Advanced Digital Communication

Source Coding

1. Consider a discrete memoryless source whose alphabet consists of K equiprobable symbols. What conditions have to be satisfied by K and the code-word length for the coding to be 100 percent? [Haykin 9.9b]

2. Consider the four codes listed below:

Symbol	Code I	Code II	Code III	Code IV
s_0	0	0	0	00
s_1	10	01	01	01
s_2	110	001	011	10
s_3	1110	0010	110	110
s_4	1111	0011	111	111

   (a) Two of these four codes are prefix codes. Identify them, and construct their individual decision trees.

   (b) Apply the Kraft-McMillan inequality to codes I, II, III, and IV. Discuss your results in light of those obtained in part (a). [Haykin 9.10]

3. A discrete memoryless source has an alphabet of seven symbols whose probabilities of occurrence are as described here:

Symbol	s_0	s_1	s_2	s_3	s_4	s_5	s_6
Probability	0.25	0.25	0.125	0.125	0.125	0.0625	0.0625

   Compute the Huffman code for this source, moving a "combined" symbol as high as possible. Explain why the computed source code has an efficiency of 100 percent. [Haykin 9.12]

4. Consider a discrete memoryless source with alphabet $\{s_o, s_1, s_2\}$ and statistics $\{0.7, 0.15, 0.15\}$ for its output.

   (a) Apply the Huffman algorithm to this source. Hence, show that the average code-word length of the Huffman code equals 1.3 bits/symbol.

   (b) Let the source be extended to order two. Apply the Huffman algorithm to the resulting extended source, and show that the average code-word length of the new code equals 1.1975 bits/symbol.

   (c) Compare the average code-word length calculated in part (b) with the entropy of the original source. [Haykin 9.13]

5. A computer executes four instructions that are designated by the code words (00, 01, 10, 11). Assuming that the instructions are used independently with probabilities (1/2. 1/8, 1/8. 1/4), calculate the percentage by which the number of bits used for the instructions may be reduced by use of an optimum source code. Construct a Huffman code to realize the reduction. [Haykin 9.15]

6. Consider the following binary sequence

   11101001100010110100…

   Use the Lempel-Ziv algorithm to encode this sequence. Assume that the binary symbols 0 and 1 are already in the codebook.

Solution

1. Source entropy is

$$H(S) = \sum_{k=0}^{K-1} p_k \log_2\left(\frac{1}{p_k}\right)$$

$$= \sum_{k=0}^{K-1} \frac{1}{K} \log_2 K = \log_2 K$$

$$\eta = \frac{H(S)}{\bar{L}} = \frac{\log_2 K}{l_0}$$

   For 100% efficiency, we have

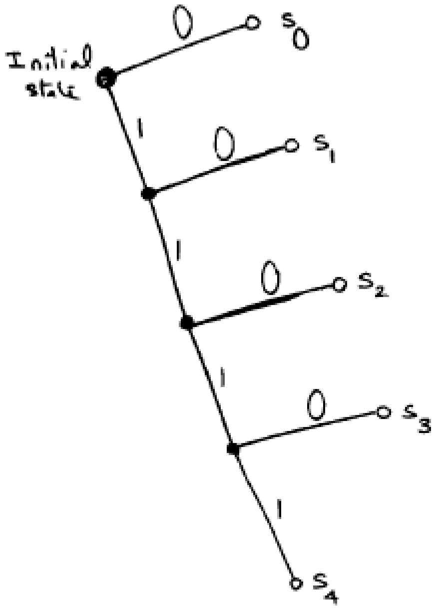
$$l_0 = \log_2 K$$

   To satisfy this equation, we choose

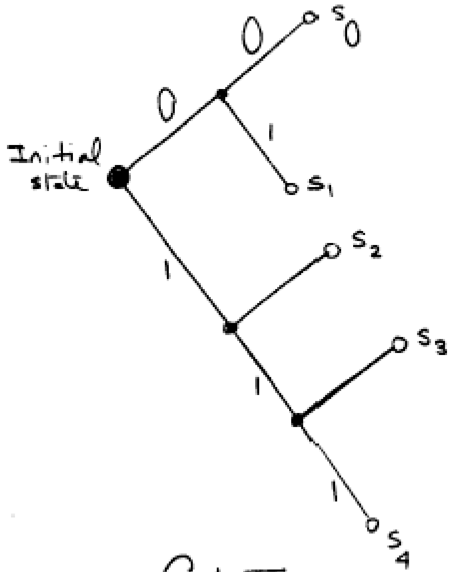
$$K = 2^{l_0}$$

2. (a)

A prefix code is defined as a code in which no code work is the prefix of any other code word. By inspection, we see therefore that codes I and IV are prefix codes, whereas codes II and III are not.

