Experiment no: 01

Name of the experiment: To implement Huffman code using symbols with their corresponding probabilities.

Objectives:

i) To learn about Huffman coding.

ii) To understand the implementation of Huffman code

Theory:

Huffman coding is an efficient method of compressing data without losing information. It is based on the concept of encoding characters with variable-length codes, where more frequently occuring characters are assigned shorter codes and less frequently occuring characters are assigned longer codes.

Algorithm steps:

1) The source symbols are arranged in the order of decreasing probability. Then the two symbols of lowest probability are assigned '0' and '1'.

2) Then combine last two symbols and move the combined symbol as high as possible.

3) Repeat the above steps until we are left with the final list of source symbols of only two for which '0' and '1' are assigned.

4) Code for each symbol is found by moving backward.

Example:

Consider a 5 symbol source with the following probability

$P(x_1) = 0.3$, $P(x_2) = 0.15$, $P(x_3) = 0.25$, $P(x_4) = 0.05$, $P(x_5) = 0.25$

Soln:

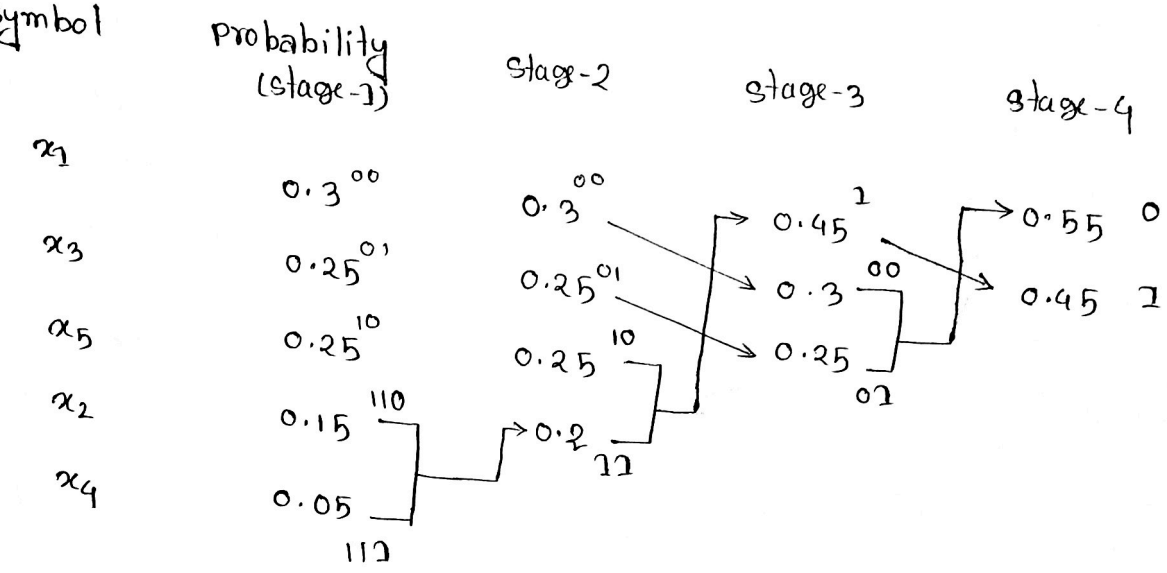


Table-1 The binary Huffman code is:

Symbol	P(Si)	Huffman code
x_1	0.3	00
x_2	0.15	110
x_3	0.25	01
x_4	0.05	111
x_5	0.25	10

Experiment no: 02

~~Problem name~~

Name of the experiment: To simulate convolutional coding based on their encoder structure.

Objectives:

1) To learn about convolutional coding.
2) To understand the implementation of convolutional coding.

Theory:

In convolutional codes, the message comprises of data streams of arbitary length and a sequence of output bits are generated by the sliding application of Boolean functions to the data stream.

Encoding by a convolutional codes:

For generating a convolutional code, the information is passed sequentially through a linear finite-state shift register. The shift register comprises of (bit) stages and Boolean function generators.

A convolutional code can be represented as (n, k, K) where —

1) 'k' is the number of bits shifted into the encoder at one time. Generally, k=1

2) n is the number of encoder output bits corresponding to k information bits.

3) The code-rate, Rc = k/n

4) The encoder memory, a shift register of size k, is the constraint length.

5) n is a function of the present input bits and the contents of k.

6) The state of the encoder is given by the value of (k-1) bits.

