a good match.
4. Few substrings of the processed input are entered into the dictionary.

Que 3.18. **Encode the following sequence using LZ78 approach : Wabba_Wabba_Wabba_Wabba_Woo_Woo_Woo.**

Answer

Dictionary

Index	Entry	Encoded output
1	W	$\langle 0, C(W)\rangle$
2	a	$\langle 0, C(a)\rangle$
3	b	$\langle 0, C(b)\rangle$
4	ba	$\langle 3, C(a)\rangle$
5	$\_$	$\langle 0, C(\_)\rangle$
6	Wa	$\langle 1, C(a)\rangle$
7	bb	$\langle 3, C(b)\rangle$
8	$a\_$	$\langle 2, C(\_)\rangle$
9	Wab	$\langle 6, C(b)\rangle$
10	$ba\_$	$\langle 4, C(\_)\rangle$
11	$Wabb$	$\langle 9, C(b)\rangle$
12	a_W	$\langle 8, C(W)\rangle$
13	o	$\langle 0, C(o)\rangle$
14	$o\_$	$\langle 13, C(\_)\rangle$
15	Wo	$\langle 1, C(o)\rangle$
16	o_W	$\langle 14, C(W)\rangle$
17	oo	$\langle 13, C(o)\rangle$

Que 3.19. **Decode the sequence**

$\langle 0, C(B)\rangle$ $\langle 0, C(A)\rangle$ $\langle 1, C(A)\rangle$ $\langle 2, C(B)\rangle$ $\langle 0, C(R)\rangle$

$\langle 5, C(R)\rangle$ $\langle 2, \ \rangle$ **using LZ78 algorithm.**

Answer

1	2	3	4	5	6
⟨0,C(B)⟩	⟨0,C(A)⟩	⟨1,C(A)⟩	⟨2,C(B)⟩	⟨0,C(R)⟩	⟨5,C(R)⟩
↓	↓	↓	↓	↓	↓
B	A	String (1) + A	String 2 + B	R	String 5 + R
		↓	↓		↓
		BA	AB		RR

⟨2, ⟩
↓
String 2
↓
A

So, the decoded string is,

BABAABRRRA

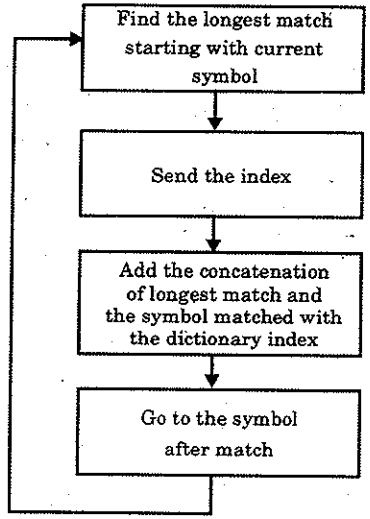
Que 3.20. Explain LZW compression algorithm.

Answer

1. Lempel – Ziu – Welch (LZW) is a universal lossless data compression algorithm created by Abrahan Lempel, Jacob Ziu and Terry Welch.
2. The algorithm is designed to be fast to implement but is not usually optimal because it performs only limited analysis of data.
3. The algorithm takes a string as input. It processes the string using the dictionary.
4. The string is encoded (and the dictionary grows) as the string is being processed.
5. Initially the dictionary contains all the possible characters (the alphabet) with their corresponding encoding.
6. The algorithm takes the longest word W (from the dictionary) that can replace the next character in the string.
7. It encode this part of the string with the encoding of W.
8. Now it takes the character C that followed W in the string and adds WC to the dictionary. This is repeated until the string is consumed.

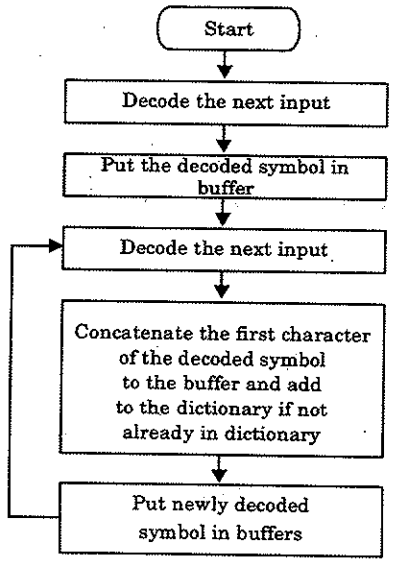
Steps for encoding :

1. Find the largest match "m", 'C' follows.
2. Encode match "m".
3. Add ("m", 'C') to dictionary.
4. Repeat from step 1, starting with 'C'.



Steps for decoding :

1. Decode an output pattern.
2. Add (previous pattern, first symbol in current pattern).
3. Go to step 2.



Que 3.21. **A sequence is encoded using LZW algorithm and the initial dictionary shown in the table.**

Index	Entry
1	*a*
2	*b*
3	*r*
4	*t*

The output of LZW encoder is the following sequence :

3	1	4	6	8	4	2	1	2	5	10	6	11	13	6

Decode this sequence. Discuss relative advantages of LZ77, LZ78 and LZW compression schemes.

UPTU 2011-12, Marks 05

UPTU 2012-13, Marks 10

Answer

Initial LZW dictionary :

Index	Entry
1	a
2	b
3	r
4	t

Output sequence : 3, 1, 4, 6, 8, 4, 2, 1, 2, 5, 10, 6, 11, 13, 6

1. The decoder starts with index values 3 which correspond to letter 'r', so we decode the first element of our sequence.
2. The next decoder input is 1, which is the index corresponding to letter a, we decode an 'a' and concatenate it with our current pattern to form the pattern 'ra'. 'ra' does not exist in the dictionary, we add it as fifth element of the dictionary.
3. Similarly, we get a new dictionary until index 13, which is as follows :

Index	Entry
1	a
2	b
3	r
4	t
5	ra
6	at
7	aa
8	ab
9	rb

10	bt
11	bta
12	btb
13	bb

Sequence after decoding is :

ratatabtbabrabtrabtabbat

Advantages of LZ77 compression scheme :

1. LZ77 algorithm uses limited window to find matches in previous text.
2. Speed of decompression is fast.
3. Simple to use.
4. Useful for compressing texts in a database or anything that we need to recover quickly.

Advantages of LZ78 compression scheme :

1. Major advantage of LZ78 algorithm is its speed.
2. It uses a variant of greedy parsing.
3. Simple to use.

Advantages of LZW compression scheme :

1. 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.
2. LZW compression is fast.
3. Royalties have to be paid to use LZW compression algorithms within applications.
4. LZW compression is the best technique for reducing the size of files containing more repetitive data.
5. LZW compression is fast and simple to apply. Since this is a lossless compression technique, none of the contents in the file are lost during or after compression.
6. LZW algorithm is efficient because it does not need to pass the string table to the decompression code.

Que 3.22. **Explain file compression : UNIX compress.**

Answer

1. The UNIX compress command is one of the earlier applications of LZW.
2. The size of the dictionary is adaptive.
3. We start with a dictionary of size 512.
4. This means that the transmitted codewords are 9 bits long. Once the dictionary has filled up, the size of the dictionary is doubled to 1024 entries.

5. The codewords transmitted at this point have 10 bits.
6. The size of the dictionary is progressively doubled as it fills up.
7. In this way, during the earlier part of the coding process when the strings in the dictionary are not very long, the codewords used to encode them also have fewer bits.
8. The maximum size of the codeword, b_{max}, can be set by the user to between 9 and 16, with 16 bits being the default.
9. Once the dictionary contains $2^{b^{max}}$ entries, compress becomes a static dictionary coding technique.
10. At this point, the algorithm monitors the compression ratio.
11. If the compression ratio falls below a threshold, the dictionary is flushed, and the dictionary building process is restarted.

Que 3.23. Discuss image compression—the Graphics Interchange Format (GIF). UPTU 2014-15, Marks 05

Answer

1. The Graphics Interchange Format (GIF) was developed by Compuserve Information Service to encode graphical images.
2. It is another implementation of the LZW algorithm and is very similar to the compress command. The compressed image is stored with the first byte being the minimum number of bits b per pixel in the original image.
3. The binary number 2^b is defined to be the clear code.
4. This code is used to reset all compression and decompression parameters to a start-up state. The initial size of the dictionary is 2^{b+1}.
5. When this fills up, the dictionary size is doubled, as was done in the compress algorithm, until the maximum dictionary size of 4096 is reached.
6. At this point, the compression algorithm behaves like a static dictionary algorithm.
7. The codewords from the LZW algorithm are stored in blocks of characters.
8. The characters are 8 bits long, and the maximum block size is 255.
9. Each block is preceded by a header that contains the block size.
10. The block is terminated by a block terminator consisting of eight 0s.
11. The end of the compressed image is denoted by an end-of-information code with a value of $2^b + 1$.
12. This codeword should appear before the block terminator.
13. GIF has become quite popular for encoding all kinds of images, both computer-generated and "natural" images.
14. While GIF works well with computer-generated graphical images, and pseudocolour or colour-mapped images, it is generally not the most efficient way to losslessly compress images of natural scenes, photographs, satellite images, and so on.

Que 3.24. What do you mean by compression over modems–V.42 bis ? UPTU 2014-15, Marks 05

Answer

1. The ITU-T Recommendation V.42 bis is a compression standard devised for use over a telephone network along with error-correcting procedures described in CCITT Recommendation V.42.
2. This algorithm is used in modems connecting computers to remote users.
3. The algorithm described in this recommendation operates in two modes, a transparent mode and a compressed mode.
4. In the transparent mode, the data are transmitted in uncompressed form, while in the compressed mode an LZW algorithm is used to provide compression.
5. The reason for the existence of two modes is that at times the data being transmitted do not have repetitive structure and therefore cannot be compressed using the LZW algorithm.
6. The V.42 bis of recommendation suggests periodic testing of the output of the compression algorithm to see if data expansion is taking place.
7. The exact nature of the test is not specified in the recommendation.
8. In the compressed mode, the system uses LZW compression with a variable-size dictionary.
9. The initial dictionary size is negotiated at the time a link is established between the transmitter and receiver.
10. The V.42 bis recommendation suggests a value of 2048 for the dictionary size.
11. It specifies that the minimum size of the dictionary is to be 512.
12. Suppose the initial negotiations result in a dictionary size of 512.
13. This means that our codewords that are indices into the dictionary will be 9 bits long.
14. Actually, the entire 512 indices do not correspond to input strings; three entries in the dictionary are reserved for control codewords.
15. These codewords in the compressed mode are shown in Table 3.24.1.
16. The V.42 bis recommendation avoids the need for an exception handler for the case, where the decoder receives a codeword corresponding to an incomplete entry by forbidding the use of the last entry in the dictionary.
17. Instead of transmitting the codeword corresponding to the last entry, the recommendation requires the sending of the codewords corresponding to the constituents of the last entry.

Table 3.24.1. Control codewords in compressed mode

Codeword	Name	Description
0	ETM	Enter transparent mode
1	FLUSH	Flush data
2	STEPUP	Increment codeword size

PART-3

Predictive Coding : Prediction with Partial Match (PPM) : The Basic Algorithm, The ESCAPE SYMBOL, Length of Context, The Exclusion Principle, The Burrows-Wheeler Transform : Move-to-Front Coding, CALIC JPEG-LS, Multiresolution Approaches, Facsimile Encoding, Dynamic Markov Compression.

CONCEPT OUTLINE : PART-3

- Predictive coding is decoding algorithm which uses the history of the sequence to predict the next candidate of the sequence.
- PPM algorithm uses large contexts to determine the probability of the symbol being encoded.
- JPEG is designed for compressing either full-colour (24 bit) or gray scale digital image of 'natural' scenes.
- In facsimile encoding, a page is scanned and converted into a sequence of black or white pixel and send over the communication channel.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.25. Define the term predictive coding.

Answer

1. If a decoding algorithm uses the history of the sequence to predict the next candidate of the sequence such coding schemes are known as predictive coding.
2. Multiresolution approach is generally used for image transmission.
3. Multiresolution model of the image regenerate the image with varying spatial resolution.

4. This multiresolution model creates a layered pyramid like structure in which each layer serve as a prediction model for the layer below it.

Que 3.26. Explain the concept of prediction with partial match algorithm.

Answer

1. The best-known context-based algorithm is the ppm algorithm.
2. The idea of the ppm algorithm is very simple.
3. We would like to use large contexts to determine the probability of the symbol being encoded.
4. However, the use of large contexts would require us to estimate and store an extremely large number of conditional probabilities, which might not be feasible.
5. Instead of estimating these probabilities ahead of time, we can reduce the burden by estimating the probabilities as the coding proceeds.
6. This way we only need to store those contexts that have occurred in the sequence being encoded.
7. This is a much smaller number than the number of all possible contexts.
8. While this mitigates the problem of storage, it also means that, especially at the beginning of an encoding, we will need to code letters that have not occurred previously in this context.
9. In order to handle this situation, the source code alphabet always contains an escape symbol, which is used to signal that the letter to be encoded has not been seen in this context.

Que 3.27. Discuss the steps involved in basic algorithm for Prediction with Partial Match (PPM).

UPTU 2011-12, 2012-13, Marks 10

UPTU 2014-15, Marks 05

Answer

1. The basic algorithm initially attempts to use the largest context.
2. The size of the largest context is predetermined.
3. If the symbol to be encoded has not previously been encountered in this context, an escape symbol is encoded and the algorithm attempts to use the next smaller context.

4. If the symbol has not occurred in this context either, the size of the context is further reduced.
5. This process continues until either we obtain a context that has previously been encountered with this symbol, or we arrive at the conclusion that the symbol has not been encountered previously in any context. In this case, we use a probability of $1/M$ to encode the symbol, where M is the size of the source alphabet.
6. For example, when coding the a of *probability*, we would first attempt to see if the string *proba* has previously occurred that is, if a had previously occurred in the context of *prob*.
7. If not, we would encode an escape and see if a had occurred in the context of *rob*.
8. If the string *roba* had not occurred previously, we would again send an escape symbol and try the context *ob*.
9. Continuing in this manner, we would try the context b, and failing that, we would see if the letter a (with a zero-order context) had occurred previously.
10. If a was being encountered for the first time, we would use a model in which all letters occur with equal probability to encode a.
11. This equiprobable model is sometimes referred to as the context of order –1.
12. As the development of the probabilities with respect to each context is an adaptive process, each time a symbol is encountered, the count corresponding to that symbol is updated.
13. The number of counts to be assigned to the escape symbol is not obvious, and a number of different approaches have been used.

Que 3.28. Write short notes on :

i. Escape symbol

ii. Length of context

Answer

i. Escape symbol :

1. Another method described by Cleary and Witten is to reduce the counts of each symbol by one and assign these counts to the escape symbol.
2. For example, suppose in a given sequence a occurs 10 times in the context of *prob*, l occurs 9 times, and o occurs 3 times in the same context (for example, problem, proboscis, etc.).

3. In Method A, we assign a count of one to the escape symbol, resulting in a total count of 23, which is one more than the number of times *prob* has occurred.
4. The situation is shown in Table 3.28.1.

Table 3.28.1. Counts using Method A

Context	Symbol	Count
prob	a	10
	l	9
	o	3
	(Esc)	1
Total Count		23

5. In the second method, known as Method B, we reduce the count of each of the symbols a, l, and o by one and give the escape symbol a count of three, resulting in the counts shown in Table 3.28.2.

Table 3.28.2. Counts using Method B

Context	Symbol	Count
prob	a	9
	l	8
	o	2
	(Esc)	3
Total Count		22

6. The reasoning behind this approach is that if in a particular context more symbols can occur, there is a greater likelihood that there is a symbol in this context that has not occurred before.
7. This increases the likelihood that the escape symbol will be used. Therefore, we should assign a higher probability to the escape symbol.

ii. Length of context :

1. It would seem that as far as the maximum length of the contexts is concerned, more is better.
2. However, this is not necessarily true.
3. A longer maximum length will usually result in a higher probability if the symbol to be encoded has a nonzero count with respect to that context.
4. However, a long maximum length also means a higher probability of long sequences of escapes, which in turn can increase the number of bits used to encode the sequence.

5. If we plot the compression performance versus maximum context length, we see an initial sharp increase in performance until some value of the maximum length, followed by a steady drop as the maximum length is further increased.
6. The value at which we see a downturn in performance changes depending on the characteristics of the source sequence.

Que 3.29. Discuss the exclusion principle.

Answer

1. The basic idea behind arithmetic coding is the division of the unit interval into subintervals, each of which represents a particular letter.
2. The smaller the subinterval, the more bits are required to distinguish it from other subintervals.
3. If we can reduce the number of symbols to be represented, the number of subintervals goes down as well.
4. This in turn means that the sizes of the subintervals increase, leading to a reduction in the number of bits required for encoding.
5. The exclusion principle used in ppm provides this kind of reduction in rate.
6. Suppose we have been compressing a text sequence and come upon the sequence *proba*, and suppose we are trying to encode the letter *a*.
7. Suppose also that the state of the two-letter context *ob* and the one-letter context *b* are as shown in Table 3.29.1.
8. First we attempt to encode *a* with the two-letter context.
9. As *a* does not occur in this context, we issue an escape symbol and reduce the size of the context.
10. Looking at the Table 3.29.1 for the one-letter context *b*, we see that *a* does occur in this context with a count of 4 out of a total possible count of 21.
11. Notice that other letters in this context include *l* and *o*.
12. However, by sending the escape symbol in the context of *ob*, we have already signalled to the decoder that the symbol being encoded is not any of the letters that have previously been encountered in the context of *ob*.
13. Therefore, we can increase the size of the subinterval corresponding to *a* by temporarily removing *l* and *o* from the Table 3.29.1.
14. Instead of using Table 3.29.1, we use Table 3.29.2 to encode *a*.
15. This exclusion of symbols from contexts on a temporary basis can result in cumulatively significant savings in terms of rate.

Table 3.29.1. Counts for exclusion example

Context	Symbol	Count
ob	*l*	10
		3
	(Esc)	2
Total Count		15
b	*l*	5
	0	3
	a	4
	r	2
	e	2
	(Esc)	5
Total Count		21

Table 3.29.2. Modified table used for exclusion example

Context	Symbol	Count
b	a	4
	r	2
	e	2
	(Esc)	3
Total Count		11

Que 3.30. Explain Burrows-Wheeler transform algorithm.

Answer

1. The Burrows-Wheeler transform also called block sorted compression, is an algorithm used in data compression technique such as bzip2.
2. It was invented by Michael Burrows and David Wheeler in 1994.
3. When a character string is transformed by the BWT, none of its character changes value.
4. The transformation permutes the order of the characters.

5. If the original string had several substrings that occurred often, then the transformed string will have several places where a single character is repeated multiple times in a row.

Algorithm :

a. Given a sequence of length N, we create N – 1 other sequences where each of these N – 1 sequences is a cyclic shift of the original sequence.

b. These N sequences are then arranged in lexicographic order.

c. The encoder then transmits the sequence of length N created by taking the last letter of each sorted, cyclically shifted sequence.

f. This sequence of last letters L, and the position of the original sequence in the sorted list, are coded and sent to the decoder.

Que 3.31. Encode the sequence "COMPRESSION" using Burrows-Wheeler transform algorithm.

Answer

1. Find the cyclic permutation of the sequence :

0	C	O	M	P	R	E	S	S	I	O	N
1	O	M	P	R	E	S	S	I	O	N	C
2	M	P	R	E	S	S	I	O	N	C	O
3	P	R	E	S	S	I	O	N	C	O	M
4	R	E	S	S	I	O	N	C	O	M	P
5	E	S	S	I	O	N	C	O	M	P	R
6	S	S	I	O	N	C	O	M	P	R	E
7	S	I	O	N	C	O	M	P	R	E	S
8	I	O	N	C	O	M	P	R	E	S	S
9	O	N	C	O	M	P	R	E	S	S	I
10	N	C	O	M	P	R	E	S	S	I	O

2. Now sort these sequences in lexicographic dictionary order :

0	C	O	M	P	R	E	S	S	I	O	N
1	E	S	S	I	O	N	C	O	M	P	R
2	I	O	N	C	O	M	P	R	E	S	S
3	M	P	R	E	S	S	I	O	N	C	O
4	N	C	O	M	P	R	E	S	S	I	O
5	O	M	P	R	E	S	S	I	O	N	C
6	O	N	C	O	M	P	R	E	S	S	I
7	P	R	E	S	S	I	O	N	C	O	M
8	R	E	S	S	I	O	N	C	O	M	P
9	S	I	O	N	C	O	M	P	R	E	S
10	S	S	I	O	N	C	O	M	P	R	E

3. The sequences is the last letter of each sequence.

4. So, L = N R S O O C I M P S E

and original sequence appear at sequence number 0 in the sorted list, so the encoded sequence consist of the sequence L and index value 10.

Que 3.32. Explain move-to-front coding and encode the sequence s s h t t h ␣ i e, assuming source alphabet is given by A = { ␣ , e, h, i, s, t}.

Answer

1. Move-to-front coding scheme take advantage of long runs of identical symbols.

2. Move-to-front coding starts with initial listing of the source alphabet.

3. The symbol in the list are assigned numbers from 0 to *n i.e.*, the symbol on the top of list is assign 0, next symbol is assigned 1 and so on.

4. The first time the particular symbol occurs, the number corresponding to the symbol in the list is transmitted and is move on top of list.

5. If we have a run of this symbol, we transmit a sequence of 0's. So, the long sequence symbol gets transformed to 0's.

Numerical :

L = ss*htth*␣*ie* and source alphabet

A = { ␣ , *e*, *h*, *i*, s, t}

1. We start with

0	1	2	3	4	5
␣	*e*	*h*	*i*	*s*	*t*

2. First symbol is *s* which is encoded as 4 and move *s* at the top of list.

0	1	2	3	4	5
s	␣	*e*	*h*	*i*	*t*

3. Next symbol *s* is encoded as 0, next symbol is *h* encoded as 3 and *h* is moved on top.

0	1	2	3	4	5
h	*s*	␣	*e*	*i*	*t*

4. Next symbol is *t* encoded as 5 and *t* moved on top.

0	1	2	3	4	5
t	*h*	*s*	␣	*e*	*i*

5. Next symbol is t encoded as 0.
   Next symbol is h encoded as 1 and moved on top.

0	1	2	3	4	5
h	t	s	$\not b$	e	i

6. Next symbol is $\not b$ encoded as 3 and moved on top.

0	1	2	3	4	5
$\not b$	h	t	s	e	i

7. Next symbol is i encoded as 5 and moved on top.

0	1	2	3	4	5
i	$\not b$	h	t	s	e

8. Next symbol is e encoded as 5 and moved on top.

0	1	2	3	4	5
e	i	$\not b$	h	t	s

9. So, the sequence of encoded symbols are :
   4 0 3 5 0 1 3 5 5

Que 3.33. Explain the idea of CALIC. Also, define the term recursive indexing.

