ADVANCED COMMUNICATION SYSTEMS

ELEN90051 (LECTURER: MARGRETA KUIJPER)

Source coding for discrete sources

1st Semester 2018

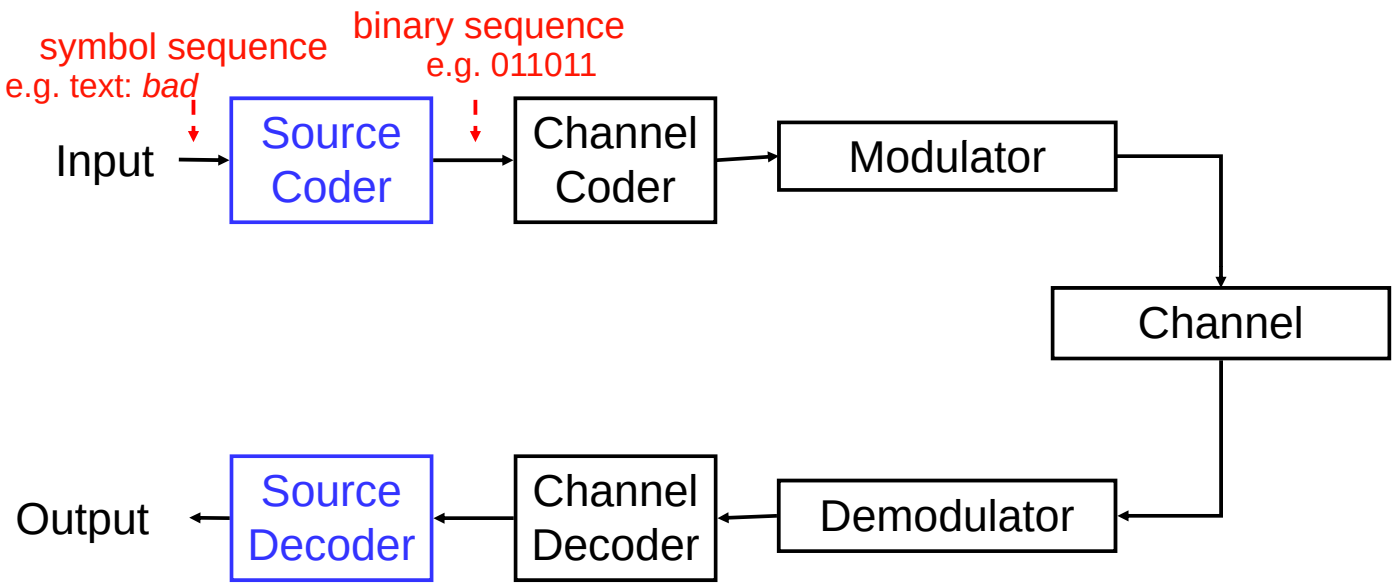
Written by Margreta Kuijper; see Chapter 6 of "Digital Communications" by Proakis & Salehi, 2008

All scanned tables and text are from the textbook "Digital Communications" by Proakis and Salehi, 2008

SOURCE CODING

- Source coding is the process by which the output of a source is efficiently converted to a sequence of bits.
- For analog sources this requires a first step of analog-to-digital conversion, involving sampling and quantization
- In this subject we assume that sampling/quantization has already happened so that we are dealing with a discrete source. Our main focus is on efficient source coding for discrete sources that is lossless; we want the encoder to produce as few bits as possible, subject to the need to reconstruct the source data without error at the decoder end.

SOURCE CODING PUT INTO CONTEXT:



EXAMPLE APPLICATIONS OF SOURCE CODING:

- Morse coding
- Facsimile coding
- Image coding
- Video coding
- Audio coding
- Speech coding
- File coding

Compression techniques that are used are: runlength coding, Huffman coding, transformation techniques of the Fourier transform type, predictive coders, perceptive coding, Lempel Ziv coding, arithmetic coding.

Later we will look at image coding and video coding in much more detail via guest lecturer Dr. Michael Biggar.

