Huffman Coding Procedure With an Example.

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

A Bottom-Up Approach

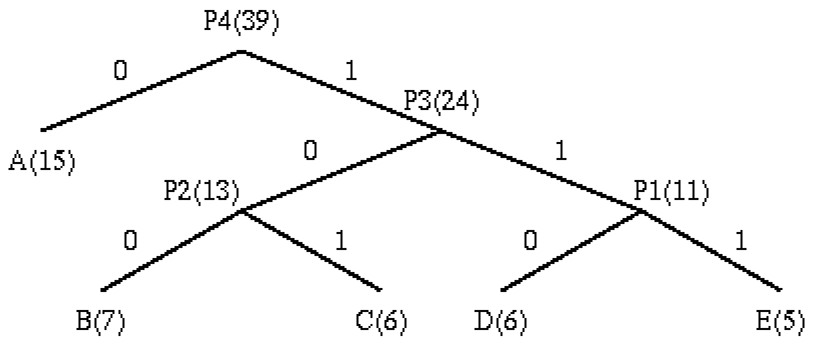
1. Initialization: Put all nodes in an OPEN list, keep it sorted at all times (e.g.,ABCDE).

2. Repeat until the OPEN list has only one node left:

(a) From OPEN pick two nodes having the lowest frequencies/probabilities, create a parent node of them.

(b) Assign the sum of the children's frequencies/probabilities to the parent node and insert it into OPEN.

(c) Assign code 0, 1 to the two branches of the tree, and delete the children from OPEN.



Symbol	Count	log(1/p)	Code	Subtotal (# ofbits)
A	15	1.38	0	15
B	7	2.48	100	21
C	6	2.70	101	18
D	6	2.70	110	18
E	5	2.96	111	15

TOTAL (# of bits): 87

a) HUFFMANN CODING:

i) Order the given symbols in the decreasing probability

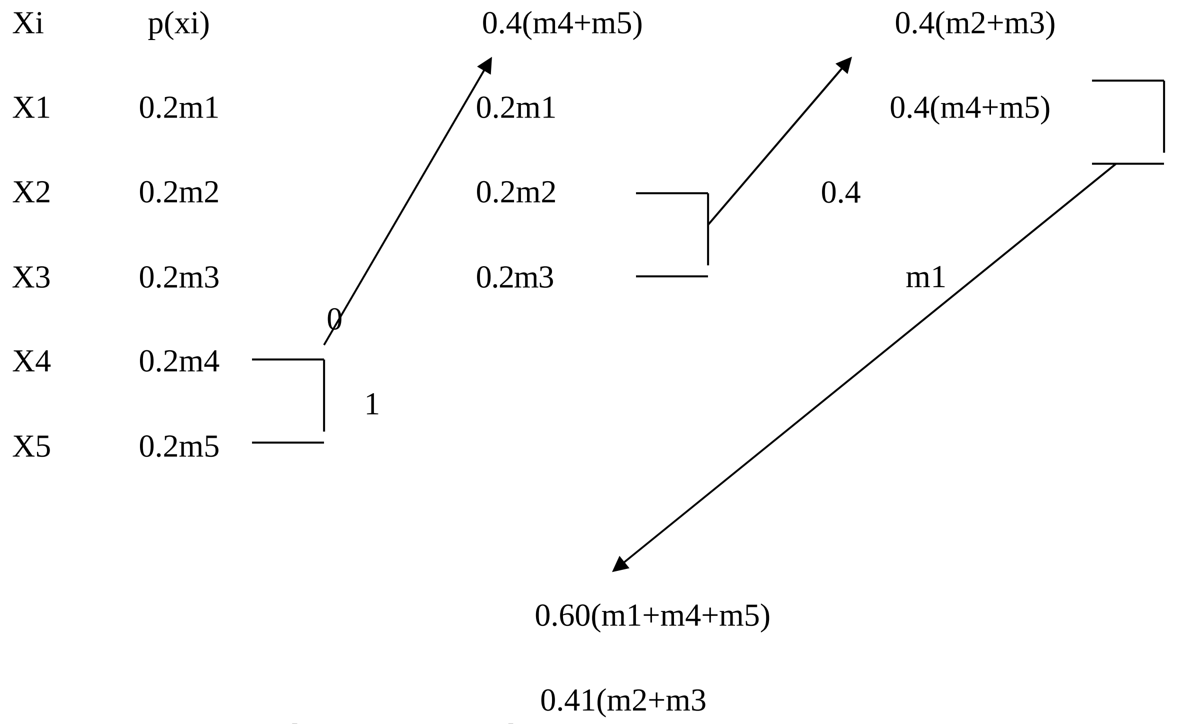
ii) Combine the bottom two or the least probability symbols into a single symbol that replaces in the next source reduction