Answer

1. The Context Adaptive Lossless Image Compression (CALIC) scheme, which came into being in response to a call for proposal for a new lossless image compression scheme uses both context and prediction of the pixel values.
2. The CALIC scheme actually functions in two modes, one for gray-scale images and another for bi-level images.
3. In an image, a given pixel generally has a value close to one of its neighbours *i.e.*, which neighbour has the closest value depends on the local structure of the image.
4. Depending on whether there is a horizontal or vertical edge in the neighbourhood of the pixel being encoded, the pixel above, or the pixel to the left, or some weighted average of neighbouring pixels may give the best prediction.
5. How close the prediction is to the pixel being encoded depends on the surrounding texture.
6. In a region of the image with a great deal of variability, the prediction is likely to be further from the pixel being encoded than in the regions with less variability.

7. In order to take into account all these factors, the algorithm has to make a determination of the environment of the pixel to be encoded. The only information that can be used to make this determination has to be available to both encoder and decoder.

		NN	NNE
	NW	N	NE
WW	W	X	

Fig. 3.33.1. Labeling the neighbours of pixel X.

8. We can get an idea of what kinds of boundaries may or may not be in the neighbourhood of X by computing,

$$d_h = |W - WW| + |N - NW| + |NE - N|$$
$$d_v = |W - NW| + |N - NN| + |NE - NNE|$$

9. The relative values of d_h and d_v are used to obtain the initial prediction of the pixel X.
10. This initial prediction is then refined by taking other factors into account.
11. If the value of d_h is much higher than the value of d_v, *i.e.*, there is a large amount of horizontal variation, and it would be better to pick N to be the initial prediction.
12. If, on the other hand, d_v is much larger than d_h, this would mean that there is a large amount of vertical variation, and the initial prediction is taken to be W. If the differences are more moderate or smaller, the predicted value is a weighted average of the neighbouring pixels.
13. The exact algorithm used by CALIC to form the initial prediction is given by the following pseudocode :

```
if d_h − d_v > 80
    X̂ ← N
else if d_v − d_h > 80
    X̂ ← W
else
{
    X̂ ← (N + W)/2 + (NE − NW)/4
    if d_h − d_v > 32
    X̂ ← (X̂ + N)/2
    else if d_v − d_h > 32
```

$\hat{X} \leftarrow (\hat{X} + W)/2$
else if $d_h - d_v > 8$
$\hat{X} \leftarrow (3\hat{X} + N)/4$
else if $d_v - d_h > 8$
$\hat{X} \leftarrow (3\hat{X} + W)/4$
}

14. Another approach used by CALIC to reduce the size of its alphabet is to use a modification of a technique called recursive indexing.
15. Recursive indexing is a technique for representing a large range of numbers using only a small set.
16. Suppose we want to represent positive integers using only the integers between 0 and 7, *i.e.*, a representation alphabet of size 8.
17. Recursive indexing works as follows : If the number to be represented lies between 0 and 6, we simply represent it by that number. If the number to be represented is greater than or equal to 7, we first send the number 7, subtract 7 from the original number, and repeat the process.
19. We keep repeating the process until the remainder is a number between 0 and 6.

We can summarize the CALIC algorithm as follows :

1. Find initial prediction $\hat{X}$.
2. Compute prediction context.
3. Refine prediction by removing the estimate of the bias in that context.
4. Update bias estimate.
5. Obtain the residual and remap it so the residual values lie between 0 and M – 1, where M is the size of the initial alphabet.
6. Find the coding context k.
7. Code the residual using the coding context.

Que 3.34. Write a short note on JPEG-LS standard.

Answer

1. The JPEG-LS standard looks more like CALIC than the old JPEG standard.
2. The initial prediction is obtained using the following algorithm :

if $NW \geq \max(W, N)$
$\hat{X} = \max(W, N)$
else
{
if NW ≤ min(W, N)

$\hat{X} = \min(W, N)$
else
$\hat{X} = W + N - NW$
}

3. This prediction approach is a variation of median adaptive prediction, in which the predicted value is the median of the N, W, and NW pixels.
4. The initial prediction is then refined using the average value of the prediction error in that particular context.
5. The contexts in JPEG-LS also reflect the local variations in pixel values.
6. However, they are computed differently from CALIC.
7. First, measures of differences D_1 D_2, and D_3 are computed as follows :

$$D_1 = NE - N$$
$$D_2 = N - NW$$
$$D_3 = NW - W$$

The values of these differences define a three-component context vector Q.

8. The components of Q (Q_1, Q_2, and Q_3) are defined by the following mappings :

$$D_i \leq -T_3 \Rightarrow Q_i = -4$$
$$-T_3 < D_i \leq -T_2 \Rightarrow Q_i = -3$$
$$-T_2 < D_i \leq -T_1 \Rightarrow Q_i = -2$$
$$-T_1 < D_i \leq 0 \Rightarrow Q_i = -1$$
$$D_i = 0 \Rightarrow Q_i = 0$$
$$0 < D_i \leq T_1 \Rightarrow Q_i = 1$$
$$T_1 < D_i \leq T_2 \Rightarrow Q_i = 2$$
$$T_2 < D_i \leq T_3 \Rightarrow Q_i = 3$$
$$T_3 < D_i \Rightarrow Q_i = 4$$

where T_1, T_2, and T_3 are positive coefficients that can be defined by the user.

9. Given nine possible values for each component of the context vector, this results in $9 \times 9 \times 9 = 729$ possible contexts.
10. In order to simplify the coding process, the number of contexts is reduced by replacing any context vector Q whose first nonzero element is negative by $-Q$.
11. Whenever this happens, a variable *SIGN* is also set to –1; otherwise, it is set to + 1.
12. This reduces the number of contexts to 365.
13. The vector Q is then mapped into a number between 0 and 364.
14. The variable *SIGN* is used in the prediction refinement step.
15. The correction is first multiplied by *SIGN* and then added to the initial prediction.

16. The prediction error r_n is mapped into an interval, that is, the same size as the range occupied by the original pixel values.
17. The mapping used in JPEG-LS is as follows :

$$r_n < -\frac{M}{2} \Rightarrow r_n \leftarrow r_n + M$$

$$r_n > \frac{M}{2} \Rightarrow r_n \leftarrow r_n - M$$

Que 3.35. What do you understand by predictive coding ? Discuss multiresolution approach. **UPTU 2013-14, Marks 10**

Answer

Predictive coding : Refer Q. 3.25, Page 90D, Unit-3.

Multiresolution approach :

1. Our final predictive image compression scheme is perhaps not as competitive as the other schemes. However, it is an interesting algorithm because it approaches the problem from a slightly different point of view.
2. Multiresolution models generate representations of an image with varying spatial resolution.
3. This usually results in a pyramid like representation of the image, with each layer of the pyramid serving as a prediction model for the layer immediately below.
4. One of the more popular of these techniques is known as HINT (Hierarchical INTerpolation).
5. The specific steps involved in HINT are as follows :
   a. First, residuals corresponding to the pixels labeled Δ in Fig. 3.35.1 are obtained using linear prediction and transmitted.

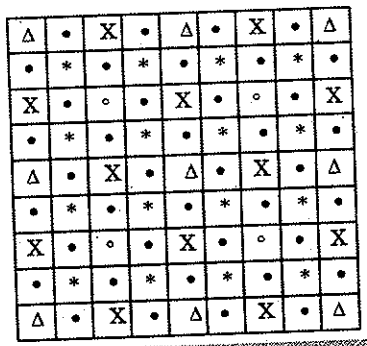


Fig. 3.35.1. The HINT scheme for hierarchical prediction.

   b. Then, the intermediate pixels (○) are estimated by linear interpolation, and the error in estimation is then transmitted.
   c. Then, the pixels X are estimated from Δ and ○, and the estimation error is transmitted.
   d. Finally, the pixels labeled * and then • are estimated from known neighbours, and the errors are transmitted. The reconstruction process proceeds in a similar manner.
6. One use of a multiresolution approach is in progressive image transmission.

Que 3.36. What is facsimile encoding ? Explain run length coding technique used earlier for facsimile.

UPTU 2011-12, 2013-14, 2014-15; Marks 05

Answer

1. Facsimile (Fon) encoding is one of the oldest application of lossless compression.
2. In facsimile transmission, a page is scanned and converted into a sequence of black or white pixel and send over the communication channel.
3. The speed of a facsimile transmission has changes in recent years. CCITT (now ITU-T) has issued number of recommendations based on the speed requirements at a given time.
4. The CCITT classifies the apparatus facsimile transmission into four groups for A4 size document :
   a. **Group 1 :** This apparatus is capable of transmitting an A4 size document in about six minutes over phone lines using an analog approach. This is standardized in recommendation T.2.
   b. **Group 2 :** This apparatus is capable of transmitting an A4 size document over phone in three minute, this also uses analog approach and, therefore, does not use data compression. The apparatus is standardized in recommendation T.3.
   c. **Group 3 :** This apparatus uses a digitized binary representation of the facsimile, A4 size data compression and is capable of transmitting because it is a digital scheme, it use document in about a minute. Standardized in recommendation T.4.
   d. **Group 4 :** This apparatus has the same speed as Group 3. The apparatus is standardized in recommendation T.6, T.503, T.521, and T.563.

Run length coding :

1. The model give rise to run length coding is the Capon-model.

2. A two-state Markov model with states S_w and S_b : S_w is a pixel encoded with white and S_b is a pixel encoded with black.
3. The transition probabilities $P(w/b)$ and $P(b/w)$, and the probability of being in each state.
4. $P(S_w)$ and $P(S_b)$, completely satisfy the model.
5. Generally, $P(w/w)$ and $P(w/b)$ are higher than $P(b/w)$ and $P(b/b)$.

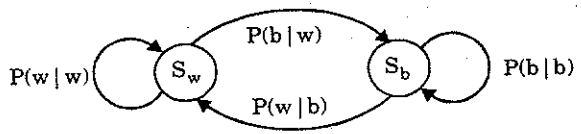


Fig. 3.36.1. Capon model for binary image.

6. The highly skewed nature of the probabilities $P(b/w)$ and $P(w/w)$, and a lesser extent $P(w/b)$ and $P(b/b)$, says that once a pixel takes on a particular color (black or white), it is highly likely that the following pixel will also be of same colours so we code the length of the run each colour rather than code the colour of each pixel separately.
7. Coding the length of runs instead of coding individual value is called the run length coding.

Que 3.37. What is facsimile encoding ? Explain run length coding technique used earlier for facsimile. Give a brief comparison of MH, MR, MMR and JBIG. **UPTU 2012-13, Marks 10**

Answer

Facsimile encoding : Refer Q. 3.36, Page 103D, Unit-3.
Run length coding : Refer Q. 3.36, Page 103D, Unit-3.
Comparison of MH, MR, MMR and JBIG : Refer Q. 3.9, Page 70D, Unit-3.

Que 3.38. Write short notes on the following :

i. Dynamic Markov Compression

ii. Graphic Interchange Format **UPTU 2011-12, Marks 05**

Answer

i. Dynamic Markov Compression :

a. Dynamic Markov compression (DMC), introduced by Cormack and Horspool uses a more general framework to take advantage of relationships and correlations, or contexts, that extend beyond a single symbol.
b. Consider the sequence of pixels in a scanned document.
c. The sequence consists of runs of black and white pixels. If we represent black by 0 and white by 1, we have runs of 0s and 1s.

d. If the current value is 0, the probability that the next value is 0 is higher than if the current value was 1.
e. The fact that we have two different sets of probabilities is reflected in the two-state model shown in Fig. 3.38.1.

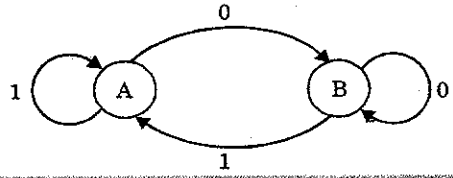


Fig. 3.38.1. A two-state model for binary sequences.

f. Consider state A. The probability of the next value being 1 changes depending on whether we reached state A from state B or from state A itself.
g. We can have the model reflect this by cloning state A, as shown in Fig. 3.38.2, to create state A.

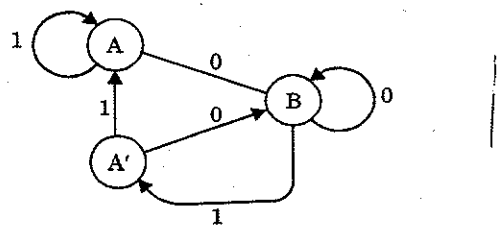


Fig. 3.38.2 A three-state model obtained by cloning.

h. Now if we see a white pixel after a run of black pixels, we go to state A.
i. The probability that the next value will be 1 is very high in this state.
j. This way when we estimate probabilities for the next pixel value, we take into account not only the value of the current pixel but also the value of the previous pixel.
k. There are a number of issues that need to be addressed in order to implement this algorithm :
   i. What is the initial number of states ?
   ii. How do we estimate probabilities ?
   iii. How do we decide when a state needs to be cloned ?
   iv. What do we do when the number of states becomes too large ?

ii. Graphic Interchange Format : Refer Q. 3.23, Page 88D, Unit-3.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Define arithmetic coding. Write its advantages and disadvantages.

Ans: Refer Q.3.1 and Q.3.5.

Q. 2. Compare Huffman and arithmetic coding.

Ans: Refer Q.3.6.

Q. 3. Write about JBIG and JBIG2 standard of bi-level image compression.

Ans: Refer Q.3.8 and 3.10.

Q. 4. Discuss about the concept of static dictionary and adaptive dictionary.

Ans: Refer Q.3.12 and Q. 3.14.

Q. 5. Write short note on :

i. Escape symbol
ii. Length of context
iii. Exclusion symbol
iv. Facsimile encoding

Ans:

i. Refer Q. 3.28.
ii. Refer Q. 3.28.
iii. Refer Q. 3.29.
iv. Refer Q. 3.36.

4 UNIT

Mathematical Preliminaries for Lossy Coding

PART-1

Distortion Criteria, Models, Scalar Quantization : The Quantization Problem.

CONCEPT OUTLINE : PART-1

- Rate distortion theory gives theoretical bounds for how much compression can be achieved using lossy compression.
- The two measure of distortion are squared error measure and the absolute difference measure.
- Lossy compression use Uniform, Gaussian, Laplacian and Gamma distribution as probability model.
- Quantization is the process of mapping a continuous range of values by a relatively small, finite set of discrete symbol or integer value.
- Scalar quantization is a mapping of an input value x into a number of output values by :

$$Q : x \rightarrow y$$

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.1. What is lossy data encoding ? Write down the distortion measure criteria's to check the fidelity of a reconstructed source sequence to the original one in such type of encoding techniques.

UPTU 2011-12, Marks 10

Answer

Lossy data encoding :

1. The limited amount of compression, available from using lossless compression scheme may be acceptable in either case – that is, resources are limited and we do not require absolute integrity or we can improve the amount of compression by accepting a certain degree of loss during the compression process.
2. Performance measure is necessary to determine the efficiency of lossy compression scheme.
3. For lossless compression only the rate of performance measure consider which is not feasible for lossy compression.

4. Thus, we need some additional performance measure, such as some measure of difference between the original and reconstructed data which is refer as distortion in the reconstructed data.
5. The rate for a discrete source is simply the entropy.
6. The study of this situation between these two extremes is called rate distortion theory.

Distortion criteria :

1. A natural thing to do when looking at the fidelity of a reconstructed sequence is to look at the differences between the original and reconstructed values. In other words, the distortion introduced in the compression process.
2. Two popular measures of distortion or difference between the original and reconstructed sequences are the squared error measure and the absolute difference measure.
3. These are called difference distortion measures.
4. If x_n is the source output and y_n is the reconstructed sequence, then the squared error measure is given by,

$$d(x, y) = (x - y)^2 \quad \ldots(4.1.1)$$

and the absolute difference measure is given by,

$$d(x, y) = |x - y| \quad \ldots(4.1.2)$$

5. In general, it is difficult to examine the difference on a term-by-term basis.
6. Therefore, a number of average measures are used to summarize the information in the difference sequence.
7. The most often used average measure is the average of the squared error measure.
8. This is called the mean squared error (mse) and is often represented by the symbol σ^2 or σ^2_d :

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2 \quad \ldots(4.1.3)$$

9. If we are interested in the size of the error relative to the signal, we can find the ratio of the average squared value of the source output and the mse.
10. This is called the signal-to-noise ratio (SNR).

$$\text{SNR} = \frac{\sigma_x^2}{\sigma_d^2} \quad \ldots(4.1.4)$$

where σ^2_x is the average squared value of the source output, or signal, and σ^2_d is the mse.

11. The SNR is often measured on a logarithmic scale and the units of measurement are decibels (abbreviated to dB).

$$SNR(dB) = 10\log_{10}\frac{\sigma_x^2}{\sigma_d^2} \quad ...(4.1.5)$$

12. Sometimes, we are more interested in the size of the error relative to the peak value of the signal x_{peak} than with the size of the error relative to the average squared value of the signal.
13. This ratio is called the peak-signal-to-noise-ratio (PSNR) and is given by,

$$PSNR(dB) = 10\log_{10}\frac{x_{peak}^2}{\sigma_d^2} \quad ...(4.1.6)$$

14. Another difference distortion measure that is used quite often, although not as often as the mse, is the average of the absolute difference, or

$$d_1 = \frac{1}{N}\sum_{n=1}^{N}|x_n - y_n| \quad ...(4.1.7)$$

15. This measure seems especially useful for evaluating image compression algorithms.
16. In some applications, the distortion is not perceptible as long as it is below some threshold.
17. In these situations, we might be interested in the maximum value of the error magnitude.

$$d_\infty = \max_n |x_n - y_n| \quad ...(4.1.8)$$

18. We have looked at two approaches to measuring the fidelity of a reconstruction.
19. The first method involving humans may provide a very accurate measure of perceptible fidelity, but it is not practical and not useful in mathematical design approaches.
20. The second is mathematically tractable, but it usually does not provide a very accurate indication of the perceptible fidelity of the reconstruction.
21. A middle ground is to find a mathematical model for human perception, transform both the source output and the reconstruction to this perceptual space, and then measure the difference in the perceptual space.

Que 4.2. Explain in brief the block diagram of a generic compression scheme. What is the distortion criteria to measure the closeness of a reconstructed source sequence to the original ?

UPTU 2013-14, Marks 10

Answer

1. The generic compression scheme has a source, a source encoder, a communication channel, and a source decoder.

2. The output generated by the source is modeled as a random variable X.
3. The source encoder take X as an input and produces a compressed representation of X, we call this compressed representation as X_c.
4. The communication channel send it to source decoder in the form of $\hat{X}$ and $X_c = \hat{X}$. The source decoder takes the compressed representation and produces a reconstruction of the source output for the end user.
5. The process can be viewed in a block diagram as shown in Fig. 4.2.1.

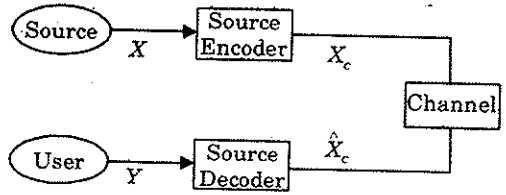


Fig. 4.2.1. Block diagram of generic compression scheme.

Distortion criteria : Refer Q. 4.1, Page 108D, Unit-4.

Que 4.3. What is conditional entropy and mutual information and average mutual information ? For two random variables X and Y show that :

i. $H(X/Y) \le H(X)$

ii. $I(X:Y) = I(Y:X)$

UPTU 2013-14, Marks 10

Answer

Conditional entropy :

1. In the case of self-information, we are generally interested in the average value of the conditional self-information.
2. This average value is called the conditional entropy.
3. The conditional entropies of the source and reconstruction alphabets are given as,

$$H(X/Y) = -\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}P(x_i|y_j)P(y_j)\log_2 P(x_i|y_j) \quad ...(4.3.1)$$

and

$$H(Y/X) = -\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}P(x_i|y_j)P(y_j)\log_2 P(y_j|x_i) \quad ...(4.3.2)$$

4. The conditional entropy $H(X|Y)$ can be interpreted as the amount of uncertainty remaining about the random variable X, or the source output, given that we know what value the reconstruction Y took.
5. The additional knowledge of Y should reduce the uncertainty about X, and we can show that,

$$H(X|Y) \le H(X) \quad ...(4.3.3)$$

Mutual and average mutual information :

1. We make use of one more quantity that relates the uncertainty or entropy of two random variables.
2. This quantity is called the mutual information and is defined as,

$$i(x_k ; y_j) = \log\left[\frac{P(x_k | y_j)}{P(x_k)}\right] \quad ...(4.3.4)$$

3. We will use the average value of this quantity, appropriately called the average mutual information, which is given by,

$$I(X ; Y) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(x_i, y_j)\log\left[\frac{P(x_i | y_j)}{P(x_i)}\right] \quad ...(4.3.5)$$

$$= \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(x_i | y_j)P(y_j)\log\left[\frac{P(x_i | y_j)}{P(x_i)}\right] \quad ...(4.3.6)$$

4. We can write the average mutual information in terms of the entropy and the conditional entropy by expanding the argument of the logarithm in equation (4.3.6).

$$I(X ; Y) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(x_i, y_j)\log\left[\frac{P(x_i | y_j)}{P(x_i)}\right] \quad ...(4.3.7)$$

$$= \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(x_i, y_j)\log P(x_i|y_j) - \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(x_i, y_j)\log P(x_i) \quad ...(4.3.8)$$

$$= H(X) - H(X|Y) \quad ...(4.3.9)$$

Derivation :

i. **Given :** Random variables X and Y.
To show : $H(X|Y) \le H(X)$ with equality if X is independent of Y.

$$\because \quad I(X; Y) = H(X) - H(X|Y) \quad ...(4.3.10)$$

and $I(X; Y) \ge 0$

$$\therefore \quad 0 \le I(X; Y) = H(X) - H(X|Y) \quad ...(4.3.11)$$

Equation (4.3.11) can also be elaborated as :

$$0 \le I(X; Y) = H(X) - \sum_y P(y)\, H(X|Y = y)$$

This theorem proves that knowing another random variable Y can only reduce the uncertainty in X. Specifically, $H(X|Y = y)$ may be greater than or less than or equal to $H(X)$, but on average,

$$H(X|Y) = \sum_y P(y)\, H(X \mid Y = y) \le H(X)$$

Hence proved.

ii. **Given :** Random variables X and Y.
To show : $I(X ; Y) = I(Y ; X)$.

