Ministry of Higher Education and Scientific Research

Al-Furat Al- Awsat Technical University

Engineering Technical college / Najaf

Communication Engineering Department

Information Technology (CE231)

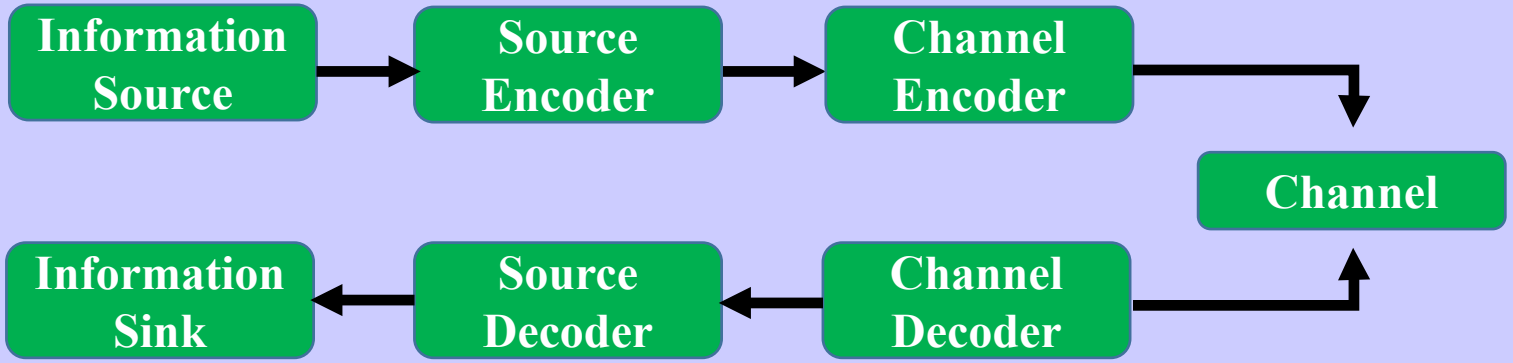
2nd Class 2018/2019

Lecturer Ali M. Alsahlany

Lecture Outlines :

- **Introduction**
- **Source coding of discrete sources**
- **Fixed Length Code**
- **Variable Length Code**
- **Minimum Code Length**
- **Code Efficiency**
- **Average Code Length**
- **Requirements for a useful symbol code**

Introduction: A general block diagram of a point-to-point digital communication system given in Figure below. The source encoder converts the sequence of symbols from the source to a sequence of binary digits, preferably using as few binary digits per symbol as possible. The source decoder performs the inverse operation.



- *Discrete sources*

The output of a discrete source is a sequence of symbols from a known discrete alphabet X. This alphabet could be the alphanumeric characters, the characters on a computer keyboard, English letters, Chinese characters, the symbols in sheet music (arranged in some systematic fashion), binary digits, etc.

- *Analog waveform sources*

The output of an analog source, in the simplest case, is an analog real waveform, representing, for example, a speech waveform. The word analog is used to emphasize that the waveform can be arbitrary and is not restricted to taking on amplitudes from some discrete set of values.

- *Simple Model of a source (Called a DISCRETE MEMORYLESS SOURCE OR DMS)*

- *Alphabet size of 4 (A, B, C, D)*
- $P(A) = p_1$, $P(B) = p_2$, $P(C) = p_3$, $P(D) = p_4$

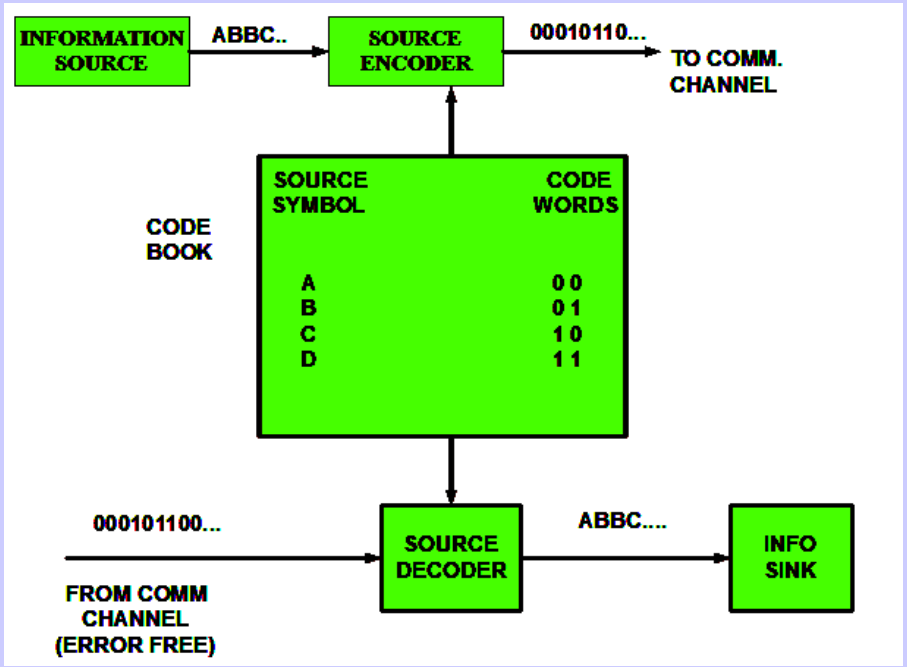
Self Information: Is a function which measures the amount of information after observing the symbol A $I(A) = \log_2 \frac{1}{P(A)}, bit$