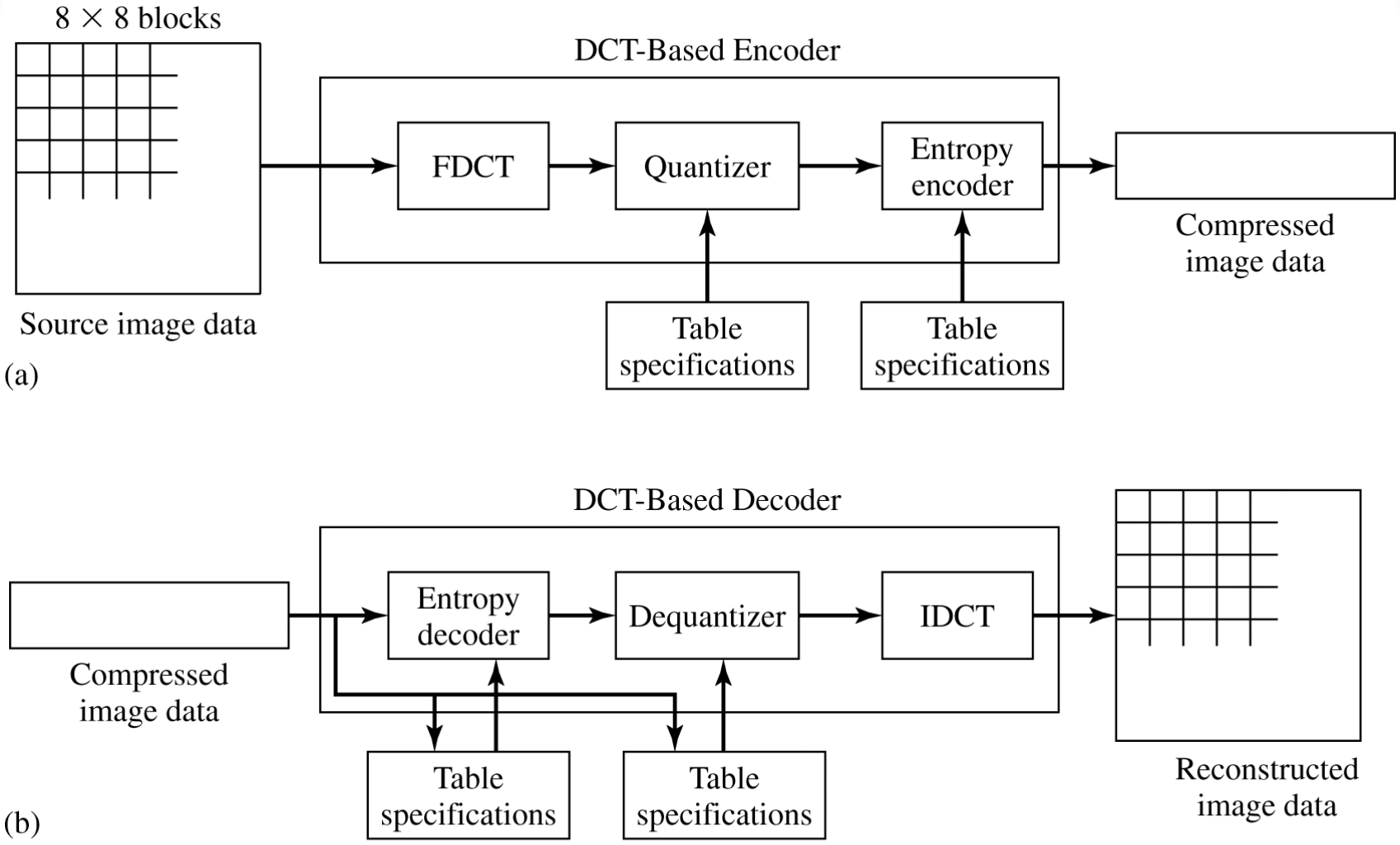


Figure 6.36

The block diagram of a JPEG encoder.

- We first focus on the source. How can we judge one source as outputting more information than another source? How can we quantitatively measure information?
- 1948 Claude Shannon (Bell Labs, USA) publishes paper "A Mathematical Theory of Communication"
- The paper is the start of the area of "Information Theory"; it contains two fundamental theorems:
  - SOURCE CODING THEOREM (also called noiseless coding theorem; in this part of the lecture notes)
  - CHANNEL CODING THEOREM (also called noisy coding theorem; in later lectures)

The year 2016 marks the 100th birthday of Claude Shannon! Just google "Shannon Centenary" to find many outreach events around the globe. Actually....Shannon should be as wellknown to the general public as Einstein as he is the founder of the theory behind the amazing communication devices that we now take for granted.

What is "information"?

Compare the following statements:

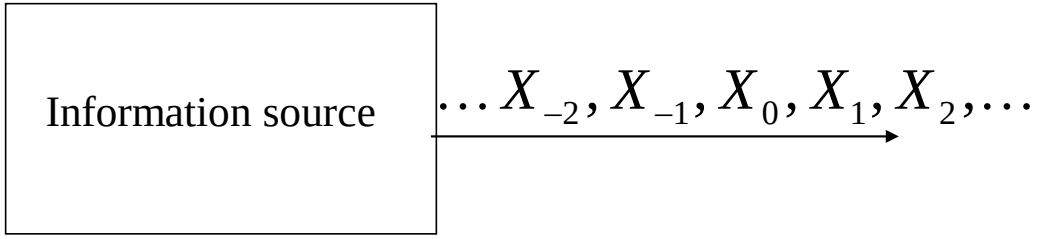
1. I will eat some food tomorrow
2. I will win 1 million dollars tomorrow

Which statement conveys more information?

- "Information" must have some uncertainty in it, otherwise it makes it makes no sense to transmit it...
- Therefore we model information as a random process

RECALL:

- Random process = collection of random variables $X(t)$
- Here we are only interested in discrete-time random processes, $\cdots, X_0, X_1, X_2, \cdots$, these are sequences of random variables
- Also, for all $i \in \mathbb{Z}$ our random variables X_i are assumed **discrete**, that is, they take their values in a finite alphabet



DEFINITION

A **discrete source** is a discrete-time random process $\cdots, X_0, X_1, X_2, \cdots$ where all X_i's are discrete random variables

Example: english text

- if all X_i's are independent from one another then = Discrete Memoryless Source (DMS)
- Is "english text" a DMS?

FIGURE: from a local Melbourne sports club, 2018

Note: from now on all X_i's are assumed identical and we use the letter X sloppily for either the random variable or the random process....

Example: Binary Source

is defined as a source X that takes values in $\{0, 1\}$ with corresponding probabilities q and $1 - q$

If $q = 0.5$ it is called a Binary Symmetric Source (BSS)

Thought experiment A: tossing a coin 10 times

$$P(X = HEAD) = q \qquad P(X = TAIL) = 1 - q$$

Think about how to encode the outcome into bits with zero error for the situations

1. $q = 0.5$ fair coin
2. $q = 0.1$ biased coin
3. $q = 0$

THOUGHT EXPERIMENT B: ROLLING A FAIR DICE MANY TIMES

Think about how to encode the outcome into bits with zero error.....

- Think of a symbol-by-symbol fixed length encoder. How many bits per symbol does your encoder use?
- How does the corresponding decoder operate?
- Is it possible to use less bits per symbol by encoding J symbols at a time for some integer $J > 1$? (**Exercise**: derive a lower bound as well as an upperbound in terms of J for the average number of bits per symbol used)
- What would you change if the dice is biased?

Consider a DMS X. How can we measure the information content of X?

Again compare:

1. I will eat some food tomorrow
2. I will win 1 million dollars tomorrow

Which statement conveys more information?