As we know, $I(X ; Y) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(x_i, y_j)\log\left[\frac{P(x_i | y_j)}{P(x_i)}\right]$ ...(4.3.12)

and $P(x_i , y_j) = P(y_j, x_i)$ ...(4.3.13)

$\therefore$ Using Bayes' rule of conditional probability,

$$\frac{P(x_i | y_j)}{P(x_i)} = \frac{P(y_j | x_i)}{P(y_j)} \quad ...(4.3.14)$$

Putting the values of equation (4.3.13) and (4.3.14) in equation (4.3.12), we get :

$$I(X ; Y) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(y_j , x_i)\log\left[\frac{P(y_j | x_i)}{P(y_j)}\right]$$

$$I(X ; Y) = I(Y ; X)$$

where $I(Y ; X) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(y_j , x_i)\log\left[\frac{P(y_j | x_i)}{P(y_j)}\right]$

Hence proved.

Que 4.4. What is rate distortion criterion ? Explain the rate distortion function for binary source and Gaussian source.

UPTU 2014-15, Marks 10

Answer

Rate distortion theory :

1. Rate distortion theory is concerned with the trade-offs between distortion and rate in lossy compression schemes.
2. Rate is defined as the average number of bits used to represent each sample value.
3. One way of representing the trade-offs is via a rate distortion function $R(D)$.
4. The rate distortion function $R(D)$ specifies the lowest rate at which the output of a source can be encoded while keeping the distortion less than or equal to D.

Rate distortion function for :

a. Binary source :

1. Suppose we have a source alphabet {0, 1}, with $P(0) = p$. The reconstruction alphabet is also binary. Given the distortion measure,

$$d(x_i, y_j) = x_i \oplus y_j \quad ...(4.4.1)$$

where $\oplus$ is modulo 2 addition, let us find the rate distortion function.

2. Assume for the moment that $p < \frac{1}{2}$.

3. For $D > p$, an encoding scheme that would satisfy the distortion criterion would be not to transmit anything and fix $Y = 1$. So, for $D \geq p$,
$$R(D) = 0 \quad \ldots(4.4.2)$$
4. We will find the rate distortion function for the distortion range $0 \leq D < p$.
Step 1 : Find a lower bound for the average mutual information :
$$I(X; Y) = H(X) - H(X|Y) \quad \ldots(4.4.3)$$
$$= H(X) - H(X \oplus Y|Y) \quad \ldots(4.4.4)$$
$$\geq H(X) - H(X \oplus Y) \text{ from } H(X|Y) \leq H(X) \quad \ldots(4.4.5)$$
Step 2 : In the second step, we have used the fact that if we know Y, then knowing X, we can obtain $X \oplus Y = X$.
Let us look at the terms on the right-hand side of $H(X|Y) \leq H(X)$
$$H(X) = -p \log_2 p - (1-p) \log_2 (1-p) = H_b(p) \quad \ldots(4.4.6)$$
where $H_b(p)$ is called the binary entropy function and is plotted in Fig. 4.4.1. Note that $H_b(p) = H_b(1-p)$.
5. Given that $H(X)$ is completely specified by the source probabilities, our task now is to find the conditional probabilities $\{P(x_i|y_j)\}$ such that $H(X \oplus Y)$ is maximized while the average distortion $E[d(x_i, y_j)] \leq D$.
6. $H(X \oplus Y)$ is simply the binary entropy function $H_b(P(X \oplus Y = 1))$, where
$$P(X \oplus Y = 1) = P(X = 0, Y = 1) + P(X = 1, Y = 0) \quad \ldots(4.4.7)$$
7. Therefore, to maximize $H(X \oplus Y)$, we would want $P(X \oplus Y = 1)$ to be as close as possible to one-half.
8. However, the selection of $P(X \oplus Y)$ also has to satisfy the distortion constraint.
9. The distortion is given by,

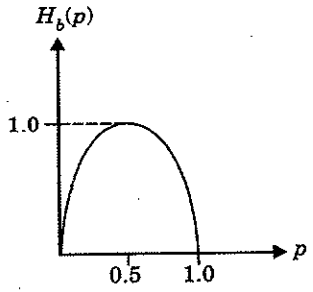


Fig. 4.4.1. The binary entropy function

$$E[d(x_i, y_j)] = 0 \times P(X = 0, Y = 0) + 1 \times P(X = 0, Y = 1) + 1 \times P(X = 1, Y = 0) + 0 \times P(X = 1, Y = 1)$$
$$= P(X = 0, Y = 1) + P(X = 1, Y = 0)$$

$$= P(Y = 1|X = 0)p + P(Y = 0|X = 1)(1-p) \quad \ldots(4.4.8)$$
10. But this is simply the probability that $X \oplus Y = 1$.
11. Therefore, the maximum value that $P(X \oplus Y = 1)$ can have is D.
12. Our assumptions were that $D < p$ and $p \leq \frac{1}{2}$, which means that $D < \frac{1}{2}$.
13. Therefore, $P(X \oplus Y = 1)$ is closest to $\frac{1}{2}$ while being less than or equal to D when $P(X \oplus Y = 1) = D$.
14. Therefore,
$$I(X; Y) \geq H_b(p) - H_b(D) \quad \ldots(4.4.9)$$
15. We can show that for $P(X = 0|Y = 1) = P(X = 1|Y = 0) = D$, this bound is achieved.
16. That is, if $P(X = 0|Y = 1) = P(X = 1|Y = 0) = D$, then
$$I(X; Y) = H_b(p) - H_b(D) \quad \ldots(4.4.10)$$
17. Therefore, for $D < p$ and $p \leq \frac{1}{2}$,
$$R(D) = H_b(p) - H_b(D) \quad \ldots(4.4.11)$$
18. Finally, if $p > \frac{1}{2}$, then we simply switch the roles of p and $1-p$.
19. Putting all this together, the rate distortion function for a binary source is,
$$R(D) = \begin{cases} H_b(p) - H_b(D) & \text{for } D < \min\{p, 1-p\} \\ 0 & \text{otherwise} \end{cases} \quad \ldots(4.4.12)$$

b. Gaussian source :

1. Suppose we have continuous amplitude source that has a zero mean Gaussian pdf with variance σ^2. Our distortion measure is given by,
$$d(x, y) = (x - y)^2 \quad \ldots(4.4.13)$$
2. Our distortion constraint is given by,
$$E[(X - Y)^2] \leq D \quad \ldots(4.4.14)$$
3. Our approach to finding the rate distortion function will be the same, that is, find a lower bound for $I(X; Y)$ given a distortion constraint, and then show that this lower bound can be achieved.
4. First we find the rate distortion function for $D < \sigma^2$.
$$I(X; Y) = h(X) - h(X|Y) \quad \ldots(4.4.15)$$
$$= h(X) - h(X - Y|Y) \quad \ldots(4.4.16)$$
$$\geq h(X) - h(X - Y) \quad \ldots(4.4.17)$$

5. In order to minimize the right-hand side of equation (4.4.17), we have to maximize the second term subject to the constraint given by equation (4.4.14).
6. This term is maximized if $X - Y$ is Gaussian, and the constraint can be satisfied if $E[(X-Y)^2] = D$.
7. Therefore, $h(X-Y)$ is the differential entropy of a Gaussian random variable with variance D, and the lower bound becomes :

$$I(X; Y) \geq \frac{1}{2}\log(2\pi e\sigma^2) - \frac{1}{2}\log(2\pi eD) \quad ...(4.4.18)$$

$$= \frac{1}{2}\log\frac{\sigma^2}{D} \quad ...(4.4.19)$$

8. This average mutual information can be achieved if Y is zero mean Gaussian with variance $\sigma^2 - D$, and

$$f_{X|Y}(x|y) = \frac{1}{\sqrt{2\pi D}}\exp\frac{-x^2}{2D} \quad ...(4.4.20)$$

9. For $D > \sigma^2$, if we set $Y = 0$, then

$$I(X ; Y) = 0 \quad ...(4.4.21)$$

and $$E[(X-Y)^2] = \sigma^2 < D \quad ...(4.4.22)$$

10. Therefore, the rate distortion function for the Gaussian source can be written as :

$$R(D) = \begin{cases} \frac{1}{2}\log\frac{\sigma^2}{D} & \text{for } D < \sigma^2 \\ 0 & \text{for } D > \sigma^2 \end{cases} \quad ...(4.4.23)$$

Que 4.5. Write a short note on probability models.

Answer

1. An important method for characterizing a particular source is through the use of probability models.
2. Probability models used for the design and analysis of lossy compression schemes differ from those used in the design and analysis of lossless compression schemes.
3. When developing models in the lossless case, we tried for an exact match.
4. The probability of each symbol was estimated as part of the modeling process.
5. When modeling sources in order to design or analyze lossy compression schemes, we look more to the general rather than exact correspondence.
6. Uniform, Gaussian, Laplacian, and Gamma distribution are four probability models commonly used in the design and analysis of lossy compression systems.

7. The shapes of these four distributions, assuming a mean of zero and a variance of one, are shown in Fig. 4.5.1.

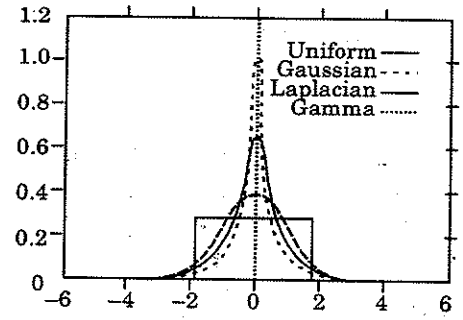


Fig. 4.5.1. Uniform, Gaussian, Laplacian, and Gamma distributions.

Types of probability models :

1. Uniform distribution :

a. As for lossless compression, this is again our ignorance model.

b. If we do not know anything about the distribution of the source output, except possibly the range of values, we can use the uniform distribution to model the source.

c. The probability density function for a random variable uniformly distributed between a and b is :

$$f_X(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \quad ...(4.5.1)$$

2. Gaussian distribution :

a. The Gaussian distribution is one of the most commonly used probability models for two reasons : it is mathematically tractable and, by virtue of the central limit theorem, it can be argued that in the limit, the distribution of interest goes to a Gaussian distribution.

b. The probability density function for a random variable with a Gaussian distribution and mean μ and variance σ^2 is :

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\frac{(x-\mu)^2}{2\sigma^2} \quad ...(4.5.2)$$

3. Laplacian distribution :

a. Many sources that we deal with have distributions that are quite peaked at zero.

b. For example, speech consists mainly of silence.

c. Therefore, samples of speech will be zero or close to zero with high probability.

d. Image pixels themselves do not have any attraction to small values.

e. However, there is a high degree of correlation among pixels.

f. Therefore, a large number of the pixel-to-pixel differences will have values close to zero.

g. In these situations, a Gaussian distribution is not a very close match to the data.

h. A closer match is the Laplacian distribution, which is peaked at zero.

i. The distribution function for a zero mean random variable with Laplacian distribution and variance σ^2 is :

$$f_X(x) = \frac{1}{\sqrt{2\sigma^2}} \exp \frac{-\sqrt{2}\,|x|}{\sigma} \qquad ...(4.5.3)$$

4. Gamma distribution :

a. A distribution that is even more peaked, though considerably less tractable than the Laplacian distribution is the Gamma distribution.

b. The distribution function for a Gamma distributed random variable with zero mean and variance σ^2 is given by :

$$f_X(x) = \frac{\sqrt[4]{3}}{\sqrt{8\pi\sigma\,|x|}} \exp \frac{-\sqrt{3}\,|x|}{2\sigma} \qquad ...(4.5.4)$$

Que 4.6. Explain linear system models.

Answer

1. A large class of processes can be modeled in the form of the following difference equation :

$$x_n = \sum_{i=1}^{N} a_i x_{n-i} + \sum_{j=1}^{M} b_j \in_{n-j} + \in_n \qquad ...(4.6.1)$$

where $\{x_n\}$ are samples of the process we wish to model, and $\{\in_n\}$ is a white noise sequence.

2. We assume that we are dealing with real valued samples.

3. A zero-mean wide-sense-stationary noise sequence $\{\in_n\}$ is a sequence with autocorrelation function,

$$R_{\in\in}(k) = \begin{cases} \sigma^2 & \text{for } k=0 \\ 0 & \text{otherwise} \end{cases} \qquad ...(4.6.2)$$

4. In digital signal processing terminology, equation (4.6.1) represents the output of a linear discrete time invariant filter with N poles and M zeroes.

5. In the statistical literature, this model is called an autoregressive moving average model of order (N, M), or an ARMA (N, M) model.

6. The autoregressive label is because of the first summation in equation (4.6.1), while the second summation gives us the moving average portion of the name.

7. If all the b_j were zero in equation (4.6.1), only the autoregressive part of the ARMA model would remain :

$$x_n = \sum_{i=1}^{N} a_i x_{n-i} + \in_n \qquad ...(4.6.3)$$

8. This model is called an N^{th} order autoregressive model and is denoted by AR(N).

9. In digital signal processing, this is an all pole filter. The AR(N) model is the most popular of all the linear models, especially in speech compression, where it arises as a natural consequence of the speech production model.

10. Notice that for the AR(N) process, knowing all the past history of the process gives no more information than knowing the last N samples of the process; that is,

$$P(x_n \mid x_{n-1}, x_{n-2}, \ldots) = P(x_n \mid x_{n-1}, x_{n-2}, \ldots, x_{n-N}) \qquad ...(4.6.4)$$

which means that the AR(N) process is a Markov model of order N.

11. The autocorrelation function of a process can tell us a lot about the sample-to-sample behaviour of a sequence.

12. A slowly decaying autocorrelation function indicates a high sample-to-sample correlation, while a fast decaying autocorrelation denotes low sample-to-sample correlation.

13. In the case of no sample-to-sample correlation, such as white noise, the autocorrelation function is zero for lags greater than zero, as seen in equation (4.6.2).

14. The autocorrelation function for the AR(N) process can be obtained as follows :

$$R_{xx}(k) = E[x_n x_{n-k}]$$

$$= E\left[\left(\sum_{i=1}^{N} a_i x_{n-i} + \in_n\right)(x_{n-k})\right] \qquad ...(4.6.5)$$

$$= E\left[\sum_{i=1}^{N} a_i x_{n-i}\, x_{n-k}\right] + E[\in_n x_{n-k}] \qquad ...(4.6.6)$$

$$= \begin{cases} \sum_{i=1}^{N} a_i R_{xx}(k-i) & \text{for } k>0 \\ \sum_{i=1}^{N} a_i R_{xx}(i) + \sigma_\in^2 & \text{for } k=0 \end{cases} \qquad ...(4.6.7)$$

Que 4.7. **What do you mean by quantization ? Describe the quantization problem with the help of an example in detail.**

UPTU 2011-12, 2014-15; Marks 10

Answer

Quantization :

1. In many lossy compression applications we are required to represent each source output using one of a small number of codewords.
2. The number of possible distinct source output values is generally much larger than the number of codewords available to represent them.
3. The process of representing a large–possibly infinite–set of values with a much smaller set is called quantization.
4. The set of inputs and outputs of a quantizer can be scalars or vectors.
5. If they are scalars, we call the quantizers scalar quantizers.
6. If they are vectors, we call the quantizers vector quantizers.

Quantization problem :

1. Quantization is a very simple process. However, the design of the quantizer has a significant impact on the amount of compression obtained and loss incurred in a lossy compression scheme.
2. The quantizer consists of two mappings : an encoder mapping and a decoder mapping.
3. The encoder divides the range of values that the source generates into a number of intervals.
4. Each interval is represented by a distinct codeword.
5. The encoder represents all the source outputs that fall into a particular interval by the codeword representing that interval.
6. As there could be many —possibly infinitely many—distinct sample values that can fall in any given interval, the encoder mapping is irreversible.
7. When the sample value comes from an analog source, the encoder is called an analog-to-digital (A/D) converter.
8. The encoder mapping for a quantizer with eight reconstruction values is shown in Fig. 4.7.1.
9. For this encoder, all samples with values between –1 and 0 would be assigned the code 011.
10. All values between 0 and 1.0 would be assigned the codeword 100, and so on.
11. On the two boundaries, all inputs with values greater than 3 would be assigned the code 111, and all inputs with values less than – 3.0 would be assigned the code 000.

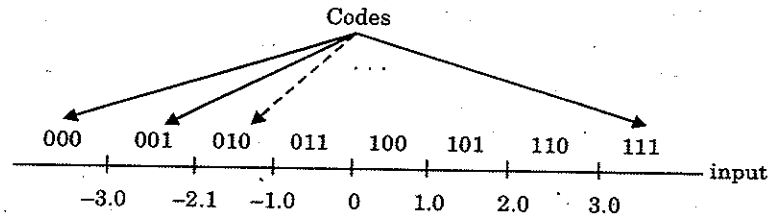


Fig. 4.7.1. Mapping of a 3-bit encoder.

Table 4.7.1. Mapping for a 3-bit D/A converter.

Input codes	Output
000	–3.5
001	–2.5
010	–1.5
011	–0.5
100	0.5
101	1.5
110	2.5
111	3.5

12. Thus, any input that we receive will be assigned a codeword depending on the interval in which it falls.
13. As we are using 3 bits to represent each value, we refer to this quantizer as a 3-bit quantizer.
14. For every codeword generated by the encoder, the decoder generates a reconstruction value.
15. Because a codeword represents an entire interval, and there is no way of knowing which value in the interval was actually generated by the source, the decoder puts out a value that, in some sense, best represents all the values in the interval.
16. Later, we will see how to use information we may have about the distribution of the input in the interval to obtain a representative value.
17. For now, we simply use the midpoint of the interval as the representative value generated by the decoder.
18. If the reconstruction is analog, the decoder is often referred to as a digital-to-analog (D/A) converter.
19. A decoder mapping corresponding to the 3-bit encoder shown in Fig. 4.7.1 is shown in Table 4.7.1.
20. Let us pose the design problem in precise terms.

21. Suppose we have an input modeled by a random variable X with pdf $f_X(x)$.
22. If we wished to quantize this source using a quantizer with M intervals, we would have to specify $M + 1$ endpoints for the intervals and a representative value for each of the M intervals.
23. The endpoints of the intervals are known as decision boundaries, while the representative values are called reconstruction levels.
24. Let us denote the decision boundaries by $\{b_i\}_{i=0}^{M}$, and the quantization operation by $Q(\cdot)$. Then,

$$Q(x) = y_i \quad \text{iff} \quad b_{i-1} < x \le b_i \qquad ...(4.7.1)$$

The mean squared quantization error is then given by,

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x - Q(x))^2 f_X(x) dx \qquad ...(4.7.2)$$

$$= \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_X(x) dx \qquad ...(4.7.3)$$

25. The difference between the quantizer input x and output $y = Q(x)$, besides being referred to as the quantization error, is also called the quantizer distortion or quantization noise.
26. But the word "noise" is somewhat of a misnomer.
27. Generally, when we talk about noise we mean a process external to the source process.
28. Because of the manner in which the quantization error is generated, it is dependent on the source process and, therefore, cannot be regarded as external to the source process.
29. One reason for the use of the word "noise" in this context is that from time to time we will find it useful to model the quantization process as an additive noise process as shown in Fig. 4.7.2.

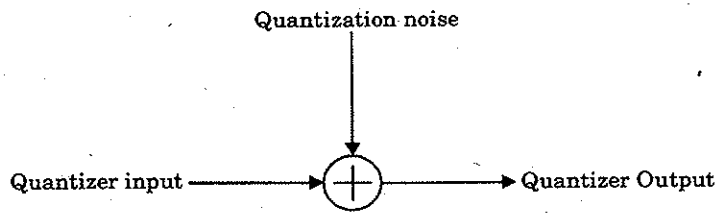


Fig. 4.7.2. Additive noise model of a quantizer.

30. If we use fixed length codewords to represent the quantizer output, then the size of the output alphabet immediately specifies the rate.
31. If the number of quantizer outputs is M, then the rate is given by,

$$R = \lceil \log_2 M \rceil \qquad ...(4.7.4)$$

32. For example, if $M = 8$, then $R = 3$. In this case, we can pose the quantizer design problem as follows :

a. Given an input pdf $f_X(x)$ and the number of levels M in the quantizer, find the decision boundaries $\{b_i\}$ and the reconstruction levels $\{y_i\}$ so as to minimize the mean squared quantization error given by equation (4.7.3).

b. However, if we are allowed to use variable-length codes, such as Huffman codes or arithmetic codes, along with the size of the alphabet, the selection of the decision boundaries will also affect the rate of the quantizer.

c. Consider the codeword assignment for the output of an eight-level quantizer shown in Table 4.7.2.

Table 4.7.2. Codeword assignment for an eight-level quantizer.

y_1	1110
y_2	1100
y_3	100
y_4	00
y_5	10
y_6	101
y_7	1101
y_8	1111

d. According to this codeword assignment, if the output y_4 occurs, we use 2 bits to encode it, while if the output y_1 occurs, we need 4 bits to encode it.

e. Obviously, the rate will depend on how often we have to encode y_4 versus how often we have to encode y_1.

f. In other words, the rate will depend on the probability of occurrence of the outputs.

g. If l_i is the length of the codeword corresponding to the output y_i, and $P(y_i)$ is the probability of occurrence of y_i, then the rate is given by :

$$R = \sum_{i=1}^{M} l_i P(y_i) \qquad ...(4.7.5)$$

h. However, the probabilities $\{P(y_i)\}$ depend on the decision boundaries $\{b_i\}$. For example, the probability of y_i occurring is given by :

$$P(y_i) = \int_{b_{i-1}}^{b_i} f_X(x) dx$$

i. Therefore, the rate R is a function of the decision boundaries and is given by the expression :

$$R = \sum_{i=1}^{M} l_i \int_{b_{i-1}}^{b_i} f_X(x)dx \qquad ...(4.7.6)$$

33. From this discussion and Eq. (4.7.3) and (4.7.6), we see that for a given source input, the partitions we select and the representation for those partitions will determine the distortion incurred during the quantization process.
34. The partitions we select and the binary codes for the partitions will determine the rate for the quantizer.
35. Thus, the problem of finding the optimum partitions, codes, and representation levels are all linked.
36. In light of this information, we can restate our problem statement :
    a. Given a distortion constraint,
    $$\sigma_q^2 \le D^* \qquad ...(4.7.7)$$
    b. Find the decision boundaries, reconstruction levels, and binary codes that minimize the rate given by Equation (4.7.6), while satisfying Equation (4.7.7).
    c. Or, given a rate constraint as :
    $$R \le R^* \qquad ...(4.7.8)$$
    d. Find the decision boundaries, reconstruction levels, and binary codes that minimize the distortion given by Equation (4.7.3), while satisfying Equation (4.7.8).