Entropy: It is the average number of bits per symbol required to describe a source $H = \sum_{i=1}^{N} P_i \, I(s_i) \quad , bit / symbol$

Simplest Code

A → 00
B → 01
C → 10
D → 11

Symbol (S_k): A
Symbol probability: $P(A)$
Codeword (c_k): 00
Codeword length (l): 2



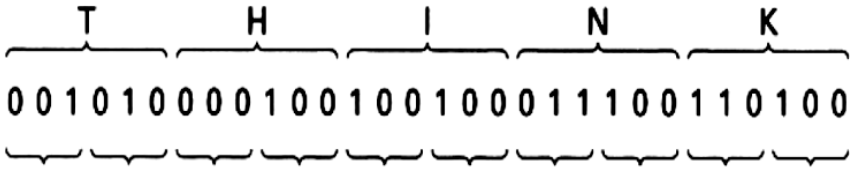
Coding:
- *Basic idea is to use as few binary digits as possible and still be able to recover the information exactly*

- *Sometimes the output of the source decoder must be an exact {replica of the information (e.g. computer data) — called NOISELESS CODING*
- *Other times the output of the source decoder can be approximately equal to the information (e.g. music, tv, speech) — called CODING WITH DISTORTION*

Fixed Length Code

- *It assigns a unique binary sequence to each input alphabet character such as 5, 6, or 7 bits*

Message (text): "THINK"



Assume N is the number of symbols

- *If N is a power of 2, the fixed codeword length* $R = l = \log_2 N$
- *If N is not a power of 2, the fixed codeword length* $R = l = \log_2 N \uparrow$

$$H \le \log_2 N \ , \qquad l \ge H$$

l = H (If the symbols have the same probability)

Variable Length Code

- *When the source symbols are not equally probable, a more efficient encoding method is to use variable length code words*
- *A significant amount of data compression can be realized when there is a wide differences in probabilities of the symbols. To achieve this compression, there must also be a sufficiently large number of symbols*

Ex: Morse Code

The codewords of letters that occur more frequently are shorter than those for letters that occur less frequently

A	.-	M	--	Y	-.--	6	-....
B	-...	N	-.	Z	--..	7	--...
C	-.-.	O	---	Ä	.-.-	8	---..
D	-..	P	.--.	Ö	---.	9	----.
E	.	Q	--.-	Ü	..--	.	.-.-.-
F	..-.	R	.-.	Ch	----	,	--..--
G	--.	S	...	0	-----	?	..--..
H	....	T	-	1	.----	!	..--.

- *The variable length codewords should depend on the probability*

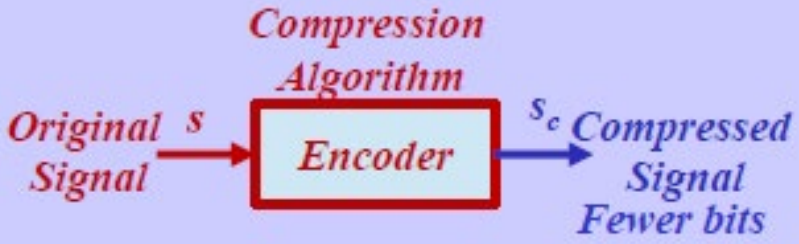
Minimum Code Length

To be efficient one using knowledge of the statistics of the source such that:

- *Frequent source symbols should be assigned SHORT code words*
- *Rare source symbols should be assigned LONGER code words*