Example:

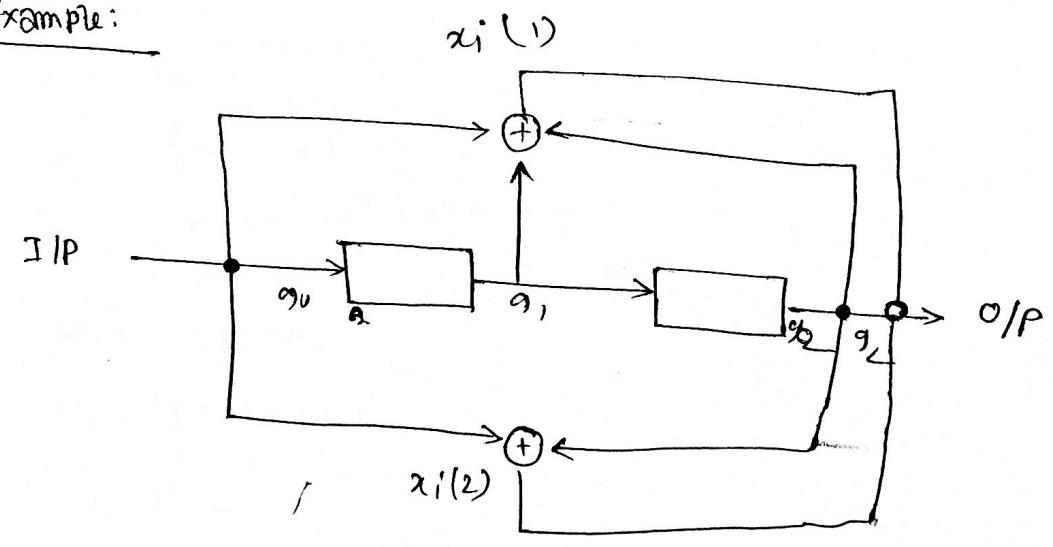


figure- Convolutional encoder

Here input :

$g_1 = 1, 1, 1$

$g_2 = 1, 0, 1$

$m = 1, 0, 0, 1, 1$

and output is v_1 and v_2

Now,

$m(x) = x^4 + x + 1$

$g_1(x) = x^2 + x + 1$

$g_2(x) = x^2 + 1$

$v_1(x) = g_1(x)\, m(x)$

$= (x^2 + x + 1)(x^4 + x + 1)$

$= x^6 + x^3 + x^2 + x^5 + x^2 + x + x^4 + x + 1$

$= x^6 + x^5 + x^4 + x^3 + 1$

$v_2(x) = g_2(x)\, m(x)$

$= (x^2 + 1)(x^4 + x + 1)$

$= x^6 + x^3 + x^2 + x^4 + x + 1$

$v_1 = 1111001$

$v_2 = 1011111$

$c = v_1 v_2$

$= 11110111010111$

Experiment no: 03

Name of the Experiment: Explain and in to write a program to implement lempel-ziv code.

Objectives:

1) To learn about lempel-ziv code.
2) To understand the implemention of Lempel-ziv code using MATLAB.

Theory:

There are two categories of compression techniques. They are

1) Lossy
2) Lossless

Lossless compression reduces bits by identifying and eliminating statistical redundancy.

On the other hand lossy compression reduces bits by removing unnecessary or less important information.

Lembel-ziv algorithm is a very common compression technique. It is lossless, meaning no data is lost when compressing. This algorithm is simple to implement and has the potential for very high throughput in hardware implementations.

Lempel-Ziv algorithm is accomplished by parsing the source data stream into segments that are the shortest subsequences not encountered previously.

Let us assumed an input binary sequence as follows:

000101111001010 0101

It is assumed that 0 and 1 are already stored

Now,

Numerical Position:	1	2	3	4	5	6	7	8	9
Subsequence :	0	1	00	01	011	10	010	100	101
Numerical Representation:			11	12	42	21	41	61	62
Binary Encoded Blocks:			0010	0011	1001	0100	1000	1100	1101

Experiment no: 04

Name of the Experiment: To implement Hamming code.

Objectives:

1) To learn about linear block code.

2) To learn about Hamming code for error correction and detection.

3) To understand the implementation of Hamming code.

Theory:

Hamming code are Linear Block codes (n, k) Hamming code for $m \geq 3$ is defined by the following equations:

1) Block length, $n = 2^m - 1$

2) Number of message bits: $k = 2^m - m - 1$

3) Number of parity bits: $(n-k) = m$

4) Minimum distance, $d_{min} = 3$

5) Efficiency $= \frac{k}{n} = \frac{2^m - m - 1}{2^m - 1}$

$$= \frac{2^m - 1}{2^m - 1} - \frac{m}{2^m - 1}$$

$$= 1 - \frac{m}{2^m - 1}$$

Hamming code is a set of error-correcting codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver.