PART-2

Uniform Quantizer, Adaptive Quantization and Non-Uniform Quantization.

CONCEPT OUTLINE : PART-2

- Uniform quantizer is a quantizer if all intervals are of same size except possibly for the two outer intervals.
- Two types of quantizer are :
  i. Midrise quantizer
  ii. Midtread quantizer
- Adaptive quantization attempts to make the quantizer design adapt to the varying input statistics in order to achieve better performance.
- Two types of adaptive quantization are :
  i. Forward adaption
  ii. Backward adaption
- A non-uniform quantizer is a quantizer if source is not uniformly distributed.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 4.8. What do you understand by uniform quantizer ? How uniform quantization of a uniformly distributed source and uniform quantization of non-uniform source is done ?

UPTU 2012-13, 2013-14; Marks 10

Answer

A quantizer is a uniform quantizer if :

a. All intervals are of the same size except possibly for the two outer intervals (decision boundaries are spaced evenly).
b. Reconstruction values are also spaced evenly, with the same spacing as the decision boundaries.

In the inner intervals, the reconstruction values are the midpoint of the intervals. The constant spacing is usually referred to as the step size and is denoted by Δ.

Fig. 4.8.1 is a uniform quantizer with $\Delta = 1$.

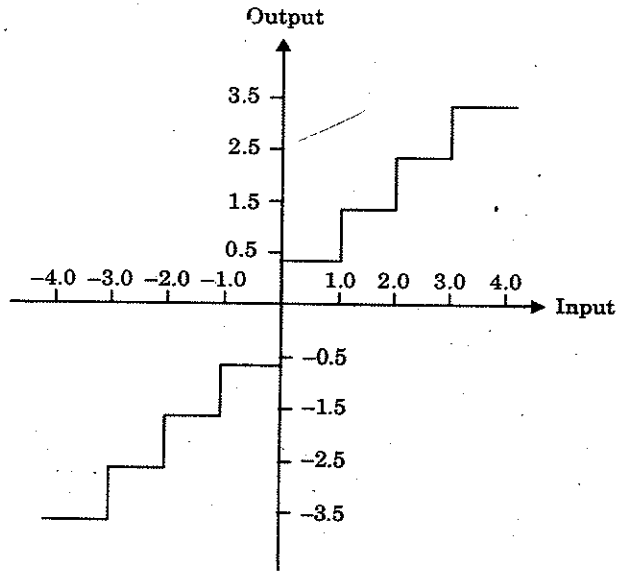


Fig. 4.8.1. Quantizer input-output map.

Two types of quantizer can be consider :

1. **Midrise quantizer :** If zero is not the representation level of the quantizer then it is called the midrise quantizer (M is even). It is shown in Fig. 4.8.2.

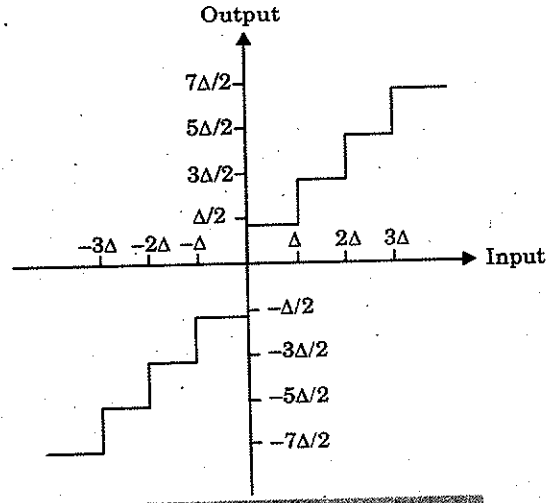


Fig. 4.8.2. A uniform midrise quantizer.

2. **Midtread quantizer :** If zero is the representation level of the quantizer *i.e.*, $\Delta = 0$ then it is called the midtread quantizer (M is odd). It is shown is Fig. 4.8.3.

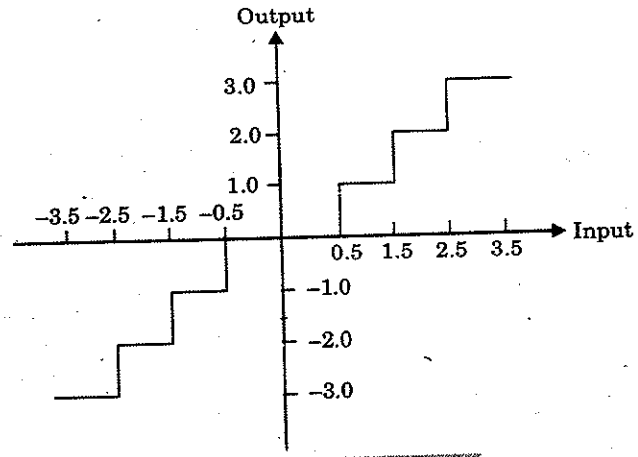


Fig. 4.8.3. A midtread quantizer.

Uniform quantization of a uniformly distributed source :

1. In uniformly distributed source, input X is uniformly distributed in $[-X_{max}, X_{max}]$.
2. For an M-level uniform quantizer, we divide $[-X_{max}, X_{max}]$ interval into M equally sized intervals each with step size Δ is given by,

$$\Delta = \frac{2X_{max}}{M}$$

3. The distortion in this case becomes :

$$\sigma_q^2 = 2\sum_{i=1}^{M/2}\int_{(i-1)\Delta}^{i\Delta}\left(x - \frac{2i-1}{2}\Delta\right)^2 \frac{1}{2X_{max}}dx$$

4. For a uniform distribution, quantization error $q = x - Q(x)$ uniformly distributed in the interval of $\left[-\Delta/2, \Delta/2\right]$.
5. The mean squared quantization error is given by,

$$\sigma_q^2 = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} q^2\,dq$$

$$= \frac{1}{\Delta}\left(\frac{1}{3}q^3\right)\Big|_{q=-\Delta/2}^{q=\Delta/2}$$

$$= \frac{1}{3\Delta}\left[\left(\frac{\Delta}{2}\right)^3 - \left(-\frac{\Delta}{2}\right)^3\right] = \frac{1}{3\Delta}\left[\frac{\Delta^3}{4}\right]$$

$$= \frac{\Delta^2}{12}$$

6. Let us also calculate the signal-to-noise ratio for this case.
7. The signal variance σ_s^2 for a uniform random variable, which takes on values in the interval $[-X_{max}, X_{max}]$, is $\frac{(2X_{max})^2}{12}$.
8. The value of the step size Δ is related to X_{max} and the number of levels M by,

$$\Delta = \frac{2X_{max}}{M}$$

9. Using a fixed length code with each code used made up of n bits, the number of codeword or level M is 2^n.
10. Combining all this we have :

$$\text{SNR (dB)} = 10\log_{10}\left(\frac{\sigma_s^2}{\sigma_q^2}\right)$$

$$= 10\log_{10}\left(\frac{(2X_{max})^2}{12}\cdot\frac{12}{\Delta^2}\right)$$

$$= 10\log_{10}\left(\frac{(2X_{max})^2}{12}\cdot\frac{12}{(2X_{max}/M)^2}\right)$$

$$= 10\log_{10} M^2$$
$$= 20\log_{10} 2^n$$
$$= 6.02n \text{ dB}$$

11. That is, for every additional bit in the quantizer, we get an increase in the signal-to-noise ratio of 6.02 dB.

Uniform quantization of non-uniform source :

1. When the distribution is no longer uniform, it is not a good idea to obtain the step size by simply dividing the range of the input by the number of levels.
2. This approach becomes totally impractical when we model our sources with distributions that are unbounded, such as the Gaussian distribution.
3. Therefore, we include the pdf of the source in the design process.
4. Our objective is to find the step size that, for a given value of M, will minimize the distortion.
5. The simplest way to do this is to write the distortion as a function of the step size, and then minimize this function.
6. An expression for the distortion, or msqe, for an M-level uniform quantizer as a function of the step size can be found by replacing the b_is and y_is in equation (4.7.3) with functions of Δ.
7. As we are dealing with a symmetric condition, we only need to compute the distortion for positive values of x; the distortion for negative values of x will be the same.
8. We see that the decision boundaries are integral multiples of Δ, and the representation level for the interval $[(k-1)\Delta, k\Delta)$ is simply $\frac{2k-1}{2}\Delta$.

   Therefore, the expression for msqe becomes :

$$\sigma_q^2 = 2\sum_{i=1}^{\frac{M}{2}-1}\int_{(i-1)\Delta}^{i\Delta}\left(x-\frac{2i-1}{2}\Delta\right)^2 f_X(x)dx$$

[Granular error]

$$+ 2\int_{\left(\frac{M}{2}-1\right)\Delta}^{\infty}\left(x-\frac{M-1}{2}\Delta\right)^2 f_X(x)\,dx$$

[Overload error]

9. To find the optimal value of Δ,

$$\frac{\delta\sigma_q^2}{\delta\Delta} = -\sum_{i=1}^{\frac{M}{2}-1}(2i-1)\int_{(i-1)\Delta}^{i\Delta}\left(x-\frac{2i-1}{2}\Delta\right)f_X(x)\,dx$$

$$-(M-1)\int_{\left(\frac{M}{2}-1\right)\Delta}^{\infty}\left(x-\frac{M-1}{2}\Delta\right)f_X(x)dx = 0$$

10. When Δ increases, size of overload region decreases and overload error decreases. When Δ increases, granular noise increases.

Que 4.9. What do you understand by adaptive quantization ? Explain the various approaches to adapting the quantizer parameters. UPTU 2012-13, Marks 10

Answer

1. For a stationary input signal, if its pdf deviates from that with which the optimum quantizer is designed, then mismatch will take place and the performance of the quantizer will deteriorate.
2. Adaptive quantization attempts to make the quantizer design adapt to the varying input statistics in order to achieve better performance. By statistics, we mean the statistic mean, variance, and type of input pdf.
3. If the mean of the input changes with time then, differential encoding is a suitable method to handle the variation. For other type of cases, adaptive quantization is found to be effective. The price paid in adaptive quantization is processing delay and an extra storage requirement.
4. There are two different types of adaptive quantization :
   i. Forward adaptation or offline approach.
   ii. Backward adaptation or online approach.

Forward adaptive quantization :

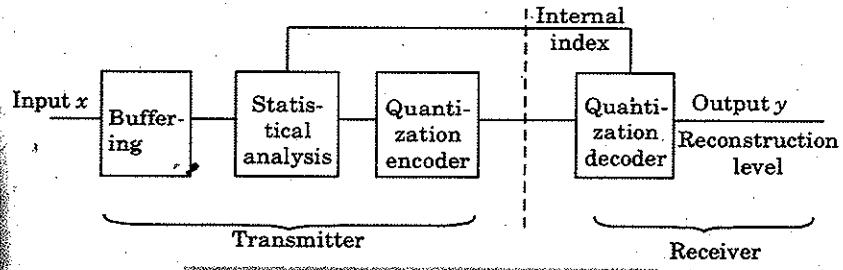


Fig. 4.9.1. Forward adaptive quantization.

1. In forward adaptive quantization, the source output is divided into blocks of data. Each block is analyzed before quantization. The selection of block size is critical issue.
2. This approach required a delay of at least the amount of time required to process a block of data.

3. If the size is small, the adaptation to the local statistics will be effective, but the side information needs to be send frequently.
4. The side information has to be transmitted more often, which means the amount of overhead per sample increase.
5. If the size is large, the bits used for side information decreases.
6. The adaptation becomes less sensitive to changing statistics, and both processing delay and storage required increases.
7. In practice, a proper compromise between quality of side information and effectiveness of adaptation produces a good selection of the block size.

Backward adaptive quantization :

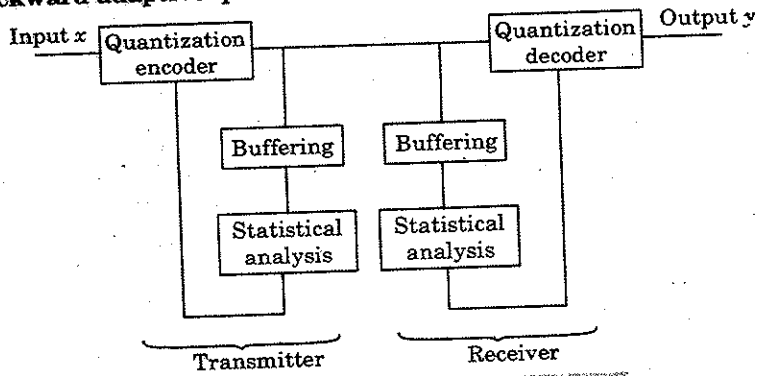


Fig. 4.9.2. Backward adaptive quantization.

1. In backward adaptive quantization, only the past quantized samples are available for use in adapting the quantizer.
2. The value of the inputs are only known to the encoder, therefore this information cannot be used to adapt the quantizer.
3. We should observe the output of the quantizer for a long period of time, then expand the quantizer step size if the input falls in the outer levels an excessive number of times, and contract the step size if the input falls in the inner levels an excessive number of times.

Que 4.10. Explain Jayant quantizer.

Answer

1. It is expected that observing a sufficient large number of input or output (quantized) data is necessary in order to track the changing statistics and then adapt the quantizer setting in adaptive quantization.
2. S. Jayant at Bell labs showed that we did not need to observe the quantizer output over a long period of time, in fact, effective adaptations can be realized with an explicit memory of only one word.

3. That is, either one input sample, x, in forward adaptive quantization or a quantized output, y, in backward adaptive quantization is sufficient. The idea is as follows :
   i. If at moment t_i the input sample x_i falls into the outer interval, then the step size at the next moment t_{i+1} will be enlarged by a factor m_i ($m_i > 1$).
   ii. If the input x_i falls into an inner interval close to $x = 0$ then the multiplier $m_i < 1$.
4. In this way, the quantizer adapts itself to the input to avoid overload as well as under load to achieve better performance.

Que 4.11. Explain uniform and non-uniform quantization with further classifications.

OR

Differentiate between uniform quantization and non-uniform quantization. **UPTU 2012-13, 2014-15; Marks 10**

Answer

1. A uniform quantizer is not optimal if source is not uniformly distributed.
2. A quantizer that has non-uniform intervals is called a non-uniform quantizer.
3. Non-uniform quantizer attempts to decrease the distortion (*i.e.*, mean square error (MSE)) by assigning more levels to more probable regions.
4. This is done by decreasing the quantization interval where probability is high and increasing the quantization interval where probability is low.

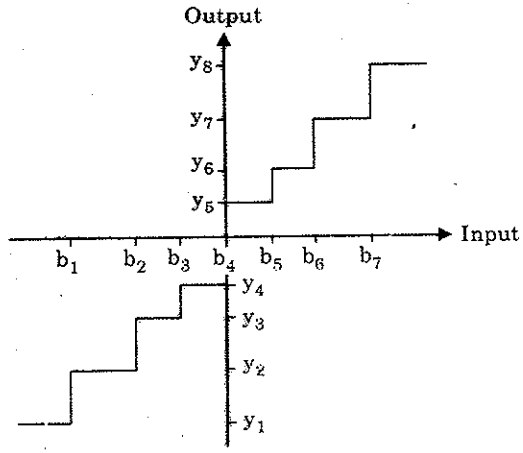


Fig. 4.11.1.

Non-uniform quantization :

1. A quantization that has non-uniform intervals is called non-uniform quantization.
2. In order to decrease the average distortion, we can try to approximate the input better in regions of high probability.
3. This can be done by making the quantization intervals smaller in those regions that have more probability mass.
4. While a non-uniform quantization provides lower average distortion, design of non-uniform quantizers is also somewhat more complex.
5. However, the basic idea is : find the decision boundaries and reconstruction levels that minimize the mean squared quantization error.

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x-\hat{x})^2 f(x)dx = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x-y_k)^2 f(x)dx \qquad ...(4.11.1)$$

5. For given M and input pdf, we need to choose $\{b_i\}$ and $\{y_i\}$ to minimize the distortion.
6. For this, we have to satisfy the Lagrangian condition :

$$\frac{\partial \sigma_q^2}{\partial y_i} = 0 \text{ and } \frac{\partial \sigma_q^2}{\partial b_i} = 0$$

$$\frac{\partial \sigma_q^2}{\partial y_i} = 0 \Rightarrow y_i = \frac{\int_{b_{i-1}}^{b_i} x\, f_X(x)\, dx}{\int_{b_{i-1}}^{b_i} f_X(x)\, dx} \qquad ...(4.11.2)$$

7. y_i is the centroid of the interval $[b_{(i-1)}, b_{(i)}]$

$$\frac{\partial \sigma_q^2}{\partial b_i} = 0 \Rightarrow b_i = \frac{y_i + y_{i+1}}{2} \qquad ...(4.11.3)$$

8. b_i is the midpoint of y_i and y_{i+1}.
   a. Given b_i, we can find corresponding optimal y_i.
   b. Given y_i, we can find the corresponding optimal b_i.
   c. And, how we can find optimal b_i and y_i simultaneously.
9. A deadlock occurs as a reconstruction levels depends on decision level and decision level depends on reconstruction level.
10. An iterative solution to the above problem is given by Lloyd. The Lloyd algorithm solves the problem of iteratively optimizing the encoder and decoder until both conditions are [illegible] with sufficient accuracy.

Lloyd algorithm :

1. Start from an initial set of reconstruction value y_i.
2. Find all decision levels as,

$$b_i = \frac{y_i + y_{i+1}}{2} \qquad ...(4.11.4)$$

3. Compute MSE :

$$\sigma_q^2 = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x-y_k)^2 f(x)dx \qquad ...(4.11.5)$$

4. Stop, if MSE changes little from last time.
5. Otherwise, update y_i.

$$y_i = \frac{\int_{b_{i-1}}^{b_i} x\, f_X(x)dx}{\int_{b_{i-1}}^{b_i} f_X(x)dx} \qquad ...(4.11.6)$$

and go to step 2.

Classifications of non-uniform quantization :

i. *pdf*-Optimized quantization :

1. A direct approach for locating the best non-uniform quantization, if we have a probability model for the source, is to find $\{b_i\}$ and $\{y_i\}$ that minimize equation (4.11.1).

$$\sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-y_i)^2\, f_X(x)\, dx \qquad ...(4.11.7)$$

2. Setting the derivative of equation (4.11.1) with respect to y_i to zero and solving for y_j, we get :

$$y_j = \frac{\int_{bj-1}^{b_j} x\, f_X(x)\, dx}{\int_{b_{j-1}}^{b_j} f_X(x)\, dx} \qquad ...(4.11.8)$$

3. The output point of each quantization interval is the centroid of the probability mass in that interval.
4. Taking the derivative with respect to b_j and setting it equal to zero, we get an expression for b_j as :

$$b_j = \frac{y_{j+1} + y_j}{2} \qquad ...(4.11.9)$$

5. Decision boundary is midpoint of two neighbouring reconstruction levels.
6. Solving these two equations will give us the values for reconstruction levels and decision boundaries that minimize the mean squared quantization error.

ii. Companded quantization :

1. Instead of making step size small, we could expand the region in which input lands with high probability in proportion to the probability with which the input lands in this region.
2. This is idea behind companded quantization.

3. This can be explained with the block diagram :

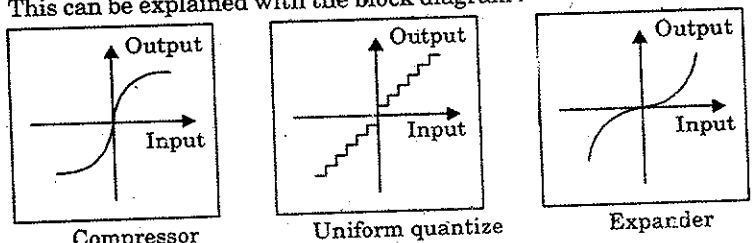


Fig. 4.11.2.

4. Input is first mapped through a compressor function.
5. This function "stretches" high probability regions close to the origin and "compresses" low probability regions away from the origin.
6. Regions close to the origin in the input to the compressor occupy greater fraction of the total region covered by compression.

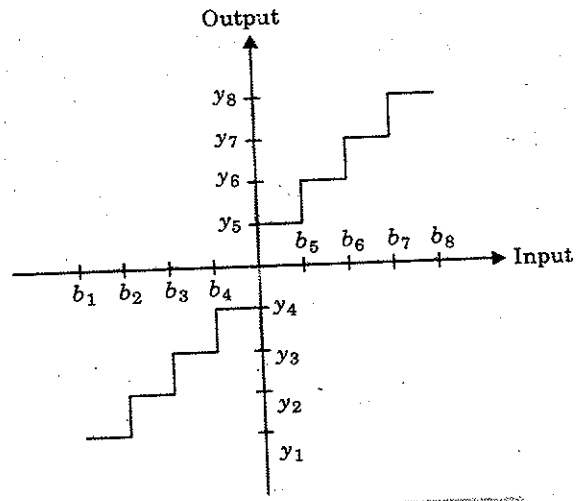


Fig. 4.11.3. Non-uniform quantization.

Uniform quantization : Refer Q. 4.8, Page 125D, Unit-4.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Describe rate distortion theory and derive the rate function for the binary source and Gaussian source.

Ans. Refer Q. 4.4.

Q. 2. Define quantization and explain quantization problem.

Ans. Refer Q. 4.7.

Q. 3. What do you understand by uniform quantizer ? How uniform quantization of a uniformly distributed source and uniform quantization of non-uniform source is done ?

Ans. Refer Q. 4.8.

Q. 4. Write about non-uniform quantization.

Ans. Refer Q. 4.11.

Q. 5. Explain Lloyd algorithm.

Ans. Refer Q. 4.11.

☺☺☺

5 UNIT

Vector Quantization

PART-1

Advantages of Vector Quantization over Scalar Quantization and The Linde-Buzo-Gray Algorithm.

CONCEPT OUTLINE : PART-1

- Vector quantization is a lossy data compression method based on the principle of block coding.
- Vector quantization is used for lossy data correction and density estimation.
- The Linde-Buzo-Gray algorithm solves the optimality criteria which require an initial codebook.
- At the encoder and decoder, there is a set of L-dimensional vectors called the codebook.
- Vectors in the codebook are called code vectors.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.1. What do you understand by vector quantization ? Also, explain the procedure of vector quantization.

UPTU 2014-15, Marks 10

Answer

1. Vector Quantization (VQ) is a lossy data compression method based on the principle of block coding. In vector quantization, we do not represent individual values but (usually small) array of them.
2. A typical example is a color map : a color picture can be represented by a 2D array of triplets (RGB).
3. In most pictures, those triplets do not cover the whole RGB space but tend to concentrate in certain areas.
4. For example, the picture of a forest will typically have a lot of green.
5. One can select a relatively small subset of representative color *i.e.*, RGB triplets and then approximate each triplet by the representative of that small set.
6. In vector quantization, we group the source output into blocks or vectors. For example, we can treat L consecutive samples of speech as the components of an L-dimensional vector.