Average Code Length

- *Consider a source has K symbols, each symbol s_k has probability P_k, and represented by a code word C_k of length l_k bits, then*



Average code word length

$$\bar{L} = \sum_{k=1}^{K} P_k \, l_k$$

Variance of the code word length

$$Variance = \sum_{k=1}^{K} P_k \, (l_k - \bar{L})^2$$

Example 1: Simple discrete source, Alphabet size of 4 (A,B,C,D), $P(A) = p_1 = P(B) = p_2 = P(C) = p_3 = P(D) = p_4$, Calculate the average and variance of the code

Solution:

$\bar{L} = 2(p_1 + p_2 + p_3 + p_4) = 2$

Example 2: Calculate the average code word length for the following symbols

Symbol S	P(s)	Code
A	0.25	11
B	0.30	00
C	0.12	010
D	0.15	011
E	0.18	10

$$\bar{L} = \sum_{k=1}^{K} P_k \, l_k$$

$\bar{L}$= 0.25(2) +0.30(2) +0.12 (3) +0.15(3) +0.18(2) = 2.27 bits

It does not mean that we have to find a way to transmit a noninteger number of bits. Rather, it means that on average the length of the code is 2.27 bits

Minimum Possible Code Word Length

$$L_{min} = H(S)$$

The outputs of an information source cannot be represented by a source code whose average length less than the source entropy

$$L_{min} \geq H(S)$$ *Shannon's First Theorem*

Code Efficiency

$$\eta = \frac{L\_min}{\bar{L}} = \frac{H(S)}{\bar{L}} = \frac{Entropy}{Average\ code\ length}$$

An efficient code means $\eta \longrightarrow 1$

Compression Ratio

$$CR = \frac{Number\ of\ bits\ of\ the\ fixed\ code\ that\ represents\ the\ symbols}{Average\ code\ length\ of\ the\ variable\ length\ code}$$

Example 3: Source has an alphabet of 26 letters (English Alphabet) with independent and identically distributed random variables: { Xi , i=1,...,26} each occurring with the same probability. Evaluate the efficiency of a fixed length binary code in which:

a. Each letter is encoded separately into a binary sequence

b. Two letters at a time are encoded into a binary sequence

a. Each letter is encoded separately into a binary sequence

$$H = \sum_{i=1}^{26} log_2 N = \sum_{i=1}^{26} log_2 26 = 4.7 \frac{bits}{symbol}$$

$$R = L = \sum_{i=1}^{26} log_2 N \uparrow = 5 \frac{bits}{symbole}$$

$$\eta = \frac{H(S)}{\bar{L}} = 94.0087\%$$

b. Two letters at a time are encoded into a binary sequence

N=26*26=676 possible sequences

$R = log_2 676 \uparrow$= 10bits/2 symbol =5bits/symbol $\eta = \frac{H(S)}{\bar{L}} = 94.0087$

Example 4: *Calculate the entropy, minimum length, average code word length, and the compression ratio for the flowing source and the given two different codes for the following symbols*

Source Symbol S_k	Symbol Probability P_k	Code I		Code II	
		Symbol Code Word c_k	Code Word Length l_k	Symbol Code Word c_k	Code Word Length l_k
S_0	1/2	00	2	0	1
S_1	1/4	01	2	10	2
S_2	1/8	10	2	110	3
S_3	1/8	11	2	1111	4

$H(s) = \sum_{k=1}^{K} P_i log_2 \left(\frac{1}{P_i}\right)$ = 1/2log_2 (2)+1/4log_2 (4)+ 1/8log_2 (8)+1/8log_2 (8)= 1.75 bits/symbol=L_{min}

$\bar{L} = \sum_{k=1}^{K} P_k l_k$

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

$\bar{L}$ = 2x(1/2+1/4+1/8+1/8)=2

$\eta = \frac{1.75}{2} = 0.875$

CR= 2/2=1

$\bar{L} = \left(1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 4 \times \frac{1}{8}\right) = 1.875$

$\eta = \frac{1.75}{1.875} = 0.9333$

CR= 2/1.875=1.0666

Uniquely Decodable

A code is not uniquely decodable if two symbols have the same codeword, i.e., if $Cs_{(i)} = Cs_{(j)}$ *for any* $i \neq j$ *or the combination of two codewords gives a third one*

It allows the user to invert mapping to the original sequence of elements of the source to (decoding)

Example 5:

Source symbol	Code 1	Code 2
	Symbol Codeword	Symbol Codeword
A	00	0
B	1	1
C	00	00
D	11	11

