Module-3
Probability Based Source Coding

Dr. Markkandan S

School of Electronics Engineering (SENSE)
Vellore Institute of Technology
Chennai

Outline

Source Coding Theorem

Introduction

Source Coding: Efficient Representation of symbols generated by a source.

1. The Primary motivation is to compress the data by efficient representation of symbols
2. A code is a set of vectors called code words
3. A discrete memoryless source (DMS) outputs a symbol selected from a finite set of symbols $x_i = 1, 2, \ldots, L$ The number of binary digits (bits) R required for unique coding, when L is a power of 2 is $R = log_2 L$
4. When L is not a power of 2, $R = \lfloor log_2 L \rfloor + 1$

Fixed Length Code (FLC) and Variable Length Code(VLC)

Let us represent 26 letters in the English alphabet using bits.

Fixed Length Code (FLC) and Variable Length Code(VLC)

Let us represent 26 letters in the English alphabet using bits.

$$R = \lfloor log_2 26 \rfloor + 1 = 5 \ bits$$

We know $2^5 = 32 > 26$

Hence, each of the letters can be uniquely represented using fixed length of 5 bits

Fixed Length Code (FLC) and Variable Length Code(VLC)

Let us represent 26 letters in the English alphabet using bits.

$$R = \lfloor log_2 26 \rfloor + 1 = 5 \ bits$$

We know $2^5 = 32 > 26$

Hence, each of the letters can be uniquely represented using fixed length of 5 bits

Allotting equal no of bits for frequently used letters and not frequently used letters is not an efficient way

We have to represent more frequently occurring letters with less number of bits using Variable Length Code (VLC)

Example: Fixed Length Code (FLC) and Variable Length Code(VLC)

Let us code First 8 letters ($A - H$) of English.

Fixed length code

Letter	*Codeword*	*Letter*	*Codeword*
A	000	E	100
B	001	F	101
C	010	G	110
D	011	H	111

Variable length code 1

Letter	*Codeword*	*Letter*	*Codeword*
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Let us Represent A BAD CAB using FLC and VLC

Example: Fixed Length Code (FLC) and Variable Length Code(VLC)

Let us code First 8 letters ($A - H$) of English.

Fixed length code

Letter	*Codeword*	*Letter*	*Codeword*
A	000	E	100
B	001	F	101
C	010	G	110
D	011	H	111

Variable length code 1

Letter	*Codeword*	*Letter*	*Codeword*
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Let us Represent A BAD CAB using FLC and VLC

Fixed Length Code	000 001 000 011 010 000 001	Total bits = 21
Variable Length Code	00 010 00 100 011 00 010	Total bits = 18

Example: Variable Length Code(VLC)

Let us code First 8 letters ($A - H$) of English.

Variable length code 1

Letter	*Codeword*	*Letter*	*Codeword*
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Variable length code 2

Letter	*Codeword*	*Letter*	*Codeword*
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

Let us Represent A BAD CAB using both VLC

Example: Variable Length Code(VLC)

Let us code First 8 letters ($A - H$) of English.

Variable length code 1

Letter	*Codeword*	*Letter*	*Codeword*
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Variable length code 2

Letter	*Codeword*	*Letter*	*Codeword*
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

Let us Represent A BAD CAB using both VLC

Variable Length Code 1	00 010 00 100 011 00 010	Total bits = 18
Variable Length Code 2	0 1001 0001	Total bits = 9

Variable Length Code(VLC) : Issues

Prefix Condition : No codeword forms prefix of another code word (VLC1 has better prefix than VLC2)

Instantaneous Codes: As soon as the sequence of bits corresponding to any one of the possible codewords is detected, symbol will be decoded

Uniquely Decodable Codes: Encoded string will be generated by only one possible input string, Have to wait unitl entire string is obtained before decoding even the first symbol

VLC2 is not a uniquely decodable code. VLC1 is uniquely decodable code

Kraft Inequality

A necessary and sufficient condition for the existence of a binary code with codewords having lengths $n_1 \leq n_2 \leq \ldots n_L$ that satisfy the prefix condition is

$$\sum_{k=1}^{L} 2^{-n_k} \leq 1$$

Proof:

Consider a binary tree $n = n_L$. This tree has 2^ntermianl nodes. Let us select any code of order n_1 as the first codeword C_1. Since no code word is prefix of any other codeword, this choice eliminates 2^{n-n_1} terminal codes. This process continues until the last codeword is assigned.

