Chapter 5. Coded Communication-Introduction

Table of Contents

Introduction to Coded Communication

Context

This unit follows on from units 3 and 4.

Introduction

While reliable data communication provides exactly the service needed for certain things, such as transferring files that have already been created from one computer to another, many applications

actually require the reliable transfer of symbols from one computer to another. Electronic mail requires the transfer of a sequence of alphanumeric symbols, sending an image requires the transfer of a sequence of colour symbols (the colours of the pixels in the image) and to send a graphic requires the transfer of a sequence of object symbols (the objects comprising the graphic, such as lines, rectangles and so on).

As long as the symbols are drawn from a finite repertoire, they can be coded. Once each symbol has been assigned a binary code, the data representing the coded message can be transferred over a data link.

This unit deals with the coding process. It begins with codes in which symbols are assigned binary codes of the same length. The ASCII code is a well-known example of such a code, and the transmission of an e-mail can be explained with its use.

Actually, such codes are, in general, not efficient. They do, however, have the essential property of unique decodability as well as the desirable one of instantaneous decodability. The Huffman algorithm is introduced as a way of devising optimal codes, in which the assigned codes have different lengths, and which also possess the requisite properties. Codes obtained in this way increase the efficiency not only of the transmission of coded symbols but also of their storage. In the case of storage, the increase in efficiency is referred to as data compression.

Objectives

At the end of this module, you should be able to:

- capture precisely the idea of a code, and the requirements of a code;
- design useful codes and optimum codes;
- carry out design exercises involving the need for coding.

Content

In parallel with this unit, you should read the relevant part of your textbooks.

Introduction

Many communication problems require the transmission of the symbols coming from some source. Computer networks, and most modern communication networks, carry data. This means that before transmission, the symbols must be transformed into data. This is done by a coding process in which each symbol is represented by a data sequence. This pattern of events is captured by the model of the communication process introduced in the first lecture. The Source emits symbols, which must be sent over the Channel which can only convey data. The Transmitter is needed to act as a coder to make this possible.

A few examples may help in illustrating this.

1. Message applications

   In a messaging application such as e-mail, a text source in effect emits a stream of alphabetic symbols that make up each item of e-mail. The symbols are coded one by one using the ASCII code, a part of which is:

symbol	**8-digit code**
A	01000001
B	01000010
C	01000011

symbol	8-digit code
D	01000100
E	01000101
...	...

This makes it possible to send e-mail over a data network.

2. Control applications

   IIn a control application, commands need to be sent to the device under control. Imagine a buggy on the moon which can be commanded to move forward, to turn right, and so on. The commands can be seen as 'command symbols' issuing from a 'command source'. They can be sent across a data network as long as each command is given a code. One possible code is:

command symbol	8-digit code
Command A	01000001
Command B	01000010
Command C	01000011
Command D	01000100
Command E	01000101
...	...

3. Monitoring applications

   In a monitoring application, various states need to be reported. Think of a remote weather station that reports once an hour that conditions are fine, rainy or stormy. The monitor acts as a source of state descriptions, or state symbols, and these can be sent over a data network as long as they are coded. One possible code is:

state symbol	8-digit code
state A	01000001
state B	01000010
state C	01000011
state D	01000100
state E	01000101
...	...

These examples show that there is a common structure to many types of application. The code used in each example is either the ASCII code itself or a variation of it.

The ASCII code is a prime example of a fixed-length code in which all the code words have the same, fixed, length. This kind of code makes things simple, but there are often alternative ways of coding symbols that are better.

To Do

Do Review Questions 1 and 2.

Sources

This section examines sources. With reference to the model of the communication process, the source is the component that produces the items to be communicated. In the case of coded communications,

these items are symbols, and, as we have seen, the communication symbols is at the heart of a wide range of applications.

It might appear that a source can be described by the number, n, of symbols it can emit, and the symbols themselves, s1, s2, . . .sn.

The description of, for example, a binary source, would be:

n=2; s1 = 0, s2 = 1

A capital letter source adequate to produce, say, e-mails typed using capital letters only would be described by:

n=27; s1 = A, s2 = B, . . . s26 = Z, s27= |space|.

