01076566 Multimedia Systems

Chapter 6: Overview of Compression

Pakorn Watanachaturaporn
pakorn.wa@KMITL.ac.th

Bachelor Program in Computer Engineering (B.Eng.)
Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Outline

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The need for compression
- Basics of information theory
- A taxonomy of compression
- Lossless compression
- Lossy compression
- Practical issues related to compression systems

- The amount of digital media data that is produced is extraordinarily large and the rate of creation increases every day.
- This growing mass of data needs to be stored, accessed, and delivered to a multitude of clients over digital networks, which have varying bandwidths.
- Data compression is a well-understood area with practical standards in place for compressing and communicating individual data types—text, images, video, audio, and graphics.

pakorn.wa@kmitl.ac.th

The need for compression

- Motivated by both storage requirements and transmission requirements.
- In earlier days of the information age, disk space was limited and expensive, and bandwidth was limited, so compression was a must.
- Today, inexpensive storage in a variety of forms is readily available, and high-bandwidth connections (DSL, fiber optics and T-1) are becoming more popular.
- However, the amount and type of information we consume has also increased many fold.

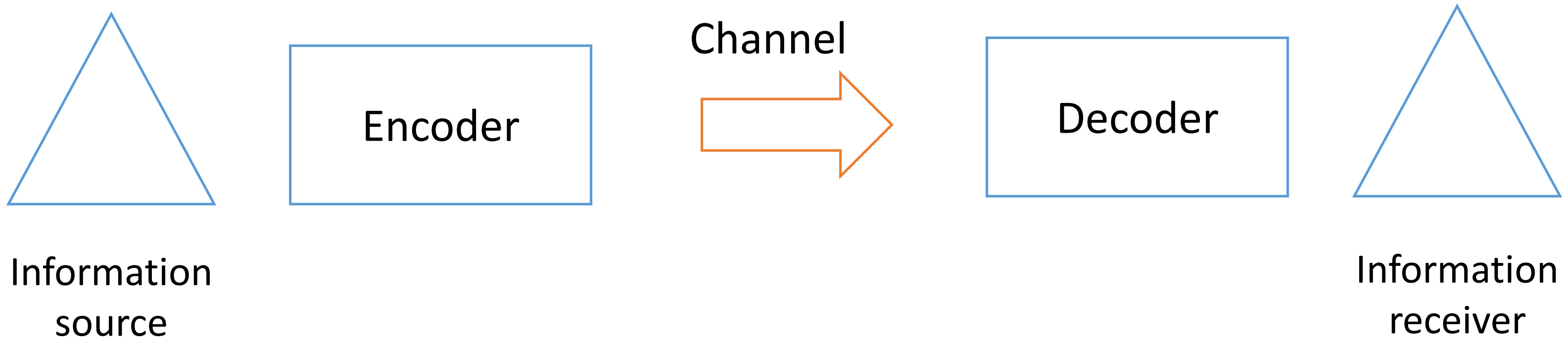
- Example
  - The interlaced HDTV (1080i) format has a frame size of 1920 × 1080. With YUV 4:2:0 subsampling, every color pixel is represented by 12 bits.
  - This brings the required bandwidth to 1080 × 1920 × 12 × 30 = 745 Mb/s.
  - Home network connections do not support this type of bandwidth and the digital signal will need to be compressed for delivery.
  - HDTV 1080i is considered to be acceptable when compressed down to 18 Mb/s.

Basics of Information Theory

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

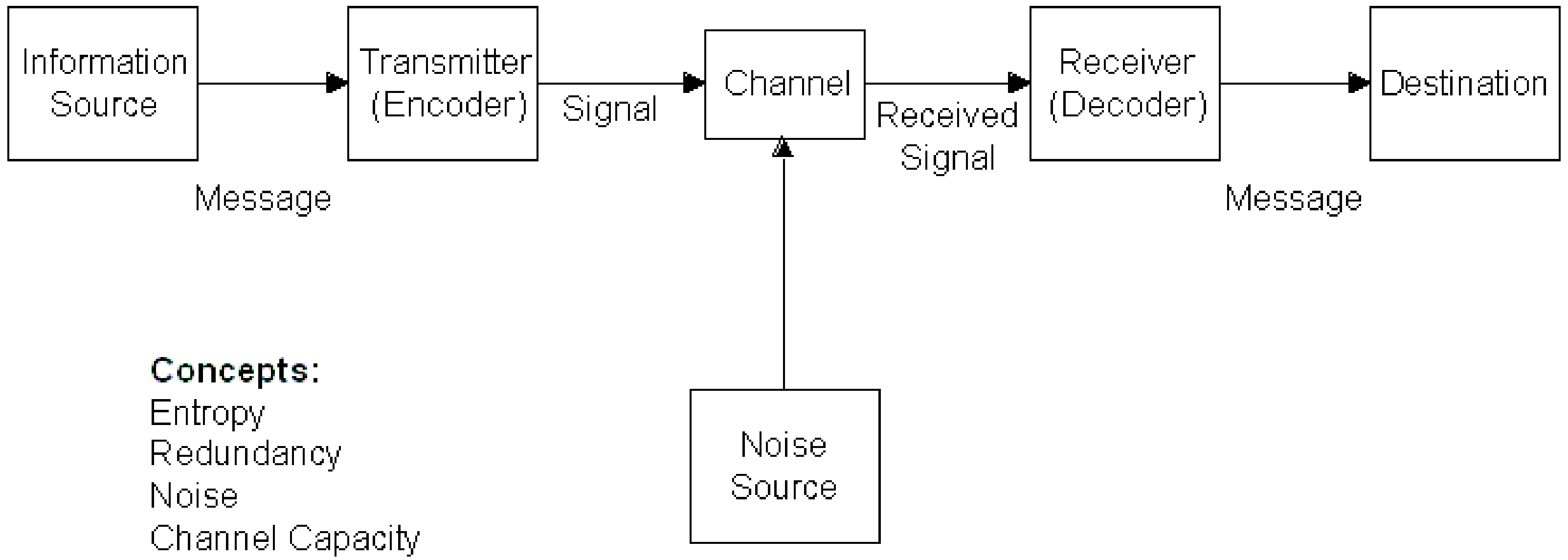
- Developed in the 1940s at Bell Labs by Claude Shannon.
- When transmitting information from a source to a destination, information theory concerns itself with the *efficiency* and *reliability* of the transmission.
  - *Efficient transmission* — How efficiently can a source transmit information to a receiver over a given channel and, thereby, communicate the information in the least amount of bits?
  - *Reliable transmission* — How reliably can a source transmit information to a receiver over a given channel, knowing that the channel is noisy and can corrupt the information during transmission.

- The first point relates to compression.
- To formalize the problem of communicating information from a source to a destination, Shannon proposed his model of inserting an encoder that encodes data produced by an information source and a decoder that decodes the information received by a receiver.



Shannon’s communication model.
Information produced by a source is encoded in binary form and decoded at the receiver.

The Shannon-Weaver Mathematical Model, 1949



Concepts:
Entropy
Redundancy
Noise
Channel Capacity

- The second point deals with ensuring reliable delivery of data across a noisy channel by adding redundant bits to the coded symbols.
- With added redundancy, the receiver is able to detect transmission errors and even correct them.

pakorn.wa@kmitl.ac.th

- Need to formalize what *information* means, and thereby have a metric to measure the amount of information a sequence of bits and bytes contains.
- Information theory allows us to describe the process to compress data without losing its *information content*.

Information theory definitions

pakorn.wa@kmitl.ac.th

- Information relates to the organization in the data as a sequence of symbols.
- If the sequence changes, so does the information.
- For example,
  - a binary data stream that consists of 80,000 bits. These bits might not need to be treated individually, but might instead be grouped into symbols.
  - the arrangement of the symbols is important

Alphabet and symbols

- Information can be thought of as an organization of individual elements called *symbols*.
- An *alphabet* is defined as a distinct and nonempty set of *symbols*.
- The number or length of symbols in the set is known as the *vocabulary*.
- Define an alphabet of symbols as $S = \{s_1, s_2, s_3, s_4, \ldots, s_n\}$
  - n can be very large
  - In practice, an alphabet is limited and finite and, hence, has a well-defined vocabulary

- For example
  - A gray image, each pixel is represented by 8 bits and can have one of 2^8 or 256 unique values.
    - Vocabulary consists of 256 symbols
    - Each symbol is represented or coded by 8 bits
  - The English alphabet has 26 unique letters
    - Vocabulary consists of 26 symbols
    - Each symbol will need 5 bits ($\log_2 26 = 4.7$)
  - CD audio data is digitized at 16 bits/sample
    - The alphabet has a vocabulary of $2^{16} = 65{,}536$ symbols

Sequence

- A series of symbols of a given alphabet form a *sequence*.
- E.g.,
  - For the alphabet $\{s_1, s_2, s_3, s_4\}$, a sample sequence is $s_1, s_2, s_1, s_2, s_2, s_2, s_1, s_2, s_3, s_4, s_1, s_2, s_3, s_3$
- A sequence of symbols is also termed a *message* produced by the source using the alphabet, and represents information.

- A change in the sequence corresponds to a change in the message and, thereby, a change in the information.
- Each symbol has a binary representation and, hence, a message translates digitally to a bit stream.
- An image, a video, and text are all examples of binary messages.

- Some symbols might occur more commonly than other symbols.
- In the English language, the letter *e* occurs more frequently than the other letters.
- The frequency of occurrence of a symbol is an important factor when coding information represented by the symbols.
- The frequency is also known as probability

Symbol probability

pakorn.wa@kmitl.ac.th

- The number of times a symbol occurs in the entire message, on the average, measures the likelihood, or probability of that symbol.
- The probability of occurrence is defined by the ratio of the number of occurrences of that symbol over the length of the entire message.

$$P_i = \frac{m_i}{N}$$

- Where m_i is the number of times symbol m_i occurs in the message of length N.

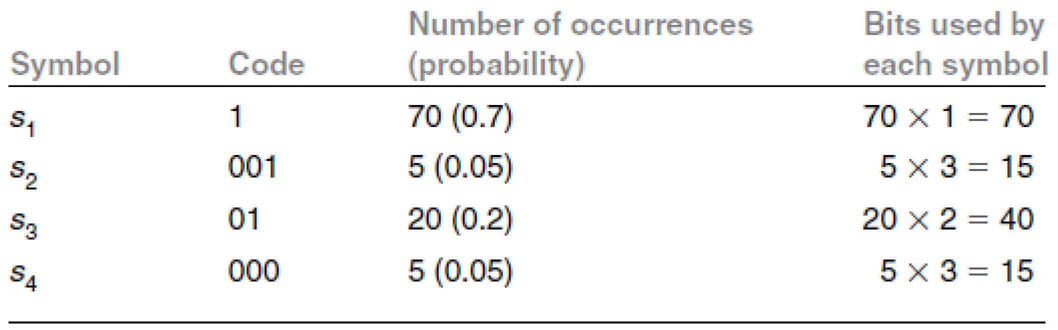
- For analysis, the probabilities can be measured for a message alone, but it might make more practical sense to measure probabilities based on a large sample set.
- For example,
  - when dealing with the English alphabet, it is better to measure the usage of the letter *e* in the language by measuring how many times it occurs over a number of sentences, rather than a single word or just one sentence.