iii) Repeat the above two steps until the source reduction is left with two symbols per probability

iv) Assign '0' to the first symbol and '1' to the second symbol

Problem:

A DMS x has five equally likely symbols P (x1) = p(x2) = P(X3) =p(x4)= P(x5) =0.2 construct the Huffman code cu calculate the efficiency.



Xi	code	Length
x1	00	2
x2	10	2
x3	11	2
x4	0 0 0	3
x5	0 0 1	3

H(x) = - Σ(5,i=1) p(xi)log(2)p(xi)
= - [(0.2
log2(0.2)X5] H(x)
= 2.32
L = Σ(5,i=1) p(xi)ni
=(0.2)(2+2+2+3+3)
L = 2.4
Therefore %q = H(x)/ Llog$_2$(M) = 2.32/2.4log2(2) = 96.7%

<u>Properties of Huffman coding:</u>

Optimum code for a given data set requires two passes.

1. Code construction complexity O(NlogN).
2. Fast lookup table based implementation.
3. Requires at least one bit per symbol.
4. Average codeword length is within one bit of zero-order entropy (Tighter bounds are known): H □ R □ H+1bit
5. Susceptible to bit errors.

ARITHMETIC CODING:

Basic idea:

Map input sequence of symbols into one single codeword

Codeword is determined and updated incrementally with each new symbol (symbol-by-symbol coding)

At any time, the determined codeword uniquely represents all the past occurring symbols. Codeword is represented by a half-open subinterval [Lc,H]=[0,1].The half-open subinterval gives the set of all code words that can be used to encode the input symbol sequence, which consists of all past input symbols. Any real number within the subinterval [Lc,Hc] can be assigned as the codeword representing the past occurring symbols.

<u>Algorithm:</u>

Let

$S = \{s_0, \ldots, s_{(N-1)}\}$ – source alphabet

$p_k = P(s_k)$ – probability of symbol s_k, $0 \leq k \leq (N-1)$

[Lc,Hc]- Interval assigned to symbol s_k, where $p_k = H_{sk} - L_{sk}$

$$L_{s_k} = \sum_{i=0}^{k-1} p_k = P_{k-1}; \quad 0 \leq k \leq (N-1)$$

$$H_{s_k} = \sum_{i=0}^{k} p_k = P_k; \quad 0 \leq k \leq (N-1)$$

- The subinterval limits can be computed as:

<u>Encoding Procedure:</u>

1. Coding begins by dividing interval [0,1] into N non-overlapping intervals with lengths equal to symbol probabilitiespk
2. Lc = 0 ; Hc = 1
3. Calculate code subinterval length: length = Hc–Lc
4. Get next input symbolsk
5. Update the code subinterval Lc= Lc+length·Lsk
   Hc= Lc+ length·Hsk
6. Repeat from Step 3 until all the input sequence has beenencoded

<u>Decoding procedure:</u>

1. Lc = 0 ; Hc = 1
2. Calculate code subinterval length: length = Hc–Lc
3. Find symbol subinterval , $0 \leq k \leq (N-1)$ suchthat
4. Output symbolsk
5. Update the code subinterval

Lc= Lc+

Length·Lsk Hc=

Lc+ length·Hsk

6. Repeat from Step 2 until last symbol isdecoded.

In order to determine when to stop the decoding:

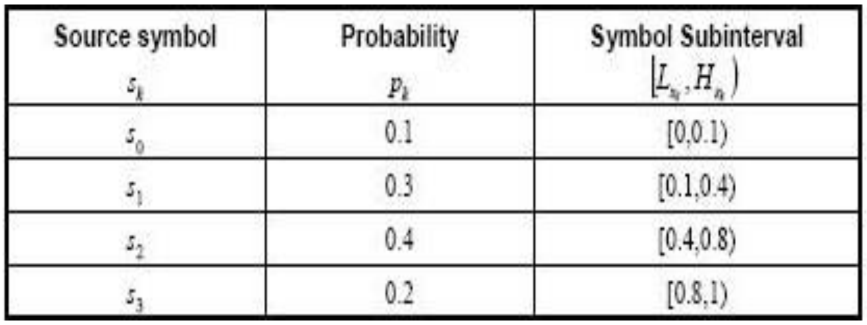
A special end-of-sequence symbol can be added to the sourcealphabet.