To Do

Do Review Question 3.

But this description is not sufficient to capture the characteristics of a source completely, as is illustrated by noting that although a fair dice and a loaded dice can both be described by:

n=6; s1 = 1, s2 = 2, . . . s6 = 6

they are not the same source. (How could someone with a loaded dice take advantage of someone who thinks it is a fair dice otherwise?)

The difference is that the frequencies of occurrence of the symbols are different in the two cases. For a fair dice, the symbols all occur with the same frequency:

fr(s1) = 1/6, fr(s2) = 1/6, . . . fr(s6) = 1/6

For a loaded dice, the frequencies are not the same. They could be, for example:

fr(s1) = 1/2, while fr(s2) = 1/10, . . . fr(s6) = 1/10

So, to specify a source we need to know how many symbols there are, what the symbols are, and how frequently each symbol occurs. {Note that the frequency can be expressed as a percentage as well as a fraction. The frequencies for the loaded dice can be written as fr(s1) = 50%, while fr(s2) = 10%, . . . fr(s6) = 10%.}

To Do

Do Review Questions 4 and 5.

Designing codes

Fixed-length codes result from a simplistic approach to code design. Their design uses only the number of code words of a source to produce the code. The design rule is: code words of length 1 will suffice if the number of code words is 2 or less; code words of length 2 will suffice if the number of code words is 4 or less; code words of length 3 will suffice if the number of code words is 8 or less; and so on. (Another way of saying this is that the length of the code words log2 n rounded up to the nearest integer.)

By way of example, consider the following command source:

Symbol	**Frequency**
Forward	25%
Back	25%

Symbol	Frequency
Right	20%
Left	20%
Reverse	10%

Since n=5, a length of 2 is not sufficient, but a length of 3 is. The code can be:

Symbol	Code Word
Forward	000
Back	001
Right	010
Left	011
Reverse	111

The length of a code is defined as the average length of its code words. In this case, the length is obviously 3 bits/symbol.

Note that the frequencies have not been used at all in designing this code, just as the frequencies of occurrence of letters and so on are not taken into account with the ASCII code. This is a way of saying that some of the information about the source has not been used in determining a code for the source. This implies that the code obtained is not as well matched to the source as it might be.

Exercise 1

Complete the design of the coding for this remote command system so that commands can be unambiguously communicated across a data link in the following form:

Forward s and Back s, where s is the number of units to be moved and takes a value from 1 to 50. Right d and Left d, where d is the number of degrees to be turned through. Reverse is a shorthand for Right 180.

You can find a discussion of Exercise 1 at the end of the chapter.

Actually, imposing a uniform rigid format like this for all the commands would be inefficient. The forward and back commands need only have numbers in the range 1 to 50, whereas the commands left and right need numbers from 1 to 180 (or, arguably 1 to 360). The reverse command, of course, needs no attached number at all. One solution to this is to devise a specific format along the lines of that in the illustration for each command. The resulting formats could be:

Command	Format
Forward	3 digits for command (000) + 6 digits for number of units
Back	3 digits for command (001) + 6 digits for number of units
Right	3 digits for command (010) + 8 digits for number of units
Left	3 digits for command (011) + 8 digits for number of units
Reverse	3 digits for command (111) only

Properties of codes

We know that a fixed-length design for a code for a source does not use all the information about the source, so that there is scope for improving on this code. Further, in the length of a code we have a

measure of its efficiency (the number of bits that must be sent for each symbol). This provides a way of comparing codes to see if one is better than another.

Codes cannot be designed arbitrarily: they need to have certain properties. They must be uniquely decodable, that is to say, once each symbol has been coded there must be only way of decoding it and that must give the original symbol. It is also desirable in some circumstances that codes should be instantaneously decodable. A code is instantaneously decodable if each of its code words can be decoded as soon as all its bits have been received.

Fixed-length codes are inherently uniquely decodable as long as each symbol is assigned a different pattern of the same length. Each distinct pattern will then decode to distinct original symbol. They are also instantaneously decodable because each code word can be decoded as soon as the fixed number of digits has been received.