Let X be a random variable with values in $\{a_1, \ldots, a_N\}$ with corresponding probabilities $p_1, p_2, \ldots, p_N$

Definitions

The **self information** in outcome a_i is defined as

$$h(a_i) := -\log_2 p_i$$

The **entropy** of X is defined as the expected self information:

$$H(X) := -\sum_{i=1}^{N} p_i \log_2 p_i$$

The entropy expresses "how random" a random variable is...

EXAMPLE: ROLLING A FAIR DICE

$$h(a_1) = \cdots = h(a_6) = \log_2 6; \qquad H(X) = \log_2 6$$

TUTE QUESTION 4.1

Consider a DMS X that takes values in $\{a_1, a_2, \ldots, a_N\}$, uniformly distributed. Compute the entropy $H(X)$.

TUTE QUESTION 4.2—BIASED DICE

Consider a DMS X that takes values in $\{a_1, a_2, a_3, a_4, a_5, a_6\}$, with corresponding probabilities $p_1 = 0.30$, $p_2 = 0.20$, $p_3 = 0.20$, $p_4 = 0.15$, $p_5 = 0.10$ and $p_6 = 0.05$. Compute the entropy and compare with the entropy of the previous example of a fair dice.

EXAMPLE: BINARY SOURCE

Consider a DMS X that takes values in $\{0, 1\}$ with corresponding probabilities p and $1 - p$. Then

$$H(X) = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

For $p = 0.1$: $H(X) \approx 0.47$.

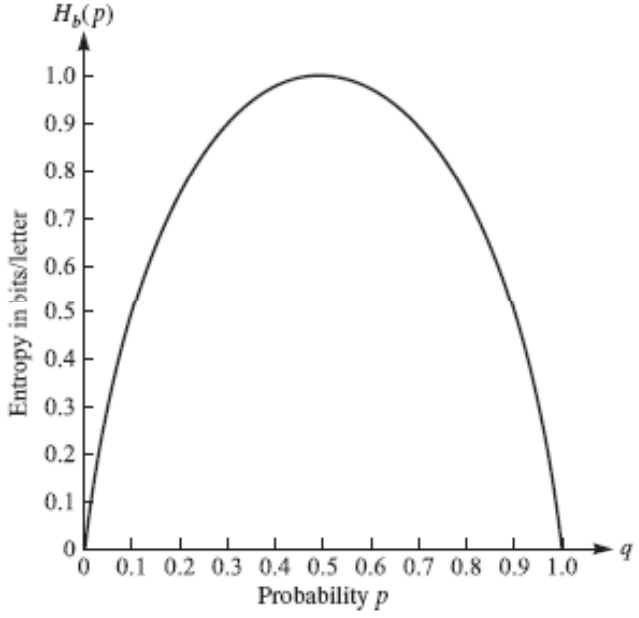


FIGURE 6.2–1
The binary entropy function.

TUTE QUESTION 4.3

Let X and Y be independent r.v's. Show that

$$H(XY) = H(X) + H(Y)$$

TUTE QUESTION 4.4

Let X be a random variable and let n be a positive integer. Then X^n is the socalled *n'th extension* of X. Show that

$$H(X^n) = nH(X)$$

So adding an extra symbol amounts to increasing the entropy by $H(X)$ bits.

Let X be a DMS with entropy $H(X)$

The average number of bits used per symbol is called the **rate** of the source code. Consider all possible rates at which X can be compressed without loss of information

Does a fundamental limit exist?

SOURCE CODING THEOREM: There exists a lossless source code for X at any rate R (in bits/symbol) if

$$R > H(X).$$

Furthermore, there exists no lossless source code for X at rates $< H(X)$; we call a code **optimal** if $R = H(X)$.

Note that the theorem only sets a fundamental limit, it doesn't tell us how to find such a source code!

For the interested reader: read sect. 6.3-1 from textbook [PS08] to get an idea about the proof....

Quiz

Consider a DMS X, taking values in $\{a_1, a_2, a_3, a_4\}$, uniformly distributed.

Question 1: Can this source be compressed beyond 2 bits/symbol?

Suppose we use a symbol-by symbol variable length source encoder:

$$\begin{aligned} a_1 &\rightarrow 0 \\ a_2 &\rightarrow 10 \\ a_3 &\rightarrow 110 \\ a_4 &\rightarrow 111 \end{aligned}$$

Question 2: Is it possible to come up with a decoder for this scheme?

Question 3: What is the expected bit length if this encoder is used? Is it a good idea to use this encoder?

Up till now we only encountered two types of good practical source coding:

- symbol-by-symbol fixed length coding
- fixed-to-fixed length coding

The advantage of these methods is that the decoder is simple: all it needs to do is to segment the received sequence into fixed length blocks. But the compression may not always be that good.....

EXAMPLE

Consider a DMS X that takes values in $\{a_1, a_2, a_3, a_4\}$, with corresponding probabilities $p_1 = 1/2$, $p_2 = 1/4$, $p_3 = 1/8$, and $p_4 = 1/8$. Then

$$\begin{aligned} H(X) &= -p_1 \log_2 p_1 - p_2 \log_2 p_2 - p_3 \log_2 p_3 - p_4 \log_2 p_4 \\ &= 1/2 + 1/2 + 3/8 + 3/8 = 1.75 \end{aligned}$$

Can fixed-to-fixed length coding lead to a compression rate near 1.75 bits/symbol?

No, we need to be looking at variable length coding. But we also need to make sure that the decoder can do its job!

The practical source coding methods that we will look into:

HUFFMAN CODING

- invented in 1952 by the student Huffman via his assignment solution
- used in for example facsimile, JPEG

LEMPEL-ZIV CODING

- invented in 1977 (another version in 1978)
- used in for example file compression (gzip, pdf conversion)

ARITHMETIC CODING

- patented by IBM in 1978
- used in for example text compression and image/video compression

Another practical method is runlength coding.

