09/01/2020

$\rho = 64$ Configuration ⎫ Grade 1
26
38 - Unused ⎭

→ Statics Huffp to achieve compression.

Nyquistic Theorem

min 64 k bits/sec

rather than sending speech, send configuration of a vocal tract.

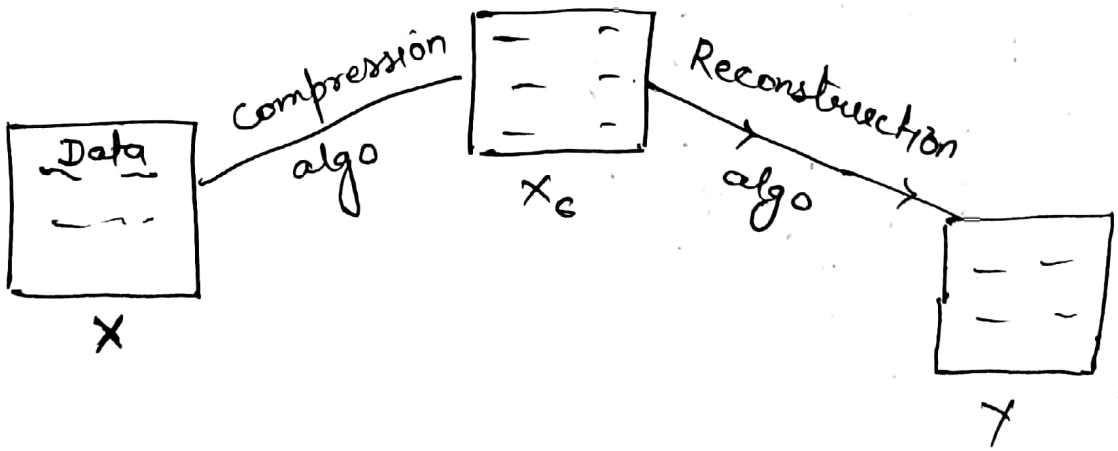
(Homer Didley)
1936

→ voice-coder

→ Exploit the useless inability to hear or process things.

Compression Techniques:-

(i) Compression Algorithm ⎫ 2 algorithm for
(ii) Reconstruction algorithm ⎭ compression.



1. Lossless Compression Algorithm

If $Y = X$ then

2. Lossy Compression Algorithm

$Y \neq X$ (still we sometime prefer bec... we get more compression compare to lossless compres...

Application of Lossless Compression Algorithm :-

1) Bank Data (Records)

Generally text data is sensetive to loss in the data.

"Do not send Money" ⟷ "Do now send Money"

2) Computer Codes etc.

3) Satellite records /data (Data obtained by satellite)

~~4) Data~~

→ Application of Lossy Compression :-

1) Transmating speech
   a) Telephony speech
   b) CD quality music

ii) Tv video

Measures of Performance:-

a) Lossless compression algorithm.

1) look at the ratio of the no of bits required to represent the data before compression to the no of bits required to represent the data after compression. This ratio is called compression ratio.

→ Eg. ~~Suppose~~ Supporting an image made up of square array of 256 x 256 pixel. requires 65, 536 Byte. The image is compressed and the compression version requires 16384 Byte. ~~This~~ We would say that the compression ratio is 4:1

→ We can also represent compression ratio by expressing the reduction in the amount of data required as the percentage of the size of the orgenal data

→ In this particular example, the compression ratio is calculated in this manner would be 75%.

→ Another way of reporting compression performance is to provide the average no of bits required to represent a single sample. It is generally referred as rate. ~~do~~ For ~~compression~~ example, in the compressed image, talked just now of we

assume. 8 bit per byte (or pixel), the avg no of bits per pixel is 2. Thus we would say that the rate is 2 bits / pixel.

b) Lossy compression Algorithm { Voice, Image (Sound, Video,) } ↓ basically for (Analog signals)

1) Distortion

2) Rate

Rate Distortion Theory

* Modelling & Coding :-

Developement of data compression algo. for a variety of data can be devide into two phases. The first phase is usually referred do, as modeling phase.

→ In this, we are trying to extract info. about any redendancy that exists in data. and describe the redendancy in the form of model.

Ex - Consider the

(b) Consider the following sequence

27 28 29 28 26 27 29 28 30 32 34 36 38

(i) 38 − 27 = 11 (4 bit)

(ii) (max no) 38 = (6 bit

(iii) 27, 1, 1, −1, −2, 1, 2, −1, 2, 2, 2, 2, 2

(c) Suppose we have the following sequence.

a. blank b. a. r, a y a r a n blank a r r a y blank r a n b f a r b f a a r b f a a a r b a w a y.

total symbol = 41

sol^n

a — 1	— 16	a — 16 — 1 bit
b — 001	— 7	r — 8 } 3 bit
b — 01100	— 1	blank — 7
f — 0100	— 3	f — 3
r — 000	— 8	y — 3 } 4 bit
n — 0111	— 2	n — 2
w — 0101	— 1	w — 1 } 5 bits
y — 0101	— 3	b — 1

most frequent has less bit codeword.

(1) Redundancy in the form of words that repeat of ten
↳ dictionary based techniques

(2) We look at groups of symbols
- redundancy in the data become more and more when we look at groups of symbols

Hybrid Model:-

Coding:-

- We mean the assignment of binary sequences to the elements of an alphabets.

a - 1
ß - 001
b - 01100
⋮
etc

→ The set of binary sequences is called a code.

{1, 001, 01100, 0101}

→ Each member of the code is called codeword.

→ An alphabets is collection of symbols called letters. $A = \{a, ß, b, f, n, \ldots y\}$

ASCII code

→ Every letters represented by 8-bits, such codes are called fixed length codes.

Fixed length code do not offer any compression. So to achieve compression, we need to resolve to variable length code.