The following table gives five different codes for the same source.

Source Codes		**Symbol Frequency**				
Symbol	Frequency	Code 1	Code 2	Code 3	Code 4	Code 5
red	50%	0	0	0	00	0
blue	25%	11	01	10	01	10
green	12.5%	00	011	110	10	110
violet	12.5%	1	0111	110	11	111

To Do

Do Review Questions 6, 7 and 8.

Notice that what causes Code 5 to be the best, even among the codes that are both uniquely and instantaneously decodable, is that the symbols that occur most frequently are assigned the shortest code words, so that the most frequently transmitted code words are also the shortest ones. The code words get longer as the frequency with which they are to be transmitted increases, so that the long code words are transmitted the least frequently. In essence, the code is the best because it is the one that is best matched to the source.

A simple test for instantaneous decodability is: given that a code is uniquely decodable, it is instantaneously decodable as long as none of its code words is the prefix of another. The prefixes of, for example, 1010 are 1, 10 and 101.

To Do

Do Review Questions 9 and 10.

Designing optimum codes

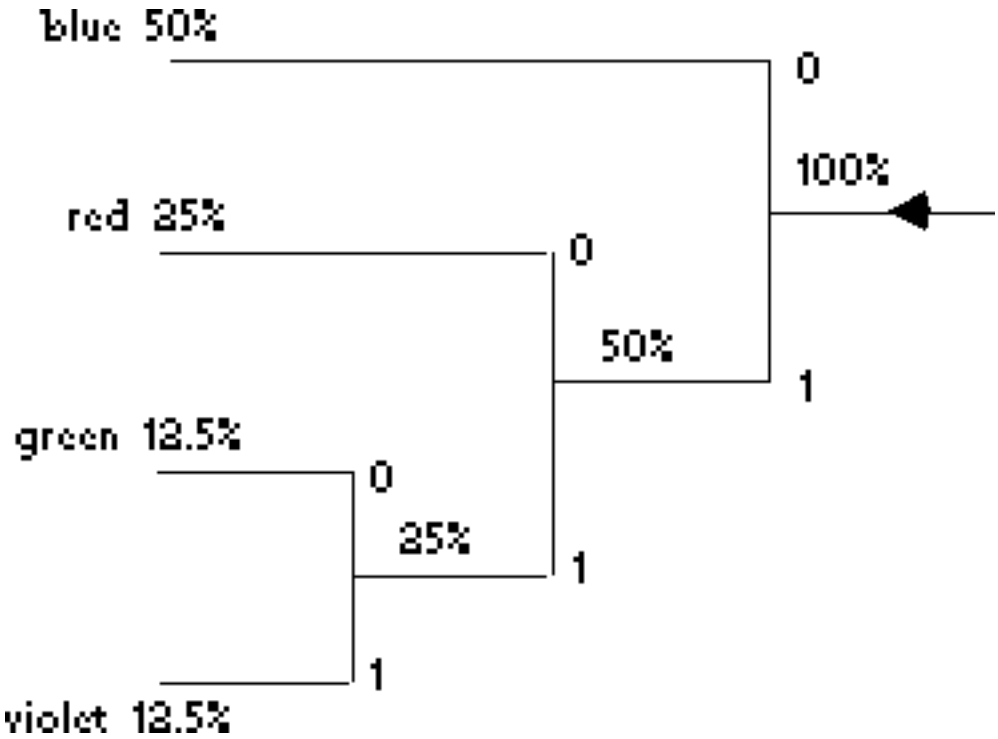
Fortunately, there is a way to design an optimum code for a given source that is both uniquely decodable and instantaneously decodable. It entails the use of the Huffman algorithm. The steps of the algorithm are:

1. Place the symbols of the source in a column as in the diagram below.
2. Repeatedly combine the two symbols with the lowest frequencies until all the symbols have been merged.
3. Label the diagram with 0 and 1 at each branching point, as shown.
4. Obtain the code for each symbol by starting at the right and reading the digits passed by the path to that symbol.