- E.g.,
  - If a source produces four symbols s_1, s_2, s_3, s_4
  - Then two bits are required; $\log_2 n$ where n is the number of symbols
  - Let us say that in a message of 100 symbols,
    - s_1 occurs 70 times
    - s_2 occurs 5 times
    - s_3 occurs 20 times
    - s_4 occurs 5 times
  - This coding method produces 200 bits for the message.

Symbol	Code	Number of occurrences (probability)	Bits used by each symbol
s_1	00	70 (0.7)	$70 \times 2 = 140$
s_2	01	5 (0.05)	$5 \times 2 = 10$
s_3	10	20 (0.2)	$20 \times 2 = 40$
s_4	11	5 (0.05)	$5 \times 2 = 10$

$\boxed{200}$

- if we distribute the code representation unequally, so that the more frequently occurring symbols have shorter code words, the entire message could be represented by fewer bits



Symbol	Code	Number of occurrences (probability)	Bits used by each symbol
s_1	1	70 (0.7)	$70 \times 1 = 70$
s_2	001	5 (0.05)	$5 \times 3 = 15$
s_3	01	20 (0.2)	$20 \times 2 = 40$
s_4	000	5 (0.05)	$5 \times 3 = 15$

$\boxed{140}$

Uniquely decodable codes

- A code c for a message set s represented by symbols is a mapping from each symbol to a bit (binary) string.
- Defined as

$$C = \{(s_1, c_1), (s_2, c_2), \dots, (s_n, c_n)\}$$

- For compression, a code needs to be implemented such that more-frequent symbols are given shorter code words than less-frequent symbols.

- These are called variable-length code words
  - If the code words are not properly chosen, the encoding from symbols to bit codes makes it ambiguous and, thus, hard or even impossible for the decoder to detect where one code word finishes and the other one starts.
- The decoder receives the binary codes and has to uniquely define the set of symbols that the binary string maps to.
- If this is possible, the code is called uniquely decodable.

pakorn.wa@kmitl.ac.th

Symbol	Code	Number of occurrences (probability)	Bits used by each symbol
s_1	0	70 (0.7)	$70 \times 1 = 70$
s_2	01	5 (0.05)	$5 \times 2 = 10$
s_3	1	20 (0.2)	$20 \times 1 = 20$
s_4	10	5 (0.05)	$5 \times 2 = 10$

$\boxed{110}$

- The code in the table maps the message "$s_1\ s_1\ s_2\ s_3$" to a binary sequence "00011"
- The sequence can be decoded as "$s_1 s_1 s_2\ s_3$" or as "$s_1\ s_1\ s_1\ s_3\ s_3$"
- Thus, the table represents a code that is **not** uniquely decodable.

Prefix codes

- A prefix code is a special kind of uniquely decodable code in which no one code is a prefix of another code.
- E.g.,
  - $C = \{(s_1, 1),\ (s_2, 001),\ (s_3, 01),\ (s_4, 000)\}$
- Prefix codes do not have to binary,
  can be used in ternary representations as well.

Information representation

- Say a source has a vocabulary of N symbols $\{s_1, s_2, \dots, s_N\}$
  - It produces a message consisting of M symbols, which are emitted at frequency of $^1/_T$ Hz, which is one symbol every T seconds.
  - Let each symbols s_i be emitted m_i times so that the frequency of s_i in the message of M symbols is m_i.
  - Let the length of each symbol s_i be given by l_i.

- The message of length M contains m_i instances of symbol s_i

$$M = \sum_{i=0}^{i=N} m_i$$

- The total number of bits that the source has produces for message M is given by

$$L = \sum_{i=1}^{i=N} m_i l_i$$

- The average per symbol length is given by

$$\lambda = \frac{\left(\sum_{i=1}^{i=N} m_i l_i\right)}{M}$$

- Additionally, in terms of symbol probabilities, λ may be expressed as

$$\lambda = \frac{\left(\sum_{i=1}^{i=N} m_i l_i\right)}{M} = \sum_{i=1}^{i=N} \left(\frac{m_i}{M}\right) l_i = \sum_{i=1}^{i=N} P_i l_i$$

where P_i is the probability of occurrence of symbol s_i.

- Given a set of symbols represented by s_i for all i in an alphabet having probabilities P_i
- The goal of compression is of **finding a binary prefixed code** $C = \{(s_1, c_1), (s_2, c_2) \ldots (s_n, c_n)\}$ where each code word c_i has code length l_i such **that the average code length λ is minimized.**

pakorn.wa@kmitl.ac.th

$$\lambda = \frac{\left(\sum_{i=1}^{i=N} m_i l_i\right)}{M} = \sum_{i=1}^{i=N} \left(\frac{m_i}{M}\right) l_i = \sum_{i=1}^{i=N} P_i l_i$$

- Since λ is a function of the product of two quantities, namely
  the probabilities of symbols P_i and their associated code lengths l_i
- As P_i increased, l_i should decreased and vice versa.
- How low can the value of λ be?

Entropy

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

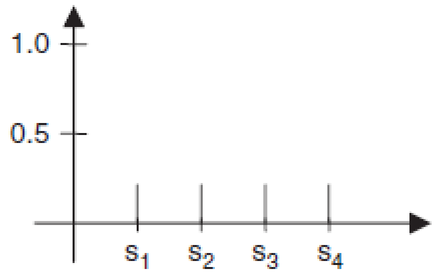
- Entropy is a measure to quantify the amount of information contained in a message of symbols given their probabilities of occurrence.
- For source-producing symbols, where each symbol i has a probability distribution P_i, the entropy is defined as

$$H = \sum P_i \log_2\left(\frac{1}{P_i}\right) = -\sum P_i \log_2 P_i$$

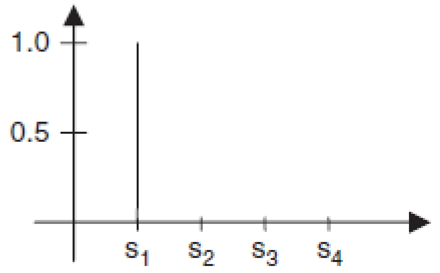
- For the symbol s_i having a probability P_i, Shannon defined the notion of self-information of the symbol given by $\log_2(\frac{1}{P_i})$.
- The self-information represents the number of bits of information contained in the symbol, and, hence, the number of bits used to send that message.
- When the probability is high, the self-information content is low and vice versa.
- Entropy becomes the weighted average of the information carried by each symbol, and, hence, the average symbol length.

- The entropy depends only on the probabilities of symbols and, hence, on the source.

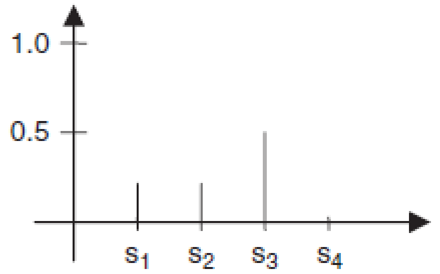
$$P_i = \{0.25,\ 0.25,\ 0.25,\ 0.25\}$$
$$H = -(4 \times 0.25 \times \log_2 0.25)$$
$$H = 2$$



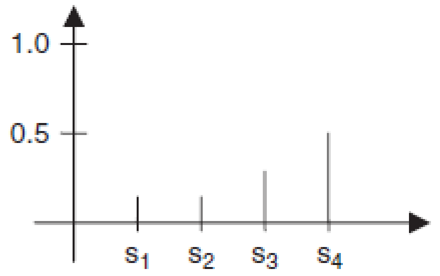
$$P_i = \{1.0,\ 0.0,\ 0.0,\ 0.0\}$$
$$H = -(1 \times \log_2 1)$$
$$H = 0$$



$$P_i = \{0.25\ \ 0.25\ \ 0.5\ \ 0.0\}$$
$$H = -(2 \times 0.25 \times \log_2 0.25 + 0.5 \times \log_2 0.5)$$
$$H = 1.5$$



$$P_i = \{0.125\ \ 0.125\ \ 0.25\ \ 0.5\}$$
$$H = -(2 \times 0.125 \times \log_2 0.125 + 0.25 \times \log_2 0.25 + 0.5 \times \log_2 0.5)$$
$$H = 1.75$$



- Information entropy is very similar to the entropy discussed in Physics: The second law of thermodynamics "entropy measures the amount of randomness in a system" – entropy increases with randomness.
- In our system of information, randomness corresponds to the arbitrary distribution of symbols. This occurs when probabilities of symbols are equal.
- When the entropy is higher (or the information content is higher), more bits per symbol are needed to encode the message.

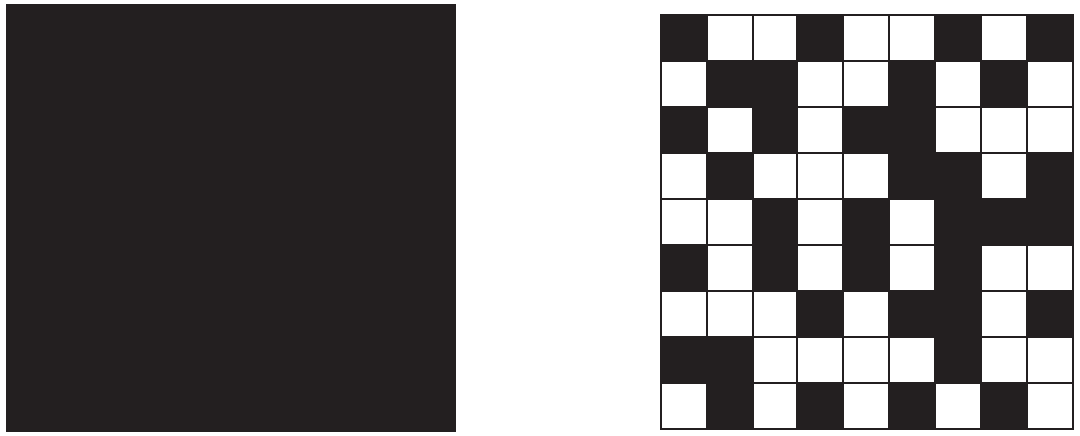


Figure 6-4 Two images are shown. The left image contains all black pixels and the information content is lower resulting in zero entropy. The right image contains roughly equal distribution of black and white pixels. The information content is higher in this case and the entropy evaluates to 1.