Ex $A = \{a_1, a_2, a_3, a_4\}$

$p(a) = \frac{1}{2}$, $p(a_2) = \frac{1}{4}$, $p(a_3) = \frac{1}{8}$ $p(a_4) = \frac{1}{8}$

		code 1	code 2	code 3	code 4
a_1	0.5	0	1	0	0
a_2	0.25	0	1	10	01
a_3	0.125	1	00	110	011
a_4	0.125	10	11	111	0111
		X	X	valid	

(Ambigous) $a_2a_3 = 100$ or 100

a_2 $\quad a_1a_1$ $\quad a_2a_3$

Avg length of code 1 $= 0.5\times1 + 0.25\times1 + 0.125\times1 + 0.125\times2$

Avg " " code 2 $=$ $= 1.125$ bits / symbol

Avg " " code 3 $= 1.75$ bits / symbol

code 4 $=$

i.e (probability * length of codeword)

⇓

Avg length of code.

* Entropy of the source, $= .175$ bits/symbol

* Decoding Rule : Accumulate ~~symbol~~ bits until you get a 0 or you get three 1's

eg $a_2a_3a_3a_1$ (code 3)

10 110 1100

$\underline{10}$ $\underline{110}$ $\underline{110}$ $\underline{0}$

a_2 a_3 a_3 a_1

Note – If 1st bit is zero while decoding, just avoid it (in code 4)

eg $a_2a_1 = \underline{01}\,|\,0$

a_2a_1

Avg length of the code

$$l = \sum_{i=1}^{4} p(a_i)\, n(a_i)$$

where $n(\cdot)$ denotes the length of the codeword for

16/01/2020

Entropy 1.75 bits / symbol

* here code3 is an instantacies code, code4 is near instantacies.

Uniquely Decodable code :-

Letters	code1	code2	code3	code4	code5
a_1	0	0	0	0	0
a_2	0	1	10	01	01
a_3	1	00	110	011	011
a_4	0	11	111	0111	
p	1.125	1.25	1.75	1.825	

Let for code 5 $\underbrace{0}_{a_1}\underbrace{1}_{a_3}\underbrace{1}_{a_3}\underbrace{1}_{a_3}\underbrace{1}_{a_3}\underbrace{1}_{a_3}\underbrace{1}_{a_3}\underbrace{1}_{a_3}$ — decoding has failed

$a_2\ a_3\ a_3\ a_3\ a_3\ a_3 \rightarrow$ successful decoding

→ code 6

Letters	codeward
a_1	0
a_2	01
a_3	10

→ so it is invalid code.

Encoding the sequence $a_1\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3$

$0\ 10\ 10\ 10\ 10\ 10\ 10\ 10\ 10$

$a_2\ a_2\ a_2\ a_2\ a_2\ a_2\ a_2\ a_2\ a_1$

or $a_1\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3\ a_3$

→ Given a ~~not~~ code, can we have a test to check, whether the code is valid code or not?

- Test for unique decoding:-

→ Suppose we have two codewords a and b, where a is k bits long and b is 'n' bits long and $k < n$.

If first k bits of b is are identical to a, then a is called a prefix of ~~bits~~. The last n-k bits of b are called the dangling suffix

→ Test:-

1.) Construct a ~~codeward~~ list of all the codewords

2.) Examine all pairs of codewords to see if any codeward is a prefix of another codeward

3.) Whenever we find such a pair add the dangling suffix to the list of codewords, unless you have added to the same dangling suffix to the list in a previous iteration.

4.) Now, repeat the procedure using this larger list.

You endup in one of the following two scenarios:

i) you get a dangling suffix i.e valid codewords

ii) There are no more unique dangling suffix, then the code which you are testing is unique decoded

The code that you are testing is not a valid code.

Ex:- code 5

1) {0, 01, 11}

0, 01 → 1
0, 11 → x
01, 11 → x

(2), {0, 01, 11, 1}

(3)

(4) {0, 01, 11, 0}

0, 01 → 1
01, 11 → x
1, 11 → 1
0, 11 → x
0 → 1 → x

01, 1, → x

↓

There are no more uniquely suffixes, so it is unique code

2) code 6

1) {0, 01, 10}

test
0, 01 → 01
0, 10 → x
01, 10 → x

2), 3) {0, 01, 10, 1}

4)

0, 01 → 1
0, 10 → x
0, 1 → x
01, 10 → x
01, 1 → x
10, 1 → 0

Since we get a dangling suffix i.e is valid codeword 0. so it is not a valid ~~codeword~~. code.

→ Can we design a code, which we can say that it is uniquely decoded by looking of it and without going for uniquely decoding tests.

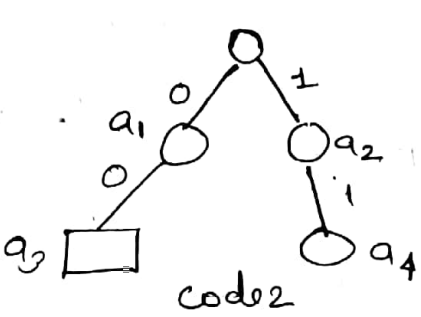
prefix code:- A code in which no codeword is a prefix to another codeword is called prefix code.

- A way of testing a code is prefix code or not is by constructing a rooted binary tree corresponding to the code.

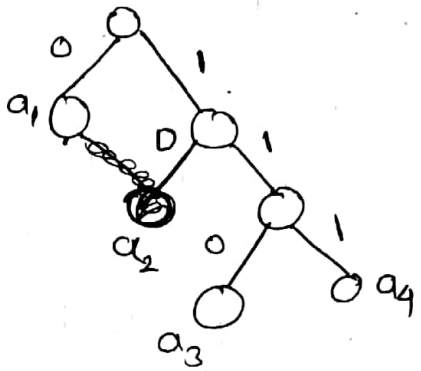
eg Code2 $a_1 - 0$, $a_2 - 1$, $a_3 - 00$, $a_4 - 11$

Code3 $a_1 \to 0$, $a_2 - 10$, $a_3 - 110$, $a_4 - 111$

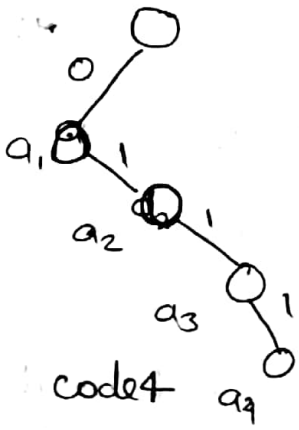
code 4 $a_1 \to 0$, $a_2 \to 01$, $a_3 \to 011$, $a_4 - 0111$



code2



code 3



code4

prefix code

→ if all the code is corresponding to the internal node of binary tree, then it is prefixcode.