If this algorithm is applied to the source described by:

Symbol	Frequency
Blue	50%
Red	25%
Green	12.5%
Violet	12.5%

it gives the following:



The code can be read as:

Symbol	Code Word
Blue	0
Red	10
Green	110
Violet	111

The length of this code is:

1*0.5 + 2*0.25 + 3*0.125 + 3*0.125 = 1.75 bits/symbol

A fixed-length code for this source could be, for example:

Symbol	Code Word
Blue	00
Red	01
Green	10
Violet	11

The fixed-length code has a length of 2 bits/symbol.

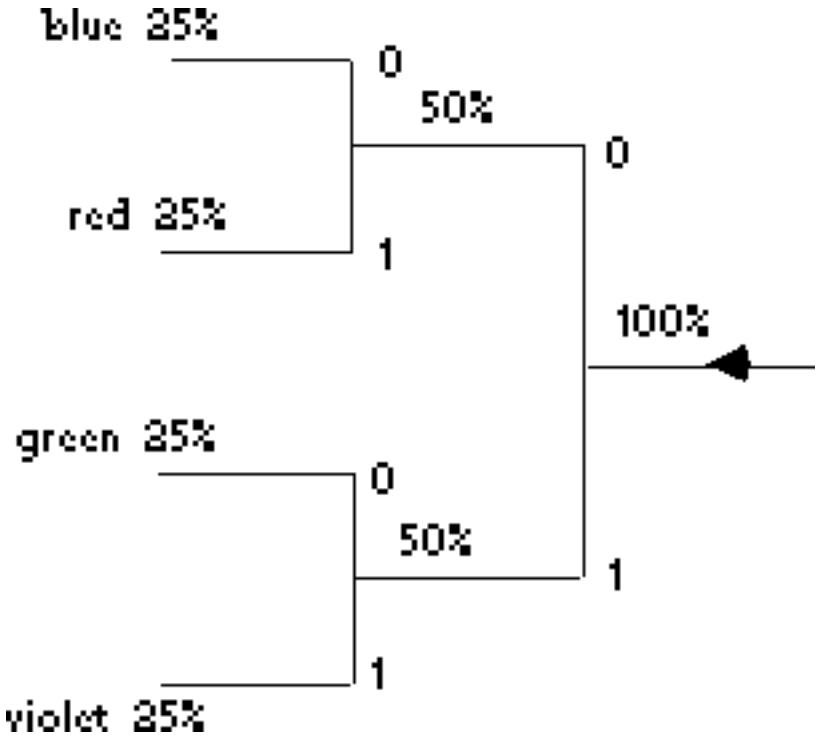
We can now compare the two codes. The ratio of the lengths is 1.75:2. Another way to put this is to say that the length of the Huffman code is 87.5% that of the fixed length code. The Huffman code, therefore, requires on average the transmission of 12.5% fewer bits to transmit the same message. By the same token, it is 12.5% more efficient than the fixed-length code.

The efficiency can also be expressed in terms of what is required to store the data. Again, the Huffman code requires on average the storage of 12.5% fewer bits to store the same message. With reference to storage, then, it is 12.5% more efficient than the fixed-length code. One could equally say that it achieves a data compression of 87.5% relative to the fixed length code.

It is important to notice that the tree structure generated by the process of successively combining symbols is driven by the frequencies of the symbols possessed by the source and that it does not always take the form shown in the example. Consider what happens with this source:

Symbol	**Frequency**
Blue	25%
Red	25%
Green	25%
Violet	25%

This time the algorithm produces:



The code resulting code is:

Symbol	**Frequency**
Blue	00
Red	01
Green	10
Violet	11

Encouragingly, the algorithm has produced code words of the same length for symbols with equal frequencies. But note carefully that the different frequencies drove the algorithm to produce a tree of a different shape.

To Do

Do Review Question 11.

Review Questions

Review Question 1

Devise a code for a number of colours.

You can find an answer/comment for this review question at the end of the chapter.