Redundant bits:

Redundant bits are extra ~~bitsary~~ binary bits that are generated and added to the information. The number of redundant bits can be calculated using the following formula:

$2^r \geq m + r + 1$ where r = redundant bit and

m = data bit

Let us consider, the number of data bits are 7. Then the number of redundant bits can be calculated using:

$2^4 \geq 7 + 4 + 1$

Thus the number of redundant bits = 4 parity bits.

A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1's in the data is even or odd. They are used for error detection.

There are two types of parity bits:

1. Even parity bit: In case of even parity, the number of 1's are counted for a given set of bits. If that count is odd, the parity bit value is set to 1, else the parity bit is zero.

2. Odd parity bit: for an odd parity ~~bit~~, ~~bi~~ if the number of 1's are counted is even, then the parity bit value is set to 1 else the parity bit's value is 0.

Hamming code structure:

Parity bits are inserted in between data bits. Commonly 7-bits Hamming code is used.

D_7	P_6	D_5	P_4	D_3	P_2	P_1

here, P_1, P_2 and P_4 are parity bits and D_7, D_6, D_5 and D_3 are Data bits

Selecting parity bits:

1) $P_1 \rightarrow 1,3,5,7$

2) $P_2 \rightarrow 2,3,6,7$

3) $P_4 \rightarrow 4,5,6,7$

Construction of G_i and H:

The matrix $G_i = (I_k | P^T)$ is called a generator matrix of a linear (n, k) code,

$H_i(P^T | I_{n-k})$ is called a parity-check matrix

Here, I_k = Identity matrix

P = Parity matrix

Let us consider a $(7, 4)$ Hamming code.

here, code vector length $n = 7$

And message length $k = 4$

$\therefore$ Number of parity bits $= n - k$

$= 7 - 4 = 3$

Let the parity matrix be, $P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

Then the Generator matrix will be,

$G_i = (I_k | P)$ here $k = 4$

$\therefore G_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

Now, $P^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$

Then the parity-check matrix ~~is~~ will be,

$$H_i = (P^T \mid I_{n-k}) \qquad I = 3$$

$$= \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Let menage, $m = 1011$

And code vector is defined as mG

$$= \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Finding of correct code word

Now let ~~the~~ a (7,4) block code generate by

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & : & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & : & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & : & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & : & 1 & 1 & 1 \end{bmatrix}$$

And Received code word is 1111010

The Syndrome vector $S = RH^T$

Here $H = [P^T : I_3]$

$$= \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Now, $H^T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Now, $S = [1\ 1\ 1\ 1\ 0\ 1\ 0] \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$= [1\ 0\ 1]$

S matches with 3rd row of H^T. Hence error is in 3rd bit.

$R = 1111010$

Error vector $e = 0010000$

Now correct codeword $C = P \oplus e$.

R = 1 1 1 1 0 1 0

e = 0 0 1 0 0 0 0

EOR - 1 1 0 1 0 1 0 = C

This will be correct code word

Source code :

```
clc; clear all; close all;
M = input('Enter 4 bit message :');
d1 = [1 0 0 0];
d2 = [0 1 0 0];
d3 = [0 0 1 0];
d4 = [0 0 0 1];
P1 = d2' + d3' + d4';
P2 = d1' + d3' + d4';
P3 = d1' + d2' + d4';
G1 = [P1 P2 P3 d1' d2' d3' d4']
P = [P1, P2, P3];
H = [eye([3 3]) P'];
~~Code_word = mod~~
Code_word = mod((m * G), 2)
E = mod((H * Code_word'), 2)
ERC = input('Give an erroneous received hamming code word:');
```

Experiment no: 05

Name of the Experiment: A binary symmetric channel has the following noise matrix with probability,

$$P(Y|X) = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix},$$

Now find the channel capacity C

Objectives:

1) To understand binary symmetric channel
2) To learn how to find channel capacity

Theory:

A symmetric channel that has two inputs and two outputs is called Binary symmetric channel.

In this model, a transmitter wishes to send a bit and the receiver will receive a bit. The bit will be "flipped" with a "cross over probability" of P and otherwise is received correctly.