CODE1, *the symbols A &C are assigned to the same codeword. Thus, the first requirement of a useful code is that each symbol be assigned to a unique binary sequence IS NOT VALID.*

CODE2 , *It is confusing to detect the code, Why? C is decoded as combination of (A, A), also D (B,B).*

Example 6:

$S_{(i)}$	P	Code 1	Code 2	Code 3
$S_{(1)}$	0.5	0	0	0
$S_{(2)}$	0.25	0	10	01
$S_{(3)}$	0.125	1	110	011
$S_{(4)}$	0.125	10	111	0111
$\bar{L}$	1	1.125	1.75	1.875

CODE1, the symbols S_1 &S_2 are assigned to the same codeword. S_4 is a combination of (S_3,S_1) or (S_3,S_2). It is non uniquely detectable code

CODE2 , all code words end with 0 except the last one is three 1s. The decoding rule is simple, accumulate bits until you get 0 or until you have three 1s. There is no ambiguity in this rule.

CODE3 , each codeword starts with 0, and the only time we see a 0 is in the beginning of a codeword. Therefore, the decoding rule is accumulate bits until you see 0. The bit before the 0 is the last bit of the previous codeword.

- *Based on the average length, code 1 appears to be the best code However, it is not useful because it is non detectable*

Example 6:

$S_{(i)}$	P	Code 1	Code 2	Code 3
$S_{(1)}$	0.5	0	0	0
$S_{(2)}$	0.25	0	10	01
$S_{(3)}$	0.125	1	110	011
$S_{(4)}$	0.125	10	111	0111
$\bar{L}$	1	1.125	1.75	1.875

CODE2 , is called INSTANTENOUS CODE because the decoder knows the moment a codeword is complete.

CODE3 , is called NOT INSTANTENOUS CODE because we have to wait till the beginning of the next codeword before we know that the current codeword is complete

This property (Instantaneous) is not a requirement for unique decodability, because it depends

How does the channel terminate the transmission?
-e.g. it could explicitly mark the end
-it could send only 0s after the end
-it could send random garbage after the end,...

How soon do we require a decoded symbol to be known? -e.g. "instantaneously" as soon as the codeword for the symbol is received.
-within a fixed delay of when its codeword is received
-not until the entire message has been received

Example 7:

Decode the string 0111111111111111111 by code1

0101010101010101 by code2

S_i	Code 1	Code 2
S_1	0	0
S_2	01	01
S_3	11	10

Using code1

The string could be decoded either $S_1 S_3 S_3 S_3 S_3 S_3 S_3 S_3 S_3$ and 1 is left which is not a codeword (Invalid decoding)

OR the string could be decoded as $S_2 S_3 S_3 S_3 S_3 S_3 S_3 S_3$ and nothing left (Valid decoding) IT IS UNIQUE DECODABLE

Using code2

The string could be decoded either $S_1 S_3 S_3 S_3 S_3 S_3 S_3 S_3 S_3$ (Valid decoding)

OR the string could be decoded as $S_2 S_2 S_2 S_2 S_2 S_2 S_2 S_2 S_1$ and left 0 which is a codeword S_1 (Valid decoding)

IT IS NOT UNIQUE DECODABLE

Test for Unique Decodability

Suppose we have two binary codewords a and b, where, a is k bits long, b is n bits long, and k < n If the first k bits of b are identical to a, then a is called a prefix of b. The last n−k bits of b are called the dangling suffix

Example 8: *if a = 010 and b = 01011, then a is a prefix of b and the dangling suffix is 11*

Steps of Unique Decodability

1-Construct a list of all the codewords.
2-Examine all pairs of codewords to see if any codeword is a prefix of another codeword .
3-Whenever you find such a pair, add the dangling suffix to the list unless you have added the same dangling suffix to the list in a previous iteration.
4-Now repeat the procedure using this larger list.
5-Continue in this fashion until one of the following two things happens:

You get a dangling suffix that is a codeword, not uniquely decodable	*There are no more unique dangling , uniquely decodable*

Example 9:

Si	Code 1	Code 2
S1	0	0
S2	01	01
S3	11	10

Code1: S_1 is a prefix of S_2 ⟶ code 1 will be [0,01,11,1]
1 is a prefix of 11 but it is already added then the last version is [0,01,11,1]
the codeword 1 is not a codeword. IT IS UNIQUELY DECODABLE