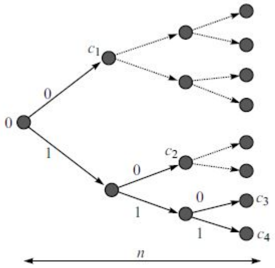


Fig. 1.9 A binary tree of order n_L.

Kraft Inequality: Example

A six symbol source is encoded in to binary codes shown below. Which of these codes are instantaneous ?

Source symbol	Code A	Code B	Code C	Code D	Code E
s_1	00	0	0	0	0
s_2	01	1000	10	1000	10
s_3	10	1100	110	1110	110
s_4	110	1110	1110	111	1110
s_5	1110	1101	11110	1011	11110
s_6	1111	1111	11111	1100	1111

Kraft Inequality: Example

A six symbol source is encoded in to binary codes shown below. Which of these codes are instantaneous ?

Source symbol	Code A	Code B	Code C	Code D	Code E
s_1	00	0	0	0	0
s_2	01	1000	10	1000	10
s_3	10	1100	110	1110	110
s_4	110	1110	1110	111	1110
s_5	1110	1101	11110	1011	11110
s_6	1111	1111	11111	1100	1111
$\sum_{k=1}^{6} 2^{-l_k}$	1	$\frac{13}{16} < 1$	1	$\frac{7}{8} < 1$	$1\frac{1}{32} > 1$

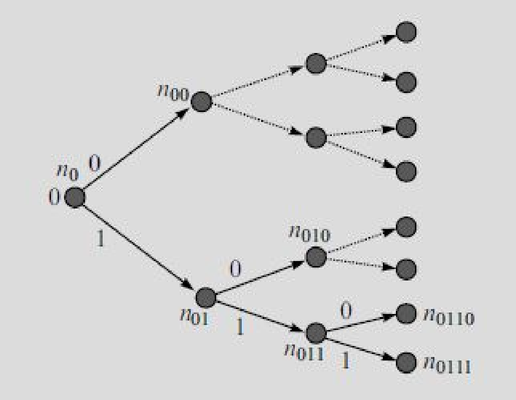
Check for Prefix Property and Kraft Inequality :

CODE E - Not Satisfies Kraft Inequality ;CODE D - Not Satisfies Prefix Property

CodeA, Code B, Code C satisfy both properties and instantaneous

Kraft Inequality: Example

Construction of a prefix code using a binary tree



$$\sum_{k=1}^{L} 2^{-n_k} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125 = 1$$

Hence Kraft inequality satisfied

Source Coding Theorem

Statement:
Let X be the ensemble of letters from a DMS with finite Entropy H(X) and the output symbols $x_k, k = 1, 2, \ldots, L$ occurring with probabilities $P(x_k), k = 1, 2, \ldots, L$.
It is possible to construct a code that satisfies the prefix condition and has an average length $\bar{R}$ that satisfies the inequality

$$H(X) \leq \bar{R} < H(X) + 1$$

The efficiency of a prefix code is

$$\eta = \frac{H(X)}{\bar{R}}$$

Redundancy of the code is

$$E = 1 - \eta$$

Example: Source Coding Theorem

Consider a Source X which generates four symbols with probabilities
$P(x_1) = 0.5, P(x_2) = 0.3, P(x_3) = 0.1 \; and \; P(x_4) = 0.1$

The entropy of the source is

$$H(X) = -\sum_{k=1}^{4} P(x_k) log_2 P(x_k) = 1.685 \; bits$$

If we use Prefix code discussed before $\{0, 10, 110, 111\}$

The average code word length $\bar{R}$ is

$$\bar{R} = \sum_{k=1}^{4} n_k P(x_k) = 1(0.5) + 2(0.3) + 3(0.1) + 3(0.1) = 1.700 \; bits$$

The efficiency of the code is

$$\eta = 1.685/1.700 = 0.9912$$

Huffman Coding

Huffman Coding Algorithm

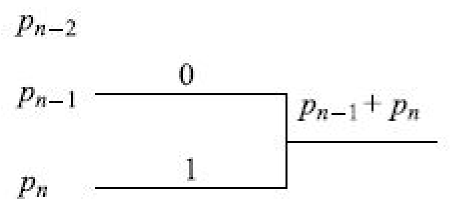
This algorithm is optimal in sense that average number of bits require to represent the source symbols is a minimum provided the prefix condition is met.