7. Or, we can take a block of L pixels from an image and treat each pixel value as a component of a vector of size or dimension L.
8. This vector of source outputs forms the input to the vector quantizer.
9. At both the encoder and decoder of the vector quantizer, there is a set of L-dimensional vectors called the codebook of vector quantizer.
10. The vectors in this codebook, known as code vectors, are selected to be representative of the vectors we generate from the source output.
11. Each code vector is assigned a binary index.
12. At the encoder, the input vector is compared to each code vector in order to find the code vector closest to the input vector.
13. The elements of this code vector are the quantized values of the source output.
14. In order to inform the decoder about which code vector was found to be the closest to the input vector, we transmit or store the binary index of the code vector.
15. Because the decoder has exactly the same codebook, it can retrieve the code vector given its binary index.
16. A pictorial representation of this process is shown in Fig. 5.1.1.
17. Although the encoder may have to perform a considerable amount of computations in order to find the closest reproduction vector to the vector of source outputs, the decoding consists of a table lookup.
18. This makes vector quantization a very attractive encoding scheme for applications in which the resources available for decoding are considerably less than the resources available for encoding.

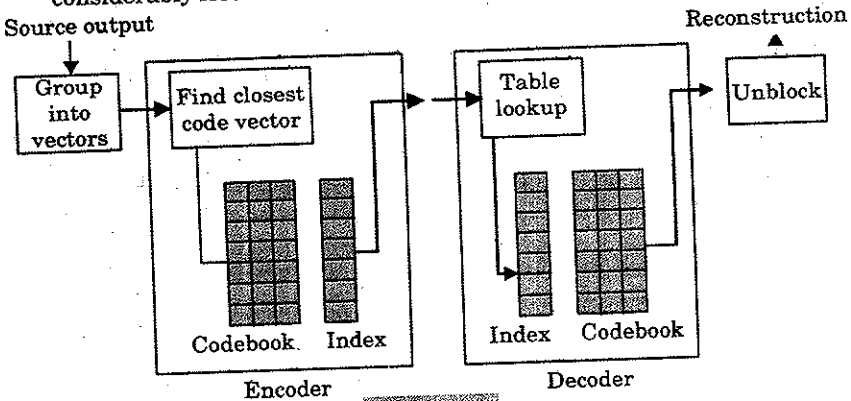


Fig. 5.1.1.

Que 5.2. **What do you mean by codebook of quantizer ?**

UPTU 2011-12, Marks 05

Answer

Refer Q. 5.1, Page 137D, Unit-5.

Que 5.3. **What is the advantage of vector quantization over scalar quantization ?**

UPTU 2012-13, Marks 10

UPTU 2015-16, Marks 02

Answer

1. Vector quantization is used for lossy data compression and the vector quantization work on a group of data or vector which provide better compression as compared to scalar quantization.
2. Vector quantization is used for lossy data correction and density estimation.
3. Lossy data correction, or prediction is used to recover data missing from some dimensions.
4. It is done by finding the nearest group with the data dimensions available, then predicting the result based on the values for the missing dimensions, assuming that they will have the same values as the groups centroid.
5. Vector quantization results in lower distortion.

Que 5.4. **Explain the steps of the Linde-Buzo-Gray algorithm.**

UPTU 2012-13, 2013-14, 2014-15, Marks 10

UPTU 2011-12, Marks 05

UPTU 2015-16, Marks 02

Answer

1. The LBG design algorithm solves the optimality criteria.
2. The algorithm requires an initial codebook.
3. The initial codebook is obtained by the splitting method. In this method, an initial code vector is set as the average of the entire training sequence.
4. This code vector is then split into two.
5. The iterative algorithm is run with these two vectors as the initial codebook.
6. The final two code vectors are splitted into four and the process is repeated until the desired number of code vector is obtained.

Steps of the algorithm are as follows :

1. Start with an initial set of reconstruction value $\left\{Y_i^{(0)}\right\}_{i=1}^{M}$ and a set of training vectors $\left\{X_n\right\}_{n=1}^{N}$. Set $k = 0$, $D^{(0)} = 0$. Select threshold $\in$.
2. The quantization regions $\left\{V_i^{(k)}\right\}_{i=1}^{M}$ are given by,

$$V_i^{(k)} = \{X_n : d(X_n, Y_i) < d(X_n, Y_j)\ \forall j \neq i\} \quad i = 1, 2, ..., M.$$

3. Compute the average distortion $D^{(k)}$ between the training vector and the representative reconstruction value.

$$D^{(k)} = \sum_{i=1}^{M} \int_{V_i(k)} \left\| X - Y_i^{(k)} \right\|^2 f_X(X)\, dX$$

4. If $\frac{(D^{(k)} - D^{(k-1)})}{D^{(k)}} < \in$ stop; otherwise, continue.

5. $k = k + 1$. Find new reconstruction values $\left\{ Y_i^{(k)} \right\}_{i=1}^{M}$ that are average value of the element of each of the quantization region $V_i^{(k-1)}$. Go to step 2.

Que 5.5. **What is quantization ? Explain additive noise model of a quantizer. Differentiate between scalar quantization and vector quantization. Discuss the advantages of vector quantization over scalar quantization.**

UPTU 2012-13, Marks 10

UPTU 2015-16, Marks 15

Answer

Quantization : Refer Q. 4.7, Page 120D, Unit-4.
Additive noise model of a quantizer : Refer Q. 4.7, Page 120D, Unit-4.
Scalar v/s vector quantization :

S. No.	Scalar quantization	Vector quantization
1.	In scalar quantization, each sample is quantized independently.	In vector quantization, each of the samples is not quantized.
2.	The scalar quantization results in higher distortion for a given rate (in bits per sample).	The vector quantization results in lower distortion for a given rate (in bits per sample).

Advantages of vector quantization over scalar quantization : Refer Q. 5.3, Page 139D, Unit-5.

PART-2

Tree Structured Vector Quantizer and Structured Vector Quantizer.

CONCEPT OUTLINE : PART-2

- The idea of tree structured vector quantization is to divide the set of output.

- Pyramid and lattice structured vector quantizer do not have storage problems.
- Regular arrangements of output points in space are called lattices.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 5.6. **Discuss the tree structured vector quantizers.**

Answer

1. Consider the two-dimensional vector quantizer shown in Fig. 5.6.1, the output points in each quadrant are the mirror image of the output points in neighbouring quadrants.
2. Given an input to this vector quantizer, we can reduce the number of comparisons necessary for finding the closest output point by using the sign on the components of the input.
3. The sign on the components of the input vector will tell us in which quadrant the input lies.
4. Because all the quadrants are mirror images of the neighbouring quadrants, the closest output point to a given input will lie in the same quadrant as the input itself.
5. Therefore, we only need to compare the input to the output points that lie in the same quadrant, thus reducing the number of required comparisons by a factor of four.
6. This approach can be extended to L dimensions, where the signs on the L components of the input vector can tell us in which of the 2^L hyperquadrants the input lies, which in turn would reduce the number of comparisons by 2^L.
7. This approach works well when the output points are distributed in a symmetrical manner.
8. However, it breaks down as the distribution of the output points becomes less symmetrical.

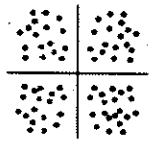


Fig. 5.6.1. A symmetrical vector quantizer in 2 dimensions.

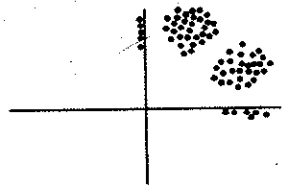


Fig. 5.6.2. Breakdown of the method using the quadrant approach.

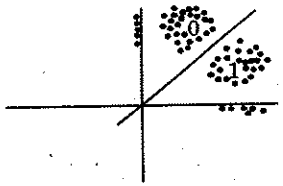


Fig. 5.6.3. Division of output points into two groups.

9. The idea of using the L-dimensional equivalents of quadrants to partition the output points in order to reduce the computational load can be extended to non-symmetrical situations, like those shown in Fig. 5.6.2.
10. Divide the set of output points into two groups, group 0 and group 1, and assign to each group a test vector such that output points in each group are closer to the test vector assigned to that group than to the test vector assigned to the other group (Fig. 5.6.3).
11. Label the two test vectors 0 and 1.
12. When we get an input vector, we compare it against the test vectors.
13. Depending on the outcome, the input is compared to the output points associated with the test vector closest to the input.
14. After these two comparisons, we can discard half of the output points.
15. Comparison with the test vectors takes the place of looking at the signs of the components to decide which set of output points to discard from contention.
16. If the total number of output points is K, with this approach we have to make $\frac{K}{2} + 2$ comparisons instead of K comparisons.
17. This process can be continued by splitting the output points in each group into two groups and assigning a test vector to the subgroups.
18. So, group 0 would be split into group 00 and group 01, with associated test vectors labeled 00 and 01, and group 1 would be split into group 10 and group 11, with associated test vectors labeled 10 and 11.
19. Suppose the result of the first set of comparisons was that the output point would be searched for in group 1.

20. The input would be compared to the test vectors 10 and 11.
21. If the input was closer to the test vector 10, then the output points in group 11 would be discarded, and the input would be compared to the output points in group 10.
22. We can continue the procedure by successively dividing each group of output points into two, until finally, if the number of output points is a power of two, the last set of groups would consist of single points.
23. The number of comparisons required to obtain the final output point would be 2 log K instead of K.
24. Thus, for a codebook of size 4096, we would need 24 vector comparisons instead of 4096 vector comparison.

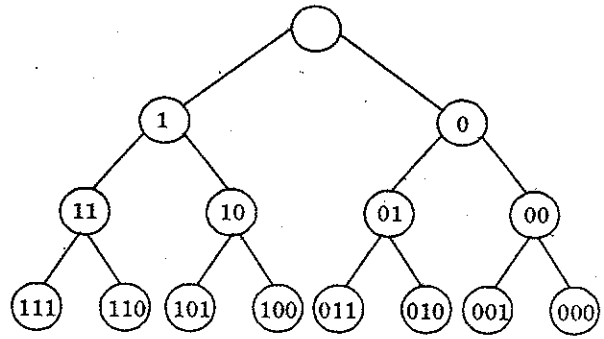


Fig. 5.6.4. Decision tree for quantization.

Que 5.7. Write a short note on structured vector quantizers.

OR

Explain the following quantization techniques in detail :

i. Structured vector quantization

ii. Pyramid vector quantization **UPTU 2013-14, Marks 10**

Answer

Structured vector quantization :

1. The tree structured vector quantizer solves the complexity problem, but acerbates the storage problem.
2. We now take an entirely different tack and develop vector quantizers that do not have these storage problems.

1. Pyramid vector quantization :

a. Suppose we are quantizing a random variable X with pdf $f_X(X)$ and differential entropy $h(X)$. Suppose we block samples of this random variable into a vector X.

b. A result of Shannon's, called asymptotic equipartition property (AEP), states that for sufficiently large L and arbitrarily small ϵ

$$\left|\frac{\log f_X(X)}{L} + h(X)\right| < \epsilon$$

for all but a set of vectors with a vanishingly small probability.

c. Almost all L-dimensional vectors lie on a contour of constant probability given by,

$$\left|\frac{\log f_X(X)}{L}\right| = -h(X)$$

d. Optimum manner to encode the source would be to distribute 2^{RL} points uniformly in this region. This is used to design a vector quantizer called pyramid vector quantizer.

e. This vector quantizer consists of the rectangular quantizer that fall on hyperpyramid and is given by,

$$\sum_{i=1}^{L} |x_i| = C$$

where C is a constant depending on the variance of input.

f. Now find the distance value,

$$r = \sum_{i=1}^{L} |x_i|$$

g. The value is quantized and transmitted to the receiver. The input is normalized by gain term and quantized using a single hyperpyramid.

h. Quantization of shape consists of finding the output points on the hyperpyramid closest to the shape and finding a binary codeword for it.

2. Lattice vector quantizers :

a. Regular arrangements of output points in space are called lattices. We can define the lattice as follows :

Let $\{a_1, a_2, \ldots, a_L\}$ be L independent L-dimensional vectors. Then the set,

$$L = \left\{X : X = \sum_{i=1}^{L} u_i a_i\right\}$$

is a lattice of $\{u_i\}$ which consists of all integers.

b. When a subset of lattice points is used as the output points of vector quantizer, the quantizer is known as lattice vector quantizer.

c. It solves the storage problem. Any lattice point can be regenerated if we know the basis set, there is no need to store the output points.

d. Any point in the lattice is given by $na_1 + ma_2$, where n and m are integers. But,

$$na_1 + ma_2 = \begin{bmatrix} n+m \\ n-m \end{bmatrix}$$

and the sum of the coefficients is $n + m + n - m = 2n$, which is even for all n. Therefore, all points in this lattice have an even coordinate sum. Lattices with these properties are called D lattices.

3. Polar and spherical vector quantizers :

a. For the Gaussian distribution, the contours of constant probability are circles in two dimensions and spheres and hyperspheres in three and higher dimensions.

b. In two dimensions, we can quantize the input vector by first transforming it into polar coordinates r and θ.

$$r = \sqrt{x_1^2 + x_2^2} \quad \ldots(1)$$

and $$\theta = \tan^{-1}\frac{x_2}{x_1}$$

r and θ can then be either quantized independently, or we can use the quantized value of r as an index to a quantizer for θ.

c. The former is known as a polar quantizer; the latter, an unrestricted polar quantizer.

d. The advantage to quantizing r and θ independently is one of simplicity.

e. The quantizers for r and θ are independent scalar quantizers.

f. However, the performance of the polar quantizer is not significantly higher than that of scalar quantization of the components of the two-dimensional vector.

g. The unrestricted polar quantizer has a more complex implementation, as the quantization of θ depends on the quantization of r.

h. However, the performance is also somewhat better than the polar quantizer.

i. The polar quantizer can be extended to three or more dimensions.

Que 5.8. Write short notes on any two :

i. Structure vector quantization

ii. Pyramid vector quantization

iii. Advantages of scalar quantization **UPTU 2014-15, Marks 10**

Answer

i. Refer Q. 5.7, Page 143D, Unit-5.

ii. Refer Q. 5.7, Page 143D, Unit-5.

iii. **Advantages of scalar quantization :**

1. Useful for analog to digital conversion.
2. With entropy coding, it yields good lossy compression.
3. Llyod algorithm works very well in practice.

Que 5.9. What is tree structured vector quantization ? Explain the following quantization technique :

i. Structured vector

ii. Pyramid vector quantization

Answer

Tree structured vector quantization : Refer Q. 5.6, Page 141D, Unit-5.

i. Refer Q. 5.7, Page 143D, Unit-5.

ii. Refer Q. 5.7, Page 143D, Unit-5.

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Write a short note on vector quantization.

Ans. Refer Q. 5.1.

Q. 2. Discuss the advantages of vector quantization over scalar quantization.

Ans. Refer Q. 5.3.

Q. 3. Differentiate vector quantization and scalar quantization.

Ans. Refer Q. 5.5.

Q. 4. Write short note on :

i. Tree structured vector quantizer

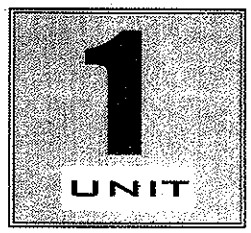
ii. Pyramid vector quantization

iii. Lattice vector quantization

Ans. i. Refer Q. 5.6.
ii. Refer Q. 5.7.
iii. Refer Q. 5.7.

Q. 5. Explain the Linde-Buzo-Gray algorithm.

Ans. Refer Q. 5.4.



Introduction (2 Marks Questions)

Memory Based Questions

1.1. Define data compression.

Ans. It is the process of encoding information using fewer bits than a decoded representation would use through use of specific encoding schemes involves encoding information using fewer bits than the original representation compression can be either loss or lossless.

1.2. Why do we need data compression ?

Ans. It needed because it helps to reduce the consumption of expensive resources such as a hard disk space or transmission bandwidth.

1.3. Describe lossless data compression.

Ans. Lossless data compression is a class of data compression that allows the exact original data to be reconstructed from the compressed data. It involves no loss of information.

1.4. What are the two ways of constructing statistical model ?

Ans. There are the two primary ways of constructing statistical models :
i. Static model ii. Adaptive model

1.5. Name some common lossless compression algorithm.

Ans. i. Run length encoding ii. LZW

1.6. Define lossy compression.

Ans. Lossy compression is a class of data compression that does not allow the exact original data to be reconstructed from the compressed data. It involves loss of information.

1.7. What are the two basic lossy compression schemes ?

Ans. i. Lossy transform codec
ii. Lossy predictive codec

1.8. Differentiate between lossless and lossy compression.

Ans.

S. No.	Lossless compression	Lossy compression
i.	Lossless compression allows the exact original data to be reconstructed from the compressed data.	Lossy compression data does not allow the exact original data to be reconstructed from the compressed data.
ii.	Lossless data compression is mainly used to compress text file or images, where loss of data is not acceptable.	Lossy data is mainly used to compress multimedia data. Better compression ratio can be achieved in lossy compression than lossless compression.

1.9. Write about compression ratio.

Ans. Compression ratio is the ratio of the size of original data to the size of the compressed data. It is a relative term.

1.10. What are the measure of data compression algorithms ?

Ans.
i. The performance can be measured by measuring the relative complexity of the algorithms.
ii. The amount of hardware resource required to implement the algorithm.
iii. The amount of compression provided by the algorithm.

1.11. Give the phases of development of data compression algorithm.

Ans. i. Modelling ii. Coding

1.12. Name various approaches for building mathematical model.

Ans. i. Physical model ii. Probability model iii. Markov model

1.13. Define composite source model.

Ans. It is a model which is a combination of several sources with only one source being active at any given time.

1.14. What are uniquely decodable code ?

Ans. A code is uniquely decodable if the mapping $C^+ : A_x^1 \rightarrow A_z^+$ is one to one that is $\forall\, x$ and x' in $A_x^+, x \neq x' \rightarrow C^+(x) \neq C^+(x')$.

1.15. Explain the rate distortion theory.

Ans. In rate distortion theory, the decompressed data does not have to be exactly the same as original data. Instead some amount of distortion D is there under tolerable limit. Shannon showed that for a given source and a given distortion measure, there is a function (RCD) called the Rate Distortion Function.

1.16. Define entropy.

Ans. It is the measure of the uncertainty associated with a random variable.

1.17. Define Markov model.

Ans. It is used to represent dependency in the data. In discrete time Markov chain model is a process which is basically used for lossless compression.

1.18. Give the various applications of lossless compression.

Ans.
i. It is used in zip file format and the Unix tool gzip.
ii. PNG or GIF use only lossless compression.

1.19. Define space saving.

Ans. It is defined as the reduction in size relative to the uncompressed size.

Application Based Questions

1.20. A representation compresses a 10MB file to 2MB. Find compression ratio.

Ans.
$$\text{Compression ratio} = \frac{\text{Compressed size}}{\text{Uncompressed size}}$$
$$= \frac{10}{2} = 5 \; i.e., \; 5:1$$

1.21. Determine whether the following code is uniquely decodable ?
{0, 01, 11, 111}
{0, 10, 110, 111}

Ans. {0, 01, 11, 111}
The codeword 0 is prefix of codeword 01 with dangling suffix 1 add it to list {0, 01, 11, 111, 1}
The codeword 1 is prefix to 111 with dangling suffix 11 which in itself is a codeword so it is not uniquely decodable.
{0, 10, 110, 111}
Since no codeword is prefix of another. So, it is uniquely decodable and is a prefix code.

1.22. Determine whether the code {01, 1, 2, 210} is uniquely decodable ?

Ans. Codeword 2 is prefix of codeword 210 with dangling suffix 10 so add 10 to the list {01, 1, 2, 210, 10}
Codeword 1 is prefix of code 10 with dangling suffix 0 so add 0 to the list {01, 1, 2, 210, 10, 0}
Codeword 0 is the prefix to codeword 01 with dangling suffix 1 and 1 itself is a codeword {case 1}
So, the above code is not uniquely decodable.

1.23. Calculate the amount of information contained in a weather forecast if the probabilities include {normal, rain, fog, hot, windy} if the respective probabilities are 0.80, 0.10, 0.04, 0.03 and 0.03.

Ans.
$$i(\text{normal, rain, fog, hot, windy}) = i(\text{normal}) + i(\text{rain}) + i(\text{fog}) + i(\text{hot}) + i(\text{windy})$$
$$= \log_2 \frac{1}{0.8} + \log_2 \frac{1}{0.10} + \log_2 \frac{1}{0.04} + \log_2 \frac{1}{0.03} + \log_2 \frac{1}{0.03}$$
$$= \log_2 1.25 + \log_2 10 + \log_2 25 + \log_2 33.3 + \log_2 33.3$$
$$= 0.32 + 3.3 + 4.6 + 5.07 + 5.07$$
$$= 18.36 \text{ bits}$$

☺☺☺

2 UNIT

Huffman Coding (2 Marks Questions)

Memory Based Questions

2.1. Define Huffman codes.

Ans: The codes generated using Huffman coding procedure is called Huffman codes. These codes are prefix codes and are optimum for a given model.

2.2. Give two observations on which Huffman procedure is based regarding optimum prefix code.

Ans:
i. In an optimum code, symbols that occur more frequently will have shorter codewords than symbols that occur less frequently.
ii. In an optimum code, the two symbols that occur least frequently will have the same length.

2.3. Write the properties of Huffman coding.

Ans:
i. **Unique prefix code :** No Huffman code is a prefix of any other Huffman code. It prevents any ambiguity in decoding.
ii. **Optimality :** Minimum redundancy code proved optimal for a given data model *i.e.*, given accurate probability distribution.

2.4. What is minimum variance Huffman code ?

Ans: It is obtained by making a small change in the conventional Huffman algorithm *i.e.*, when two or more symbols have the same probabilities as that of newly combined probability of the symbols.

2.5. What are the necessary conditions for an optimal variable length binary code ?

Ans:
i. Given any two letters a_j and a_k, if $p[a_j] \geq p[a_k]$, then $l_j \leq l_k$, where l_j is the number of bits in the codeword for a_j.
ii. The two least probable letters have codewords with the same maximum length l_m.

2.6. What do you mean by coding ?

Ans: Assigning codewords to the modeled data is called coding.

2.7. Write about the concept of adaptive Huffman coding.

Ans: In this, neither transmitter nor receiver knows anything about the statistics of the source sequence at the start of transmission. As transmission progresses, nodes corresponding to symbols transmitted will be added to the tree, and the tree is reconfigured using an update procedure.