Code2: S_1 is a prefix of S_2 ⟶ code 2 will be [0,01,10, 1]
In new list, 1 is a prefix for 10, the dangling suffix 0 , which is the codeword. then the last version is [0,01,10,1]
IT IS NOT UNIQUELY DECODABLE

- *The variable length codewords should DEPEND ON PROBABILITY*
- *Code should be UNIQUELY DECODABLE*

Prefix Code

- Is a code in which no codeword is the beginning of another codeword
- Is a code in which no codeword is a prefix to another codeword

Since no codeword is a prefix of the other, we will never face the possibility of a dangling suffix being a codeword. In this case, the set of dangling suffixes is the null set, and we do not have to worry about finding a dangling suffix that is identical to a codeword

A prefix code is uniquely decodable but the converse is not true

Example 10:

$S_{(i)}$	Code 1	Code 2	Code 3
$S_{(1)}$	0	0	0
$S_{(2)}$	1	10	01
$S_{(3)}$	00	110	011
$S_{(4)}$	11	111	0111
	Not Uniquely Decodable Nor Prefix Code	Uniquely Decodable Codes	

Prefix

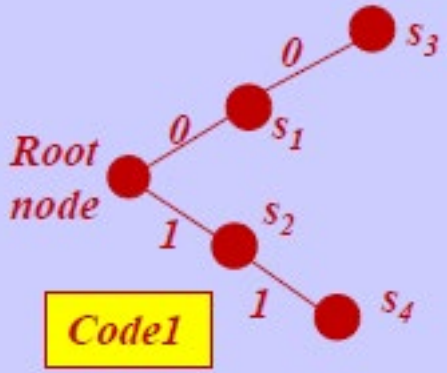
Homework 1: *Determine the prefix codes and the uniquely decodable codes of the following codes*

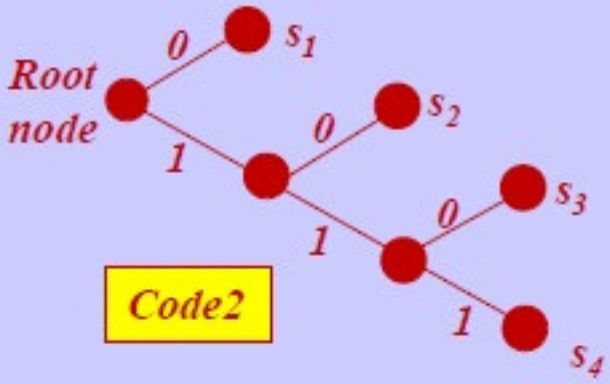
Symbol	Prob.	Code 1	Code 2	Code 3	Code 4
A	P[A]=1/2	1	1	0	00
B	P[B]=1/4	01	10	10	01
C	P[C]=1/8	001	100	110	10
D	P[D]=1/16	0001	1000	1110	11
E	P[E]=1/16	00001	10000	1111	110
Average Length		31/16	31/16	30/16	33/16

Test for Prefix Code

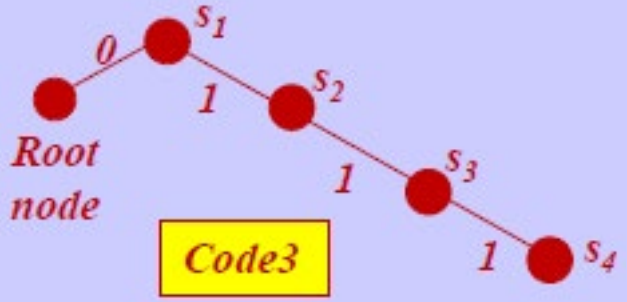
1. Draw the rooted binary tree corresponding to the code (Decoding Tree) Starts from a single node (the root node) and has a maximum of two possible branches at each node. One of these branches corresponds to a 1 and the other branch corresponds to a 0
2. The code for any symbol can be obtained by traversing the tree from the root to the external node corresponding to that symbol .