Steps:

1. Arrange the source symbols in a decreasing order of their probabilities
2. Take the bottom two symbols and tie them together. Add the probabilities of the two symbols and write it on the combined branches with a '1' and '0'.
3. Treat this sum of probabilites as a new probability associated with a new symbol. Again pick the two smallest probabilities tie tham together. Each time we perform this, total number of symbols is reduced by one
4. Continue this procedure until only one probability is left . This completes the construction of Huffman Tree
5. To find the prefix codeword for any symbol, follwo the branches from the final node back to the symbol

Huffman Coding

Combining probabilities



Number of stages required for encoding operation

$$n = \frac{N - r}{r - 1}$$

Here N= Total Number of symbols in source alphabet

Binary Huffman Coding r=2

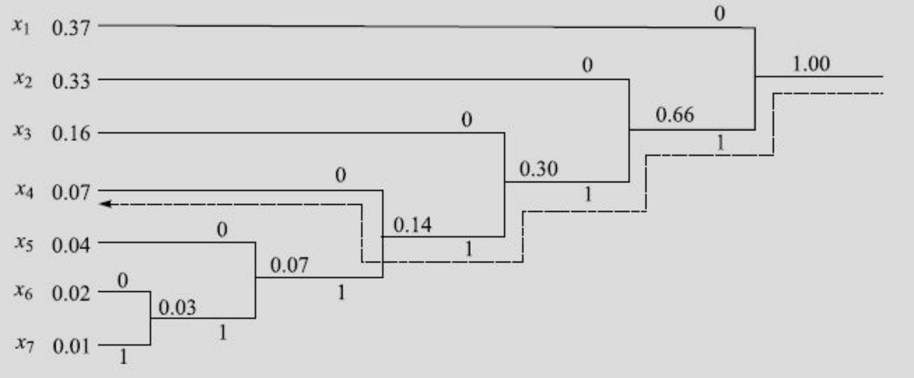
Ternary Huffman coding r=3

Quarternary Huffman Coding r=4

Example 1: Binary Huffman Coding

Consider a DMS with seven possible symbols $x_i, i = 1, 2, \ldots, 7$ and the corresponding probabilities $P(x_1) = 0.37, P(x_2) = 0.33, P(x_3) = 0.16, P(x_4) = 0.07, P(x_5) = 0.04, P(x_6) = 0.02, P(x_7) = 0.01$

Letus construct the Huffman Tree



Example 1: Binary Huffman Coding

Symbol	*Probability*	*Self Information*	*Codeword*
x_1	0.37	1.4344	0
x_2	0.33	1.5995	10
x_3	0.16	2.6439	110
x_4	0.07	3.8365	1110
x_5	0.04	4.6439	11110
x_6	0.02	5.6439	111110
x_7	0.01	6.6439	111111

The entropy of the source is

$$H(X) = -\sum_{k=1}^{7} P(x_k) log_2 P(x_k) = 2.1152 \ bits$$

The average number of binary digits per symbol is

$$\bar{R} = \sum_{k=1}^{7} n_k P(x_k) = 1(0.37) + 2(0.33) + 3(0.16) + 4(0.07) + 5(0.04) + 6(0.02) + 6(0.01) = 2.17 \ bits$$

Example 2: Non Binary Huffman Coding - Quarternary Huffman Coding

Construct a quarternary Huffman code for the following set of message symbols with the respective probabilities

A	B	C	D	E	F	G	H
0.22	0.20	0.18	0.15	0.10	0.08	0.05	0.02

Step-1: No of Stages $n = \frac{N-r}{r-1} = \frac{8-4}{4-1} = \frac{4}{3}$ Not an integer