Efficiency

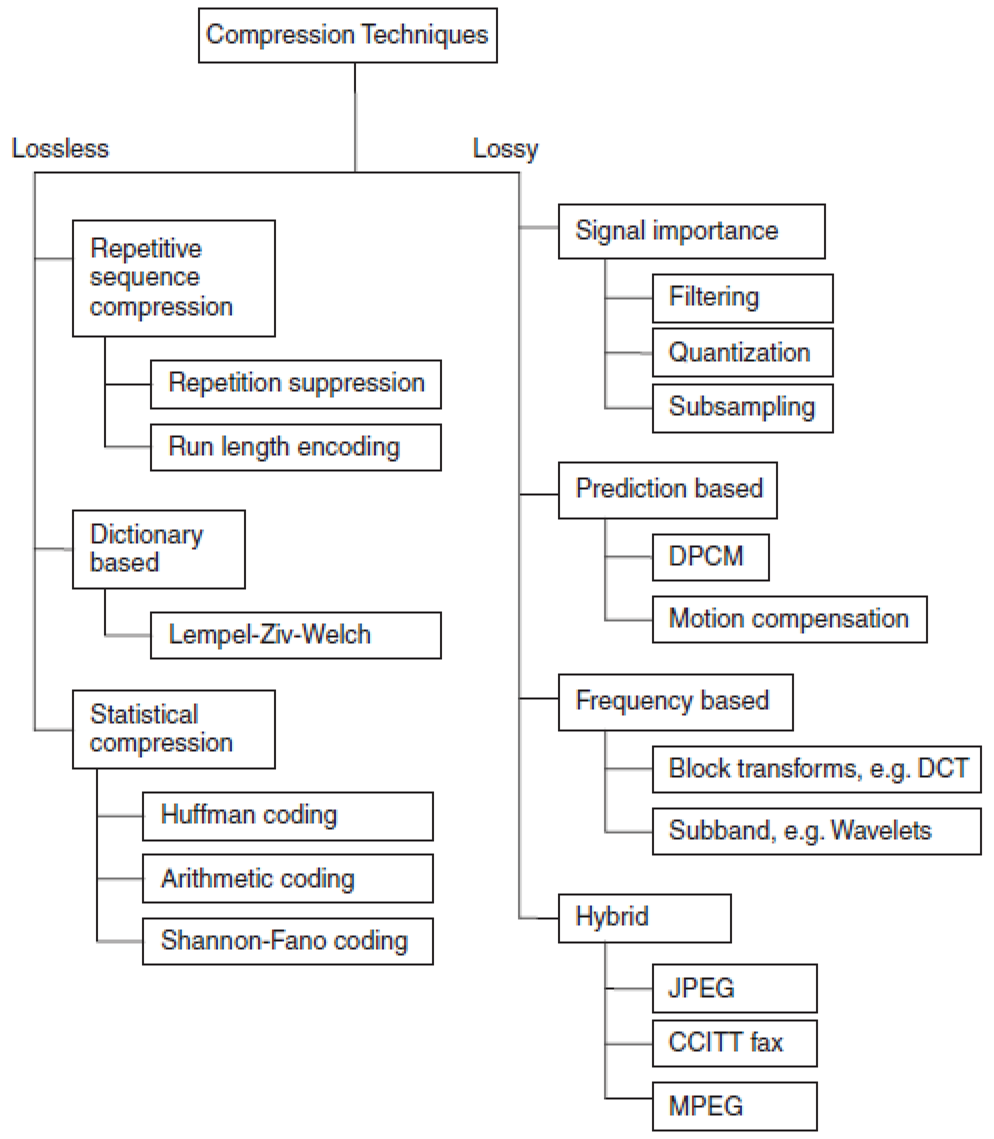
- Given an alphabet of symbols, the goal of any coding system is to transmit the information contained in a string composed of these symbols in the most efficient manner, that is, in **the least number of bits**.
- The coding system needs to evaluate the frequency of occurrence of each symbol in the string and then decide how many bits to code each symbol so that the overall number of bits used for that string is the least

- *Efficiency* is an important metric to evaluate the coding system's ability of compression and is defined as

$$Efficiency = \frac{Entropy}{Average\ Symbol\ Length}$$

A taxonomy of compression

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



Compression Metrics

- Two commonly used metrics
  - Compression rate – relates to the transmission of data
  - Compression ratio – relates to storage of data

- Lossless compression
  - Normally assigns different codes to the original samples or symbols by studying the statistical distribution of those symbols in the signal.
  - Thus, the bit rate produced by the compressed signal, or the compression rate, is bound to vary depending on how the signal's symbols are distributed.
  - This produces a variable bit rate for the compressed signal.
- However, variable rates are undesirable when delivering or streaming real-time data.
- Using lossy techniques, the compressed signal can be distorted to achieve a constant bit rate and are useful for streaming video and audio and for real-time delivery of media across networks.

Rate distortion

- In lossy compression, compression corrupts the information; therefore, the reconstructed signal is not the same as the original signal.
- An important metric of such lossy techniques is the amount of distortion caused by compression.

pakorn.wa@kmitl.ac.th

- Assume that an original signal y is compressed, and then reconstructed as $\hat{y}$
- The distortion or error in this case can be measured by the

  $$error = f(y - \hat{y})$$

  where f is some function that measures the difference

- Other measures such as
  - Mean square error (MSE)

$$\sigma_d^2 = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

  - Signal-to-noise ratio (SNR)

$$SNR = \frac{\sigma_{\hat{y}}^2}{\sigma_d^2}, \qquad SNR(db) = 10 \times \log\left(\frac{\sigma_{\hat{y}}^2}{\sigma_d^2}\right)$$

  - Peak signal-to-noise ratio (PSNR)

$$PSNR(db) = 10 \times log\left(\frac{\sigma_{peak}^2}{\sigma_d^2}\right)$$

Lossless compression

- Lossless compression techniques achieve compression by removing the redundancy in the signal.
- This is normally achieved by assigning new codes to the symbols based on the frequency of occurrence of the symbols in the message.
- In most practical cases, probabilistic models have to be assumed.

- For example
  - In the English language
    - *e* is the most frequently occurring letter—13% on average
    - *z* is the least frequently occurring letter—0.07% on average
  - However, in a certain sentence "Zoos display zebras from Zambia, Zimbabwe, and Zaire," there are more occurrences of *z* than *e*.
  - In this case, using the probabilistic model would not result in an efficient compression, and sometimes might even do the opposite.

- Thus, lossless coding techniques based on a probabilistic model might not necessarily guarantee compression when the distribution of symbols in the message does not conform to the probabilistic model used.

Run Length Encoding

- Run length encoding is the simplest form of redundancy removal.
- The runs of the same symbol are encoded using two entities—a count suggesting the number of repeated symbols and the symbol itself.

- For example,

```
BBBBEEEEEEEECCCCDAAAAA
```

  can be represented as

```
4B8E4C1D5A
```

- The original 22 byte message has been reduced to 10 bytes, giving a compression ratio of 2.2
- How would we know whether the number in the encoded string was the length of a run or part of the string content?

pakorn.wa@kmitl.ac.th

- Run length encoding is used in a variety of tools involving text, audio, images, and video.
- Run length encoding has also become a part of compression standards—it is supported by most bitmap file formats such as TIFF, BMP, and PCX and a variant of it is even adopted in the JPEG compression pipeline

Repetition Suppression

- Repetition suppression works by reserving a specific symbol for a commonly occurring pattern in a message.
- For example, a specific series of successive occurrences of the letter *a* in the following example is replaced by the flag ψ:
  Abdcbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
  aaaaa
  is replaced by

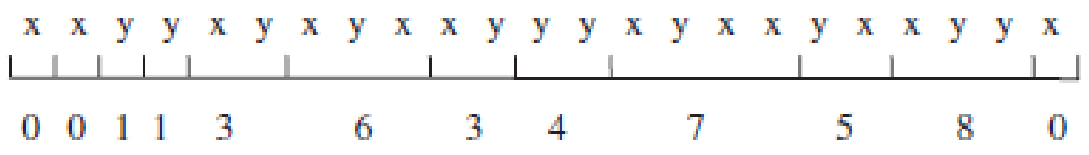
  Abdcbψ
- Repetition suppression can be considered as an instance of run length encoding where successive runs of specific characters are replaced

Pattern substitution

- Based on the dynamic creation of a dictionary of strings.
- Each string is a subsequence of symbols, as they appear in the message.
- Derived in the late 1970s by Lempel and Ziv and later refined by Welch, which is now popularly known as the LZW compression algorithm.
- The compression is based on a simple idea that attempts to build a group of individual symbols into strings.
- The strings are given a code, or an index, which is used every time the string appears in the input message.

- Here is the process:
  - Initialize the dictionary to contain all initial symbols. The vocabulary forms the initial dictionary entries. Every entry in the dictionary is given an index.
  - While scanning the message to compress, search for the longest sequence of symbols that has appeared as an entry in the dictionary. Call this entry **E**.
  - Encode **E** in the message by its index in the dictionary.
  - Add a new entry to the dictionary, which is **E** followed by the next symbol in the scan.
  - Repeat the process of using the dictionary indexes and adding until you reach the end of the message.

pakorn.wa@kmitl.ac.th



Iteration number	Current length symbol(s)	Dictionary code used	Next symbol in string	New dictionary code generated	New dictionary index generated
				x	0
				y	1
1	x	0	x	x x	2
2	x	0	y	x y	3
3	y	1	y	y y	4
4	y	1	x	y x	5
5	x y	3	x	x y x	6
6	x y x	6	x	x y x x	7
7	x y	3	y	x y y	8
8	y y	4	x	y y x	9
9	x y x x	7	y	x y x x y	10
10	y x	5	x	y x x	11
11	x y y	8	x	x y y x	12
12	x	0	END		

Index	Entry	Index	Entry
0	x	7	x y x x
1	y	8	x y y
2	x x	9	y y x
3	x y	10	x y x x y
4	y y	11	y x x
5	y x	12	x y y x
6	x y x		

- Decompression works in the reverse manner. Given the encoded sequence and
- the dictionary, the original message can be obtained by a dictionary lookup.
- Note that transmitting the entire dictionary to the decoder is not necessary.
  - The dictionary can be re-created from the coded message itself, based on the initial vocabulary (x and y, in this example).
  - The decoder knows that the *last* symbol of the most recent dictionary entry is the *first* symbol of the next parse block. This knowledge is used to resolve possible conflicts in the decoder.

- Practical implementation needs to handle a few issues for efficient running.
  - How long should the dictionary grow?
  - Along with the dictionary growth, the number of bits required to represent the index also grows (bits required $b = \log_2 N$, where N is the dictionary size).
  - It is impractical to have a large N, and this value is normally limited to 4096, which gives 12 bits per index or it might also be an input parameter to the algorithm.
  - Once the limit is reached, no more dictionary entries can be added.