Review Question 2

How many colours can be represented using an eight-digit code such as that given in the answer to Review Question 1?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 3

Describe the following information sources by giving the number of symbols they possess and the symbols themselves:

1. Semaphore source. A semaphore symbol is made with two flags each of which can, independently, assume one of eight positions.

2. Punch card source. Holes can be punched in any of 10 rows and 80 columns on the card, and any combination of punched holes is possible.

3. A colour television source. The source emits pictures. Assume that a picture consists of 625 lines, each with 800 distinguishable horizontal positions, and that each position can assume one of 16 colours.

You can find an answer/comment for this review question at the end of the chapter.

Review Question 4

Give complete descriptions of the following sources:

1. The punch card source of the previous question when all possible punched cards are equally likely to occur.

2. An alphabetic source that emits the characters necessary to create an alphabetic text.

You can find an answer/comment for this review question at the end of the chapter.

Review Question 5

Given a typical alphabetic text, how could you determine from it the frequencies of the alphabetic symbols?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 6

Which of Codes 1-5 are uniquely decodable?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 7

Which of Codes 2-5 are instantaneously decodable?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 8

Which of the codes that are both uniquely decodable and instantaneously decodable is the best?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 9

Explain why the no-prefix condition ensures that a uniquely decodable code is also instantaneously decodable.

You can find an answer/comment for this review question at the end of the chapter.

Review Question 10

Technically, it could be claimed that the prefixes of 1010 are 1, 10, 101 and 1010. Why, in producing a test of whether uniquely decodable codes are instantaneously decodable, is there no need to consider a code word as a prefix of itself?

You can find an answer/comment for this review question at the end of the chapter.

Review Question 11

Determine a Huffman code for each of the following sources.

1. n=5; fr(x1)=1/2, fr(x2)=fr(x3)=fr(x4)=fr(x5) = 1/8
2. n=5; fr(x1)=1/2, fr(x2)=1/4, fr(x3)=1/8, fr(x4)=fr(x5) = 1/16

In each case, calculate its advantage over an equal-length code.

You can find an answer/comment for this review question at the end of the chapter.

Discussion Topics

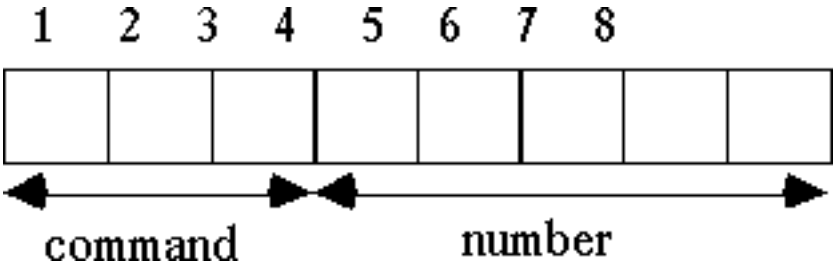
1. Discuss whether a Huffman code is uniquely decodable.
2. Discuss the sense in which a Huffman code is an optimum code, why it is an optimum code and the way in which achieves efficiency whether of transmission or storage.
3. To what extent is a Huffman code unique?

Answers and Comments

Exercise 1

We already have a code for the commands, but now some of the commands need to have extra information added to them. If the information is added in a consistent way at the sending end, it can

be interpreted in a consistent way at the receiving end. This can be taken care of by designing a format such as:



Here, the first three digits carry the code for the command and digits 5 to 8 the code for the attached number.

Review Question 1

The code could be:

Colour Symbol	**8-digit code**
Colour A	01000001
Colour B	01000010
Colour C	01000011
Colour D	01000100
Colour E	01000101
....	...

Review Question 2

2^8 = 256 colours.

Review Question 3

1. n=64: s1 = flag1 vertical and flag2 vertical, etc.

2. n=2^800: s1 = blank card, s2 = card punched in row1, column 1, etc

3. n=16^(800*625): s1 = screen all in colour 1, s2 = screen all in colour 1 except for colour 2 in row1, column 1, etc.

Review Question 4

1. n, sk as before; fr(sk) = 1/n for all k.