Next value to get integer is N=10

$n = \frac{N-r}{r-1} = \frac{10-4}{4-1} = 2$

Example 2: Non Binary Huffman Coding - Quarternary Huffman Coding

Symbol	Probability		Stage 1		Stage 2	
A	0.22	1	0.22	1	**0.40**	0
B	0.20	2	0.20	2	0.22	1
C	0.18	3	0.18	3	0.20	2
D	0.15	00	0.15	00	0.18	3
E	0.10	01	0.10	01		
F	0.08	02	0.08	02		
G	0.05	030	**0.07**	03		
H	0.02	031				
D_1	0	032				
D_2	0	033				

Example 2: Non Binary Huffman Coding - Quarternary Huffman Coding

Symbol	Probability (p_i)	Code	Length (l_i)
A	0.22	1	1
B	0.20	2	1
C	0.18	3	1
D	0.15	00	2
E	0.10	01	2
F	0.08	02	2
G	0.05	030	3
H	0.02	031	3

Example 3: Huffman Coding Based on Frequency

Problem-

A file contains the following characters with the frequencies as shown. If Huffman Coding is used for data compression, determine-

1. Huffman Code for each character
2. Average code length
3. Length of Huffman encoded message (in bits)

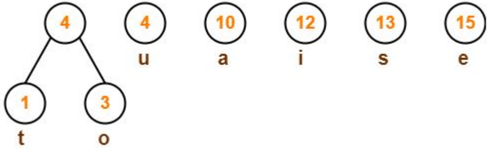
Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

Example 3: Huffman Coding Based on Frequency

Step-01:

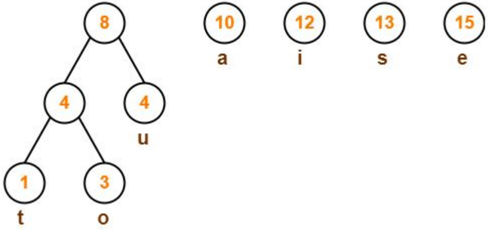
Example 3: Huffman Coding Based on Frequency

Step-02:



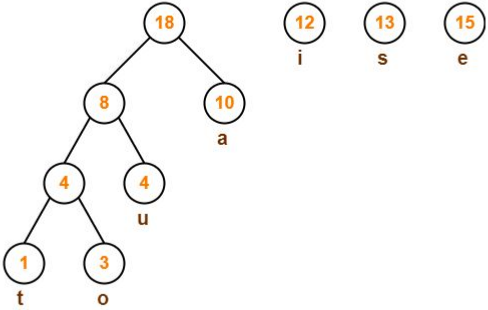
Example 3: Huffman Coding Based on Frequency

Step-03:



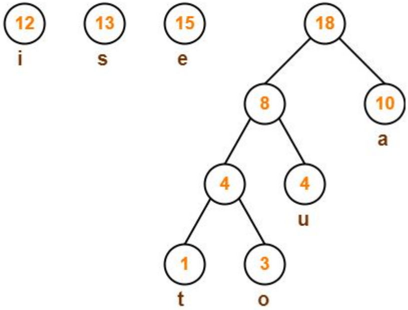
Example 3: Huffman Coding Based on Frequency

Step-04:

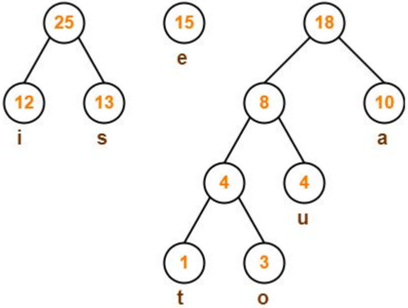


Example 3: Huffman Coding Based on Frequency

Step-05:

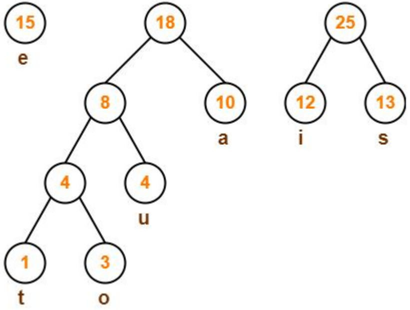


Example 3: Huffman Coding Based on Frequency

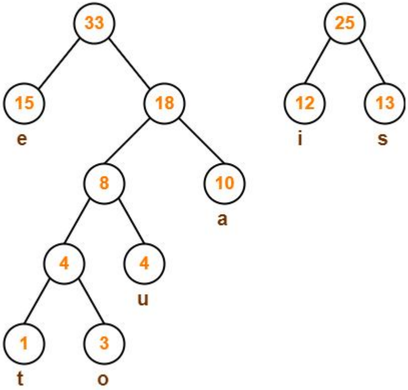


Example 3: Huffman Coding Based on Frequency

Step-06:

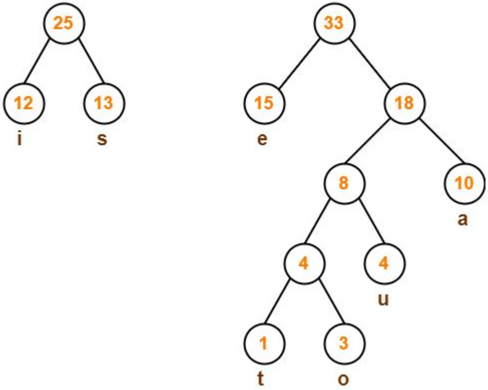


Example 3: Huffman Coding Based on Frequency

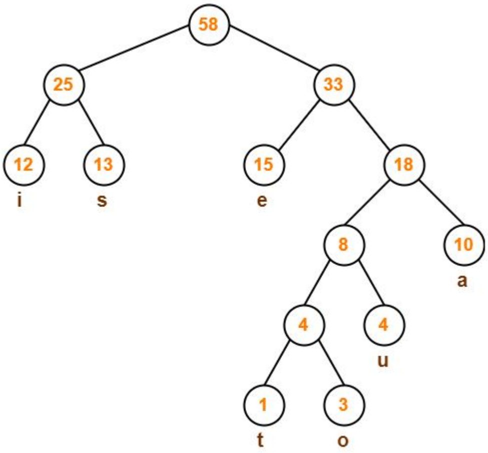


Example 3: Huffman Coding Based on Frequency

Step-07:



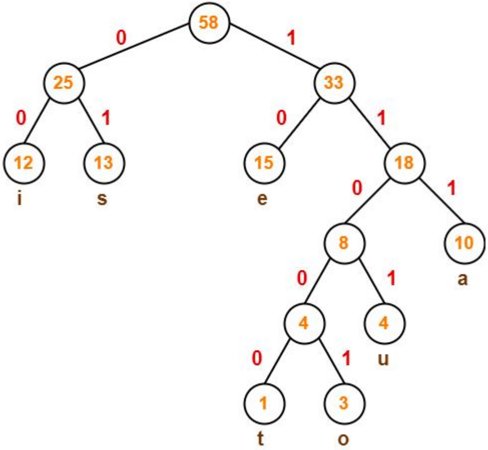
Example 3: Huffman Coding Based on Frequency



Huffman Tree

Example 3: Huffman Coding Based on Frequency

After assigning weight to all the edges, the modified Huffman Tree is-



Example 3: Huffman Coding Based on Frequency

1. Huffman Code For Characters-

To write Huffman Code for any character, traverse the Huffman Tree from root node to the leaf node of that character.

Following this rule, the Huffman Code for each character is-

- a = 111
- e = 10
- i = 00
- o = 11001
- u = 1101
- s = 01
- t = 11000

From here, we can observe-

- Characters occurring less frequently in the text are assigned the larger code.
- Characters occurring more frequently in the text are assigned the smaller code.

Example 3: Huffman Coding Based on Frequency

2. Average Code Length-

Using formula-01, we have-

Average code length

$= \sum (\text{frequency}_i \times \text{code length}_i) / \sum (\text{frequency}_i)$

$= \{ (10 \times 3) + (15 \times 2) + (12 \times 2) + (3 \times 5) + (4 \times 4) + (13 \times 2) + (1 \times 5) \} / (10 + 15 + 12 + 3 + 4 + 13 + 1)$

$= 2.52$

Extended Huffman Coding

Consider a DMS with three possible symbols x_i, $i = 1, 2, 3$ and the corresponding probabilities $P(x_1) = 0.4$, $P(x_2) = 0.35$, $P(x_3) = 0.25$