Huffman Coding

pakorn.wa@kmitl.ac.th

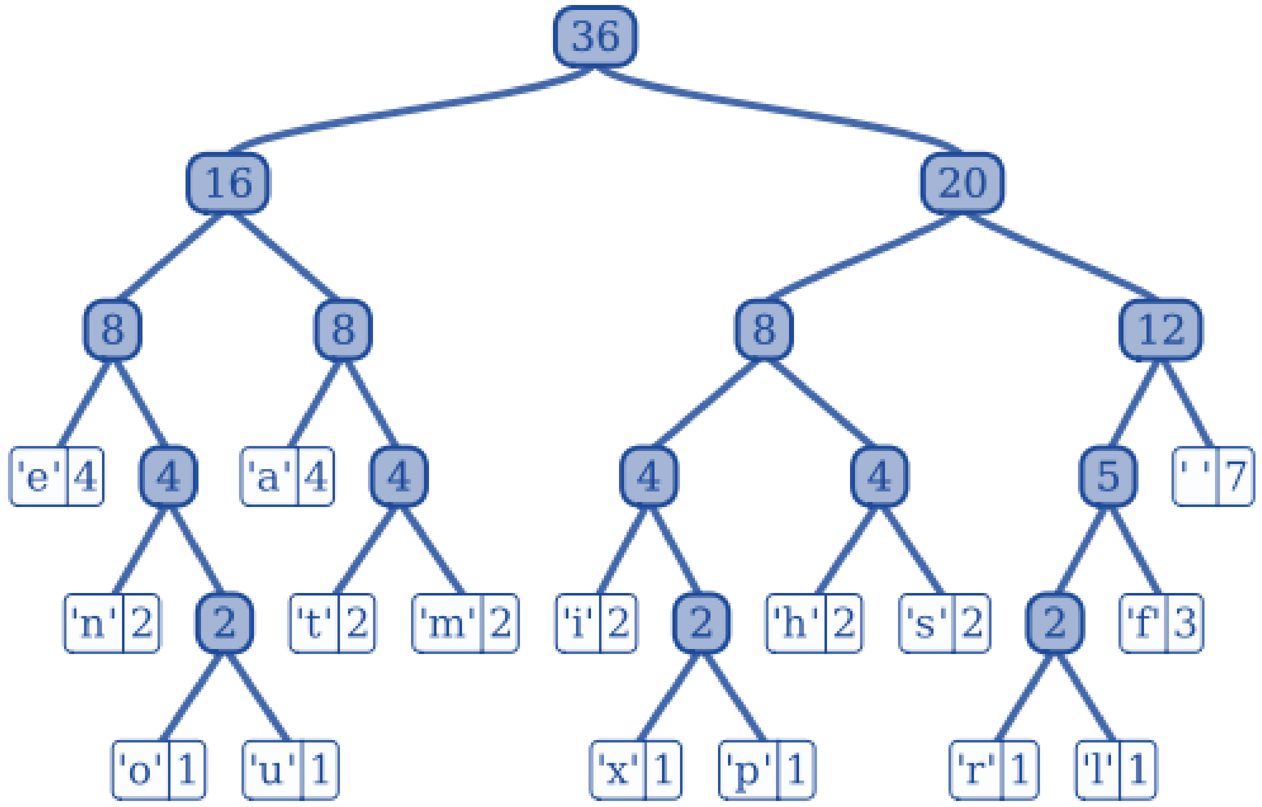
- When data is initially sampled, each sample is assigned the same number of bits
  - The length is $\log_2 n$ where n is the number of distinct samples or quantization intervals.
- Assigning an equal number of bits to all samples might not be the most efficient coding scheme because some symbols occur more commonly than others.
- The goal is to have more frequent symbols that have smaller code lengths and less frequent symbols that have longer code lengths.

- This variable-length representation is normally computed by performing statistical analysis on the frequency of occurrence of each symbol.
- Huffman coding provides a way to efficiently assign new codes to each symbol, depending on the frequency of occurrence of the symbol.
- The process starts by computing or statistically determining the probability of occurrence of each symbol.
- The symbols are then organized to form the leaves of a tree, and a binary tree is built such that the most frequent symbol is closer to the root and the least frequent symbol is farthest from the root.

- This can be outlined as follows:
  - The leaves of the binary tree are associated with the list of probabilities.
  - The leaves can be sorted in increasing/decreasing order, though this is not necessary.
  - The two smallest probabilities are made sibling nodes by combining them under one new parent node with a probability equal to the sum of the child node probabilities.
  - Repeat the process by choosing two least probability nodes (which can be leaf nodes or new parent nodes) and combining them to form new parents until only one node is left. This forms the root node and results in a binary Huffman tree.
  - Label the branches with a 0 and 1 starting from the root and going to all intermediary nodes.
  - Traverse the Huffman tree from the root to a leaf node noting each branch label (1 or 0). This results in a Huffman code representation for that leaf node symbol. Perform the same for all the symbols. Note that the symbols closer to the root and, hence, most frequent have smaller bit code lengths than the symbols farther away from the root.

Char	Freq
Space	7
A	4
E	4
F	3
H	2
I	2
M	2
N	2
S	2
T	2
L	1
O	1
P	1
R	1
U	1
X	1

pakorn.wa@kmitl.ac.th



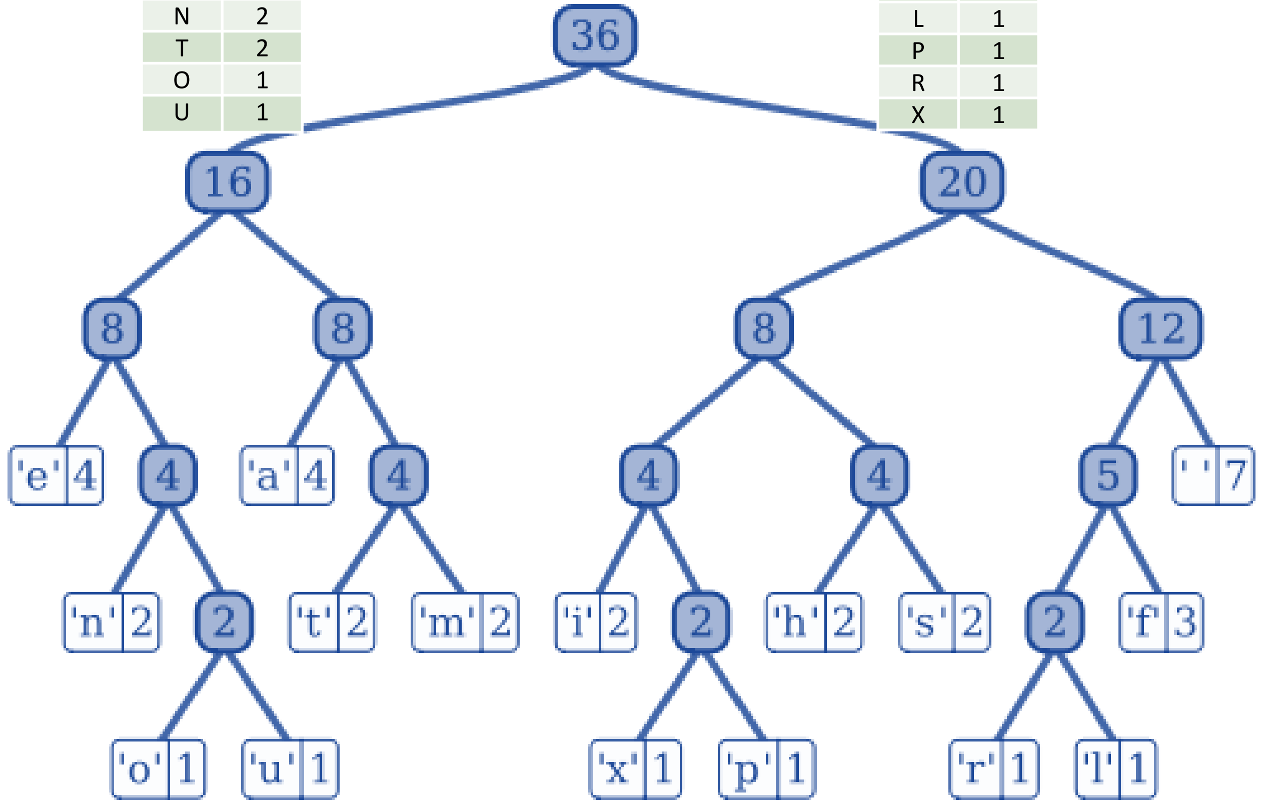
Char	Freq
Space	7
A	4
E	4
F	3
H	2
I	2
M	2
N	2
S	2
T	2
L	1
O	1
P	1
R	1
U	1
X	1

KMITL พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Char	Freq
A	4
E	4
M	2
N	2
T	2
O	1
U	1

Char	Freq
Space	7
F	3
H	2
I	2
S	2
L	1
P	1
R	1
X	1



Char	Freq
Space	7
A	4
E	4
F	3
H	2
I	2
M	2
N	2
S	2
T	2
L	1
O	1
P	1
R	1
U	1
X	1

pakorn.wa@kmitl.ac.th

Char	Freq
A	4
E	4
M	2
N	2
T	2
O	1
U	1

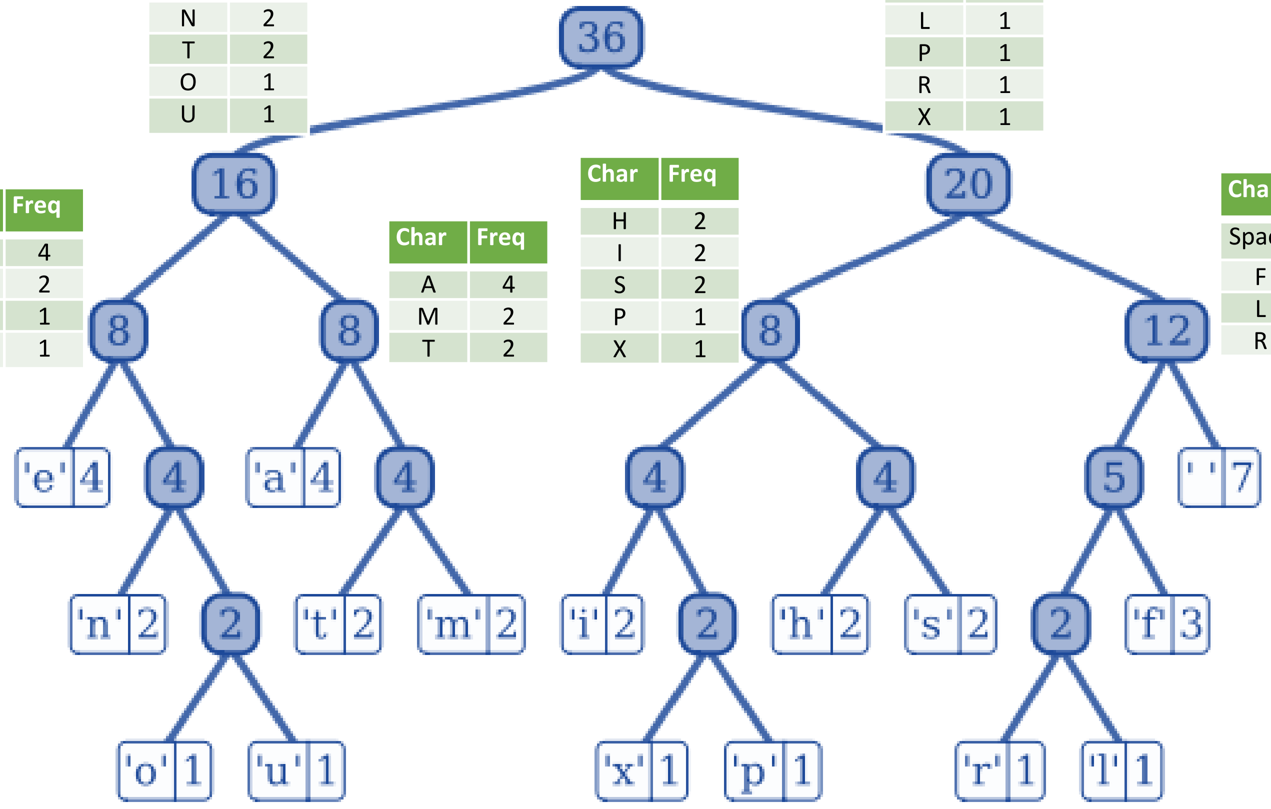
Char	Freq
Space	7
F	3
H	2
I	2
S	2
L	1
P	1
R	1
X	1