Quiz

We require the decoder to be **instantaneous**, that is, not have any decision delays.

Answer the following questions for all source codes in the table (so Code I, Code II and Code III):

Question 1: Is it possible to come up with a decoder for this code?

Question 2: What is the expected bit length if this code is used? Is it a good idea to use this code?

TABLE 6.3–1
Variable-Length Codes.

Letter	$P[a_k]$	Code I	Code II	Code III
a_1	$\frac{1}{2}$	1	0	0
a_2	$\frac{1}{4}$	00	10	01
a_3	$\frac{1}{8}$	01	110	011
a_4	$\frac{1}{8}$	10	111	111

QUIZ OUTCOMES

- Code I is not uniquely decodable, try decoding the received sequence 001001.....
- Code II has the tree structure below. It is a **prefix code**, that is, none of its codewords is the prefix of another codeword. This makes the code **uniquely decodable** and **instantaneous**.
- Code III is clearly not a prefix code. In fact it is uniquely decodable

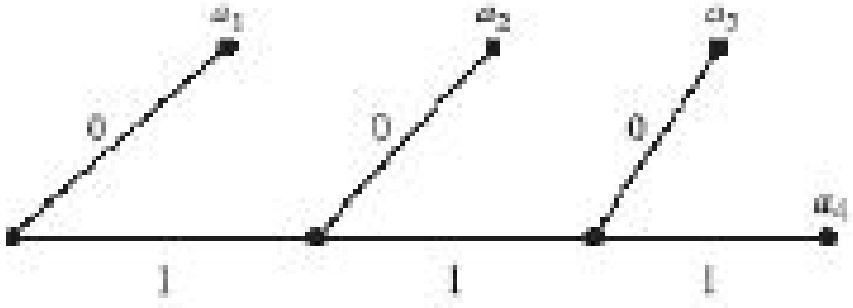


FIGURE 6.3–1
Code tree for code II in Table 6.3–1.

Let X be a DMS with entropy $H(X)$

Because of the source coding theorem we know that we can get a compression rate arbitrarily close to $H(X)$. But can we do this via a prefix code?

SOURCE CODING THEOREM FOR PREFIX CODES:
There exists a prefix code for X with average bit length $\bar{R}$ satisfying

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

The above theorem drives the search for good variable length prefix codes, such as Huffman codes. It turns out that Huffman codes satisfy the above theorem.

For the interested reader: read sect. 6.3-2 from textbook [PS08] to get an idea about the proof....

SYMBOL-BY-SYMBOL HUFFMAN CODING

The main idea behind Huffman codes is already in older codes, such as Morse code:

MAIN IDEA

Choose variable length codewords such that more probable symbols have shorter codewords.

The ingenuity of Huffman's design is the use of a tree structure.

EXAMPLE

Consider the following DMS; it has entropy 2.11.

EXAMPLE 6.3–1. Consider a DMS with seven possible symbols $x_1, x_2, \ldots, x_7$ having the probabilities of occurrence illustrated in Figure 6.3–4. We have ordered the source symbols in decreasing order of the probabilities, i.e., $P(x_1) > P(x_2) > \cdots > P(x_7)$. We begin the encoding process with the two least probable symbols x_6 and x_7. These two symbols are tied together as shown in Figure 6.3–4, with the upper branch assigned a 0 and the lower branch assigned a 1. The probabilities of these two branches are added together at the node where the two branches meet to yield the probability 0.01. Now we have the source symbols $x_1, \ldots, x_5$ plus a new symbol, say x_6', obtained by combining x_6 and x_7. The next step is to join the two least probable symbols from the set $x_1, x_2, x_3, x_4, x_5, x_6'$. These are x_5 and x_6', which have a combined probability of 0.05. The branch from x_5 is assigned a 0 and the branch from x_6' is assigned a 1. This procedure continues until we exhaust the set of possible source letters. The result is a code tree with branches that contain the desired code words. The code words are obtained by beginning at the rightmost node in the tree and proceeding to the left. The resulting code words are listed in Figure 6.3–4. The average number of binary digits per symbol for this code is $\bar{R} = 2.21$ bits per symbol. The entropy of the source is 2.11 bits per symbol.

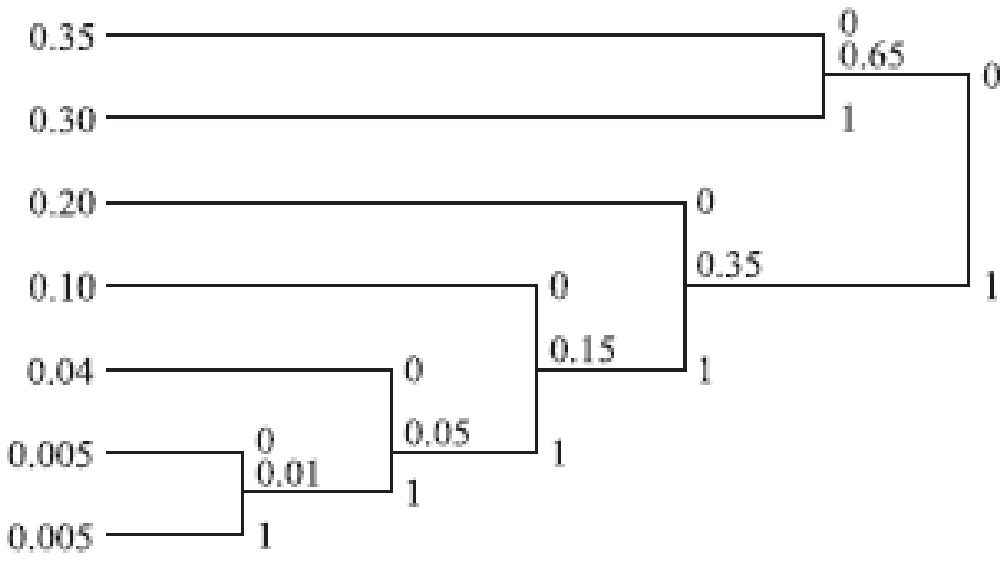


FIGURE 6.3–4
An example of varia
encoding for a DMS

Note that for one particular source X there may be several alternative Huffman codes—a Huffman code is not unique