Codewords using Huffman Algorithm

Symbol	Probability	Self Information	Codeword
x_1	0.40	1.3219	1
x_2	0.35	1.5146	00
x_3	0.25	2.0000	01

The entropy of this source is

$$H(X) = -\sum_{k=1}^{3} P(x_k) log_2 P(x_k) = 1.5589 \ bits$$

The average number of binary digita per symbol is

$$\bar{R} = \sum_{k=1}^{3} n_k P(x_k) = 1(0.40) + 2(0.35) + 2(0.25) = 1.60 \ bits$$

The Efficiency of this code is $\eta = 1.5589/1.6 = 0.9743$

Extended Huffman Coding

Group the symbols for 2^{nd} order extension

Symbol Pairs	Probability	Self Information	Codeword
x_1x_1	0.1600	2.6439	10
x_1x_2	0.1400	2.8365	001
x_2x_1	0.1400	2.8365	010
x_2x_2	0.1225	3.0291	011
x_1x_3	0.1000	3.3219	111
x_3x_1	0.1000	3.3219	0000
x_2x_3	0.0875	3.5146	0001
x_3x_2	0.0875	3.5146	1100
x_3x_3	0.0625	4.0000	1101

The entropy of this source is

$$2H(X) = -\sum_{k=1}^{9} P(x_k) log_2 P(x_k) = 3.1177 \ bits \implies H(X) = 1.5589 \ bits$$

The average number of binary digita per symbol is

$$\bar{R_B} = \sum_{k=1}^{9} n_k P(x_k) = 3.1775 \ bits \ per \ symbol \ pair \implies \bar{R} = \bar{R_B}/2 = 1.5888$$

The Efficiency of this code is $\eta = 1.5589/1.5888 = 0.9812$

Shannon - Fano Coding

Shannon's First Encoding Algorithm

Codes that uses codeword lengths of $l(x) = \lceil log \frac{1}{P(x)} \rceil$ are called Shannon Codes. Shannon codeword lengths satisfy the kraft inequality.

Steps:

1. Given the source alphabet S and the corresponding probabilities P for a given information source
2. Arrange the probabilities in the non increasing order
3. Compute the length of l_i for the codeword corresponding to each symbol s_i from probabilit $p_i$is given by

$$l_i \geq log_2 \frac{1}{P_i}$$

Shannon's First Encoding Algorithm

4. Define the following parameters from the probability set $q_1 = 0$
   $q_2 = p_1 = q_1 + p_1$
   $q_3 = p_1 + p_2 = q_2 + p_2$
   $q_4 = p_1 + p_2 + p_3 = q_3 + p_3$
   $\ldots$
   $\ldots$
   $q_{N+1} = 1$
5. Expand q_i in binary till l_i number of places after decimal point
6. The numbers after decimal places in the binary representation of q_i are the codewords for the corresponding symbol s_i

Example : Shannon's First Encoding Algorithm

Consider a source with alphabets $S = \{A, B, C, D\}$ with corresponding probabilities $P = (0.1, 0.2, 0.3, 0.4)$.Find the codewords for symbols using Shannon's algorithm. Also find the source efficiency and redundancy

1. Arrange the probabilities in the non increasing order

$$P = (0.4, 0.3, 0.2, 0.1) \quad S = (D, C, B, A)$$

2. Find the minimum value of l_i such that $l_1 \geq log_2 \frac{1}{p_1} = log_2 \frac{1}{0.4} \implies l_1 = 2$

   $l_2 \geq log_2 \frac{1}{p_2} = log_2 \frac{1}{0.3} \implies l_3 = 2$

   $l_3 \geq log_2 \frac{1}{p_3} = log_2 \frac{1}{0.2} \implies l_3 = 3$

   $l_4 \geq log_2 \frac{1}{p_4} = log_2 \frac{1}{0.1} \implies l_4 = 4$

Example: Shannon's First Encoding Algorithm

3. Calcualte the parameters q_i

   $q_1 = 0$

   $q_2 = p_1 = 0.4$

   $q_3 = p_1 + p_2 = 0.7$

   $q_4 = p_1 + p_2 + p_3 = 0.9$

   $q_5 = p_1 + p_2 + p_3 + p_4 = 1$

4. Represent q_1, q_2, q_3, q_4 in binary up to l_i places after decimal points

   $q_1 = (0.0)_{10} = (0.00)_2$

   $q_2 = (0.4)_{10} = (0.01)_2$

   $q_3 = (0.7)_{10} = (0.101)_2$

   $q_4 = (0.9)_{10} = (0.1110)_2$

Example: Shannon's First Encoding Algorithm

5. The codewords are

Symbol	Probability	Code Word	Length
D	0.4	00	2
C	0.3	01	2
B	0.2	101	3
A	0.1	1110	4