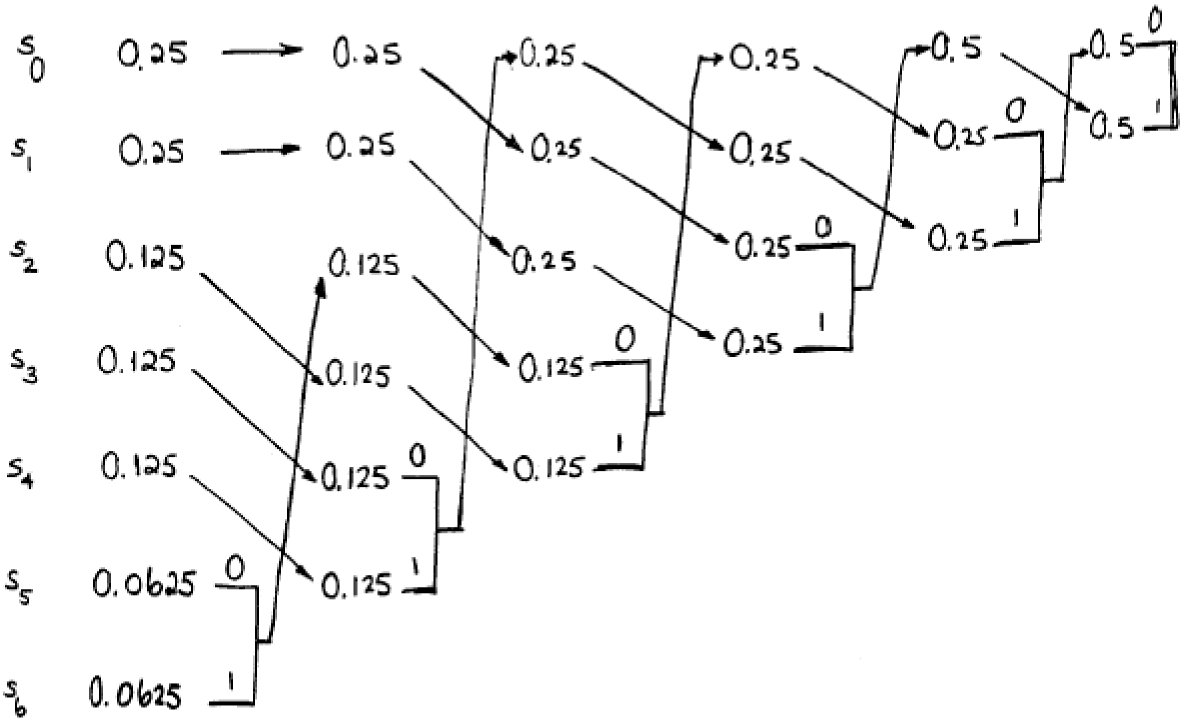


Code I



Code IV

3.



The Huffman code is therefore

s_0	1 0
s_1	1 1
s_2	0 0 1
s_3	0 1 0
s_4	0 1 1
s_5	0 0 0 0
s_6	0 0 0 1

$$\bar{L} = \sum_{k=0}^{6} p_k l_k$$

$$= 0.25(2)(2) + 0.125(3)(3) + 0.0625(4)(2)$$

$$= 2.625$$

$$H(S) = \sum_{k=0}^{6} p_k \log_2\left(\frac{1}{p_k}\right)$$

$$= 0.25(2) \log_2\left(\frac{1}{0.25}\right) + 0.125(3) \log_2\left(\frac{1}{0.125}\right)$$

$$+ 0.0625(2) \log_2\left(\frac{1}{0.0625}\right)$$

$$= 2.625$$

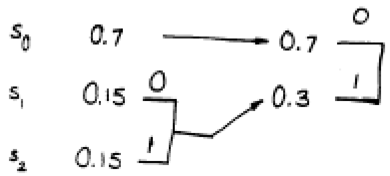
$$\eta = \frac{H(S)}{\bar{L}} = \frac{2.625}{2.625} = 1$$

We could have shown that the efficiency of the code is 100% by inspection since

$$\eta = \frac{\sum_{k=0}^{6} p_k \log_2(1/p_k)}{\sum_{k=0}^{6} p_k l_k}$$

where $l_k = \log_2(1/p_k)$.