Char	Freq
E	4
N	2
O	1
U	1

Char	Freq
A	4
M	2
T	2

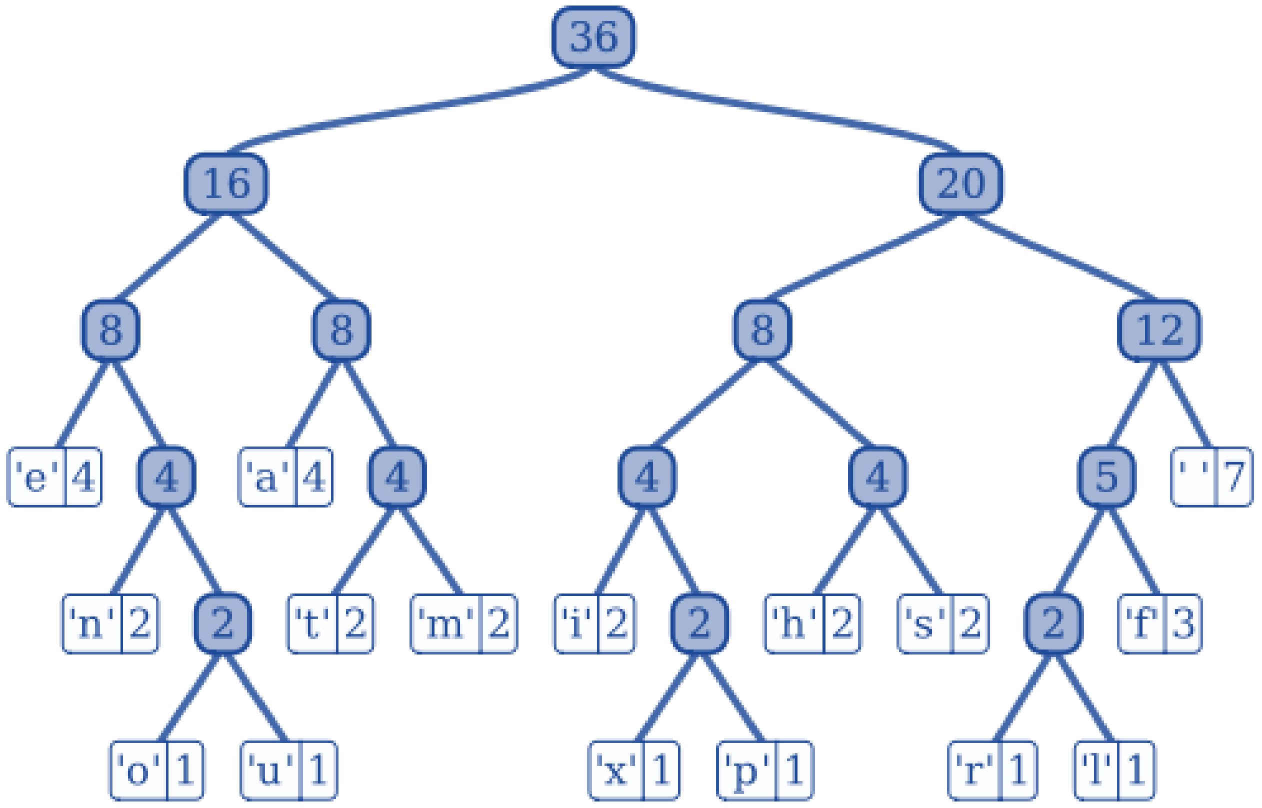
Char	Freq
H	2
I	2
S	2
P	1
X	1

Char	Freq
Space	7
F	3
L	1
R	1



KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Char	Freq	Code
Space	7	111
A	4	010
E	4	000
F	3	1101
H	2	1010
I	2	1000
M	2	0111
N	2	0010
S	2	1011
T	2	0110
L	1	11001
O	1	00110
P	1	10011
R	1	11000
U	1	00111
X	1	10010



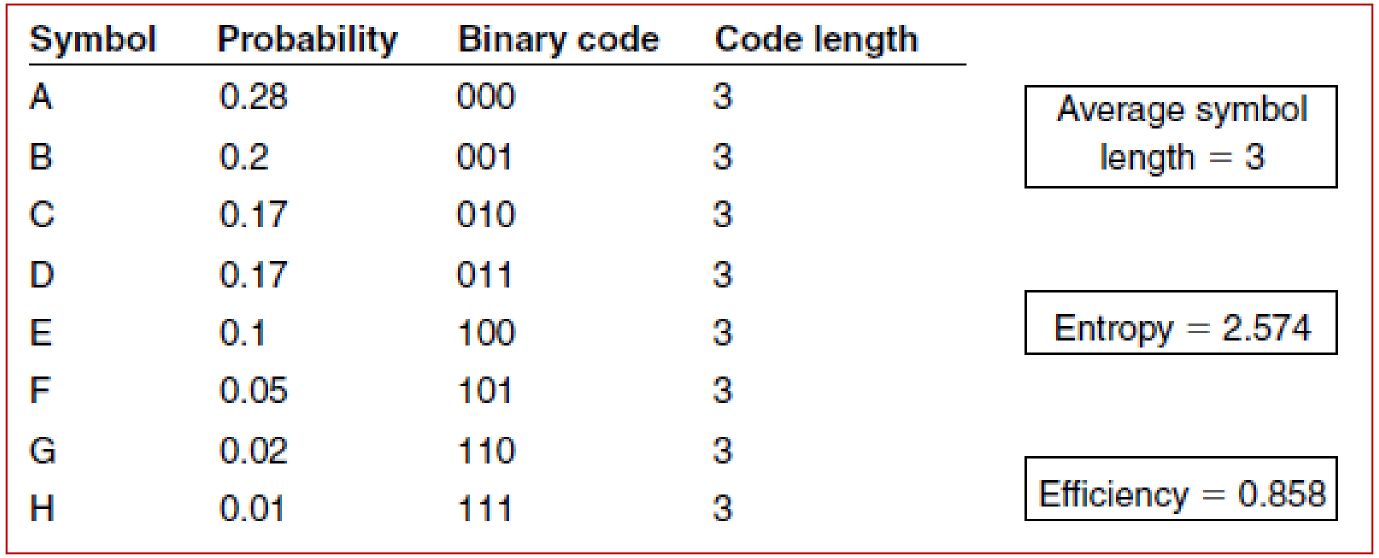
Arithmetic Coding

pakorn.wa@kmitl.ac.th

- Arithmetic coding overcomes this constraint by mapping an entire message to a real number between zero and one.
- This real number representing the entire message is coded as a binary number.
- Arithmetic coding, thus, encodes a message entirely without assigning a fixed binary code to each symbol and, thereby, tends to produce better compression ratios.
- The coding process uses a one-dimensional table of probabilities.
- Given an alphabet of *n* symbols, there are an infinite number of messages that are possible.
- Each message is mapped to a unique real number in the interval [0,1).
- The interval contains an infinite amount of real numbers, so it must be possible to code any message uniquely to one number in the interval.

- The interval is first set at [0,1) for the first symbol, and then partitioned according to the symbol probabilities.
- Depending on the first symbol of the message, an interval is chosen, which is then further partitioned depending on probabilities.

- Suppose we have a vocabulary of n symbols:
  - Divide the interval [0,1) into n segments corresponding to the n symbols; the segment of each symbol has a length proportional to its probability. Each segment i has an upper bound U and lower bound L corresponding to the start of the segment and the end of the segment ($U - L = P$).
  - Choose the segment that corresponds to the first symbol in the message string. This is the new current interval with its computed new upper and lower bounds.
  - Divide the new current interval again into n new segments with length proportional to the symbols probabilities and compute the new intervals accordingly.
  - From these new segments, choose the one corresponding to the next symbol in the message.
  - Continue Steps 3 and 4 until the whole message is coded.
  - Represent the segment's value by a binary fraction in the final interval.



Symbol	Probability	Binary code	Code length
A	0.28	000	3
B	0.2	001	3
C	0.17	010	3
D	0.17	011	3
E	0.1	100	3
F	0.05	101	3
G	0.02	110	3
H	0.01	111	3

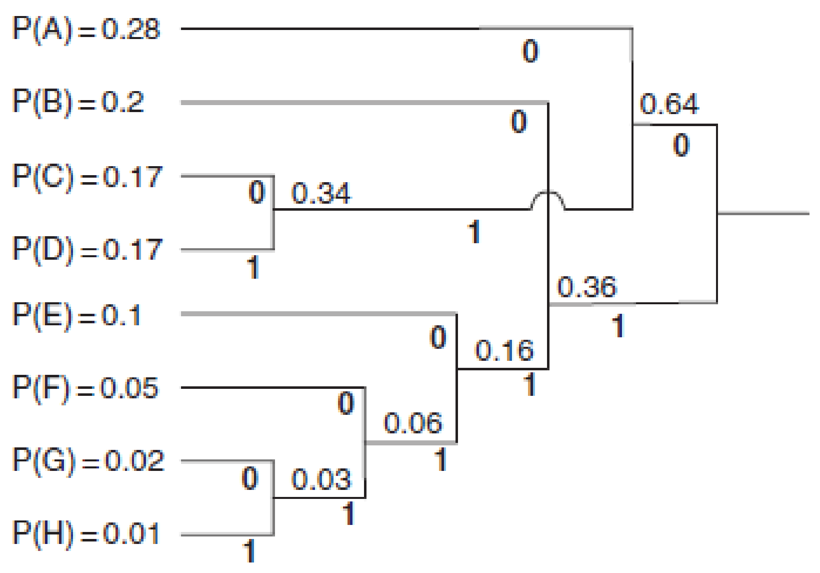
Average symbol length = 3

Entropy = 2.574

Efficiency = 0.858

- The symbols used with their probability distribution and original binary code length.
- Each symbol has the same code length 3.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