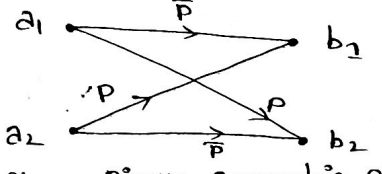


figure: Binary symmetric channel

Here a_1 and a_2 are transmitted signals and b_1, b_2 are Received signal

P and $\bar{P}$ are tranition probabilities

$$P(B|A) = \begin{matrix} a_1 \\ a_2 \end{matrix} \begin{bmatrix} b_1 & b_2 \\ \bar{P} & P \\ P & \bar{P} \end{bmatrix}$$

$$P + \bar{P} = 1$$

$$\bar{P} = 1 - P$$

As in symmetric channel,

$$C = \log_2 S - h$$

$$h = \sum_{i=1}^{S} P_i \log_2 \left(\frac{1}{P_i}\right)$$

$$= \bar{P} \log_2 \frac{1}{\bar{P}} + P \log_2 \frac{1}{P}$$

here, $S = 2$

$$C = \log_2 2 - \left(\bar{P} \log_2 \frac{1}{\bar{P}} + P \log_2 \frac{1}{P}\right)$$

$$= 1 - \left(\bar{P} \log_2 \frac{1}{\bar{P}} + P \log_2 \frac{1}{P}\right)$$

$$= 1 - h$$

here C is the channel capacity

and h is the conditional probability

Experiment no: 06

Name of the Experiment: write a program to check the optimality of Huffman code.

Objectives:

1) To learn about Huffman coding.

2) To checks optimality of Huffman coding.

Theory:

To check the optimality of a Huffman code, we can use the kraft inequality.

Uniquely Decoded code satisfies kraft inequality. It is defined as,

$$\sum_{i=1}^{m} 2^{-n_i} \leq 1$$

here, n_i is the no. of bits in a code word

m is no. of code words in a code

for example let

	Code A
m_1	00
m_2	01
m_3	10
m_4	11

$\sum_{i=1}^{4} 2^{-m_i} = 2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = 4 \cdot \frac{1}{2^2} = 1$

$\therefore$ optimal

	Code B
m_1	0
m_2	10
m_3	11
m_4	110

$\sum_{i=1}^{4} 2^{-m_i} = 2^{-1} + 2^{-2} + 2^{-2} + 2^{-3}$

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

$= 9/8 > 1$ so, no optimal

Example:

Consider a 5 symbol source with the following probability

'a'	'b'	'c'	'd'	'e'
25	25	20	15	15

Then Huffman codeg.

a	1	0	
b	0	1	
c	1	1	
d	0	0	1
e	0	0	0

- Inequality $= \sum_{i=1}^{5} 2^{-mi}$

$= 2^{-2} + 2^{-2} + 2^{-2} + 2^{-3} + 2^{-3}$

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

$= \frac{3}{4} + \frac{2}{4}$

$= \frac{3}{4} + \frac{1}{4}$

$= 1 \leq 1$

So the code is optimal

Another way to check the optimality of Huffman code is to verify that the code is prefix-free, meaning no codeword is a prefix of any other codeword. A prefix-free code is always optimal.

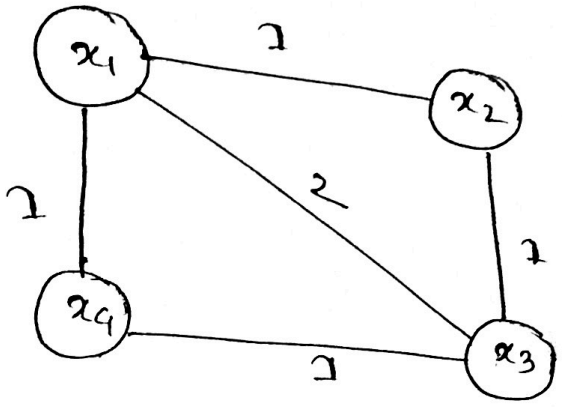
A third way to check the optimality of Huffman code is by comparing the average length of the Huffman code with the entropy of the source. The Huffman code is optimal if its average length is equal to the entropy of the source.

Source code:

```
clc; clear all; close all;
x = {'a', 'b', 'c', 'd', 'e'};
disp('symbols:')
disp(x)
frequency = [25 25 20 15 15];
disp('corresponding frequency of symbols:')
disp(frequency)
if length(frequency) == length(x)
    disp('Equal length is expected.')
else
```

Experiment no: 07

Name of the Experiment: To find the entropy rate of a random walk on the following weighted graph.