2.8. How adaptive Huffman coding overcome the drawbacks of conventional Huffman coding ?

Ans:
i. Huffman coding assigns an output code to each symbol, with the output codes being as short as 1 bit, or considerably longer than the input symbols, strictly depending on their probabilities.
ii. Huffman coding suffers from the fact that the uncompressor need same knowledge of the probabilities of the symbols in the compressed file, providing this information need more bit to encode the file.

2.9. Explain Golomb code.

Ans: Golomb coding is a data compression scheme based on entropy encoding and is optimal for alphabets following a geometric distribution, making it highly suitable for situations in which the occurrence of small value in the input stream is significantly more likely than large value.

2.10. Mention any three applications of Huffman coding.

Ans:
i. Lossless image compression
ii. Text compression
iii. Audio compression

2.11. Write about Tunstall code.

Ans: It uses equal number of bits to represent all codewords, however each Tunstall codeword represents a different number of letters. The main advantage of a Tunstall code is that error in codeword does not propagate.

2.12. To design a Tunstall code what are the conditions to be follows ?

Ans: To design a Tunstall code that has a fixed codeword length but have variable number of symbols per codeword, should satisfy following conditions :
i. We should be able to parse a source output sequence into sequence of symbol that appears in the code book.
ii. We should maximize the average number of source symbol represented by each codeword.

2.13. Write about Rice code for lossless compression.

Ans. It is an adaptive version of Golomb code. In Rice code, a sequence of non-negative integers is divided into blocks of k integer piece. Each block is then coded.

2.14. What basically adaptive Huffman tree do ?

Ans. It basically do two things :

i. Increment the frequency count of the symbol.

ii. Update the configuration of the tree.

2.15. How text compression is an application of Huffman coding ?

Ans. In text, we have a discrete alphabet that, in a given class, has relatively stationary probability. It is known that for two documents that are substantially different, the two set of probabilities are very much alike.

Application Based Questions

2.16. Design a Golomb code for m = 5 given that $[\log_2 5] = 3$ and $[\log_2 5] = 2$.

Ans.

Table 2.1. Golomb code for m = 5.

n	q	r	Codeword	n	q	r	Codeword
0	0	0	000	8	1	3	10110
1	0	1	001	9	1	4	10111
2	0	2	010	10	2	0	11000
3	0	3	0110	11	2	1	11001
4	0	4	0111	12	2	2	11010
5	1	0	1000	13	2	3	110110
6	1	1	1001	14	2	4	110111
7	1	2	1010	15	3	0	111000

2.17. Draw the minimum variance Huffman encoding procedure.

Ans.

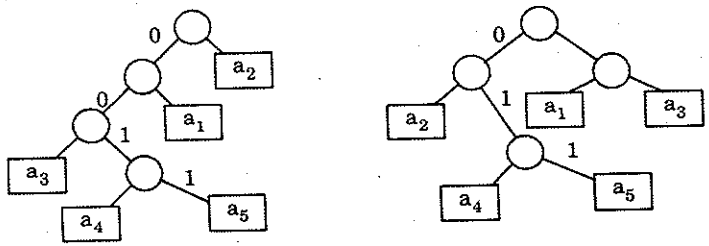


Fig. 2.1.

2.18. Give an example of 2 bit, Tunstall code.

Ans.

Sequence	Codeword
AAA	00
ABB	01
AB	10
B	11

2.19. Draw the flowchart to explain the encoding procedure.

Ans.

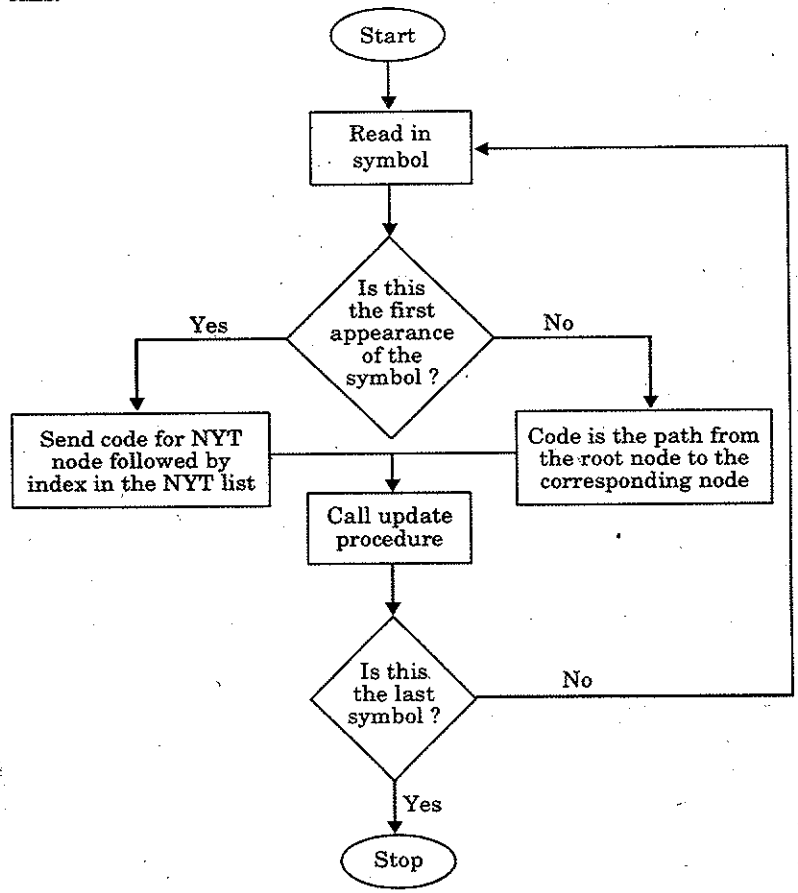


Fig. 2.2.

☺☺☺

3 UNIT

Arithmetic Coding (2 Marks Questions)

Memory Based Questions

3.1. Explain arithmetic coding.

Ans: The fundamental of arithmetic coding is to utilize the probability and entropy. In comparison to the well known Huffman coding algorithm, arithmetic coding overcomes the constraint that the symbol to be encoded has to be encoded by a whole number of bits and the constraints of Huffman code that is to find the Huffman codeword.

3.2. What are the applications of arithmetic coding ?

Ans: The application of arithmetic coding is in area of image, audio and video compression.

3.3. Compare Huffman and Arithmetic coding.

Ans:

S. No.	Huffman coding	Arithmetic coding
1.	Huffman algorithm requires building the entire code for all possible sequences of length m.	Arithmetic coding does not need to build entire code book instead we simply obtain the code for the tag corresponding to a given sequence.
2.	It is not easy to implement multiple Huffman codes in a system.	It is easy to implement multiple arithmetic codes in a system than Huffman codes.

3.4. How a binary code is generated ?

Ans: The first step for generating binary code is to find a tag for a given sequence. The next step is to find the unique and efficient binary code for the tag by taking the binary representation of the tag and truncating it to,

$$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1 \text{ bits}$$

3.5. Write any two advantages of arithmetic coding.

Ans:
i. Arithmetic coding is naturally suitable for adaptation strategies.
ii. Arithmetic coding is close to optimal compression performance for sources with very low entropies.

3.6. Write any two disadvantages of arithmetic coding.

Ans:
i. Unlike Huffman coding, the decoder requires knowledge of the number symbols to be decoded.
ii. Infinite precision is required.

3.7. Define the concept of static dictionary.

Ans: It is static in nature *i.e.*, the entry in the dictionary are predefined. It is mainly useful when prior knowledge of the output of the source is available.

3.8. Name two applications where LZW compression scheme is used.

Ans: i. UNIX compression ii. Compression over modems

3.9. Define uniqueness of arithmetic code.

Ans: It means it is uniquely decodable *i.e.*, no codeword is a prefix of another codeword because a prefix code is always uniquely decodable.

3.10. What do you understand by efficiency of the arithmetic code ?

Ans: It is measured by how close is the code to the entropy. Closer the code to the entropy more efficient it is.

3.11. Explain the JBIG standard of Bi-level image compression.

Ans: Joint Bi-level Image Processing Group (JBIG) recommends arithmetic coding as the coding scheme for coding binary images. The JBIG standard is a combination of lossless compression algorithm and progressive transmission algorithm.

3.12. Discuss about JBIG2.

Ans: It is an advance version of JBIG algorithm, it uses the same arithmetic coding scheme as JBIG. JBIG2 takes into consideration the structure of bi-level images which enhances the compression performance.

3.13. What are the problems in LZ77 ?

Ans: The problems in LZ77 are :
i. LZ77 to work efficiently, patterns should occur close together. This assumption is removed in LZ78.
ii. The coding of $<o, l, c>$ is not efficient.
iii. Performance of LZ77 increases with the size of buffer. But large buffer requires efficient search strategies.
iv. Using a triplet to encode a single character is inefficient.

3.14. Write about the concept of Prediction with Partial Match algorithm.

Ans: It is a context-based algorithm. PPM algorithm uses a large contexts to determine the probability of the symbol being encoded. The probabilities of symbols are estimated as they are encoded, this reduces the burden of storing them.

3.15. Write about the exclusion principle.

Ans. The basic idea of arithmetic coding is the division of the unit interval into sub intervals, each of which represents a particular letter which is smaller the sub interval, more bits are required to distinguish it from other sub intervals. If we can reduce the number of symbols to be represented the number of sub intervals goes down. This in turn means that size of sub intervals increases, leading to a reduction in number of bits required for encoding. The exclusion principle used in PPM provides this kind of reduction in rate.

3.16. Discuss facsimile encoding.

Ans. It is one of the oldest applications of lossless compression. In facsimile transmission, a page is scanned and converted into a sequence of black or white pixel and send over the communication channel.

3.17. Explain Dynamic Marker Model.

Ans. It was introduced by Cormack and Horspool. It generalizes the Capon model and takes into account the relationship of the past symbols to the symbols to be encoded.

3.18. What do you mean by GIF ?

Ans. It is a Graphic Interchange Format which is a platform independent image format which is suitable for transfer across slow connections. Compressed format that uses the LZW compression and compresses at a ratio of 3 : 1 and 5 : 1.

3.19. Define PNG.

Ans. It is a Portable Network Graphic format. It is one of the first standards to be collaboratively developed over the internet. The compression algorithm used in PNG is based on LZ77. The number of people bonded together and created a completely patent free graphics format called PNG. It is superior to GIFs in that it has better compression and support millions of colours.

3.20. Describe the idea of CALIC.

Ans. The Context Adaptive Losses Image Compression (CALIC) scheme, which came into being in response to a call for proposal for a new lossless image compression scheme in 1994. CALIC scheme functions in two modes, one for gray-scale images and another for bi-level images.

3.21. What are the weaknesses of LZ78 algorithms ?

Ans. LZ78 compression algorithm suffers from following weakness :

i. Dictionary grows without bound.
ii. Long phrases appear late.
iii. Inclusion of first non-matching symbol may prevent a good match.
iv. Few substrings of the processed input are entered into the dictionary.

Application Based Questions

3.22. Write the encoding algorithm for LZ77.

Ans. The encoding algorithm for LZ77 is as follows :

```
while (look ahead buffer not empty) {
go backward in search buffer to find longest match of the look
ahead buffer
if (length > 0)
{
output (position, length, next symbol);
shift the window length + 1 position along;
}
else
{
output (0, 0, first symbol in look ahead buffer);
shift the window 1 position along;
}
}
```


3.23. Encode the string BABAABRRRA using the LZ78 algorithm.

Ans. BABAABRRRA

Dictionary

Index	Entry	Encoded output
1	B	⟨0, C (B)⟩
2	A	⟨0, C (A)⟩
3	BA	⟨1, C (A)⟩
4	AB	⟨2, C (B)⟩
5	R	⟨0, C (R)⟩
6	RR	⟨5, C (R)⟩
7	A	⟨2,⟩

The compressed message is

⟨0, C (B)⟩ ⟨0, C (A)⟩ ⟨1, C (A)⟩ ⟨2, C (B)⟩
⟨0, C (R)⟩ ⟨5, C (R)⟩ ⟨2,⟩

3.24. Draw the flowchart for the encoding process of LZW compression algorithm.

Ans.

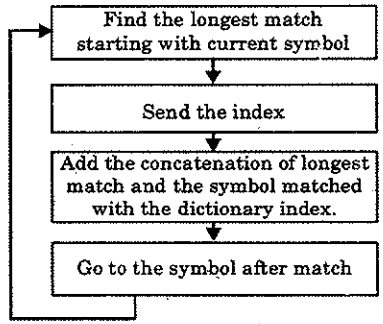


Fig. 3.1.

☺☺☺

4 UNIT

Mathematical Preliminaries for Lossy Coding (2 Marks Questions)

Memory Based Questions

4.1. Define rate distortion theory.

Ans. It is a major branch of information theory which provides the theoretical foundations for lossy data compression. It addresses the problem of determining the minimal amount of entropy R that should be communicated over channel. So, the source can be approximately reconstructed at the receiver without exceeding a given distortion D.

4.2. What is probability model used in lossy compression ?

Ans. The probability model corresponds to general rather than exact correspondence in contrast to lossless compression where probability model determines the exact match.

4.3. Describe quantization.

Ans. It is a process of mapping a continuous range of values by a relatively small, finite set of discrete symbol or integer values. For example, rounding a real number in the interval [0, 100] to an integer.

4.4. Define quantizer distortion or quantization noise.

Ans. The mean squared quantization error is given by :

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x - Q(x))^2 f_x(x)dx$$

$$= \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_x(x)dx$$

The difference between the quantizer input x and output $y = Q(x)$ referred as quantization error, is also called quantizer distortion or quantization noise.

4.5. What is scalar quantization ?

Ans. It is a mapping of an input value x into a number of output values y :

$$Q : x \rightarrow y$$

Example : Any real number x can be rounded off to the nearest integer say

$$q(x) = \text{round}(x)$$

4.6. State the quantization problem.

Ans. Quantization problem *i.e.,* problem of finding optimum scalar quantizer state that :

i. Given a distortion constraint $\sigma_q^2 \leq D^*$. Find the decision boundaries, reconstruction levels and binary code that minimize the rate *i.e.,* given an input pdf $f_x(x)$ and M, find adaptive $\{b_i\}$ and $\{y_i\}$ so as to minimize σ_q^2.

ii. Given a rate constraint $R \leq R^*$.Find the decision boundaries, reconstruction levels and binary codes that minimize the distortion.

4.7. Mention the two types of quantizer.

Ans.

i. **Midrise quantizer :** If zero is not reconstruction level of the quantizer it is called the midrise quantizer.

ii. **Midthread quantizer :** If zero is the reconstruction level of the quantizer *i.e,* $\Delta = 0$, it is called the midthread quantizer.

4.8. Define adaptive quantization.

Ans. Adaptive quantization attempts to make the quantizer design adapt to the varying input statistics in order to achieve better performance By statistics we mean the statistics mean, variance; and type of input pdf.

4.9. Briefly explain Jayant quantizer.

Ans. It is expected that observing a sufficient large number of input or output data is necessary in order to track the changing statistics and then adapt the quantizer setting in adaptive quantization. S.Jayant at Bell labs showed that we did not need to observe the quantizer output over a long period of time, in fact, effective adaptations can be realized with an explicit memory of only one word.

4.10. Define non-uniform quantization.

Ans. A uniform quantizer is not optimal if source is not uniformly distributed. A quantizer that has non-uniform intervals is called a non-uniform quantizer.

Non-uniform quantizer attempts to decrease the distortion by assigning more levels to more portable regions.

4.11. Write about Gaussian distribution as a probability model in lossy compression.

Ans. The Gaussian distribution is one of the most commonly used probability model as it is mathematically tractable. The probability density function for a random variable with a Gaussian distribution

and mean μ and variance σ^2 is

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(x-\mu)^2}{2\sigma^2}$$

4.12. Write about uniform distribution as a probability model in lossy compression.

Ans. It is used to model the source when we do not know anything about the distribution of the source output except possibly the range of values. The probability density function for a random variable uniformly distributed between a and b is given by

$$f_x(x) = \begin{cases} \frac{1}{b-a} & for \quad a \le x \le b \\ 0 & otherwise \end{cases}$$

4.13. What is encoder mapping ?

Ans. It divides the range of value that the source generates into a number of intervals. Each interval is then mapped to a codeword. It is many-to-one irreversible mapping. The codeword only identifies the intervals, not the original value. If the source or sample value comes from an analog source it is called an A/D converter.

4.14. What is decoder mapping ?

Ans. The decoder gives an estimated value that the source might have generated. Usually it is the midpoint of the interval but a more accurate estimate will depend on the distribution of the values in the interval. In estimating the values, the decoder might generate some errors. If the reconstruction is analog the decoder is referred to as D/A converter.

4.15. Name the two different types of adaptive quantization.

Ans. i. Forward quantization
ii. Backward quantization

Application Based Questions

4.16. Draw the block diagram of generic compression scheme.

Ans.

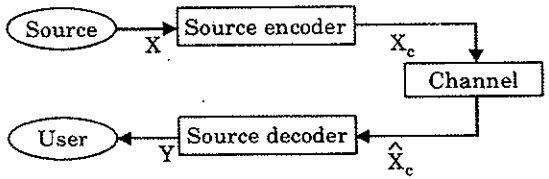


Fig. 4.1.

4.17. Depict the additive noise model of a quantizer.

Ans.

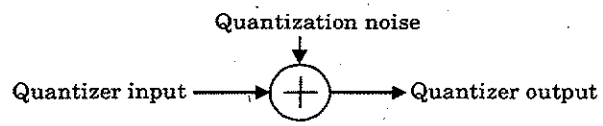


Fig. 4.2.

4.18. Draw the mapping of a 3-bit encoder.

Ans.

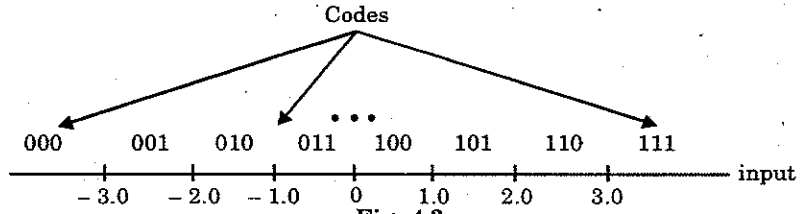


Fig. 4.3.

4.19. Draw the mapping of a 3-bit D/A converter

Ans.

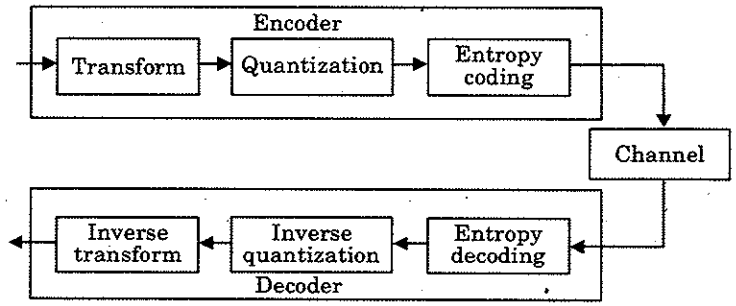


Fig. 4.4.

4.20. Show forward adaptive quantization with the help of block diagram only.

Ans.

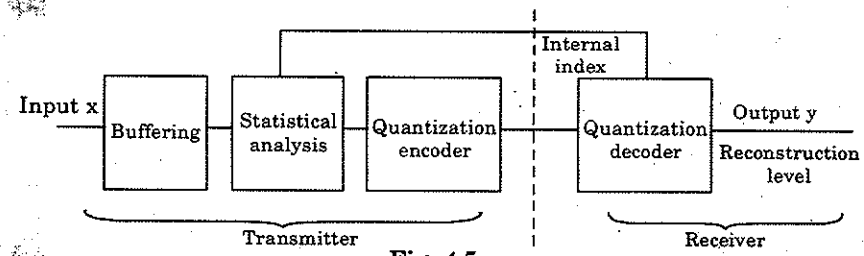


Fig. 4.5.

4.21. Show backword adaptive quantization with the help of block diagram only.

Ans:

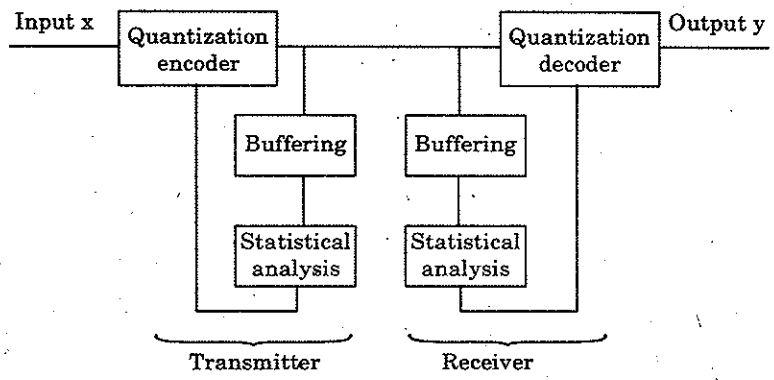


Fig. 4.6.

4.22. Write the Lloyd algorithm.

Ans: **Lloyd algorithm :**

i. Start from an initial set of reconstruction value y_i.

ii. Find all decision levels

$$b_i = \frac{y_i + y_{i+1}}{2}$$

iii. Compute MSE :

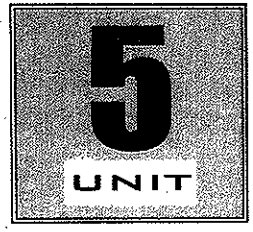
$$\sigma_q^2 = \sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} (x - y_k)^2 f(x) dx$$

iv. Stop if MSE changes little from last time.

v. Otherwise, update y_i.

$$y_i = \frac{\int_{b_{i-1}}^{b_i} x f(x)\, dx}{\int_{b_{i-1}}^{b_i} f(x)\, dx}$$

and go to step 2.



Vector Quantization (2 Marks Questions)

Memory Based Questions

5.1. Define vector quantization.

Ans: It is a lossy data compression method based on the principle of block coding. In vector quantization, we do not represent individual values but array of them.

5.2. What is the advantage of vector quantization over scalar quantization ?

Ans: Vector quantization is used for lossy data compression and the vector quantization work on a group of data or vector which provide better compression as compared to scalar quantization.

5.3. Write the drawbacks of tree structured vector quantizer.

Ans: i. Possible increase in distortion.
ii. TSVQ requires approximately twice the memory required by conventional VQ.

5.4. Discuss Gain-shape vector quantization.

Ans: In some application, the dynamic range of the input is quite large. To represent the various vectors from the source, we need a very large codebook. This requirement can be reduced by normalizing the source output vectors, then quantizing the normalized vector and normalization factor separately. Variations due to dynamic range are represented by the normalization factor (Gain). This technique is called Gain-shape VQ.