In fact, for the previous example we can easily come up with an equivalent Huffman code: here's an example (check for yourself that this code has the same efficiency as the previous Huffman code)

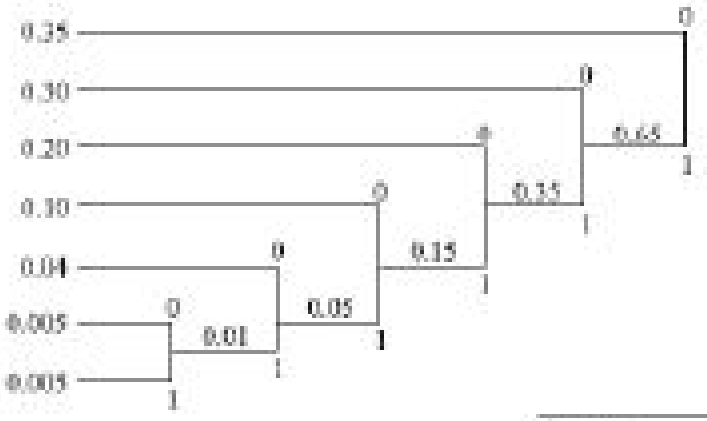


FIGURE 6.3–5
An alternative code for the DMS in Example 6.3–1.

Letter	Code
x_1	0
x_2	10
x_3	110
x_4	1110
x_5	11110
x_6	111110
x_7	111111

$\bar{R} = 2.21$

TUTE QUESTION 4.5

(I) Is the Huffman coding in the above example optimal?

(II) Can you think of a DMS Y with 4 symbols for which the Huffman code is optimal?

(III) Let N be a positive integer. Can you think of a DMS Z with N symbols for which the Huffman code is optimal?

TUTE QUESTION 4.6 (*advanced*)

Consider a DMS X that takes values in $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, with corresponding probabilities $p_1 = 0.22$, $p_2 = 0.18$, $p_3 = 0.17$, $p_4 = 0.15$, $p_5 = 0.13$, $p_6 = 0.10$ and $p_7 = 0.05$. Design a ternary Huffman code, using 0, 1 and 2 as letters. What is the resulting average codeword length? Compare the average codeword length with the entropy in a meaningful way.

AN APPLICATION: JPEG

- lossy compression of still images
- first uses Discrete Cosine Transform (DCT)
- then quantization
- then (optional) Huffman encoding
- image quality can be increased by increasing bits/pixel rate

Later in this subject, you will learn many more details on JPEG by guest lecturer Dr. Michael Biggar.