6. The Entropy $H(x) = \sum_{k=1}^{L} p(x_k) log_2 \frac{1}{p_k} = 1.8464 \; bits/sym$
7. The Average Code Length $\bar{R} = \sum_{k=1}^{L} n_k p(x_k) = 2.4 \; bits/sym$
8. The efficiency is $\eta = \frac{1.8464}{2.4} = 0.7693 \implies 76.93\%$
9. The redundancy is $E = 1 - \eta = 0.2307$

Shannon-Fano Encoding Algorithm

This is an improvement over Shannon's first algorithm. It offers better coding efficiency compared to Shannon's algorithm.

Steps:

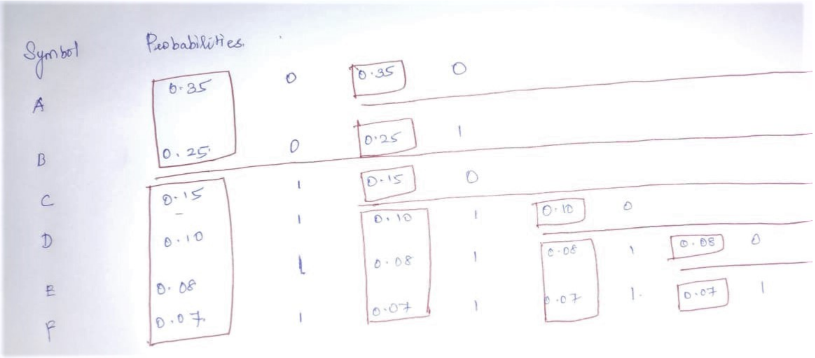
1. Arrange the probabilities in the non-increasing order
2. Group the probabilities in to exactly two sets such that the sum of probabilities in both the groups is almost equal.
3. Assign bit '0' to all elements of the first group and bit'1' to all elements of group 2
4. Repreat Step-2 by dividing each group in two sub groups till no further division is possible

Example: Shannon-Fano Encoding Algorithm

Consider the following source: $S = (A, B, C, D, E, F)$ with following probabilities $P = \{0.1, 0.15, 0.25, 0.35, 0.08, 0.07\}$.

Steps:

1. Arrange the given probabilities in the non-increasing order. Divide the probabilites in to two almost equiprobable groups.



Example:Shannon-Fano Encoding Algorithm

2. The codewords using Shannon-Fano Algoritm is

Symbol	Probability (P_i)	Codeword	Length (l_i)
A	0.35	00	2
B	0.25	01	2
C	0.15	10	2
D	0.10	110	3
E	0.08	1110	4
F	0.07	1111	4

Example:Shannon-Fano Encoding Algorithm

3. The entropy of the code is

$$H(x) = \sum_{k=1}^{L} p(x_k) log_2 \frac{1}{p_k} = 2.33\ bits/sym$$

4. The Average Code Length is
$\bar{R} = \sum_{k=1}^{L} n_k p(x_k)$
$\bar{R} = (0.35)2 + (0.25)2 + (0.15)2 + (0.10)3 + (0.08)4 + (0.07)4$
$\bar{R} = 2.4\ bits/sym$
5. The Efficiency is $\eta = 2.33/2.4 = 0.978 \implies 97.8\%$
6. The redundancy is $E = 1 - \eta = 1 - 0.978 = 0.292 \implies 2.92\%$