5.5. Define lattice vector quantizer.

Ans: When a subset of lattice points is used as the output points of vector quantizer, then the quantizer is known as lattice vector quantizer.

5.6. What are the advantages of pyramid and lattice structured vector quantizer over the tree structured vector quantizer ?

Ans: Tree structured vector quantizer solves the complexity problem, but arises the storage problem. Pyramid and lattice structured vector quantizer do not have storage problem. They also improve the rate distortion performance problem.

5.7. Write about the optimality criteria of vector quantization.

Ans: Let C and P are the solution to the minimization problem, then it must satisfy the following criteria :

i. Nearest neighbour condition :

$$S_n = \{x : | | X - C_n | |^2 \leq | | X - C_{n'} | |^2 \ \forall \ n' = 1, 2, \ldots\ldots N\}$$

ii. Centroid condition :

$$C_n = \frac{\sum X_{m \in S_n} \cdot X_m}{\sum X_{m \in Sn} \cdot 1}$$

$$n = 1, 2, \ldots\ldots N$$

5.8. What do you understand by codebook and codevectors ?

Ans: At the encoder and decoder there is a set of L-dimensional vectors called the codebook. Vectors in the codebook are called codevectors or output points.

Application Based Questions

5.9. Define the idea of tree structured vector quantization.

Ans:

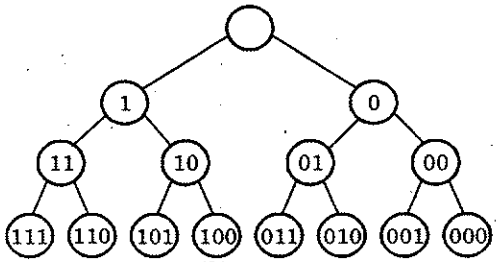


Fig. 5.1.

5.10. Draw the procedure of vector quantization.

Ans:

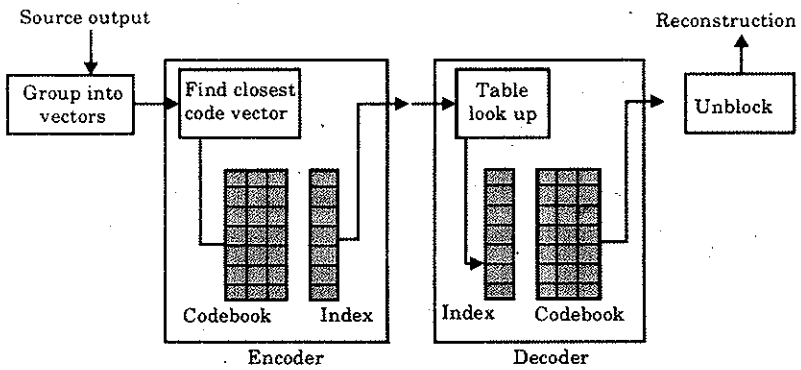


Fig. 5.2.

5.11. Draw a three-stage vector quantizer.

Ans:

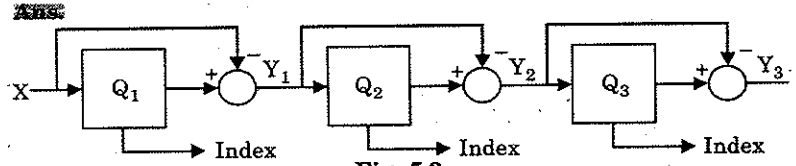


Fig. 5.3.

5.12. Draw a D_2 lattice.

Ans:

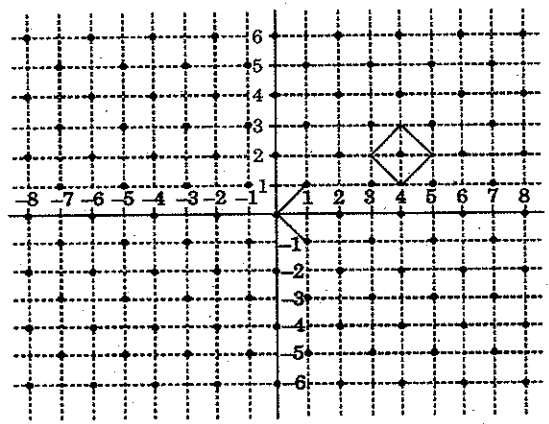


Fig. 5.4.

5.13. Draw the reconstruction levels for a 2-bit trellis coded quantizer.

Ans:

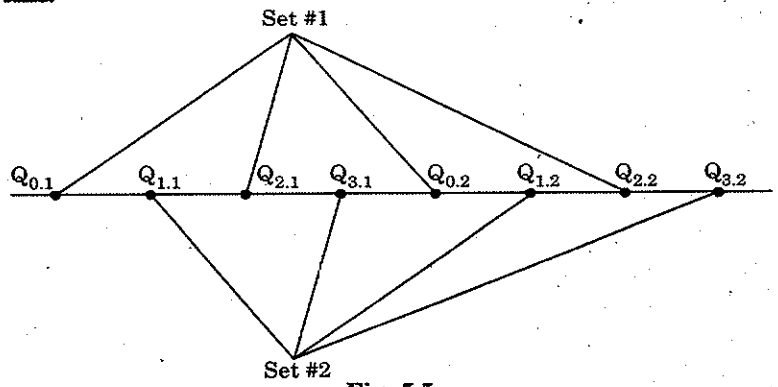


Fig. 5.5.

☺☺☺

B. Tech.
(SEM. VII) ODD SEMESTER THEORY EXAMINATION, 2011-12
DATA COMPRESSION

Time : 3 Hours **Total Marks : 100**

Note : Attempt all questions.

1. Attempt any **four** parts of the following : **(5 × 4 = 20)**

a. What do you mean by data compression ? Write some of the application where it is used.

Ans: Refer Q. 1.2, Page 6D, Unit-1.

b. What do you mean by lossless compression ? Compare lossless compression with lossy compression.

Ans: Refer Q. 1.3, Page 7D, Unit-1.

c. Differentiate between static length and variable length coding scheme. Explain with the help of an example.

Ans: Refer Q. 1.12, Page 15D, Unit-1.

d. What is average information ? What are the properties used in measure of average information ?

Ans: Refer Q. 1.13, Page 17D, Unit-1.

e. Explain Markov model and composite source models.

Ans: Refer Q. 1.14, Page 18D, Unit-1.

f. What do you understand by uniquely decodable codes ?

Ans: Refer Q. 1.24, Page 27D, Unit-1.

2. Attempt any **four** parts of the following : **(5 × 4 = 20)**

a. What is the limitation of Huffman coding ? Explain by an example.

Ans: Refer Q. 2.3, Page 33D, Unit-2.

b. Explain the update procedure of adaptive Huffman coding algorithm with the help of a flowchart.

Ans: Refer Q. 2.11, Page 43D, Unit-2.

c. Write down the application of Huffman coding in text compression and audio compression.

Ans: Refer Q. 2.23, Page 58D, Unit-2.

d. Explain Golomb codes and Tunstall codes.

Ans: Refer Q. 2.16, Page 53D, Unit-2.

e. Prove that the average codeword length l of an optimal code for a source S is greater than or equal to entropy $H(S)$.

Ans: Refer Q. 1.10, Page 14D, Unit-1.

f. Explain lossless image compression with an example.

Ans: Refer Q. 2.23, Page 58D, Unit-2.

3. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. i. What do you mean by binary code ? Compare binary code with Huffman code.

Ans: Refer Q. 3.7, Page 69D, Unit-3.

ii. Explain the run length coding with the help of suitable example.

Ans: Refer Q. 3.36, Page 103D, Unit-3.

b. i. Compare Huffman coding and arithmetic coding. How a tag is generated in arithmetic coding ?

Ans: Refer Q. 3.6, Page 68D, Unit-3.

ii. A sequence is encoded using LZW algorithm and the initial dictionary shown in table :

Index	Entry
1	a
2	b
3	r
4	t

The output of LZW encoder is the following sequence. 3, 1, 4, 6, 8, 4, 2, 1, 2, 5, 10, 6, 11, 13, 6. Decode this sequence.

Ans: Refer Q. 3.21, Page 86D, Unit-3.

c. i. Give differences between static and adaptive dictionary coding scheme.

Ans: Refer Q. 3.15, Page 77D, Unit-3.

ii. Write short notes on the following :
1. Dynamic Markov compression
2. Graphic Interchange Format

Ans: Refer Q. 3.38, Page 104D, Unit-3.

4. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. What is rate distortion theory ? Derive the rate distortion function for the :
i. Binary source ii. Gaussian source

Ans: Refer Q. 4.4, Page 113D, Unit-4.

b. Describe the steps involved in basic algorithm for Prediction with Partial Match (PPM).

Ans: Refer Q. 3.27, Page 91D, Unit-3.

c. What is lossy data encoding ? Write down the distortion measure criteria to check the fidelity of a reconstructed source sequence to the original one in such type of encoding techniques.

Ans: Refer Q. 4.1, Page 108D, Unit-4.

5. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. What is quantization ? Describe the quantization problem with the help of an example in detail.

Ans: Refer Q. 4.7, Page 120D, Unit-4.

b. What do you mean by codebook of quantizer ? Explain the steps of the Linde-Buzo-Gray algorithm.

Ans: Refer Q. 5.2, Page 138D and Q. 5.4, Page 139D; Unit-5.

c. What is tree structured vector quantization ? Explain the following quantization technique :
i. Structured vector quantization
ii. Pyramid vector quantization

Ans: Refer Q. 5.9, Page 146D, Unit-5.

B. Tech.
(SEM. VII) ODD SEMESTER THEORY EXAMINATION, 2012-13
DATA COMPRESSION

Time : 3 Hours **Total Marks : 100**

Note: Attempt all questions.

1. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. What is data compression and why we need it ? Explain compression and reconstruction with the help of block diagram. What are the measures of performance of data compression algorithms ?

Ans. Refer Q. 1.5, Page 9D, Unit-1.

b. What do you understand by information and entropy ? Given an alphabet A = $\{a_1, a_2, a_3, a_4\}$, find the first-order entropy in the following cases :
i. $P(a_1) = 1/2, P(a_2) = 1/4, P(a_3) = P(a_4) = 1/8$.
ii. $P(a_1) = 0.505, P(a_2) = 1/4, P(a_3) = 1/8$ and $P(a_4) = 0.12$.
And also differentiate between static length and variable length coding schemes. Explain with the help of examples.

Ans. Refer Q. 1.12, Page 15D, Unit-1.

c. What do you understand by prefix code ? Determine whether the following codes are uniquely decodable :
i. {0, 01, 110, 111} ii. {1, 10, 110, 111}
And also explain modeling and coding with the help of suitable examples.

Ans. Refer Q. 1.21, Page 24D, Unit-1.

2. Attempt any **two** questions of the following : **(10 × 2 = 20)**

a. How Rice code can be viewed ? Explain the implementation of the Rice code in the recommendation for lossless compression from the Consultative Committee on Space Data Standard. Explain adaptive Huffman coding. How is it different from conventional Huffman coding ?

Ans. Refer Q. 2.15, Page 50D, Unit-2.

b. For an alphabet $A = \{a_1, a_2, a_3, a_4, a_5\}$ with probabilities $P(a_1) = 0.15$, $P(a_2) = 0.04$, $P(a_3) = 0.26$, $P(a_4) = 0.05$ and $P(a_5) = 0.50$.
i. Calculate the entropy of this source.
ii. Find a Huffman code for this source.
iii. Find the average length of the code in (ii) and its redundancy.
And also design a 3-bit Tunstall code.
For an alphabet $A = \{a_1, a_2, a_3\}$ with probabilities $P(a_1) = 0.7$, $P(a_2) = 0.2$, $P(a_3) = 0.1$. Design a 3-bit Tunstall code.

Ans. Refer Q. 2.17, Page 54D, Unit-2.

c. Explain minimum variance Huffman code and encoding procedure taking a suitable example. What are the various applications of Huffman coding ?

Ans. Refer Q. 2.24, Page 58D, Unit-2.

3. Attempt any **two** questions of the following : **(10 × 2 = 20)**

a. A sequence is encoded using LZW algorithm and the initial dictionary shown in the table.

Index	Entry
1	a
2	b
3	r
4	t

The output of LZW encoder is the following sequence :

3	1	4	6	8	4	2	1	2	5	10	6	11	13	6

Decode this sequence. Discuss relative advantages of LZ77, LZ78 and LZW compression schemes.

Ans. Refer Q. 3.21, Page 86D, Unit-3.

b. What do you mean by binary code ? Compare binary code with Huffman code. What are adaptive compression schemes ? What is the basic difference between adaptive and statistical compression scheme ? Discuss with the model of adaptive compression.

Ans. Refer Q. 3.11, Page 73D, Unit-3.

c. What is Facsimile encoding ? Explain run length coding technique used earlier for Facsimile. Give a brief comparison of MH, MR, MMR and JBIG.

Ans. Refer Q. 3.37, Page 104D, Unit-3.

4. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. What do you understand by uniform quantizer ? How uniform quantization of a uniformly distributed sources and uniform quantization of non-uniform sources is done ?

Ans. Refer Q. 4.8, Page 125D, Unit-4.

b. What do you understand by adaptive quantization ? Explain the various approaches to adapting the quantizer parameters.

Ans. Refer Q. 4.9, Page 129D, Unit-4.

c. Discuss the steps involved in basic algorithm for Prediction with Partial Match (PPM).

Ans. Refer Q. 3.27, Page 91D, Unit-3.

5. Attempt any two parts of the following : (10 × 2 = 20)

a. What is quantization ? Explain additive noise model of a quantizer. Differentiate between scalar quantization and vector quantization. Discuss the advantages of vector quantization over scalar quantization.

Ans. Refer Q. 5.5, Page 140D, Unit-5.

b. Discuss the Linde-Buzo-Gray algorithm in detail.

Ans. Refer Q. 5.4, Page 139D, Unit-5.

c. Explain uniform and non-uniform quantization with further classifications.

Ans. Refer Q. 4.11, Page 131D, Unit-4.

B. Tech.
(SEM. VII) ODD SEMESTER THEORY EXAMINATION, 2013-14
DATA COMPRESSION

Time : 3 Hours **Total Marks : 100**

Note : Attempt all questions.

1. Attempt any **four** parts of the following : **(5 × 4 = 20)**

a. What is data compression and why we need it ? Explain compression and reconstruction with the help of block diagram.

Ans: Refer Q. 1.1, Page 5D, Unit-1.

b. Explain modeling and coding with the help of suitable examples.

Ans: Refer Q. 1.6, Page 10D, Unit-1.

c. What do you understand by information and entropy ? Find the first order entropy over an alphabet $A = \{a_1, a_2, a_3, a_4\}$ where $P(a_1) = P(a_2) = P(a_3) = P(a_4) = 1/4$.

Ans: Refer Q. 1.9, Page 13D, Unit-1.

d. What is zero frequency model in Markov models in text compression ?

Ans: Refer Q. 1.15, Page 21D, Unit-1.

e. Determine whether the following codes are uniquely decodable :

i. {0, 10, 110, 111}

ii. {1, 10, 110, 111}

Ans: Refer Q. 1.19, Page 24D, Unit-1.

f. What are the measures of performance of data compression algorithms ?

Ans: Refer Q. 1.5, Page 9D, Unit-1.

2. Attempt any **four** parts of the following : **(5 × 4 = 20)**

a. What is redundancy of code ? How it can be defined and calculated ?

Ans: Refer Q. 2.21, Page 56D, Unit-2.

b. Design Golomb code for $m = 5$ where values of n are 0, 1, ..., 10.

Ans: Refer Q. 2.14, Page 49D, Unit-2.

c. For an alphabet $A = \{a_1, a_2, a_3\}$ with probabilities $P(a_1) = 0.7$, $P(a_2) = 0.2$, $P(a_3) = 0.1$
Design a 3-bit Tunstall code.

Ans: Refer Q. 2.18, Page 55D, Unit-2.

d. Prove that the average codeword length l of an optimal code for a source S is greater than or equal to entropy $H(S)$.

Ans: Refer Q. 1.10, Page 14D, Unit-1.

e. Write down Huffman coding algorithm. How this algorithm is used to design Huffman code for a source that takes letter from an alphabet set $A = \{a_1, a_2, a_3, a_4, a_5\}$.

Ans: Refer Q. 2.1, Page 29D, Unit-2.

f. Write short notes on the following :

i. Rice code

ii. Non binary Huffman code

Ans: Refer Q. 2. 19, Page 56D, Unit-2.

3. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. i. Where we use the dictionary techniques of encoding ? Also, explain various types of dictionary techniques.

Ans: Refer Q. 3.14, Page 77D, Unit-3.

ii. Find the real valued tag for the sequence $a_1a_2a_3a_2a_3a_1$ over letter $\{a_1, a_2, a_3\}$ with probabilities {0.2, 0.3, 0.5}.

Ans: Refer Q. 3.4, Page 66D, Unit-3.

b. i. How LZ77 algorithm works ? What are the problems with LZ77 ? Explain with an example.

Ans: Refer Q. 3.16, Page 77D, Unit-3.

ii. What is facsimile encoding ? Explain run length coding technique used earlier for facsimile.

Ans: Refer Q. 3.36, Page 103D, Unit-3.

c. i. Explain update procedure for the adaptive Huffman coding and encoding algorithm/flowchart.

Ans: Refer Q. 2.11, Page 43D, Unit-2.

ii. The joint probabilities of the transmitted and received messages of a communication system is given as :

$P(X, Y) =$

	Y_1	Y_2	Y_3	Y_4
X_1	1/4	0	1/10	0
X_2	0	1/4	0	1/20
X_3	0	0	1/10	1/20
X_4	0	1/20	0	1/10
X_5	0	0	0	1/20

Calculate $H(X)$ and $H(Y)$.

Ans. Refer Q. 1.11, Page 14D, Unit-1.

4. Attempt any **two** parts of the following : **(10 × 2 = 20)**

a. What do you understand by uniform quantizer ? How uniform quantization of a uniformly distributed source and uniform quantization of non-uniform sources is done ?

Ans. Refer Q. 4.8, Page 125D, Unit-4.

b. Discuss generic compression scheme with the help of block diagram. What are the distortion criteria for lossy coding ?

Ans. Refer Q. 4.2, Page 110D, Unit-4.

c. What is conditional entropy and mutual information and average mutual information ? For two random variables X and Y show that :

i. $H(X/Y) \leq H(X)$

ii. $I(X : Y) = I(Y : X)$.

Ans. Refer Q. 4.3, Page 111D, Unit-4.

5. **Attempt any two parts of the following : (10 × 2 = 20)**

a. Explain the steps of the Linde-Buzo-Gray algorithm.

Ans. Refer Q. 5.4, Page 139D, Unit-5.

b. What do you understand by predictive coding ? Discuss multiresolution approaches.

Ans. Refer Q. 3.35, Page 102D, Unit-3.

c. Explain the following quantization techniques in detail :

i. Structured vector quantization

ii. Pyramid vector quantization

Ans. Refer Q. 5.7, Page 143D, Unit-5.

B. Tech.

(SEM. VII) ODD SEMESTER THEORY EXAMINATION, 2014-15

DATA COMPRESSION

Time : 3 Hours **Total Marks : 100**

Note : Attempt all questions.

1. Attempt any **two** questions : **(10 × 2 = 20)**

a. What is data compression and why we need it ? Describe two applications where lossy compression technique is necessary for data compression.

Ans. Refer Q. 1.4, Page 8D, Unit-1.

b. Explain modeling and coding with the help of examples. What do you understand by prefix code ?

Ans. Refer Q. 1.23, Page 27D, Unit-1.

c. Explain different approaches for building mathematical model. Also, define two state Markov model for binary images.

Ans. Refer Q. 1.14, Page 18D, Unit-1.

2. Attempt any **two** questions : **(10 × 2 = 20)**

a. i. Differentiate between conventional Huffman coding and adaptive Huffman coding.

Ans. Refer Q. 2.10, Page 42D, Unit-2.

ii. Write short notes on the following :

x. Golomb codes

y. Rice codes

Ans. Refer Q. 2.20, Page 56D, Unit-2.

b. Draw the Huffman tree for the following symbols whose frequency occurrence in a message text is started along with their symbol below :
A : 15, B : 6, C : 7, E : 25, F : 4, G : 6, H : 10, I : 15
Decode the message 1110100010111011

Ans. Refer Q. 2.2, Page 31D, Unit-2.

c. Explain redundancy code with the help of one example.

Ans. Refer Q. 2.21, Page 56D, Unit-2.

d. What are the various application of Huffman coding and also give various steps required in encoding procedure ?

Ans. Refer Q. 2.22, Page 57D, Unit-2.

3. Attempt any **four** questions : **(5 × 4 = 20)**

a. Explain run length encoding technique with the help of suitable example.

Ans: Refer Q. 3.36, Page 103D, Unit-3.

b. Explain the JBIG standard of bi-level image compression.

Ans: Refer Q. 3.8, Page 69D, Unit-3.

c. Give LZ77 approach for adaptive dictionary based encoding.

Ans: Refer Q. 3.16, Page 77D, Unit-3.

d. Explain graphic interchange format (GIF) and where it is used.

Ans: Refer Q. 3.23, Page 88D, Unit-3.

e. Write short notes on following :

i. Compression over modems

ii. V.42 bits

Ans: Refer Q. 3.24, Page 89D, Unit-3.

f. Discuss the steps involved in basic algorithm for Prediction with Partial Match (PPM).

Ans: Refer Q. 3.27, Page 91D, Unit-3.

4. Attempt any **two** questions : **(10 × 2 = 20)**

a. What do you mean by quantization ? Describe the quantization problem with the help of an example in detail.

Ans: Refer Q. 4.7, Page 120D, Unit-4.

b. Differentiate between uniform quantization and non-uniform quantization.

Ans: Refer Q. 4.11, Page 131D, Unit-4.

c. What is rate distortion criterion ? Explain the rate distortion function for binary source and Gaussian source.

Ans: Refer Q. 4.4, Page 113D, Unit-4.

5. Attempt any two questions : **(10 × 2 = 20)**

a. What is vector quantization ? Explain procedure for vector quantization.

Ans: Refer Q. 5.1, Page 137D, Unit-5.

b. Explain the basic steps for Linde-Buzo Gray algorithm.

Ans: Refer Q. 5.4, Page 139D, Unit-5.

c. Write short notes on any two :

i. Structured vector quantization

ii. Pyramid vector quantization

iii. Advantages of scalar quantization

Ans: Refer Q. 5.8, Page 145D, Unit-5.