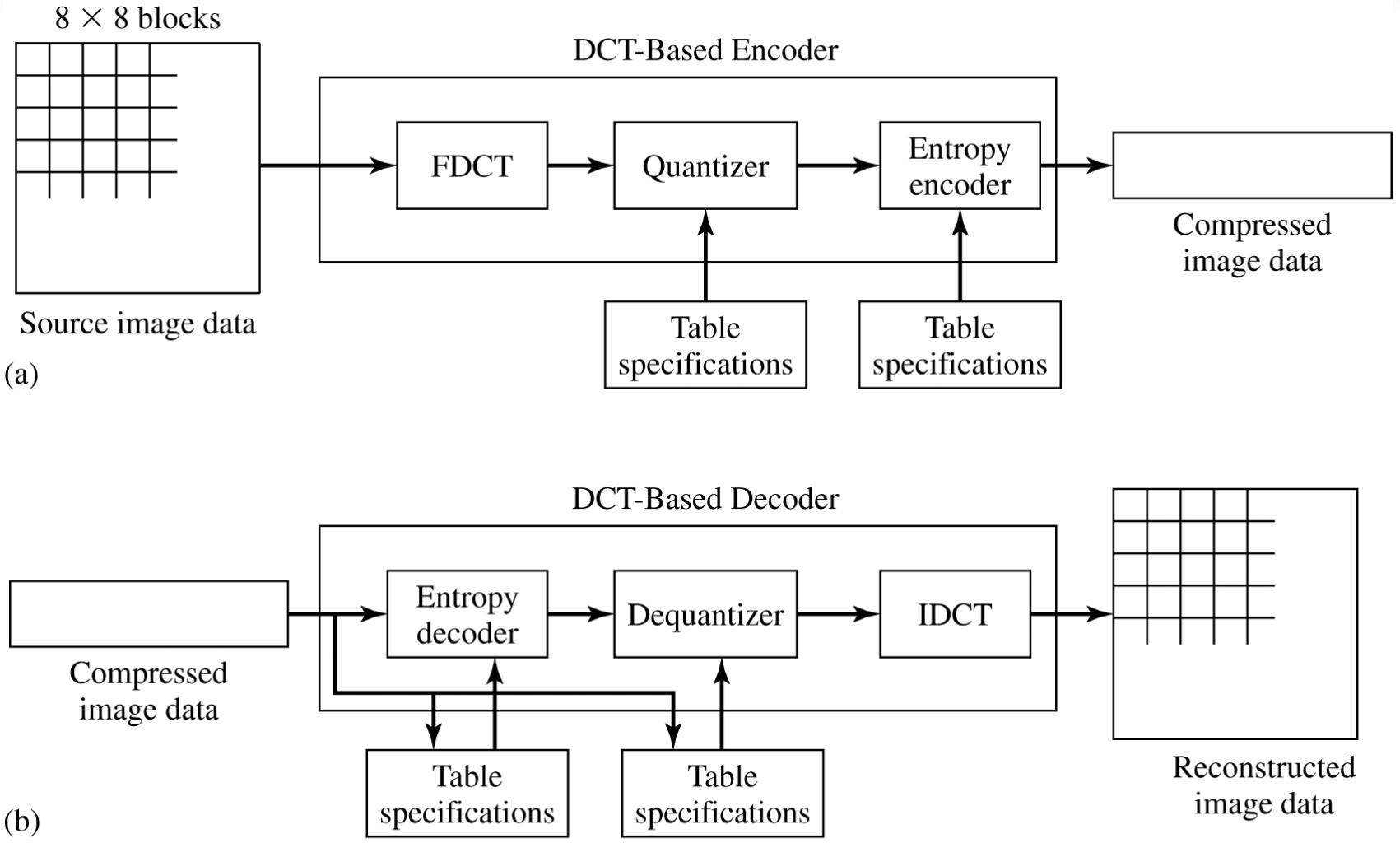


Figure 6.36

The block diagram of a JPEG encoder.

TUTE QUESTION 4.7

Consider the DMS X that takes values in the alphabet $\{-5,-3,-1,0,1,3,5\}$, with corresponding probabilities $0.05, 0.10, 0.10, 0.15, 0.05, 0.25, 0.30$.

1. Calculate $H(X)$.
2. Assume that the source is quantized as

$$\begin{aligned} q(-5) &= q(-3) = -4 \\ q(-1) &= q(0) = q(1) = 0 \\ q(3) &= q(5) = 4 \end{aligned}$$

Calculate the entropy of the quantized source.

Quiz

Consider a DMS X that takes values in $\{a_1, a_2, a_3\}$, with corresponding probabilities $p_1 = 0.45$, $p_2 = 0.35$ and $p_3 = 0.20$. Compute $H(X)$ and construct a Huffman code. Is your code optimal?

QUIZ OUTCOMES

$H(X) = 1.513$ whereas the code's average bit length $\bar{R}_1 = 1.55$ bits/symbol. The code's rate comes close to the entropy but is not optimal. The code efficiency is 97.6%.

How to improve on the efficiency?

TABLE 6.3–2
Huffman code for Example 6.3–3

Letter	Probability	Self-information	Code
x_1	0.45	1.156	1
x_2	0.35	1.520	00
x_3	0.20	2.330	01

$H(X) = 1.513$ bits/letter
$\bar{R}_1 = 1.55$ bits/letter
Efficiency = 97.6%

Fixed-to-variable Huffman coding

The efficiency of Huffman coding is increased by encoding blocks of J symbols at a time.

Reasoning why

Then, since Huffman codes satisfy the Source Coding Theorem for Prefix Codes, we have

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

Since $H(X^J) = JH(X)$ by Tute question 4.4, this implies that

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

In terms of $\bar{R} :=$ the average number of bits per symbol this translates as

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

Clearly $\bar{R}$ comes arbitrarily close to $H(X)$ by selecting J sufficiently large.

EXAMPLE

Again consider the DMS X of the previous example; it takes values in $\{a_1, a_2, a_3\}$, with corresponding probabilities $p_1 = 0.45$, $p_2 = 0.35$ and $p_3 = 0.20$. Recall that $H(X) = 1.513$.

Encoding pairs of symbols (so $J = 2$) gives the following result:

TABLE 6.3-3
Huffman code for encoding pairs of letters

Letter pair	Probability	Self-information	Code
x_1x_1	0.2025	2.312	10
x_1x_2	0.1575	2.676	001
x_2x_1	0.1575	2.676	010
x_2x_2	0.1225	3.039	011
x_1x_3	0.09	3.486	111
x_3x_1	0.09	3.486	0000
x_2x_3	0.07	3.850	0001
x_3x_2	0.07	3.850	1100
x_3x_3	0.04	4.660	1101

$2H(X) = 3.026$ bits/letter pair
$\bar{R}_2 = 3.0675$ bits/letter pair
$\frac{1}{2}\bar{R}_2 = 1.534$ bits/letter
Efficiency = 98.6%

For $J = 1$ the efficiency was 97.6%; now, for $J = 2$ the efficiency is increased to 98.6%

QUESTION:

What happens to the efficiency as J increases further? And can you think of a disadvantage of using a large value of J?

TUTE QUESTION 4.8

Consider a DMS X that takes values in $\{0, 1\}$ with corresponding probabilities 0.9 and 0.1

Recall that $H(X) = 0.47$. How to approach $H(X)$ by J'th extension Huffman coding? Construct a table for $J = 1$, $J = 2$ and $J = 3$.

TUTE QUESTION 4.9

Consider a DMS X that takes values in $\{a_1, a_2, a_3, a_4\}$, with corresponding probabilities $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.2$ and $p_4 = 0.1$.

1. Compute the entropy $H(X)$.
2. What is the minimum required average codeword length to represent this source for error-free reconstruction?
3. Design a (symbol-by-symbol) Huffman code and compare its average codeword length with $H(X)$.
4. Design a Huffman code for the 2nd extension of the source ($J = 2$, taking two symbols at a time). What is the average codeword length? What is the average number of bits per source symbol?
5. Which scheme is more efficient, the one in (3) or the one in (4)?

ARITHMETIC CODING

Just like Huffman coding, source statistics are required. Unlike Huffman coding, arithmetic coding is stream coding, its encoder uses previous information in the symbol sequence.

MAIN IDEA

Map the received symbol sequence of probability p into a subinterval of the real-number interval [0,1] of length p. Specify this subinterval by its binary representation bit sequence.

WHY DOES THIS LEAD TO COMPRESSION?

It takes more bits to specify a small subinterval (<–> unlikely symbol sequences) than a large subinterval (<–> likely symbol sequences).

- Arithmetic coding has very good performance, approaching entropy. Unlike Huffman coding, the number of bits per symbol is not required to be an integer value, which is an advantage.
- Arithmetic coding lends itself well to sources with memory because it can easily incorporate conditional probabilities. Similarly it can also easily handle time-varying source statistics.