→ For any non-prefix uniquely decodable code, we can always find a prefix code, with the same codewords length.

Huffman Coding:-

Observations:-

1) In an optimization code, symbol that occur more frequently (~~than~~ have a ~~shorter~~ higher probability of occurance) will have shorter codeward than symbols that occur less

frequently. optimum

2) In a huffman coding, the two symbol that occurs lesser frequently will have same length.

Suppose,

That is, the length of the codewords of the least frequent symbols are not same.

→ Assume, the two least frequent symbols are a & b, also assume that the length of codeword of a is k bit and the length of codeword of b is n, and $k < n$.

→ Because the code under consideration is a prefix code, no codeword is a prefix of another codeword.

→ Since no codeword is prefix of another codeword it follows that, the codeword corresponding to a is not a prefix to a codeword corresponding to b.

→ chop of the last $(n-k)$ bits of the codeword corresponding to b.

→ Even after chopping of the last $(n-k)$ bits of the codeword corresponding to 'b', the two codeword corresponding to 'a' and 'b' are still different...

→ The new codeword of 'b' may be a prefix another codeword

Claims:- This can't happen.

Because a & b are two least frequent fellows and all others symbol are more frequent than a & b. so by our first observation, the lengths of codewords of other symbol is smaller than the length of codeword of a & b. Hence the shorting codeword of b cant be a prefix of another codeword.

Requirements:—

The codeword corresponding to the lowest probable symbol differs only the in the last bit.

That is if Y and S are the two least probable symbols in an alphabet, if the codeword of Y is m*0, the codeword of S is would be m*1.

Where m is a steing 1's and 0's and * denotes concatenation.

Ex:- Design of Huffman code:—

Let us design a huffman code for a source that puts out letters from another set of

$A = \{a_1, a_2, a_3, a_4, a_5\}$ bit

$p(a_1) = p(a_3) = 0.2$, $p(a_2) = 0.4$

$p(a_4) = a\, p(a_5) = 0.1$

The entropy for this source is 2.122 bits/symbol

Letters	probability	codeword
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a_4	0.1	$c(a_4)$
a_5	0.1	$c(a_5)$

Assume $c(a_4) = \alpha_1 * 0 = 0010$, $c(a_5) = \alpha_1 * 1 = 0011$ } codeword of both are same except the last bit

Now,

a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a_4'	0.2	α_1

Assume

$c(a_3) = \alpha_2 * 0 = 000$

$c(a_4') = \alpha_1 = \alpha_2 * 1$

$= 001$

Now

a_2	0.4	$c(a_2)$
a_3'	0.4	α_2
a_1	0.2	$c(a_1)$

$c(a_3') = \alpha_2 = \alpha_3 * 0 \Rightarrow 01$

$c(a_1) = \alpha_3 * 1 \Rightarrow 01$

Now

a_1'	0.6	α_3	$= 0$
a_2	0.4	$c(a_2)$	$= 1$

Now

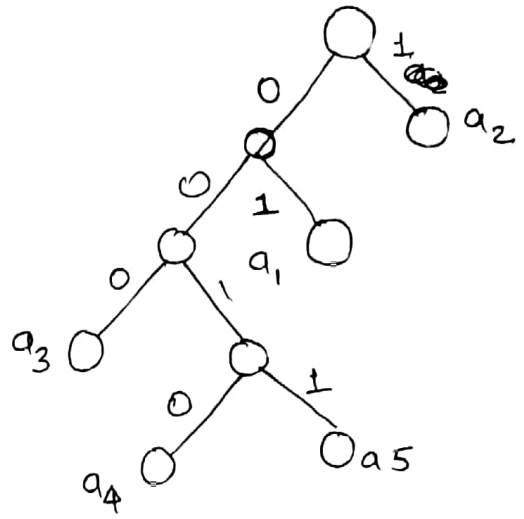
$c(a_2) = 1$

$c(a_1) = 01$

$c(a_3) = 000$

$c(a_4) = 0010$

$c(a_5) = 0011$

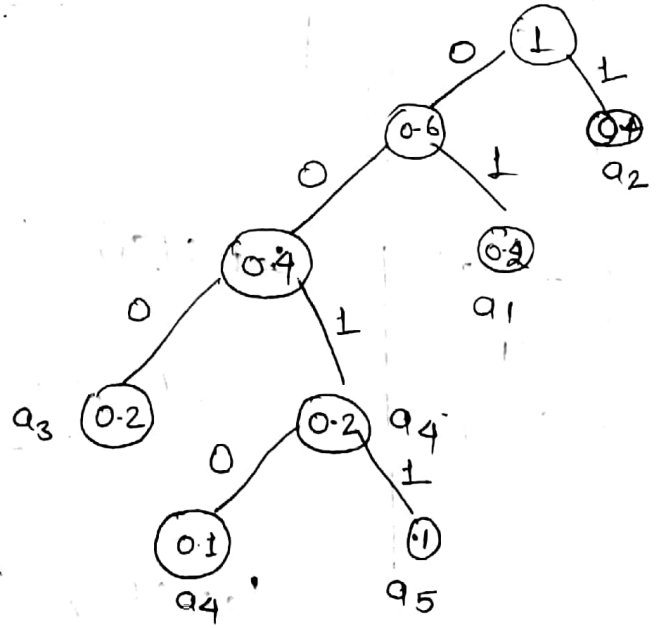


Avg length of code = $0.4 * 1 + 0.2 * 2 + 0.2 * 3 + 0.1 \times 4 + 0.1 \times 4$

$= 0.4 + 0.4 + 0.6 + 0.4 + 0.4 = 2.2$ bits / symbol.