B. Tech.
(SEM. VII) ODD SEMESTER THEORY EXAMINATION, 2015-16
DATA COMPRESSION

Time : 3 Hours **Total Marks : 100**

Note: Attempt questions from all sections as per directions.

Section – A

1. Attempt **all** parts of this section. Answer in brief. **(2 × 10 = 20)**

a. What is data compression ?

Ans: Refer Q. 1.1, Page 5D, Unit-1.

b. Discuss dynamic Markov compression with suitable example.

Ans: Refer Q. 3.38, Page 104D, Unit-3.

c. Differentiate between lossy and lossless compression.

Ans: Refer Q. 1.3, Page 7D, Unit-1.

d. Explain the predictive coding techniques in data compression.

Ans: **The three types of predictive coding are :**

1. **Median edge detector (MED) :** Median Edge Detector (MED) belongs to the group of switching predictors that select one of the three optimal sub predictors depending on whether it found the vertical edge horizontal edge, or smooth region.
2. **Gradient adjusted predictor (GAP) :** Gradient Adjusted Predictor (GAP) is based on gradient estimation around the current pixel.
3. **Gradient estimation detector (GED) :** The number of contextual pixel is also a compromise between the MED and GAP predictors, and in contrast to the GAP predictor, which has three predefined threshold, the proposed predictor is based on one threshold.

e. Differentiate between LZ77 and LZ78 data compression techniques.

Ans: Refer Q. 3.16, Page 77D and Q. 3.17, Page 81D; Unit-3.

f. Write advantages of vector quantization over scalar quantization.

Ans: Refer Q. 5.3, Page 139D, Unit-5.

g. How the Linde-Buzo-Gray algorithm works ?

Ans: Refer Q. 5.4, Page 139D, Unit-5.

h. Compare and contrast JPEG and MPEG.

Ans: Both, JPEG and MPEG are two different types of compressing formats. The main difference between the two is that JPEG is mainly used for image compression, while MPEG has various standards for audio and video compression. JPEG stands for Joint Photographic Expert Group. The file name for a JPEG image is .jpg or .jpeg. MPEG, on the other hand, stands for the Moving Picture Experts Group. It is a working group of experts that was formed in 1988 by ISO and IEC.

i. What benefits are offered by compression schemes in designing systems ?

Ans. 1. Less disk space (more data in reality).
2. Faster writing and reading. 3. Faster file transfer.
4. Variable dynamic range. 5. White order independent.

j. What are the advantages of using specialized multimedia servers ?

Ans. 1. Better user experience 2. Scalability
3. Content protection 4. Higher bandwidth
5. Uploading and storing 6. Options for delivery

Section – B

Attempt any **five** questions. **(10 × 5 = 50)**

2. Why we need data compression ? Explain compression and reconstruction with the help of block diagram.

Ans. Refer Q. 1.1, Page 5D, Unit-1.

3. Differentiate between static length and variable length coding schemes. Explain with the help of examples.

Ans. Refer Q. 1.12, Page 15D, Unit-1.

4. Based upon the requirements of reconstruction, how data compression techniques are broadly classified. Explain these classifications in brief.

Ans. Refer Q. 1.3, Page 7D, Unit-1.

5. What are the measures of performance of data compression algorithms ?

Ans. Refer Q. 1.5, Page 9D, Unit-1.

6. What is average information ? What are the properties used in measurement of average information ?

Ans. Refer Q. 1.13, Page 17D, Unit-1.

7. Discuss generic compression scheme with the help of block diagram. What are the distortion criteria for lossy coding ?

Ans. Refer Q. 4.2, Page 110D, Unit-4.

8. Explain uniform and non-uniform quantization with further classifications.

Ans. Refer Q. 4.11, Page 131D, Unit-4.

9. Explain the procedure for the adaptive Huffman coding and encoding algorithm with the help of flowchart.

Ans. Refer Q. 2.12, Page 45D, Unit-2.

Section – C

Attempt any **two** questions : **(15 × 2 = 30)**

10. What do you understand by Markov model ? Discuss the role of Markov models in text compression.

Ans. Refer Q. 1.14, Page 18D and Q. 1.15, Page 21D; Unit-1

11. Explain minimum variance Huffman code and encoding procedure with suitable example.

Ans. Refer Q. 2.24, Page 58D, Unit-2.

12. What is quantization ? Explain additive noise of a quantizer. Differentiate between scalar quantization and vector quantization. What are the advantages of vector quantization over scalar quantization ?

Ans. Refer Q. 5.5, Page 140D, Unit-5.

B.Tech.
(SEM. VIII) EVEN SEMESTER THEORY EXAMINATION, 2016-17
DATA COMPRESSION

Time : 3 Hours **Max. Marks : 100**

Note : Be precise in your answer. In case of numerical problem assume data wherever not provided.

SECTION-A

1. Explain **all** parts : **(2 × 10 = 20)**

a. What do you understand by entropy ?

Ans. Refer Q. 1.16, Page 148D, Unit-1, Two Marks Questions.

b. What do you mean by lossless compression ?

Ans. Refer Q. 1.3, Page 147D, Unit-1, Two Marks Questions.

c. Define data compression.

Ans. Refer Q. 1.1, Page 147D, Unit-1, Two Marks Questions.

d. Define compression ratio.

Ans. Refer Q. 1.9, Page 148D, Unit-1, Two Marks Questions.

e. Differentiate between fidelity and quality.

Ans.

S. No.	Fidelity	Quality
1.	Fidelity means accuracy or exact correspondence to some given quality or fact.	Quality means the level of excellence.
2.	It is the quality of faithfulness or loyalty.	It is the standard of something as measured against other things of a similar kind.

f. Discuss binary code.

Ans. A binary code is a way of representing text or computer instructions by the use of the binary number system 0 and 1 by assigning a bit string to each particular symbol or instruction.

g. Discuss Huffman code.

Ans: Refer Q. 2.1, Page 150D, Unit-2, Two Marks Questions.

h. Define distortion.

Ans: Distortion is defined as the measure of difference between the actual source samples $\{x_k\}$ and the corresponding quantized value $\{\hat{x}_k\}$.

i. Define the term PPM.

Ans: Refer Q. 3.14, Page 155D, Unit-3, Two Marks Questions.

j. Discuss Golomb coding.

Ans: Refer Q. 2.9, Page 151D, Unit-2, Two Marks Questions.

SECTION-B

2. Attempt any **five** of the following questions : **(10 × 5 = 50)**

a. Explain rice coding and its implementation.

Ans: Refer Q. 2.15, Page 50D, Unit-2.

b. Explain minimum variance Huffman code.

Ans: Refer Q. 2.4, Page 34D, Unit-2.

c. Explain encoding and decoding in LZW algorithm.

Ans: Refer Q. 3.20, Page 84D, Unit-3.

d. Explain adaptive quantization.

Ans: Refer Q. 4.9, Page 129D, Unit-4.

e. Explain prediction with partial match.

Ans: Refer Q. 3.26, Page 91D, Unit-3.

f. Explain scalar and vector quantization.

Ans: **Scalar quantization :**

1. In scalar quantization, each sample is quantized independently.
2. The set of inputs and outputs of a quantizer in scalar quantization are scalars.
3. The scalar quantization results in higher distortion for a given rate (in bits per sample).
4. The quantizer used in the quantization consists of two mappings : an encoder mapping and a decoder mapping.

Vector quantization : Refer Q. 5.1, Page 137D, Unit-5.

g. Explain modeling and coding with the help of example. What do you understand by prefix code ?

Ans: Refer Q. 1.21, Page 24D, Unit-1.

h. What are two observations on which Huffman procedure is based regarding optimum prefix code ? What are the various applications of Huffman coding ?

Ans: Two observations on which Huffman procedure is based regarding optimum prefix code are :

1. In an optimum code, symbols that occur more frequently (have a higher probability of occurrence) will have shorter codewords than symbols that occur less frequently.
2. In an optimum code, the two symbols that occur least frequently will have the same length.

Applications of Huffman coding : Refer Q. 2.22, Page 57D, Unit-2.

SECTION-C

Attempt any **two** of the following questions : **(15 × 2 = 30)**

3. What do you understand by adaptive quantization ? Explain the various approaches to adapting the quantizer parameters ?

Ans: Refer Q. 4.9, Page 129D, Unit-4.

4. What is facsimile encoding ? Explain run-length coding technique used earlier for facsimile.

Ans. Refer Q. 3.36, Page 103D, Unit-3.

5. What do you understand by uniform quantizer ? How uniform quantization of uniformly distributed sources and uniform quantization of non-uniform sources is done ?

Ans. Refer Q. 4.8, Page 125D, Unit-4.

B.Tech.
(SEM. VIII) EVEN SEMESTER THEORY EXAMINATION, 2017-18
DATA COMPRESSION

Time : 3 Hours **Max. Marks : 100**

Note : 1. Attempt **all** Sections. If require any missing data; then choose suitably.

SECTION-A

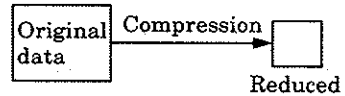
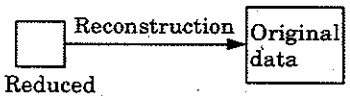
1. Attempt **all** questions in brief. **(2 × 10 = 20)**

a. Define data compression and why we need it.

Ans. **Data compression :** Refer Q. 1.1, Page 147D, Unit-1, Two Marks Questions.
Need : Refer 1.2, Page 147D, Unit-1, Two Marks Questions.

b. Differentiate between compression and reconstruction.

Ans.

S. No.	Compression	Reconstruction
1.	It reduces the original data into compressed form.	It reconstructs the compressed data into original form.
2.		

c. What are the limitations of Huffman coding ?

Ans. Refer Q. 2.3, Page 33D, Unit-2.

d. Write down the application of Huffman Coding in text compression and audio compression.

Ans. Refer Q. 2.22, Page 57D, Unit-2.

e. What do you mean by Binary Code ? Compare Binary Code with Huffman Code.

Ans. Refer Q. 3.7, Page 69D, Unit-3.

f. Define Graphic Interchange Format.

Ans. Refer Q. 3.18, Page 156D, Unit-3, Two Marks Questions.

g. What is rate distortion criterion ?

Ans. Refer Q. 4.1, Page 158D, Unit-4, Two Marks Questions.

h. Differentiate between uniform and non-uniform quantization.

Ans:

S. No.	Uniform quantization	Non-uniform quantization
1.	It is a type of quantization in which the quantization levels are uniformly spaced is termed.	It is a type of quantization in which quantization levels are unequal and mostly the relation between them is logarithmic, is termed as non uniform quantization.
2.	It has some amount of quantization errors.	It reduces quantization error.

i. What is predictive coding ?

Ans: Refer Q. 3.25, Page 90D, Unit-3.

j. Write down the merits and demerits of vector quantization.

Ans: **Merits of vector quantization :**

1. Vector Quantization can lower the average distortion D with number of reconstruction levels held constant.

Demerits of vector quantization :

1. Inevitable loss in rate-distortion performance.

SECTION-B

2. Attempt any **three** parts of the following : **(10 × 3 = 30)**

a. What do you understand by information ? Give an alphabet $A = \{a1, a2, a3, a4, a5\}$, find the first order entropy of the following :
$P(a1) = 1/2, P(a2) = 1/4, P(a3) = P(a4) = 1/8, P(a5) = 1/2$.

Ans: **Information :** Refer Q. 1.9, Page 13D, Unit-1.
Numerical : First-order entropy is given by,

$$h = -\left[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{2}\log_2\frac{1}{2}\right]$$

$$= \frac{1}{2} + \frac{1}{4} + \frac{3}{8} + \frac{3}{8} + \frac{1}{2} = \frac{4+2+3+3+4}{8} = \frac{16}{8} = 2 \text{ bits.}$$

b. For an alphabet $A = \{a1, a2, a3, a4, a5\}$ with probabilities $P(a1) = 0.15$, $P(a2) = 0.04$, $P(a3) = 0.26$, $P(a4) = 0.05$ and $P(a5) = 0.50$

i. Calculate the entropy of this source

ii. Find a Huffman Code for this source

iii. Find the average length of the code

Ans: Refer Q. 2.17, Page 54D, Unit-2.

c. What is the basic difference between adaptive and statistical compression scheme ? Discuss with the model of adaptive compression.

Ans: Refer Q. 3.11, Page 73D, Unit-3.

d. Discuss the steps involved in basic algorithm for Prediction with Partial Match (PPM).

Ans: Refer Q. 3.27, Page 91D, Unit-3.

e. What is vector quantization ? Explain procedure for vector quantization.

Ans: Refer Q. 5.1, Page 137D, Unit-5.

SECTION-C

3. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Explain physical, probability, Markov and composite source model in detail.

Ans: Refer Q. 1.14, Page 18D, Unit-1.

b. Determine whether the following codes are uniquely decodable or not :

i. {0, 01, 11, 111} **ii. {0, 01, 110, 111}**

iii. {1, 10, 110, 111} **iv. {0, 01, 10}**

Ans:

i. Refer Q. 1.20, Page 24D, Unit-1.
ii. Refer Q. 1.21, Page 24D, Unit-1.
iii. Refer Q. 1.21, Page 24D, Unit-1.
iv. In the list {0, 01, 10}, the codeword 0 is prefix to 01 with dangling suffix 1, 1 is prefix to 10 with dangling suffix 0. Now, add all the suffix to the list *i.e.*, {0, 01, 10, 0, 1}. Here codeword 0 is already present in the list, so, it is not a uniquely decodable.

4. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Design 3-bit Tunstall code for a memory less source with the following alphabet :
$S = \{A, B, C\}$ with their $P(A) = 0.6$, $P(B) = 0.3$, $P(C) = 0.1$

Ans: For 3-bit Tunstall code, the maximum number of codewords in the codebooks is

$2^3 = 8$

Symbol	Initial Probability
A	0.6
B	0.3
C	0.1

Taking symbol with highest probability (*i.e.*, A) and concatenate it with every other symbol and remove A from list, we get :

Sequence	Probability
B	0.3
C	0.1
AA	0.36
AB	0.18
AC	0.06

As number of codewords is less than maximum value, again apply the same procedure with AA (maximum probability).

Sequence	Probability
B	0.3
C	0.1
AB	0.18
AC	0.06
AAB	0.108
AAC	0.036
AAA	0.216

We have to stop here as if we apply one more iteration, the number of codeword increase the maximum limit. Thus the sequence will become :

Sequence	Codeword
B	000
C	001
AB	010
AC	011
AAB	100
AAC	101
AAA	110

b. Design Golomb code for $m = 5$ and $n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10$.

Ans. Refer Q. 2.14, Page 49D, Unit-2.

5. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. What is Facsimile encoding ? Explain run-length coding technique used earlier for Facsimile. Give a brief comparison of MH, M& MMR and JBIG.

Ans. Refer Q. 3.37, Page 104D, Unit-3.

b. Explain the JBIG standard of bi-level image compression.

Ans. Refer Q. 3.8, Page 69D, Unit-3.

6. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. What do you understand by adaptive quantization ? Explain the various approaches to adapting the quantizer parameters.

Ans. Refer Q. 4.9, Page 129D, Unit-4.

b. What is lossy data encoding ? Write down the distortion measure criteria's to check the fidelity of a reconstructed source sequence to the original one in such type of encoding techniques.

Ans. Refer Q. 4.1, Page 108D, Unit-4.

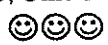
7. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Explain the steps of the Linde-Buzo-Gray algorithm.

Ans. Refer Q. 5.4, Page 139D, Unit-5.

b. What is Quantization ? Explain additive noise model of a quantizer.

Ans. Refer Q. 5.5, Page 140D, Unit-5.



B.Tech.
(SEM. VIII) EVEN SEMESTER THEORY EXAMINATION, 2018-19
DATA COMPRESSION

Time : 3 Hours **Max. Marks : 100**

Note : Attempt **all** Section. If require any missing data, then choose suitably.

SECTION-A

1. Attempt **all** questions in brief : **(2 × 10 = 20)**

a. Is Huffman coding is a lossless or lossy compression ? Write applications of Huffman coding.

Ans. Huffman coding is a lossless compression.
Application of Huffman coding : Refer Q. 2.10, Page 151D, Unit-2, Two Marks Questions.

b. What is a composite source model ?

Ans. Refer Q. 1.13, Page 148D, Unit-1, Two Marks Questions.

c. What are prefix codes ?

Ans. Refer Q. 1.21, Page 24D, Unit-1.

d. Explain JBIG standard.

Ans. Refer Q. 3.11, Page 155D, Unit-3, Two Marks Questions.

e. Explain entropy.

Ans. Refer Q. 1.16, Page 148D, Unit-1, Two Marks Questions.

f. Define compression ratio.

Ans. Refer Q. 1.9, Page 148D, Unit-1, Two Marks Questions.

g. Determine whether the code {0, 10, 110, 111} is uniquely decodable or not.

Ans. This code is uniquely decodable because there is no dangling suffix.

h. Which compression technique is used in "compress" command of Unix operating systems ?

Ans. The Lempel – Ziv algorithm is used to compress command of Unix operating system.

i. Explain uniform quantizer.

Ans: Refer Q. 4.8, Page 125D, Unit-4.

j. What is entropy coded quantization ?

Ans: The anticipating efficient source coding of the quantized outputs, to minimize the MSE for a given entropy rather than a given number of representation points. This approach is called entropy coded quantization and is almost implicit in the layered approach to source coding.

SECTION-B

2. Attempt any **three** of the following : **(10 × 3 = 30)**

a. What are the advantages of vector quantization over scalar quantization ? Explain with the help of an example.

Ans: Refer Q. 5.3, Page 139D, Unit-5.
Example : Refer Q. 5.1, Page 137D, Unit-5.

b. What is data compression ? Why we need it ? Explain compression and reconstruction with the help of block diagram.

Ans: Refer Q. 1.1, Page 5D, Unit-1.

c. Write short note on Golomb codes and Tunstall codes.

Ans: Refer Q. 2.16, Page 53D, Unit-2.

d. What do you mean by quantization ? Describe the quantization problem with the help of an example in detail.

Ans: Refer Q. 4.7, Page 120D, Unit-4.

e. Explain various types of dictionary based coding techniques.

Ans: Refer Q. 3.12, Page 75D, Unit-3.

SECTION-C

3. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. What do you mean by lossless compression and lossy compression ? Compare lossless compression with lossy compression.

Ans: Refer Q. 1.3, Page 7D, Unit-1.

b. What do you understand by information ? Give an alphabet A = {a, a2, a3, a4}, find the first order entropy of the following : P(a1) = 1/2, P(a2) = 1/4, P(a3) = P(a4) = 1/8.

Ans: Refer Q. 1.12, Page 15D, Unit-1.

4. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Given the eight symbols A, B, C, D, E, F, G, and H with probabilities 1/30, 1/30, 1/30, 2/30, 3/30, 5/30, 5/30, and 12/30 :

i. Draw the Huffman tree for these symbols.

ii. Compute the average no. of bits / symbol.

Ans:

i.

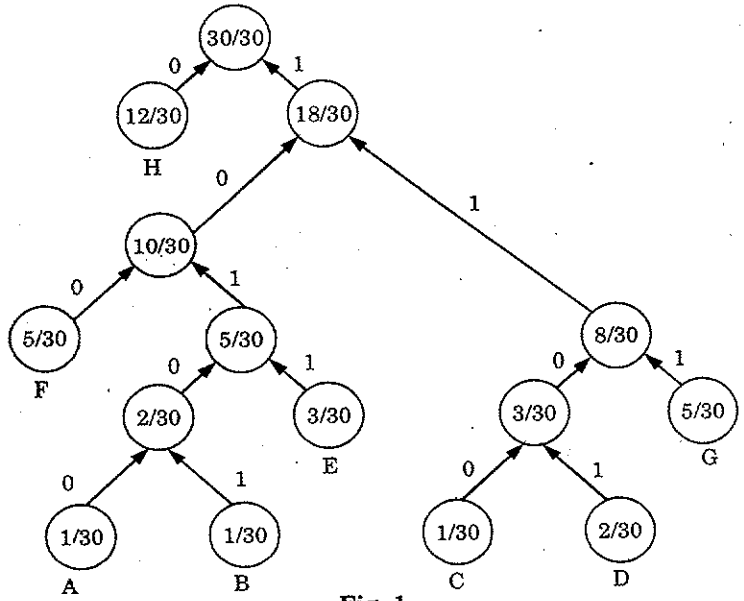


Fig. 1.

A : 10100 = 5 bits
B : 10101 = 5 bits
C : 1100 = 4 bits
D : 1101 = 4 bits
E : 1011 = 4 bits
F : 100 = 3 bits
G : 111 = 3 bits
H : 0 = 1 bit

ii. Average Huffman code size = $\left(5 \times \frac{1}{30}\right) + \left(5 \times \frac{1}{30}\right) + \left(4 \times \frac{1}{30}\right) +$

$\left(4 \times \frac{2}{30}\right) + \left(4 \times \frac{3}{30}\right) + \left(3 \times \frac{5}{30}\right) + \left(3 \times \frac{5}{30}\right) + \left(1 \times \frac{12}{30}\right) = \frac{76}{30}$

b. Differentiate between adaptive Huffman coding and Huffman coding ?

Ans: Refer Q. 2.10, Page 42D, Unit-2.

5. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Compare and contrast LZ77 and LZ78 with examples.

Ans:

S. No.	LZ77	LZ78
1.	The LZ77 algorithm works on past data.	The LZ78 algorithm attempts to work on future data.
2.	The LZ77 is slower.	LZ78 is faster than LZ77.
3.	The output format of LZ77 is triplet < 0, l, c> where 0 = offset, l = length of the match, c = next symbol to be encoded.	The output format of LZ78 is pair < i, c> where i = index and c = next character.
4.	This algorithm is open source and used in ZIP, and by the formats PNG, TIFF, PDF and many others, LZ77 is used in gzip, Squeeze, LHA, PKZIP, and ZOO.	LZ78 has various applications in the field of information theory such as random number generation, hypothesis testing parsing of the string etc. LZ78 is used in compress, GIF, CCITT (modems), ARC, PAK.

b. Discuss the steps involved in basic algorithm for prediction with partial match (PPM).

Ans: Refer Q. 3.27, Page 91D, Unit-3.

6. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Explain the various distortion criteria used in lossless schemes.

Ans: Refer Q. 4.1, Page 108D, Unit-4.

b. Differentiate between uniform and non uniform quantization.

Ans: Refer Q. 4.11, Page 131D, Unit-4.

7. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Differentiate between scalar quantization and vector quantization.

Ans: Refer Q. 5.5, Page 140D, Unit-5.

b. Explain the steps of Linde-Buzo-Gray algorithm.

Ans: Refer Q. 5.4, Page 139D, Unit-5.