Objectives:

1) To learn about entropy rate of a random walk.
2) To understand the implementation of entropy rate of a random walk on a weighted graph.

Theory:

Here, in this example,

$x_i = x_1, x_2, x_3, x_4$

and $\omega_i = \omega_1, \omega_2, \omega_3, \omega_4$

here ω_i is the weight of the edges

we know, $\omega_i = \sum_j \omega_{ij}$

$\therefore \omega_1 = 1 + 1 + 2 = 4$

$\omega_2 = 1 + 1 = 2$

$\omega_3 = 1+1+2 = 4$

$\omega_4 = 1+1 = 2$

Again we know,

$$\sum_i \omega_i = 2\omega$$

$$\Rightarrow \omega = \frac{\sum \omega_i}{2}$$

$$= \frac{\omega_1+\omega_2+\omega_3+\omega_4}{2}$$

$$= \frac{4+2+4+2}{2}$$

$$= 6$$

Then the stationary distribution is,

$$\mu_i = \frac{\omega_i}{2\omega}$$

$$= \frac{\omega_1 \cdot \omega_2 \cdot \omega_3 \cdot \omega_4}{2\omega}$$

$$= \frac{4 \cdot 2 \cdot 4 \cdot 2}{2 \times 6_3}$$

$$= .$$

$$\mu_1 = \frac{\omega_1}{2\omega} = \frac{4}{2\times 6} = 0.333$$

$$\mu_2 = \frac{\omega_2}{2\omega} = \frac{2}{12} = 0.167$$

$$\mu_3 = \frac{\omega_3}{2\omega} = \frac{4}{12} = 0.333$$

$$\mu_4 = \frac{\omega_4}{2\omega} = \frac{2}{12} = 0.167$$

... are possible.

$\frac{\omega_i}{2\omega} = M_i$

$H\left(\frac{\omega_i}{2\omega}\right) \Rightarrow \frac{\omega_i}{2\omega} = (0.333, 0.167, 0.333, 0.167)$

$H\left(\frac{\omega_i}{2\omega}\right) = -\left[0.333\log_2 0.333 + 0.167\log_2 0.167 + 0.333\log_2 0.333 + 0.167\log_2 0.167\right]$

$= -\left[-0.528 - 0.431 - 0.528 - 0.431\right]$

$= 1.918$

$\frac{\omega_{ij}}{2\omega} = \left[\left(\frac{0}{2\times6}, \frac{1}{2\times6}, \frac{2}{2\times6}, \frac{1}{2\times6}\right), \left(\frac{1}{2\times6}, \frac{0}{2\times6}, \frac{1}{2\times6}, \frac{0}{2\times6}\right), \left(\frac{2}{2\times6}, \frac{1}{2\times6}, \frac{0}{2\times6}, \frac{1}{2\times6}\right), \left(\frac{1}{2\times6}, \frac{0}{2\times6}, \frac{1}{2\times6}, \frac{0}{2\times6}\right)\right]$

$H\left(\frac{\omega_{ij}}{2\omega}\right) = (0, 0.083, 0.167, 0.083), (0.083, 0, 0.083, 0), (0.167, 0.083, 0, 0.083), (0.083, 0, 0.083, 0)$

$= -\left[0.083\log_2 0.083 + 0.167\log_2 0.167 + 0.083\log_2 0.25 + 0.083\log_2 0.083 + 0.083\log_2 0.083 + 0.167\log_2 0.167 + 0.083\log_2 0.083 + 0.083\log_2 0.083 + 0.083\log_2 0.083 + 0.083\log_2 0.083\right]$

$= -\left[-0.298 - 0.431 - 0.298 - 0.298 - 0.298 - 0.431 - 0.298 - 0.298 - 0.298 - 0.298\right]$

$= 3.425$

Entropy rate,

$$H(x) = H\left(\frac{\omega_{ij}}{2\omega}\right) - H\left(\frac{\omega_i}{2\omega}\right)$$

$$= 3.25 - 1.918$$

$$= 1.932$$

Source code:

```
a = [0 1 2 1 : 1 0 1 0 : 2 1 0 1 : 1 0 1 0];
Sum=0; w1=0; w2=0; w3=0; w4=0;
for i = 1:4
    w1 = w1 + a(1,i);
    w2 = w2 + a(2,i);
    w3 = w3 + a(3,i);
    w4 = w4 + a(4,i);
    for j = 1:4
        Sum = Sum + a(i,j);
    end
end
```