ELEC S347F Multimedia Technologies

Data Compression Basics

Multimedia

- Text
  - Plain Text, Rich-Formatted Text, Hyper Text
- Audio
  - Speech, Music

What are their major differences?

- Image
  - Graphic, Picture
- Video
  - Animation, Motion Picture

Properties of Multimedia Data

- Raw CD-Quality Audio
  - 44,100 samples per second
  - 16 bits per sample
  - Stereo (2 channels)
  - Requires 1,411,200 bits per second
  - A 3-min song requires 1.4 Mbps x 180 s = 242 Mb = 30 MB
- Raw High Definition Television (HDTV)
  - 10 bits @ 1920 x 1080 @ 60fps = 445 MB per second
  - A two-hour movie requires 3.2 TB (including audio)
- Raw 4K Ultra High Definition Television (4K UHDTV)
  - 10 bits @ 3840 x 2160 @ 60fps = 1.3 GB per second
  - A two-hour movie requires 12.3 TB (including audio)

Properties of Multimedia Data

- The multimedia data can be very large
  - Video >> Image >> Audio >> Text
  - Large size not favor for storage, communication and processing
  - Relying on higher storage space and bandwidth is not a good option
  - Data/traffic will always increase to fill the current storage/bandwidth limit whatever this is
- Solution: reduce the size of multimedia data
  - Compression: find any redundancy and remove it
  - Synthesis: using high-level representation to describe the multimedia data

Course Content

- The course will mainly discuss
  - Compression of multimedia data
  - Synthesis of multimedia data
  - Delivery of multimedia data

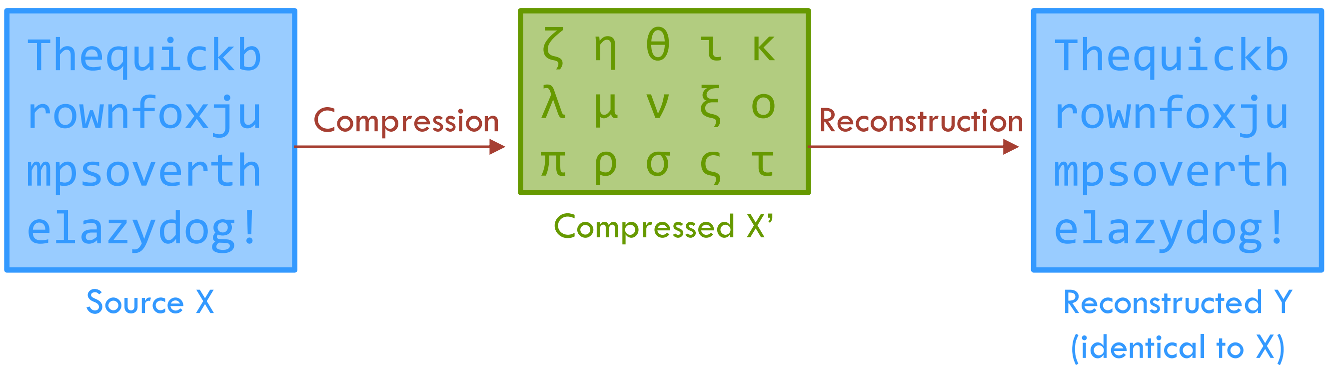
Compression

Compression Techniques

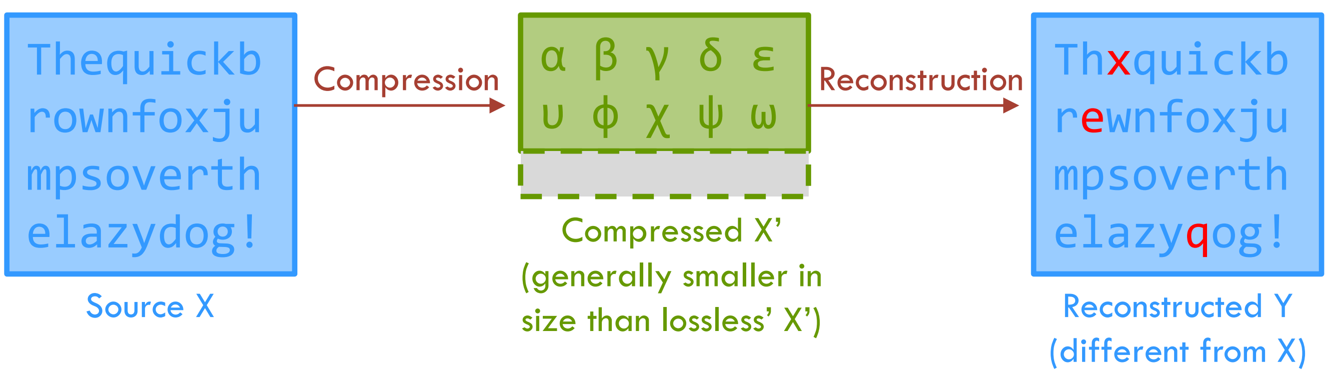
- When speak of a compression technique, actually referring to two kinds of algorithms
  - Compression algorithm, and
  - Reconstruction algorithm
- There is the compression algorithm that takes an input X and generates a representation X' that requires fewer bits
- There is a reconstruction algorithm that operates on the compressed representation X' to generate the reconstruction Y
- The convention: compression algorithm refers to both the compression and reconstruction algorithms