Shannon-Fano-Elias Coding

Codes that uses codeword lengths of $l(x) = \lceil log \frac{1}{P(x)} \rceil$ are called Shannon Codes.
Shannon-Fano-Elias Coding uses Cumulative Distribution Function to allocate Code Words.
The Cumulative Distribution Function is Defined as

$$F(x) = \sum_{z \leq x} P(z)$$

Where P(z) is probability of occurance of z.
The modified Cumulative Distributive Function is

$$\bar{F}(x) = \sum_{z \leq x} P(z) + \frac{1}{2} P(x)$$

Where, $\bar{F}(x)$ represents the sum of probabilities of all symbols less than x plus half the probability of the symbols x.
Note: In this code, No need to arrange the probabilities in descending order

Example : Shannon-Fano-Elias Coding

PROBLEM:

Construct Shannon-Fano-Elias coding for the source symbols x_1, x_2, x_3, x_4 with probabilites $\frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \frac{1}{2^3}$.

STEPS:

1. Find $F(x) = \sum_{z \leq x} P(z)$ (Add all previous and current probabilities of the symbol)

Symbol	Probability	F(x)
x_1	$\frac{1}{2}$	0.5
x_2	$\frac{1}{2^2}$	0.75
x_3	$\frac{1}{2^3}$	0.875
x_4	$\frac{1}{2^3}$	1

For Example, To find $F(x_4) = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = 1$

Example : Shannon-Fano-Elias Coding

2. Find $\bar{F}(x) = \sum_{z<x} P(z) + \frac{1}{2}P(x)$ (Add all previous probabilities of less than x and half the current probability of the symbol)

Symbol	Probability	F(x)	$\bar{F}(x)$
x_1	$\frac{1}{2}$	0.5	0.25
x_2	$\frac{1}{2^2}$	0.75	0.625
x_3	$\frac{1}{2^3}$	0.875	0.8125
x_4	$\frac{1}{2^3}$	1	0.9375

For Example, To find $\bar{F}(x_3) = \frac{1}{2} + \frac{1}{2^2} + \frac{\frac{1}{2^3}}{2} = 0.8125$

3. Find $\bar{F}(x)$ in binary form (Convert the decimal floating values in to binary)

Symbol	Probability	F(x)	$\bar{F}(x)$	$\bar{F}(x)_{binary}$
x_1	$\frac{1}{2}$	0.5	0.25	0.01
x_2	$\frac{1}{2^2}$	0.75	0.625	0.101
x_3	$\frac{1}{2^3}$	0.875	0.8125	0.1101
x_4	$\frac{1}{2^3}$	1	0.9375	0.1111

For Example, To find $\bar{F}(x_3) = 0.8125$ *in to* $\bar{F}(x_3)_{binary}$

$0.8125X2 = 1.6250$

$0.625X2 = 1.250$

$0.25X2 = 0.50$

$0.5X2 = 1 \implies (0.1101)_2$

Example : Shannon-Fano-Elias Coding

4. Determine the length of the codeword using $l(x) = \lceil log \frac{1}{P(x)} \rceil + 1$

Symbol	Probability	F(x)	$\bar{F}(x)$	$\bar{F}(x)_{binary}$	$l(x)$
x_1	$\frac{1}{2}$	0.5	0.25	0.01	2
x_2	$\frac{1}{2^2}$	0.75	0.625	0.101	3
x_3	$\frac{1}{2^3}$	0.875	0.8125	0.1101	4
x_4	$\frac{1}{2^3}$	1	0.9375	0.1111	4

For Example, To find $l_3 = \lceil log \frac{1}{\frac{1}{2^3}} \rceil + 1 = 4$

Example : Shannon-Fano-Elias Coding

5. Write the code word from $\bar{F}(x)_binary$ for the length of $l(x)$

Symbol	Probability	F(x)	$\bar{F}(x)$	$\bar{F}(x)_{binary}$	$l(x)$	code
x_1	$\frac{1}{2}$	0.5	0.25	0.01	2	01
x_2	$\frac{1}{2^2}$	0.75	0.625	0.101	3	101
x_3	$\frac{1}{2^3}$	0.875	0.8125	0.1101	4	1101
x_4	$\frac{1}{2^3}$	1	0.9375	0.1111	4	1111

For Example, To find codeword for x_3, $\bar{F}(x_3)_{binary} = 0.1101$ with l(3) is 4. Hence Code word is 1101

6. Entropy for this code is 1.75 bits
7. Average Code Word Length is 2.75 bits