Redundancy = Avg length of code − entropy

$= 2.200 - 2.122 = 0.028$ bits/symbol.



here $\mu = 2.22$

Variance = $(1-2.2)^2 + (2-2.2)^2 + (3-2.2)^2 + (4-2.2)^2 + (4-2.2)^2$

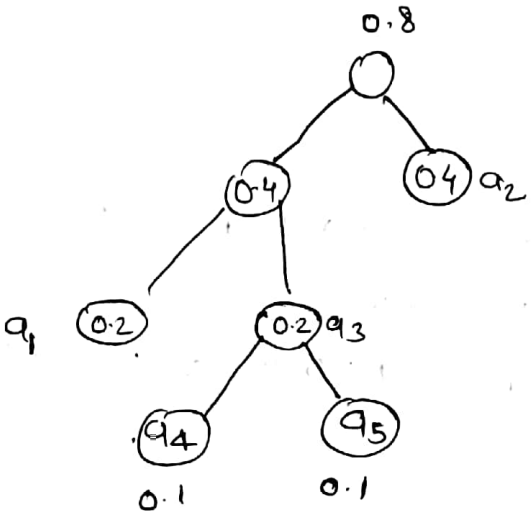
$= 1.44 + 0.04 + 0.64 + 3.24 + 3.24$

$= 8.6$

20 Jan 2020

Q. Design a Huffman code for a source that puts out letters from an alphabets

$A = \{a_1, a_2, a_3, a_4, a_5\}$ with $P(a_1) = P(a_3) = 0.2$, $P(a_2) = 0.4$, $P(a_4) = P(a_5) = 0.1$



$c(a_2) = 1$

$c(a_1) = 00$

$c(a_3) = 010$

$c(a_4) = 0110$

$c(a_5) = 0111$

Letters	Prob	codeword
a_2	0.4	$c(a_2)$
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$
a_4	0.1	$c(a_4)$
a_5	0.1	$c(a_5)$

Let $c(a_4) = \alpha_1 * 0$
$= 010$

$c(a_5) = \alpha_1 * 1$
$= 011$

a_2	0.4	$c(a_2)$
a_4'	0.2	α_1
a_1	0.2	$c(a_1)$
a_3	0.2	$c(a_3)$

Let $c(a_1) = \alpha_2 * 0 = 10$

$c(a_3) = \alpha_2 * 1$
$= 11$

a_3'	0.4	α_2
a_2	0.4	$c(a_2)$
a_4'	0.2	α_1

Let $c(a_2) = \alpha_3 * 0$

$c(a_4') = \alpha_1 = \alpha_3 * 1$
$= 01$

a_2'	0.6	α_3 = 0
a_1'	0.4	α_2 = 1

$c(a_2) = 00$

$c(a_1) = 10$

$c(a_3) = 11$

$c(a_4) = 010$

$c(a_5) = 011$

avg length of codeword $= 0.4*2 + 0.2*2 + 0.2*2 + 0.1*3 + 0.1*3$

$= 0.8 + 0.4 + 0.4 + 0.3 + 0.3$

$= 2.22$ bits / symbol (Mean)

Variance of Code (which is min is taken as good)

$= (2-2.2)^2 + (2-2.2)^2 + (2-2.2)^2 + (3-2.2)^2 + (3-2.2)^2$

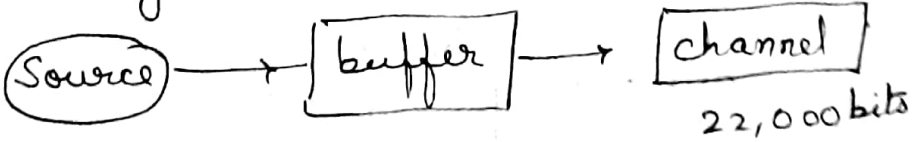
$= 1.4$ (min variance code as compared to 1 method)

→ Suppose your code (source) is emiting 10,000 symbol per second.

per second, on and avg source is emiting 22,000 bits.

Buy the channel, saying, please provide me the bandwidth required to transmits 22,000 bits / sec.

For first few second, your source is emiting a_4 & a_5 symbol.



→ Huffman code is better.

Optimality of Huffman code :-

The necessary condition for optimal variable length binary code are as follow

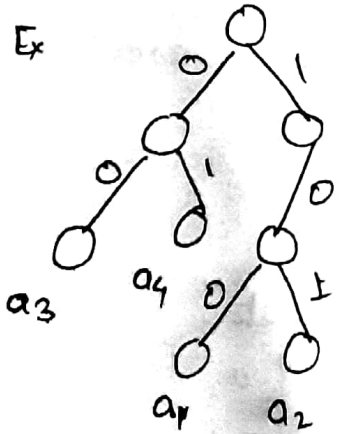
Conditions: ① Given any two letters a_j and a_k. if $P(a_j) \geq P(a_k)$, then $l_j(a_j) \leq l_k$ where l_j is the no of bits in the codeword for a_j

② The two least probable letters have codeword with same max^m length l_m.

③ In the tree corresponding to optimum code, there must be two branches ~~or~~ stemming from each intermediate node.

④ Suppose we change an intermediate node into a leaf node by combining all the leaf descending from it into a composite code of a reduced alphabets, then if the original tree was optimal for the original alphabets, the reduced tree is optimal for the reduced alpha.