2. n=53: s1 = a, s2 =b, and so on, with the frequencies as in English texts, where e occurs most frequently and letters such as x, z and q occur least frequently.

Review Question 5

Count the number of symbols in the text, and call the number N. Count the number of occurrences of sk and call it N(sk). Then fr(sk) N(sk)/ N.

Review Question 6

Code 1 is not. Consider the reception of 00, obtained by coding 'green'. How would it be decoded? Codes 2-5 are.

Review Question 7

Code 2 is not. A 0 zero signals the beginning of a code word. This means that you don't know that a code word has ended until the next one has begun, so decoding cannot be instantaneous. Code 3 is: receipt of the 0 at the end of a code word indicates reception of the complete code word. Code 4 is: reception of two digits indicates reception of a complete code word. Code 5 is: receipt of the 0 at the end of a code word, or of three successive 1s, indicates reception of a complete code word.

Review Question 8

Code 5, as it is the shortest, and so the most efficient.

Review Question 9

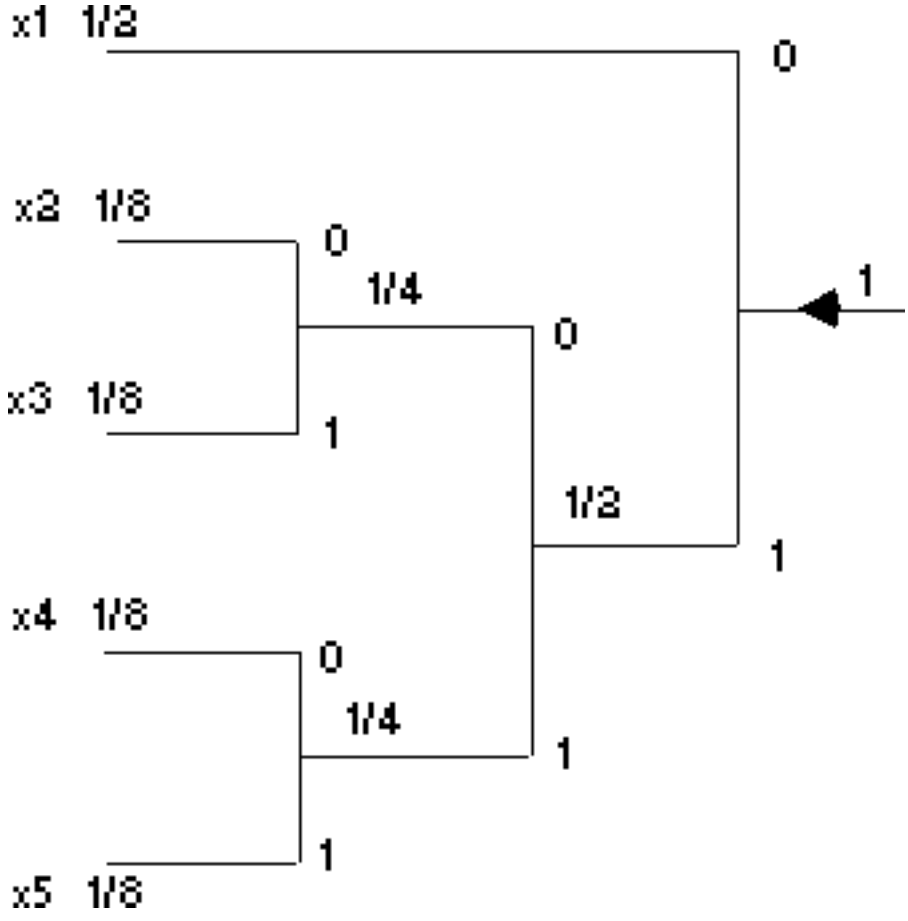
If one code word is a prefix of another, reception of the shorter code word cannot be recognised until at least one more digit has been received. The extra digit(s) will resolve the issue of whether the shorter code word has been received, but then decoding is not instantaneous. Conversely, if no code word is a prefix of another, there is no possibility of confusion at the point of receiving a code word so that decoding can be instantaneous.