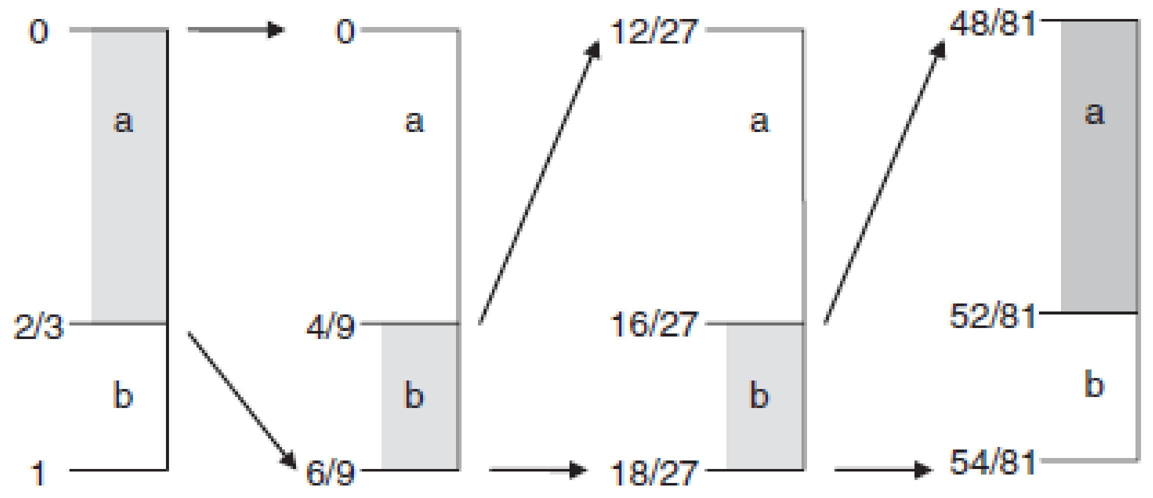


Symbol	Probability	Huffman code	Code length
A	0.28	00	1
B	0.2	10	3
C	0.17	010	3
D	0.17	011	3
E	0.1	110	3
F	0.05	1110	4
G	0.02	11110	5
H	0.01	11111	5

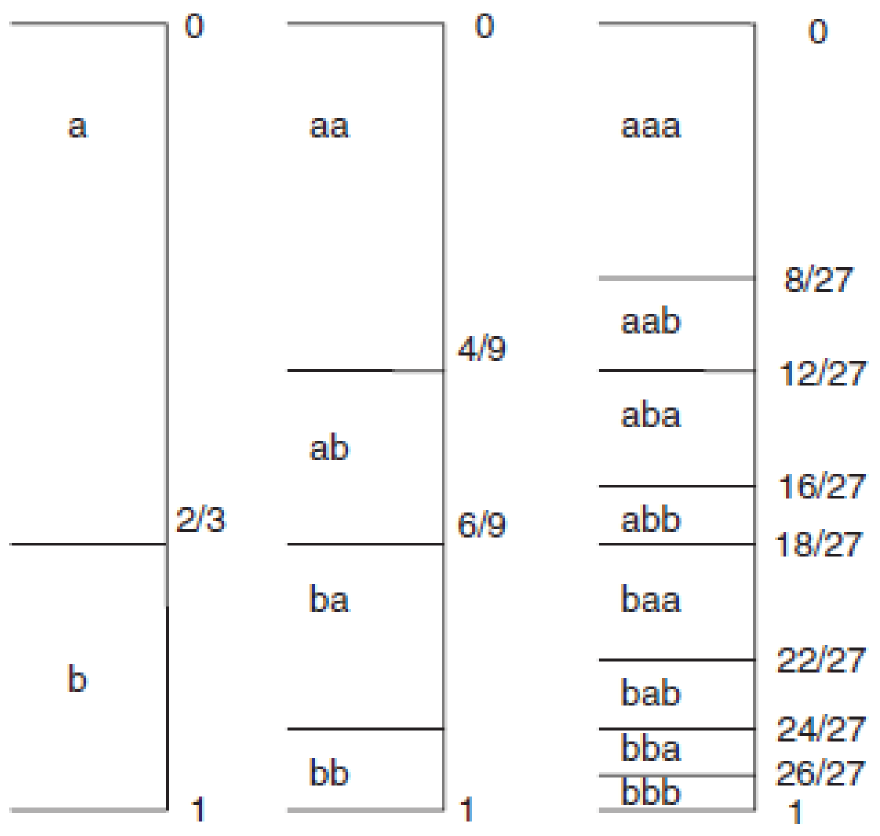
Average symbol length = 2.63

Entropy = 2.574

Efficiency = 0.9792



- Any number in the interval can be chosen as a binary representation of the message.
- The number that is typically chosen is one that has the least number of bits in that interval.
- For the interval [48/81, 52/81], the binary representation of this range (up to eight binary decimals) is [0.10010111, 0.1010010].
- This interval can be represented by 0.101, which corresponds to the decimal 0.625.
- Thus, 101 should suffice as a code for "*abba*"



0
a
2/3
b
1
0
aa
4/9
ab
6/9
ba
bb
1
0
aaa
8/27
aab
12/27
aba
16/27
abb
18/27
baa
22/27
bab
24/27
bba
26/27
bbb
1

Lossy compression

pakorn.wa@kmitl.ac.th

- Entropy-coding or lossless compression has theoretical limits on the amount of compression that can be achieved.
- It might be necessary and appropriate to sacrifice some amount of information and thereby increase the compression obtained.
- For instance, human visual experiments on image perception have shown distortion introduced by image compression is still perceptually acceptable.
- In such cases, the original signal cannot be recovered in its entirety and there is a loss or distortion introduced.
- Most lossy compression schemes introduce a distortion because of quantizing the symbol code representations.
- Quantization is inherently lossy.

pakorn.wa@kmitl.ac.th

Differential PCM

- PCM data consists of digitally coded samples. Signals contain all kinds of frequencies, which cause it to change over time.
- When a signal does not have high frequencies in it, it changes slowly, causing successive samples to be similar in value.
- If successive samples are similar in value, they can be predicted using the current and past samples.
- The closer the prediction is to the actual value, the lesser the prediction error.

- Differential pulse code modulation (DPCM) works by doing the simplest of prediction.
- The simplest prediction is the signal itself.
- If the $(n-1)^{th}$ sample is represented by $y(n-1)$, then the n^{th} sample can be predicted to be $y(n)=y(n-1)$, causing the prediction error to be the difference between the predicted and actual samples as

$$d(n) = y(n) - y(n-1)$$

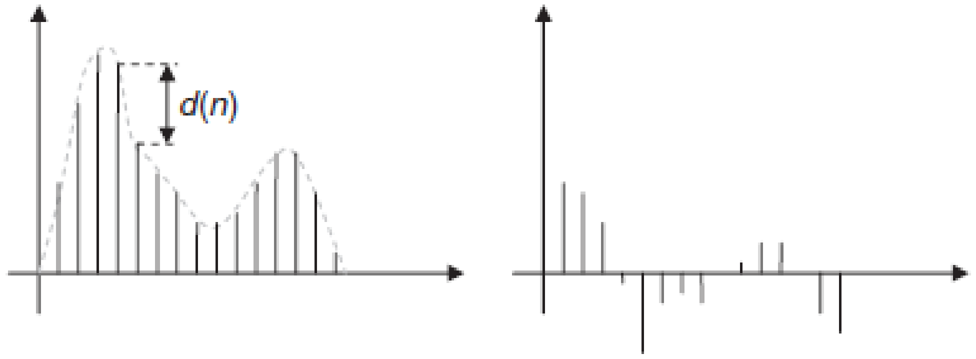
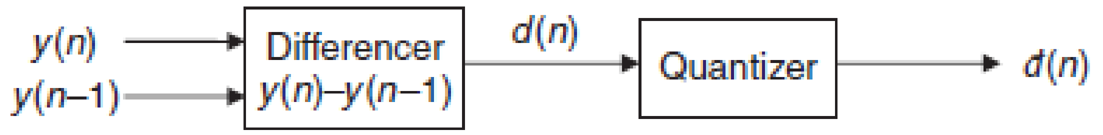


Figure 6-10 Differential pulse code modulation. The left signal shows the original signal. The right signal samples are obtained by computing the differences between successive samples. Note the first difference d(1) is the same as y(1); all the successive differences d(n) are computed as y(n) − y(n − 1).

pakorn.wa@kmitl.ac.th

- At the encoder, DPCM works by sending quantized value $d(n)$ of the signal $d(n)$.
- The decode works by recovering the lossy signal $\hat{y}(n)$ from $\hat{d}(n)$ as follows

if $n = 1,$ $\quad\hat{y}(n) = \hat{d}(n)$
else $\quad\hat{y}(n) = y(n-1) + \hat{d}(n)$



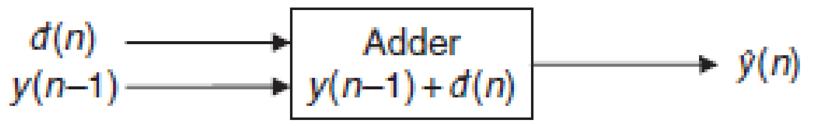
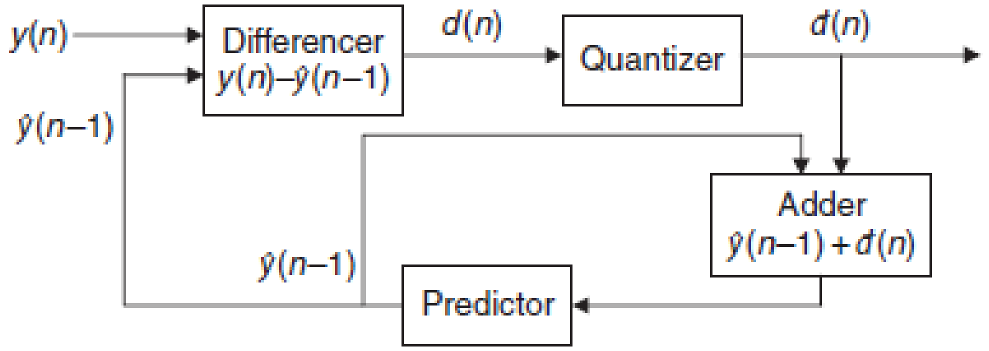


Figure 6-11 Open loop differential pulse code modulation. Encoder (top) works by predicting difference $d(n)$ and quantizing it. Decoder (bottom) works by recovering quantized signal $\hat{y}(n)$ from $\bar{d}(n)$.



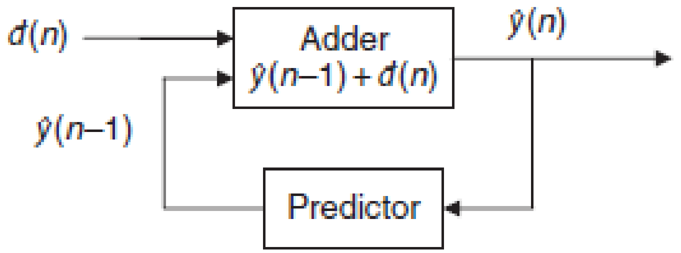


Figure 6-12 Closed loop differential pulse code modulation. At the encoder (top), the difference d(n) is computed as y(n) − ŷ(n-1). This reduces the error that accumulates under the open loop case. A decoder (bottom) is simulated at the encoder; the decoded values are used for computing the difference.