Ex



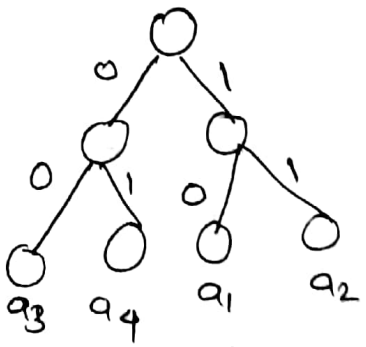
optimal code

$c(a_1) = 100$

$c(a_2) = 101$

$c(a_3) = 00$

$c(a_4) = 01$



$c(a_1) = 01$

$c(a_2) = 11$

$c(a_3) = 00$

$c(a_4) = 01$

→ If there were any intermediate node with only one branch coming from that node, we could remove it of the code without effecting the decipherability, while reducing its avg length.

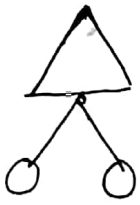
for 4 condition eg.

optimal for the original alphabets

Assuming, it is not optimal for the reduced alphabets.

→ That is, there is optimal tree for the reduced alphabet.

Take that optimal tree for the reduced alphabets.



→ This tree has less avg length than the orgenal tree, so it is optimal than orgenal tree, which is not possible, so contradiction goes wrong.

→ In order to satisfy 1, 2, 3 condition the two least probable letters would have to be assigned the codeword of minimum length lm.

23 Jan 2020

Optimality of Huffman Code:-

Condition 1: Given any two letters are ~~is~~ leave

→ Further more, these corresponding to the from the same intermediate code. This is same as saying that the codewords for these letters are identical for last bit.

→ Conseder the common prefix as the codeword for the composite letters of a reduced alphabet.

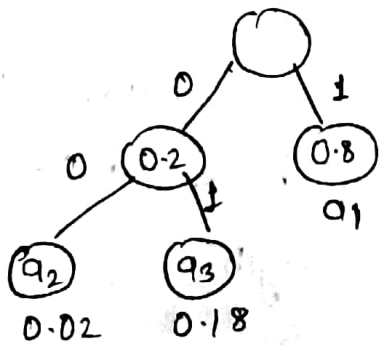
→ Since the code for the reduced alphabets needs ~~means~~ to be optimum for the code of original alphabets, we follow the same procedure again.

Extended Huffman Code:-

Ex:- $A = \{a_1, a_2, a_3\}$

$p(a_1) = 0.8 \quad p(a_2) = 0.02, \quad p(a_3) = 0.18$

Entropy of the source is given as 0.816 bits/symbol.



$c(a_1) = 1$

$c(a_2) = 00$

$c(a_3) = 01$

avg length of the code is = 0.8 * 1 + 0.02 * 2 + 0.18 * 2

= 0.80 + 0.04 + 0.36 = 1.2 bits/symbol

0.04
× 0
36
1.20

Redundancy = 1.2 − 0.816 = 0.384 bits / symbol.

→ Wastage around 50%. i.e 47% more bits

2nd Method:-

Soln: taken into consider two symbol at a time

	Probability		code		
(a_1a_1) =	0.64	——	0	1	0.64
a_1a_2 =	0.016	——	10101	5	
a_1a_3 =	0.144	——	11	2	
a_2a_1 =	0.016	——	101000	6	
a_2a_2 =	0.0004	——	10100101	8	
a_2a_3 =	0.0036	——	1010011	7	
a_3a_1 =	0.144	——	100	3	
a_3a_2 =	0.0036	——	10100100	8	
a_3a_3 =	0.0324	——	1011	4	

avg length of code = 1.7228 bits / symbol

for comparing = $\frac{1.7228}{2}$ = 0.8614 bits / symbol.

Non- Binary Huffman Codes:—

1. Symbols that occurs more frequently will have shorter codewords, there symbols that occurs less frequently.
2. Two symbols that occur least frequently will have the same codeword length.

2 m-ary code

{0, 1, 2 ... , m-1}

Ternary (3 ary) code

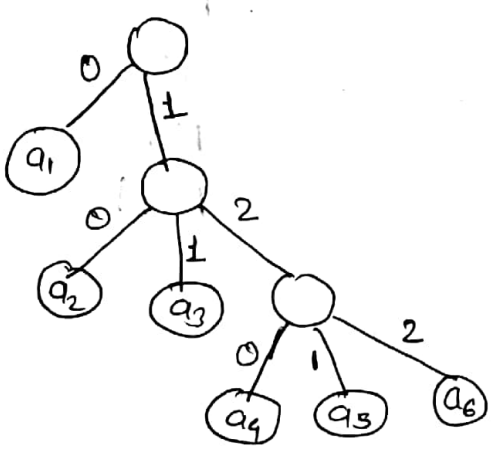
{0, 1, 2}

more ← · → least

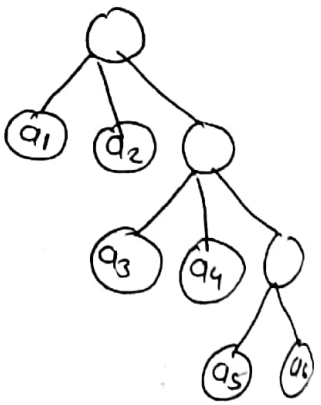
Let A = {$a_1, a_2, a_3, a_4, a_5, a_6$} and

Ternary code (3-ary)

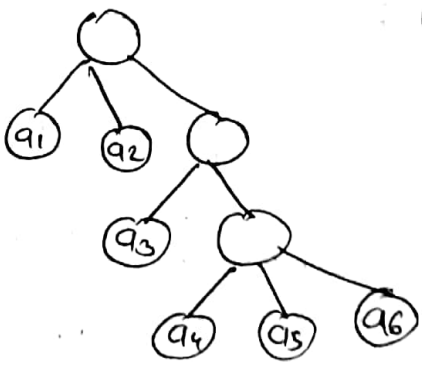
① a_1, a_2, a_3, [a_4, a_5, a_6] ⇒ a_1, a_2, a_3, [a_4'] ⇒ a_1, a_2'



② a_1, a_2, a_3, a_4, [a_5, a_6] ⇒ a_1, a_2, [a_3, a_4, a_5'] ⇒ a_1, a_2, a_3'



③ a_1, a_2, a_3, [a_4, a_5, a_6] ⇒ a_1, a_2, [a_3, d_4] ⇒ a_1, a_2, a_3'



→ M is the size of the alphabet
m is the size of the code symbol.