4. (a)



s_0 0
s_1 1 0
s_2 1 1

The average codeword length $L_{av} = \sum_{i=0}^{2} p_i L_i = 0.7 \times 1 + ... = 1.3$ bits/symbol

(b) For the extended source, we have

$p(s_0 s_0) = 0.7 \times 0.7 = 0.49, p(s_0 s_1) = 0.105, p(s_0 s_2) = 0.105,$
$p(s_1 s_0) = 0.105, p(s_1 s_1) = 0.0225, p(s_1 s_2) = 0.0225,$
$p(s_2 s_0) = 0.105, p(s_2 s_1) = 0.0225, p(s_2 s_2) = 0.0225$

Applying the Huffman algorithm to the extended source, we obtain the following source code:

$s_0 s_0$: 1
$s_0 s_1$: 001
$s_0 s_2$: 010
$s_1 s_0$: 011
$s_1 s_1$: 000100
$s_1 s_2$: 000101
$s_2 s_0$: 0000
$s_2 s_1$: 000110
$s_2 s_2$: 000111

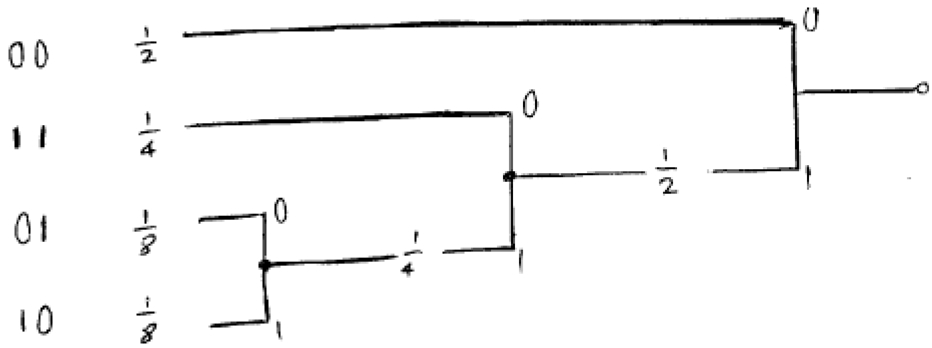
The average codeword length $L_{av} = 0.49 \times 1 + ... = 2.395$ bits/symbol

$$L_{av} / 2 = 1.1975 \text{ bits/symbol}$$

(c) source entropy $H(X) = -0.7 \log_2 0.7 - 0.15 \log_2 0.15 - 0.15 \log_2 0.15 = 1.18$ bits/symbol

Therefore, $H(X) \le \frac{L_{av}}{n} \le H(X) + \frac{1}{n}$

5.



Computer code	Probability	Huffman Code
0 0	$\frac{1}{2}$	0
1 1	$\frac{1}{4}$	1 0
0 1	$\frac{1}{8}$	1 1 0
1 0	$\frac{1}{8}$	1 1 1

The number of bits used for the instructions based on the computer code, in a probabilistic sense, is equal to

$$2\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8}\right) = 2 \text{ bits}$$

On the other hand, the number of bits used for instructions based on the Huffman code, is equal to

$$1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 3 \times \frac{1}{8} = \frac{7}{4}$$

The percentage reduction in the number of bits used for instruction, realized by adopting the Huffman code, is therefore

$$100 \times \frac{1/4}{2} = 12.5\%$$

6.

Initial step

Subsequences stored: 0

Data to be parsed: 1 1 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 0 0 ...

Step 1

Subsequences stored: 0, 1, 11

Data to be parsed: 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 1 0 0 ..

Step 2

Subsequences stored: 0, 1, 11, 10

Data to be parsed: 1 0 0 1 1 0 0 0 1 0 1 1 0 1 0 0 ....

Step 3

Subsequences stored: 0, 1, 11, 10, 100

Data to be parsed: 1 1 0 0 0 1 0 1 1 0 1 0 0 ...

Step 4

Subsequences stored: 0, 1, 11, 10, 100, 110

Data to be parsed: 0 0 1 0 1 1 0 1 0 0 ...

Step 5

Subhsequences stored: 0, 1, 11, 10, 100, 110, 00

Data to be parsed: 1 0 1 1 0 1 0 0 ....

Step 6

Subsequences stored: 0, 1, 11, 10, 100, 110, 00, 101

Data to be parsed: 1 0 1 0 0 ...

Step 7

Subsequences stored: 0, 1, 11, 10, 100, 110, 00, 101, 1010

Data to be parsed: 0 ....

Numerical positions	1	2	3	4	5	6	7	8	9
Subsequences	0,	1,	11,	10,	100,	110,	00,	101,	1010
Numerical representations			22,	21,	41,	31,	11,	42,	81
Binary encoded blocks			0101,	0100,	0100,	0110,	0010,	1001,	10000