Review Question 10

If one code word formed a complete prefix of another, the code would not be uniquely decodable.

Review Question 11

1. The Huffman algorithm gives:

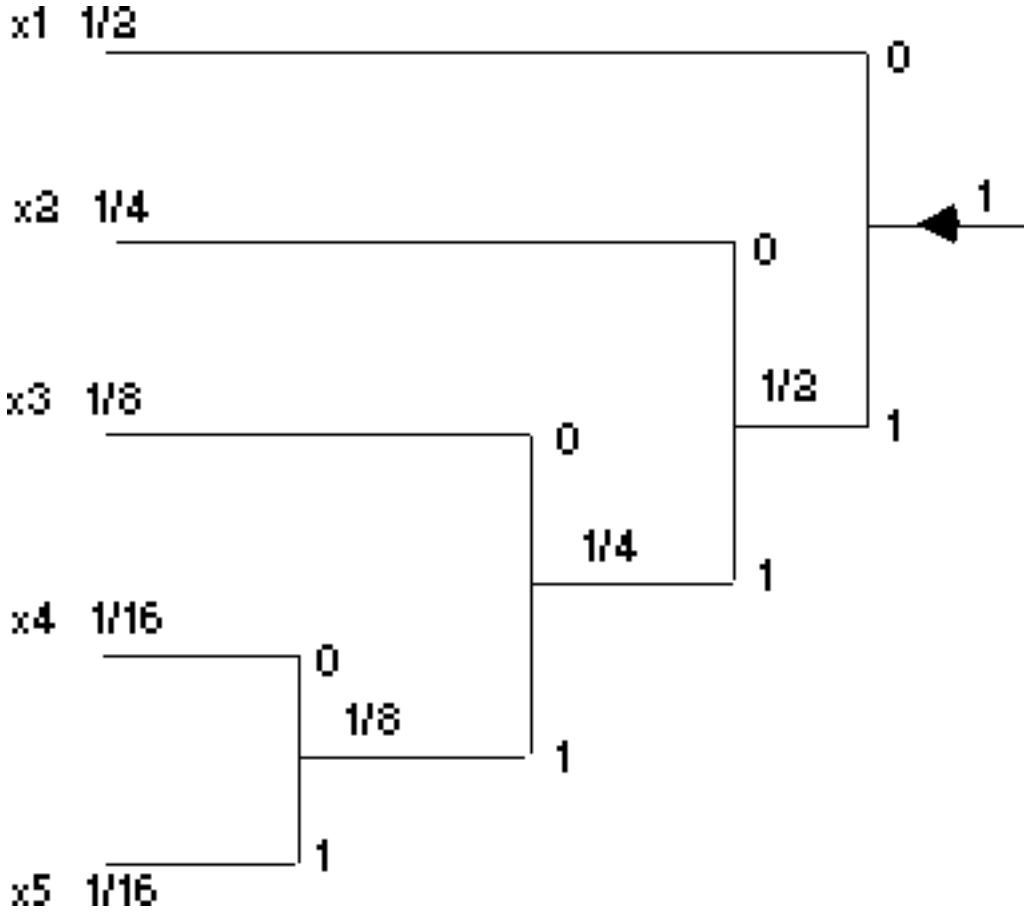


The code is:

Symbol	Code word
x1	0
x2	100
x3	101
x4	110
x5	111

The length of the code is 1*0.5 + 4*3*0.125 = 2.5 bits/symbol. A fixed length code requires 3 bits/symbol. The advantage of the Huffman code over the fixed length code is in requiring only 2.5/3 = 83.33% of its digits to achieve the same effect.

2. The Huffman algorithm gives:



The code is:

Symbol	Code word
x1	0
x2	10
x3	110
x4	1110
x5	1111

The length of the code is 1*0.5 + 2*0.25 + 3*0.125 + 2*4*0.0625 = 1.875 bits/symbol. A fixed length code requires 3 bits/symbol. The advantage of the Huffman code over the fixed-length code is in requiring 1.875/3 = 62.5% of its digits to achieve the same effect.