EXAMPLE ARITHMETIC ENCODING

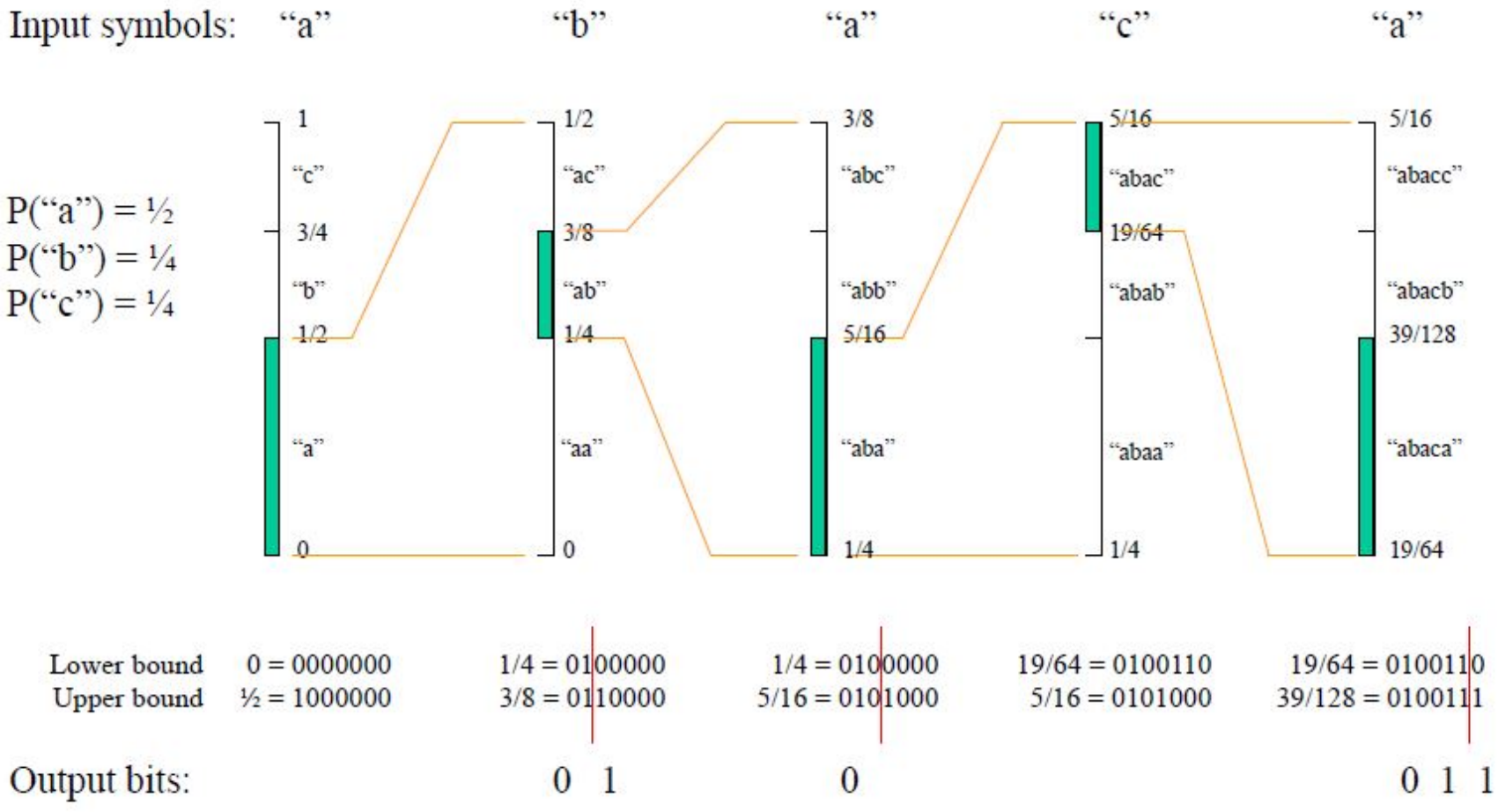
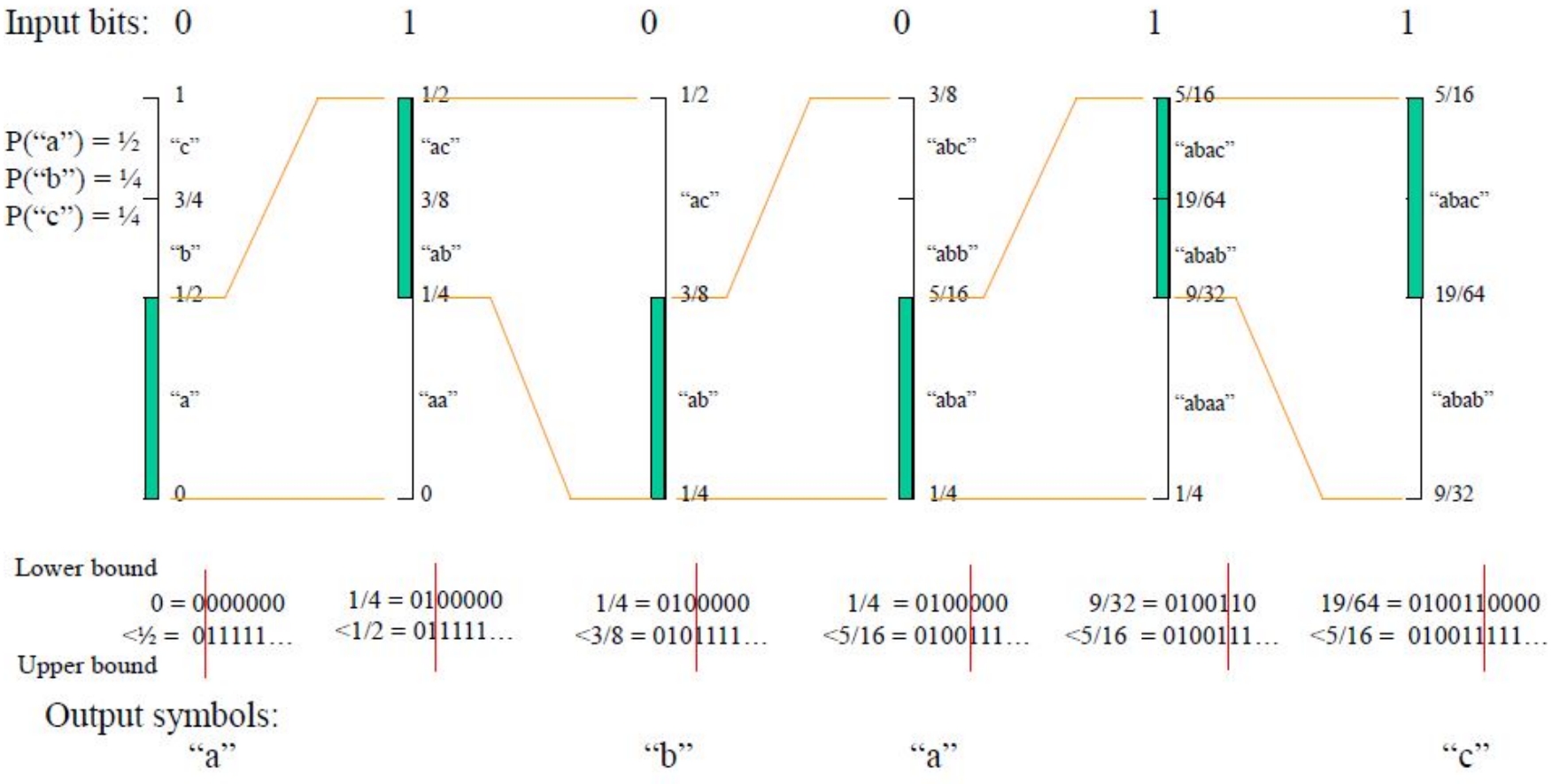