Vector quantization

pakorn.wa@kmitl.ac.th

- Quantization is normally performed on individual samples where the original signal value is quantized to fall in a range, and this range is represented by an index or number.
- Quantization causes a lossy compression and most compression techniques operate by quantizing each symbol taken individually.

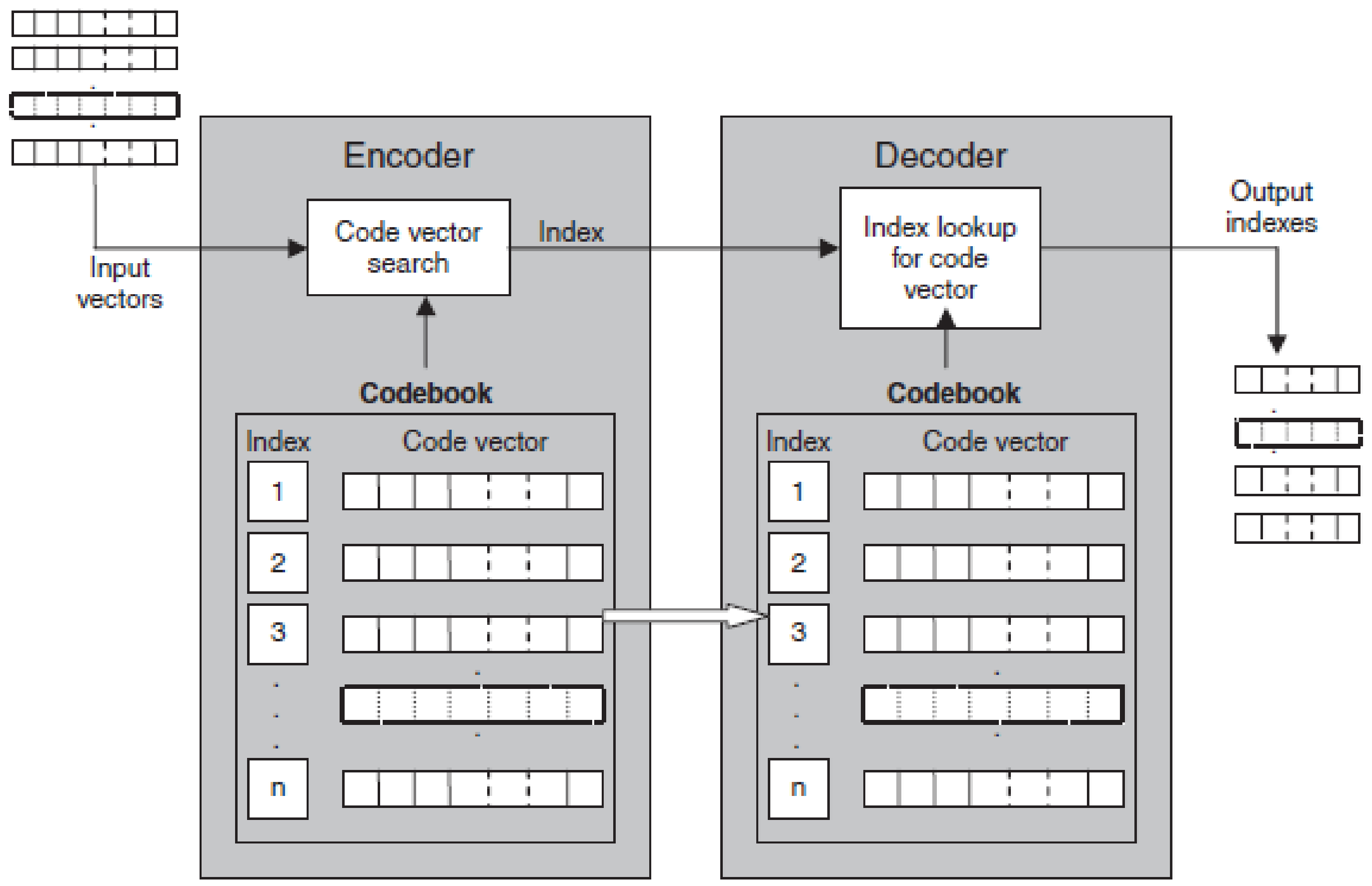
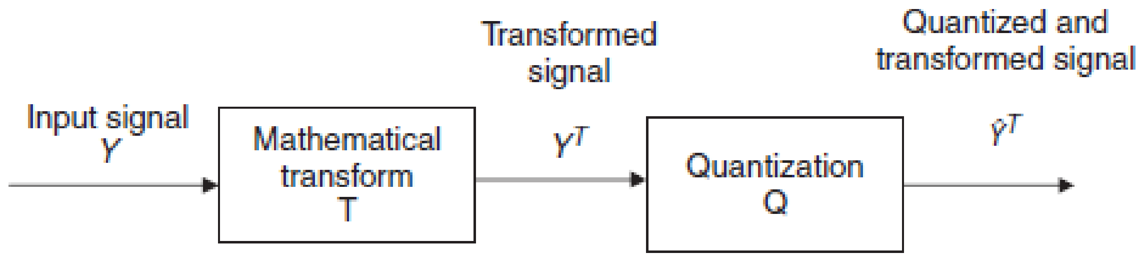
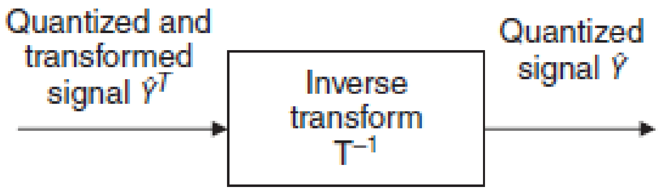


Figure 6-13 Vector quantization. On the left, input data is organized as vectors. In the encoder, a search engine finds the closest vector from the codebook and outputs a codebook index for each input vector. The decoding process (right) proceeds as an index lookup into the codebook and outputs the appropriate vector. Along with the coded indexes, the encoder needs to communicate the codebook to the decoder.