$2 \le n \le m$

here $n \bmod (m-1) = M \bmod (m-1)$

eg $M = 6 \quad m = 3$ and $n \in \{2, 3\}$

$2 \bmod 2 = 0 \qquad 6 \bmod 2 = 0$

$3 \bmod 2 = 1 \qquad \therefore n = 2$

Ex:- Generate a ternary huffman code for a source of six letter alphabet and probability model

$p(a_1) = p(a_3) = p(a_4) = 0.2$; $p(a_5) = 0.25$, $p(a_6) = 0.1$

$p(a_2) = 0.05$

$M = 6 \quad m = 3$
$n \in \{2, 3\}$
$n = 2$

Letters	Prob.	codeword
a_5	0.25	$c(a_5)$
a_1	0.20	$c(a_1)$
a_3	0.20	$c(a_3)$
a_4	0.20	$c(a_4)$
a_6	0.10	$c(a_6)$
a_2	0.05	$c(a_2)$

a_5	0.25	$c(a_5)$
a_1	0.20	$c(a_1)$
a_3	0.20	$c(a_3)$
a_4	0.20	$c(a_4)$
a_6'	0.15	~~$c(a_6)$~~ α_1

$c(a_6) = \alpha_1 * 0 = 020$

$c(a_2) = \alpha_1 * 1 = 021$

a_5	0.25	$c(a_5)$
a_1	0.20	$c(a_1)$
a_3'	0.55	α_2

$c(a_3) = \alpha_2 * 0 = 00$

$c(a_4) = \alpha_2 * 1 = 01$

$c(a_1) = \alpha_1 = \alpha_2 * 2 = 02$

a_3'	0.55	$\alpha_2 = 0$
a_5	0.25	$c(a_5) = 1$
a_1	0.20	$c(a_1) = 2$

$c(a_1) = 2$
$c(a_2) = 021$
$c(a_3) = 00$
$c(a_4) = 01$
$c(a_5) = 1$
$c(a_6) = 020$

* Mathematical Preliminaries for Lossless Compression

event = set of sample space

self information of an event.

→ Let A be an event and P(A) be the probability of event A occurs. Then the self information event A is denoted by i(A) and is defined as

$$\boxed{i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A)}$$

if b = 2 information is measured in BITS

= e " " " NATS

= 10 " " " HARTLEY

→ Let A & B be two independent events, Then

then $i(AB) = -\log_p P(AB) = -\left[\log_b [P(A)\,P(B)]\right]$

$= -\left[\log P(A) + \log P(B)\right]$

$= -\log P(A) - \log P(B)$

$= i(A) + i(B)$

Ex:- Let H & T be outcomes of flipping a coin.

If the coins is fair then $P(H) = P(T) = \frac{1}{2}$

and $i(H) = -\log P(H)$

$= -\log\left(\frac{1}{2}\right)$

$= -\log 1 + \log 2$

$= \log 2 = 1$ bit

$i(T) = -\log P(T)$

$= -\log\left(\frac{1}{2}\right)$

$= -\log 1 + \log 2$

$= 1$ bit.

⇒ If the coin is not fair and let $P(H) = \frac{1}{8}$, $P(T) = \frac{7}{8}$

Then $i(H) = -\log P(H)$

$= -\log\left(\frac{1}{8}\right)$

$= -\log 1 + \log 8$

$= 3$ bits

$i(T) = -\log P(T)$

$= -\log\left(\frac{7}{8}\right)$

$= -\log 7 + \log 8$

$= 0.153$ bits

⇒ If we have a set of independent events A_i, which are set of outcomes of some experiment such that $\cup A_i = S$ (sample space)

then avg self information of an event associate with random experiment is given by

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

This quantity is called the entropy of associated with the experiment.

First Order Entropy

→ The set of the symbol A is often called the alphabet for the source and the symbol are referred to as letters

For a general source S with alphabet $A = \{1, 2, 3, \ldots, m\}$ that generates a sequence $\{x_1, x_2, \ldots\}$, the entropy is given by

$$H(S) = \lim_{n \to \infty} \frac{1}{n} G_n$$

$$G_n = -\sum_{i_1=1}^{m} \sum_{i_2=1}^{m} \cdots \sum_{i_n=1}^{m} P(x_1 = i_1, x_2 = i_2, x_3 = i_3 \ldots x_n = i_n) \times \log_b P(x_1 = i_1, x_2 = i_2 \ldots x_n = i_n)$$

→ If each element in the sequence is independent and identically distributed (iid) then, we can show that

$$G_n = -n \sum_{i_1=1}^{m} P(x_1 = i_1) \log_b P(x_1 = i_1)$$

then entropy

$$H(S) = \lim_{n \to \infty} \frac{1}{n} \left(-n \sum_{i_1=1}^{m} P(x_1 = i_1) \log_b P(x_1 = i_1) \right)$$

$$H(S) = -\sum_{i_1=1}^{m} P(x_1 = i_1) \log_b P(x_1 = i_1)$$

→ If S is iid

$$H(S) = -\sum_{i=1}^{m} p(x = i) \log P(x = i)$$

↑
First order entropy

→ Generally we don't the entropy of a source what we can do is we can only estimate the entropy of a source.

27/01/2020.

Ex:- Consider the following sequence

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

$P(1) = \frac{1}{16} = P(10) = P(6) = P(7)$

$P(2) = \frac{2}{16} = P(3) = P(4) = P(5) = P(8) = P(9)$

Entropy of source $H(S) = -\sum_{i=1}^{10} P(i) \log P(i)$

$= -4\left(\frac{1}{16} \log \frac{1}{16}\right) + 6\left(\frac{2}{16} \log \frac{2}{16}\right)$

$= 3.25$ bits / symbol.