FIGURE: from "Video Processing and Communications" by Wang, Ostermann and Zhang, Prentice Hall 2002

EXAMPLE ARITHMETIC DECODING



(figure by Michael Biggar)

LEMPEL-ZIV CODING

Just like arithmetic coding, LZ coding belongs to the family of stream codes. Unlike Huffman coding and arithmetic coding LZ coding does not require any source statistics.

There are two main versions of Lempel-Ziv coding: LZ77 and LZ78.

MAIN IDEA

Encode a sequence of variable length blocks into a sequence of fixed-length codewords by using previous information.

HOW?

By constructing a dictionary tailored to the received symbol sequence.

WHY DOES THIS LEAD TO COMPRESSION?

The dictionary assigns bits in a "stingy" manner, only when it encounters new symbol information, thereby implicitly getting a grasp on the source statistics.

EXAMPLE LZ78

Consider a binary DMS X of which we don't know the statistics. Suppose that the source outputs the following binary sequence of length 44

$$10101101001001110101000011001110101100011011$$

LZ78 parses the sequence as

$$1\ 0\ 10\ 11\ 01\ 00\ 100\ 111\ 010\ 1000\ 011\ 001\ 110\ 101\ 10001\ 1011$$

It constructs a dictionary as in the Table on the next page (codeword = address of previous matching phrase + new bit).

Question: how long is the encoded sequence in the above example? What is the average bit length per source bit?

Note: Lempel-Ziv's efficiency improves with the length of the source sequence

TABLE 6.3–4
Dictionary for Lempel-Ziv algorithm

	Dictionary location	Dictionary contents	Code word
1	0001	1	00001
2	0010	0	00000
3	0011	10	00010
4	0100	11	00011
5	0101	01	00101
6	0110	00	00100
7	0111	100	00110
8	1000	111	01001
9	1001	010	01010
10	1010	1000	01110
11	1011	011	01011
12	1100	001	01101
13	1101	110	01000
14	1110	101	00111
15	1111	10001	10101
16		1011	11101

TUTE QUESTION 4.10

Perform LZ78 encoding on the binary source sequence

000100100000011000010000000100000010100001000000110100000001100

Hint: you require two passes through the sequence to decide on the size of the dictionary

A FEW FINAL WORDS ON SOURCE CODING

- We talked about discrete sources. What about continuous sources? How many bits per symbol required for lossless compression? Infinitely many, so loss of information must occur in practice!
- If a continuous source is sampled, quantized and then losslessly encoded then, overall, there is loss of information.
- There is a trade-off between compression rate and level of distortion. This leads to rate distortion theory (not covered here):

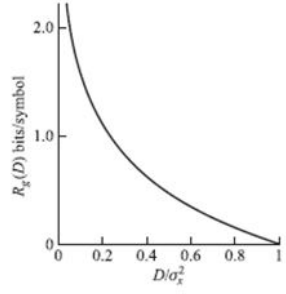
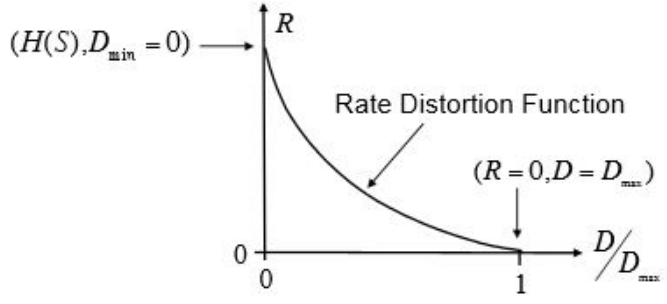


FIGURE 6.4–1
Rate distortion function for a continuous-amplitude, memoryless Gaussian source.



- Many applications tolerate lossy source coding, for example image coding and video coding, see next lectures by guest lecturer Dr. Michael Biggar.