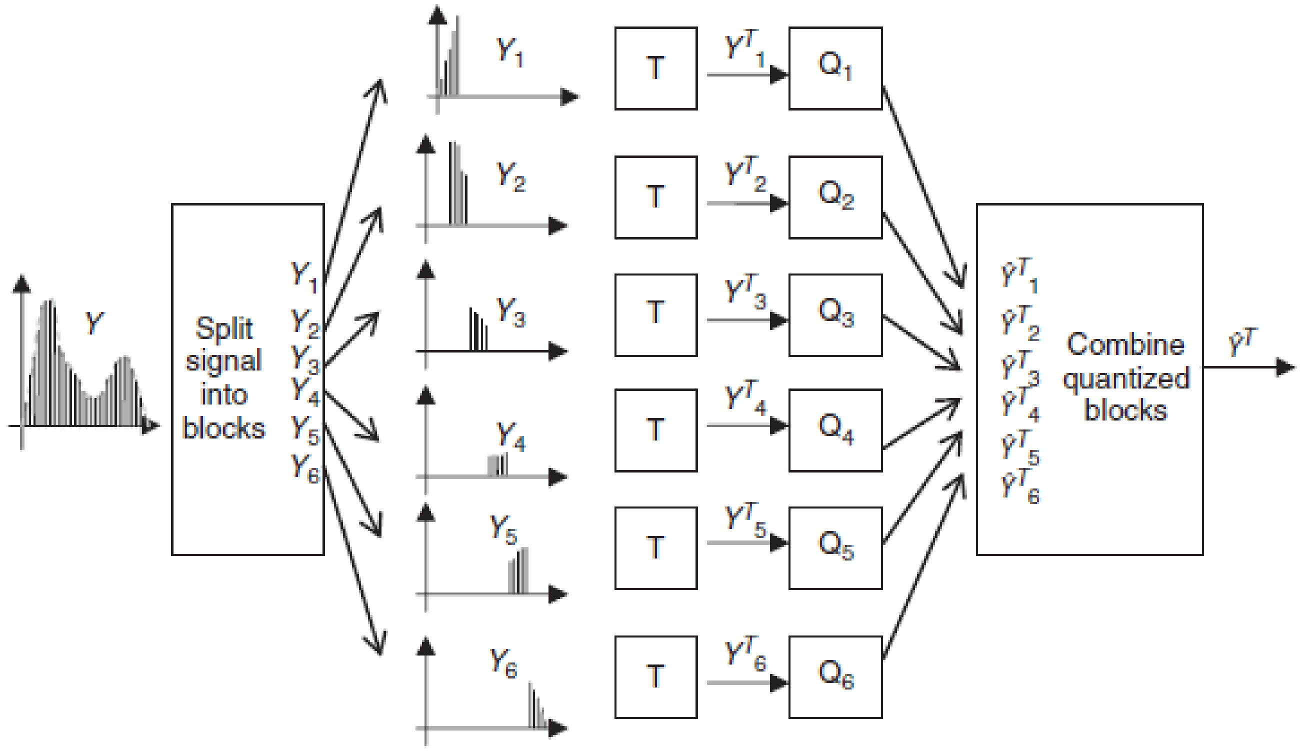
Transform coding

pakorn.wa@kmitl.ac.th

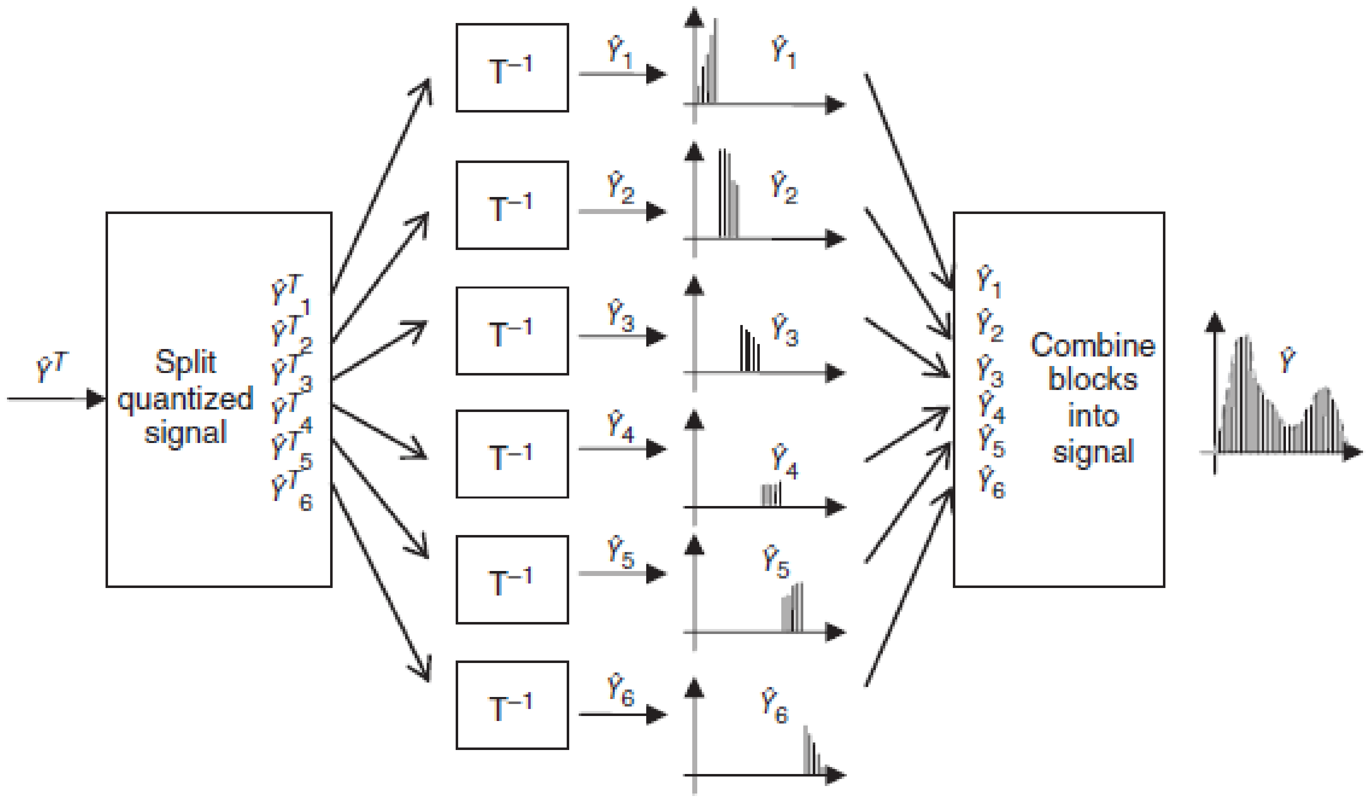



Quantized and transformed signal $\hat{Y}^T$
Inverse transform T^{-1}
Quantized signal $\hat{Y}$

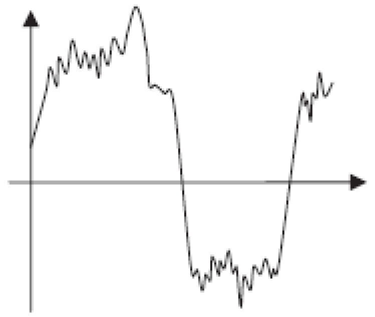
- *Frequency transforms*—Discrete Fourier transforms, Hadamard transforms, Lapped Orthogonal transforms, Discrete Cosine transforms
- *Statistical transforms*—Karhunen-Loeve transforms
- *Wavelet transforms*—While similar to frequency transforms, these transforms work more efficiently because the input is transformed to a multiresolution frequency representation.

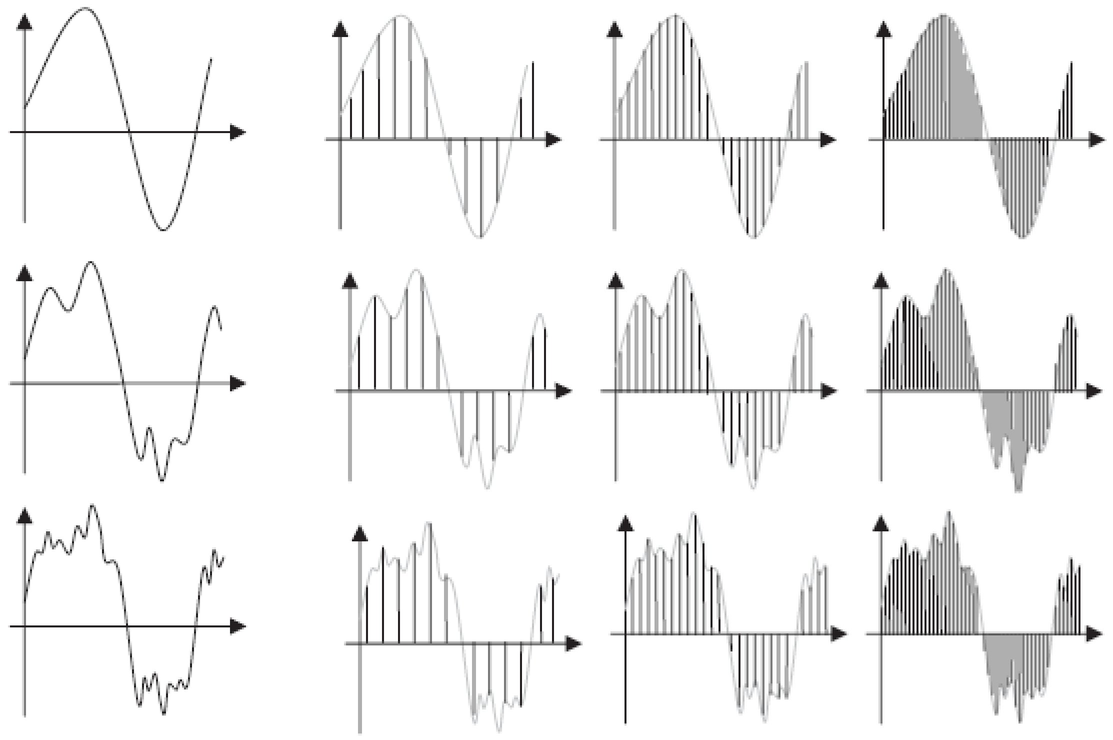

Y
Split signal into blocks
Y_1
Y_2
Y_3
Y_4
Y_5
Y_6
T
Y^T_1
Y^T_2
Y^T_3
Y^T_4
Y^T_5
Y^T_6
Q_1
Q_2
Q_3
Q_4
Q_5
Q_6
$\hat{Y}^T_1$
$\hat{Y}^T_2$
$\hat{Y}^T_3$
$\hat{Y}^T_4$
$\hat{Y}^T_5$
$\hat{Y}^T_6$
Combine quantized blocks
$\hat{Y}^T$

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



Subband coding





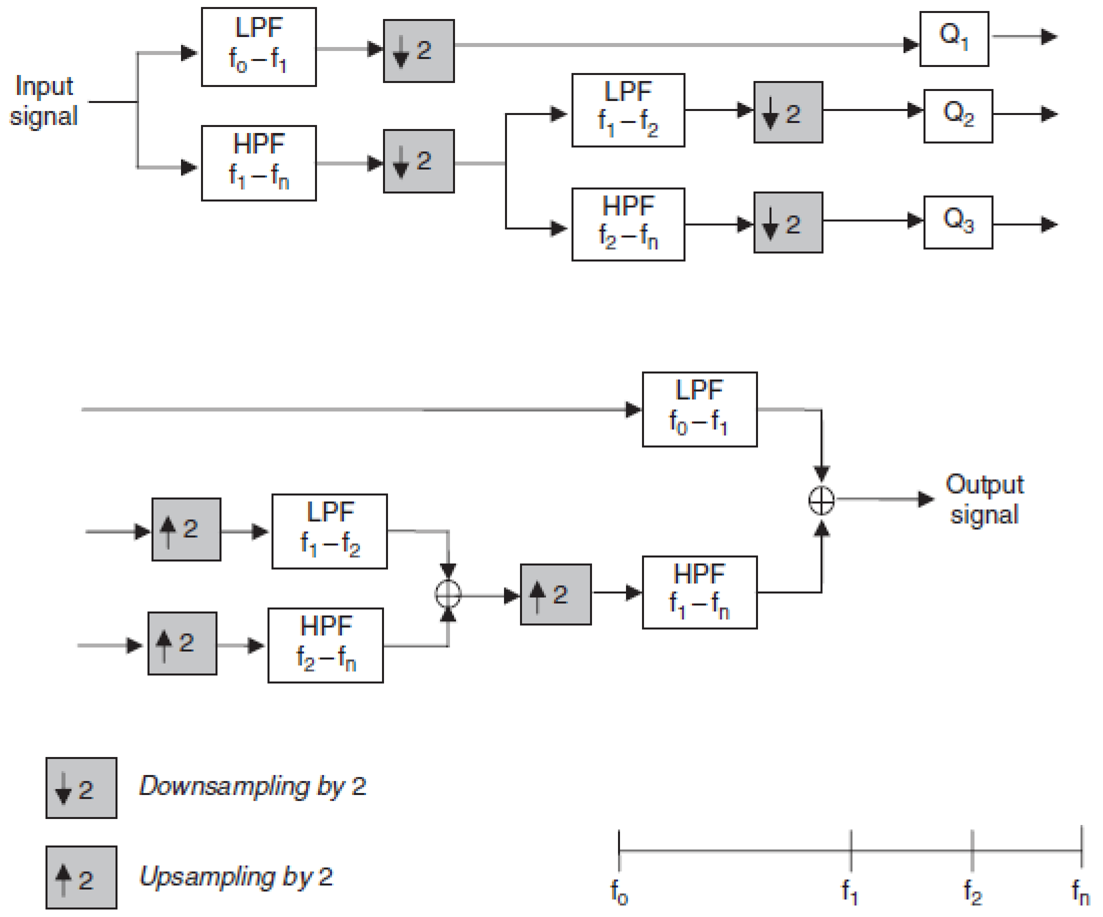


Figure 6-17 Subband coding. The top figure shows a subband encoder till level 2—the input signal is passed through low- and high-pass filters, with the low-pass outputs downsampled. The resulting signals are quantized differently. The bottom figure shows the decoder, which adds together the different filtered signals to produce the final output.

Hybrid compression techniques

pakorn.wa@kmitl.ac.th

- The main goal of compression, whether lossy or lossless, is to minimize the data used to represent information.
  - In the case of lossy compression, the process is subject to a distortion evaluation so that the signal perception remains relatively unaffected when evaluated by the human perceptual system.
  - In the case of lossless compression, there is no distortion, but the method is subject to theoretical limits of how much compression can be achieved.
- Hybrid methods combine the good properties of these various processes.

Practical issues related to compression systems

pakorn.wa@kmitl.ac.th

- Encoder speed and complexity
  - Performance requirements
  - Data available for analysis
- Rate control
  - In traditional media communication applications, the signal is compressed **offline** and stored on disc.
  - A server then serves or streams this data through a network to the end viewer.
  - the major constraint for the encoding/decoding system is the storage space generated by the encoded files and transmission bandwidth available to serve the end client.

- Symmetric and asymmetric compression
  - Symmetric compression refers to the case when the decoder's execution is the exact reverse of the encoder's execution.
  - An asymmetric compression method is one where either the encoder or the decoder is more complex than the other, consequently having unequal encoding and decoding times.
- Adaptive and non-adaptive compression
  - Is it capable of adapting its encoding technique depending on the data that is produced by the source.

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Q & A