→ If we assume that the source is not iid, rather if we assume there is a sample to sample correlation

we remove this correlation by taking the difference between the adjacent sample

1 1 -1 -1 1 1 1 -1 1 1 1 1 -1 1 1

now this sequence is independent

$P(1) = \frac{13}{16}$ $\quad P(-1) = 3/16$

Entropy of source $H(S) = -\frac{13}{16} \log \frac{13}{16} - \frac{3}{16} \log \frac{3}{16}$

$= 0.70$ bits / symbol

→ If you know more about the source & data, better you get the compression.

→ Consider the following controlled sequence

1 2 1 2 3 3 3 3 1 2 3 3 3 3 1 2 3 3 1 2

$P(1) = \frac{5}{20} = P(2)$

$P(3) = \frac{10}{20}$

$H(S) = -\frac{5}{25}\log\frac{5}{25} - \frac{5}{20}\log\frac{5}{20} - \frac{10}{20}\log\frac{10}{20}$

$= 2\left(\frac{-1}{4}\log\frac{1}{4}\right) - \frac{1}{2}\log\frac{1}{2}$

$= \frac{-1}{2}\log\frac{1}{16} - \frac{1}{2}\log\frac{1}{2}$

$= -\frac{1}{2}\left(\log\frac{1}{16} + \log\frac{1}{2}\right) = 1.5$ bits/symbol

$1.5 * 20$

30 bits

→ Now we take two symbol as one symbol

$P(12) = \frac{5}{10}$; $P(33) = \frac{5}{10}$

$H(S) = -2\left(\frac{5}{10}\log\frac{5}{10}\right) = -2\left(\frac{1}{2}\log\frac{1}{2}\right)$

$= 1$ bit/symbol

$1 * 10 = 10$ bits

→ Models:-

(i) Physical model :- If you know the data generated process, we can use that info to construct a model. eg. in speech related application, knowledge about the physics of speed production can be used to construct a mathematical model for the sample speech process.

(ii) Models are for certain data can also be obtained through knowledge of the underlying process.

2) ~~Statical~~ Probability Model

i) Ignorance Model (if you don't know abt source)

we assume that each letter that is generated is independent to every other letter and each occurs with equal probability.

(ii) Next step in ~~probability~~ complexity is to keep independence assumption but remove equal probability (if you know sth abt source)

$$p = \{p(a_1), p(a_2) \ldots p(a_m)\}$$

probability of each element is different.

(iii) Remove the independence

↓

Markov Model :-

Discrete Markov Chain

Let $\{x_n\}$ be a sequence of ~~location~~ observation

The sequence is said to follow a k^{th} order Marker Model if

$$p(x_n | x_{n-1}\, x_{n-2} \ldots x_{n-k}) = p(x_n | x_{n-1} x_{n-2} \ldots x_1)$$

→ In other words, knowledge of the past k symbols is equivalent to the knowledge of the entire past history of the process.

The value taken as

$\{x_{n-1}, x_{n-2}, x_{n-3}, \ldots x_{n-k}\}$ are called the states of process

30/1/2020

→ If the size of the source alphabet is l, then no of states = l^K.

→ The most commonly used Marker Model is the first order Marker Model. for which

$$P(x_n | x_{n-1}) = P(x_n | x_{n-1}, \ldots, x_1)$$

→ $x_n = e x_{n-1} + c_n$

Ex. Preceeding

Suppose we have already proceed (received) precedin$g$ ↓ 1/26

precedi$n$

precedin

* bigger the content, probability is greater

Shannon :- 2nd Order content

i) Alphabets consists of 26 english letter + space = 27

He found the entropy of english text is = 3.1 bit / symbol

ii) 2nd order content (2 words)

He found the entropy of english text is 2.4 bit/symbol.

* entropy of source can only be estimated

→ Lower and upper bound prediction done by people

100 order context (block size = 100)
0.6 bits/letters ↳ entropy fall down
1.3 bits/letters

Composite Source Model

A source is written as composite of several sources $S \rightarrow S_1 \cup S_2 \cup S_3 \ldots \cup S_k$
$\downarrow M_1$ $\downarrow M_2$ $\downarrow M_k$
and each may have its own model. and their is switch
$\downarrow$
P_i

The Kraft – Mc. Millan Inequality :-

Theorem: Let C_e be a code with N codewords having length $l_1, l_2 \ldots l_N$. If C_e is uniquely decodable then,

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

and its not for unique decodeable. i.e necessary condition, and vice versa is not possible ie if some code is satisfy this rule, it doesnot mean it is uniquely decodeable

proof:- Consider $(K(C_e))^n$ → if it less than 1, it does not go exponentially

$$= \left(\sum_{i=1}^{N} 2^{-l_i} \right)^n$$

$$= \left(\sum_{i=1}^{N} 2^{-l_i}\right)\left(\sum_{i=1}^{N} 2^{-l_i}\right) \quad \ldots\ldots \text{ n times}$$

$$= \sum_{i_1=1}^{N}\sum_{i_2=2}^{N}\cdots\sum_{i_n=1}^{N} 2^{-(l_{i_1}+l_{i_2}\cdots\cdots l_{i_n})}$$

$\because$ small length of codeword = 1

$\therefore \min(l_{i_1} + l_{i_2} + \ldots\ldots + l_{i_n}) \geq n$

$l = \max\{l_1, l_2 \ldots l_N\}$

then $\max(l_{i_1} + l_{i_2} \ldots + l_{i_n}) \leq nl$

$$\therefore \quad [K(C_e)]^n \leq \sum_{k=n}^{nl} A_k 2^{-k}$$

where A_k is no of combination of n codeword whose sum of the length is equal to k.

claim: $A_k \leq 2^k$

We know that the no of binary sequence of length k is 2^k. Since the code that we have consider is uniquely decodable a binary sequence can at most represent one combination of the codewords.