Example 11: Draw the rooted binary tree for the following codes

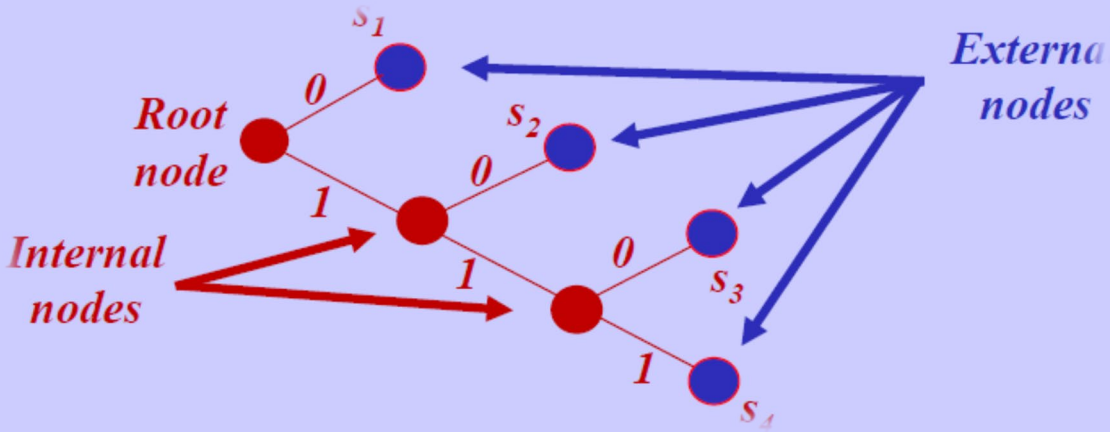




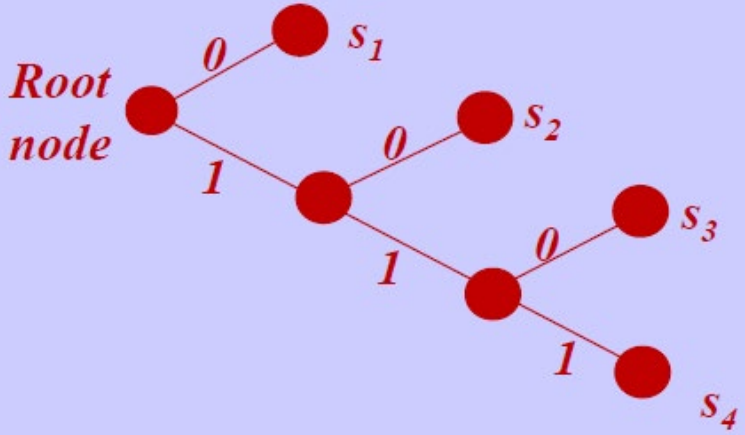
S_i	Code 1	Code 2	Code 3
S_1	0	0	0
S_2	1	10	01
S_3	00	110	011
S_4	11	111	0111



Determine the type of nodes apart from the root node :
Internal nodes that give rise to other nodes
External nodes or leaves that terminated



The prefix code has the codewords are only associated with the external nodes



Kraft-McMillan Inequality

$$\sum_{k=1}^{K} 2^{-l_k} \leq 1$$

A prefix code must satisfy the Kraft McMillan's inequality
But a code satisfies Kraft McMillan's inequality is not necessarily be a prefix code

Example 12:

s_k	Code	
	C_k	l_k
s_0	0	1
s_1	10	2
s_2	110	3
s_3	11	2

For this code $2^{-1} + 2^{-2} + 2^{-3} + 2^{-2} = 9/8$
which means that the code IS NOT A PREFIX CODE

i.e., Kraft-McMillan Inequality can determine that a given code IS NOT A PREFIX CODE

Example 13: *For this code*

$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$

s_k	Code	
	c_k	l_k
s_0	0	1
s_1	100	3
s_2	110	3
s_3	11	2

Is the code a PREFIX code?
NO WHY?
s_3 is a beginning of s_2

Hence, if we have a uniquely decodable code that its codeword lengths satisfy the Kraft- McMillan inequality , then we can always find a prefix code with those codeword lengths

Thus, by restricting ourselves to prefix codes, no need for nonprefix uniquely decodable codes that have a shorter average length

Code Redundancy

Is the difference between the average length and the entropy

$$\rho = \bar{L} - H = \sum_{k=1}^{K} P_k (l_k - \log P_k(s_k))$$

Example 6: The code $C_1 = \{0; 101\}$ is a prefix code because 0 is not a prefix of 101, nor is 101 a prefix of 0.

Exercise 1: Is the code $C_2 = \{0; 10; 110; 111\}$ prefix or not?

Exercise 2: Is the code $C_3 = \{00; 01; 10; 11\}$ prefix or not?

Exercise 3: Is the code $C_4 = \{1; 101\}$ prefix or not?

Homework 2: : *Calculate the entropy, minimum length, average code word length, and the compression ratio for the flowing source*

	S_k	P_k	C_k	l_k
C	A	1/2	0	1
	B	1/4	10	2
	C	1/8	110	3
	D	1/8	111	3