If fixed-length block of symbols are encoded, the decoder can simply keep a count of the number of decoded symbols

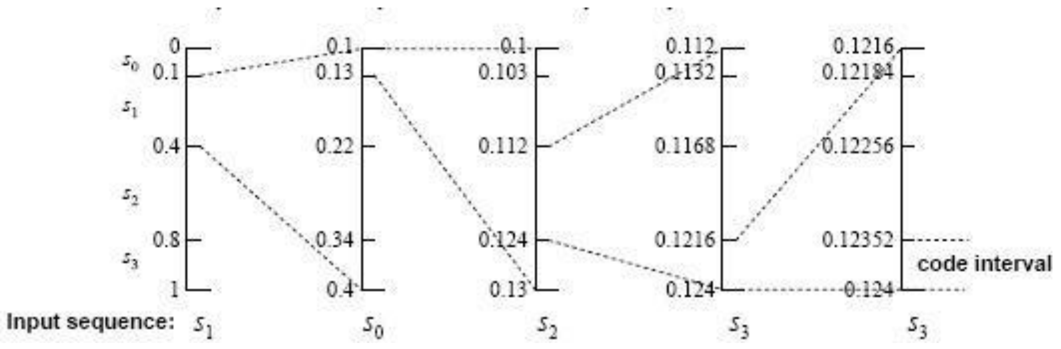
Example:

1. Construct code words using arithmetic coding for the sequence s0,s1,s2,s3 with probabilities

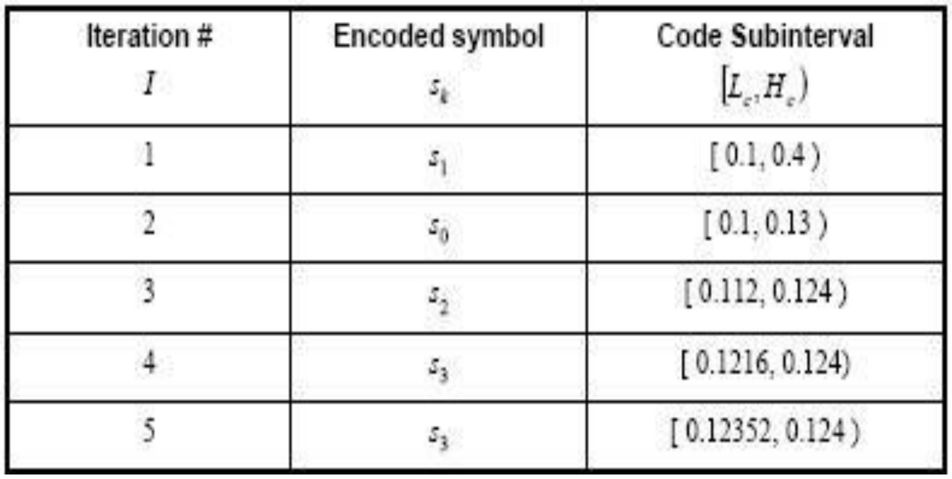
{0.1,0.3,0.4,0.2}



Source symbol s_k	Probability p_k	Symbol Subinterval $[L_{s_k}, H_{s_k})$
s_0	0.1	[0,0.1)
s_1	0.3	[0.1,0.4)
s_2	0.4	[0.4,0.8)
s_3	0.2	[0.8,1)



Answer:



Iteration # I	Encoded symbol s_k	Code Subinterval $[L_c, H_c)$
1	s_1	[0.1, 0.4)
2	s_0	[0.1, 0.13)
3	s_2	[0.112, 0.124)
4	s_3	[0.1216, 0.124)
5	s_3	[0.12352, 0.124)