Therefore $A_k \leq 2^k$.

$$[K(C_e)]^n = \sum_{k=n}^{nl} A_k 2^{-k} \leq \sum_{k=n}^{nl} 2^k \times 2^{-k} = (nl - n + 1)$$

That is $K(C_e)$ is growing linearly with n and not exponentially with n. Hence $k(c)^n \leq 1$

Ex: code 3 code 4 code 5 code 6

	code 3	code 4	code 5	code 6
a_1	0	0	$a_1 - 0$	0
a_2	10	01	$a_2 - 01$	01
a_3	110	011	$a_3 - 11$	10 ↓ not
a_4	111	0111		

uniquely decodable.

for code 3

$l_1 = 1,\ l_2 = 2\ \ l_3 = 3\ \ l_4 = 3$

$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125$

$= 0.75 + 0.250 = 1$ ✓

for code 4

$l_1 = 1\ \ l_2 = 2\ \ l_3 = 3\ \ l_4 = 4$

$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16}$

$= 0.5 + 0.25 + 0.125 + \frac{0.125}{2} = \frac{1 + .50 + .250 + 0.125}{2}$

$= \frac{1.875}{2} = 0.937$

Theorem:- Given a set of integer $l_1, l_2 \ldots l_N$ that satisfy the inequality

$$\sum 2^{-l_i} \leq 1$$

we can always find a fixed codeword lengths $l_1, l_2 \ldots l_N$

Proof:- Without loss of generality, assume that

$l_1 \leq l_2 \leq \ldots \leq l_N$

Define a sequence of number

$$w_1 = 0$$

$$w_j = \sum_{i=1}^{j-1} 2^{l_j - l_i}$$

$$w_2 = 2^{l_2 - l_1}$$

$$w_3 = 2^{l_3 - l_2} + 2^{l_3 - l_1}$$

$$w_4 = 2^{l_4 - l_3} + 2^{l_4 - l_2} + 2^{l_4 - l_1}$$

consider for $j \geq 1$

$$\log_2(w_j) = \log_2 \left(\sum_{i=1}^{j-1} 2^{l_j - l_i} \right)$$

$$\log_2(w_j) = \log_2 \left(2^{l_j} \sum_{i=1}^{j-1} 2^{-l_i} \right)$$

$$= \log_2 2^{l_j} + \log_2 \left(\sum_{i=1}^{j-1} 2^{-l_i} \right)$$

$$= l_j + \log_2 \underbrace{\left(\sum_{i=1}^{j-1} 2^{-l_i} \right)}_{\pi \leq 1}$$ $\log(\leq 1) = -ve$

$$\leq l_j$$

→ If $\lceil \log(w_j) \rceil = l_j$ then the j^{th} codeword c_j is the binary representation of w_j

→ If $\lceil \log(w_j) \rceil < l_j$, then c_j is the binary representation of w_j with $l_j - \lceil \log_2(w_j) \rceil$ zeros appened to the right.

$$C_e = \{ c_1, c_2 \ldots c_N \}$$

Claims : C_e is the prefix code.

Assume not, then there j and k such that $j<k$ and c_j is prefix of c_k

This means that the l_j most significant bits of w_k form the binary representation of w_j.

That is,

$$w_j = \left\lfloor \frac{w_k}{2^{l_k - l_j}} \right\rfloor$$

$\rightarrow$ Consider $\frac{w_k}{2^{l_k - l_j}} = \frac{\sum_{i=1}^{k-1} 2^{l_k - l_i}}{2^{l_k - l_j}} = \sum_{i=1}^{k-1} 2^{l_k - l_i - l_k + l_j}$

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

$$= \underbrace{\sum_{i=1}^{k-1} 2^{l_j - l_i}}_{w_j} + \sum_{i=1}^{k-1} 2^{l_j - l_i} \cdots$$

$$= w_j + 2^{l_j - l_j} + \sum_{i=i+1}^{k-1} 2^{l_j - l_i}$$

$$= w_j + 1 + \sum_{i=i+1}^{k-1} 2^{l_j - l_i}$$

$$\left\lfloor \frac{\log k}{2^{l_k - l_j}} \right\rfloor \geq w_j + 1$$

Hence, the assumption that c_j is a prefix of c_k is wrong. Hence the code c_e is a prefix code.

Length of Huffman Code.

$S \rightarrow H(S)$

Huffman code

$(l_1, l_2 \ldots l_N) \rightarrow \bar{l}$.

$$\boxed{H(S) \leq \bar{l} < H(S) + 1}$$

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

$$H(S) = -\sum_{i=1}^{k} p(a_i) \log_2 p(a_i)$$

$$H(S) - \bar{l} = -\sum_{i=1}^{k} p(a_i) \log_2 p(a_i) - \sum_{i=1}^{k} p(a_i) l_i$$

$$= \sum_{i=1}^{k} p(a_i) \log_2 \frac{1}{p(a_i)} - \sum_{i=1}^{k} p(a_i) l_i$$

$$= \sum_{i=1}^{k} p(a_i) \log_2 \frac{1}{p(a_i)} - \sum_{i=1}^{k} p(a_i) \log_2 2^{l_i}$$

$$= \sum_{i=1}^{k} p(a_i) \log_2 \frac{1}{p(a_i)} + \sum_{i=1}^{k} p(a_i) \log_2 2^{-l_i}$$

$$= \sum_{i=1}^{k} p(a_i) \left[\log_2 \frac{1}{p(a_i)} + \log_2 2^{-l_i} \right]$$

$$= \sum_{i=1}^{k} p(a_i) \log_2 \left(\frac{2^{-l_i}}{p(a_i)} \right)$$

Jer moins Lemma:—

If f is a concave function, then $E(f(x)) \leq f(E(x))$

$$\leq \log_2 \left(\sum_{i=1}^{k} P(q_i) \frac{2^{-l_i}}{P(q_i)} \right)$$

$$\leq \log_2 \left(\sum_{i=1}^{k} 2^{-l_i} \right)$$

$\sum 2^{-l_i} \leq 1$ (Kraft inequality) $\rightarrow \leq 0$.

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

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