Compression Schemes

- Lossless Compression Schemes



- Lossy Compression Schemes



Compression Schemes

- Based on the requirements of reconstruction, data compression schemes can be divided into two broad classes
  - Lossless compression scheme, and
  - Lossy compression scheme
- Lossless Compression Schemes: Y is identical to X
- Lossy Compression Schemes: Y is different from X, but generally X' of lossy schemes are smaller in size than that generated by lossless scheme

Compression in Multimedia Data

- Compression basically employs redundancy in the data
- Temporal Redundancy
  - Exploit correlation of data in time domain
- Spatial Redundancy
  - Exploit correlation of data in space domain
- Spectral Redundancy
  - Exploit correlation of data in frequency domain
- Psycho-Visual Redundancy
  - Exploit perceptual properties of the human visual system

Measures of Performance

- A compression algorithm can be evaluated in a number of different ways
  - Complexity
    - The amount of time and memory required to run the algorithm
    - Highly related to the cost of implementation
  - Efficiency
    - The amount of compression in size
  - Distortion
    - How closely the reconstruction resembles the original

Compression Efficiency

- There are several ways to express how efficient of a compression algorithm
- Code Rate
  - The average no. of bits required to represent a sample
  - The smaller the value, the better the rate
- Code Efficiency (= Entropy / Code Rate)
  - The ratio between the best possible code rate and the algorithm's code rate
  - The larger the value, the better the compression algorithm
- Code Redundancy (= Code Rate – Entropy)
  - The difference between the algorithm's code rate and the best possible code rate
  - The smaller the value, the better the compression algorithm

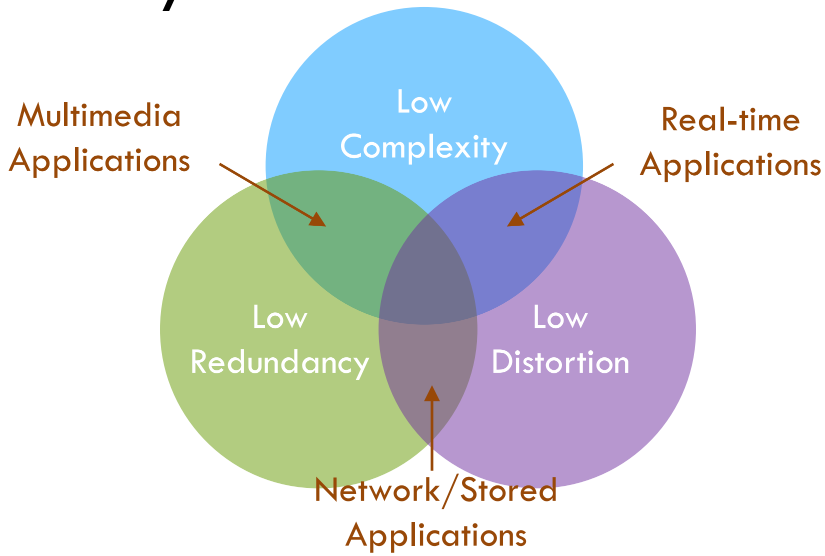
(The meaning of Entropy will be defined later)

Distortion

- Measurement for lossy compressions only
- The commonly used measurements
  - Root Mean Square Error (RMSE)
    - $RMSE = \sqrt{\frac{\sum_{i=1}^{n}[Y(i)-X(i)]^2}{n}}$
  - Normalized Mean Square Error (NMSE)
    - $NMSE = \frac{\sum_{i=1}^{n}[Y(i)-X(i)]^2}{\left[\sum_{i=1}^{n} X(i)\right]^2}$
  - Peak Signal-to-Noise Ratio (PNSR)
    - $PSNR = \left(\frac{Peak\ Value}{RMSE}\right)^2$

Complexity vs. Efficiency vs. Distortion

- A compression algorithm is considered as good if it is in low complexity, low redundancy and low distortion
  - However, these three cannot be achieved at the same time very often



Modeling and Coding

- The development of data compression algorithm for a variety of data can be divided into two phases
  - The first phase: modeling
  - The second phase: coding
- Modeling
  - Extract information about any redundancy that exists in the data
  - Describe the redundancy in the form of a model
- Coding
  - Encode the description of the model and the data

Modeling: Example 1

- Consider the following sequence of numbers $\{x_1, x_2, x_3, \ldots\}$:

$\{x_1, x_2, x_3, ..\}$	9	11	11	11	14	13	15	17	16	17	20	21

  - If the data are represented as natural binary numbers, code rate = 5 bits per sample

$\{x'_1, x'_2, x'_3, ..\}$	01001	01011	01011	01011	01110	01101	01111	10001	10000	10001	10011	10101

  - The whole sequence requires 12 x 5 = 60 bits
  - Can we use fewer bits to representation the sequence?

Modeling: Example 1

- Let's exploit the structure in the data
  - The sequence is close to a linear function
  - Define model $x'_n = n + 8$, residual $e_n = x_n - x'_n$

$\{x_1, x_2, x_3, ..\}$	9	11	11	11	14	13	15	17	16	17	20	21
$\{x'_1, x'_2, x'_3, ..\}$	9	10	11	12	13	14	15	16	17	18	19	20

  - The residual (difference between the data and the model) consists of only three numbers -1, 0, 1

$\{e_1, e_2, e_3, ..\}$	0	1	0	-1	1	-1	0	1	-1	-1	1	1

  - Only require 2 bits to represent each element of the residual sequence
  - With the description of the model and the residual sequence, it is possible to reconstruct the x_n sequence

Modeling

- Once we recognize the structure of the data, we can make use of the structure to predict the value of each element in the sequence
- There exists different types of structures
- See below example

Modeling: Example 2

- Consider the following sequence of numbers

$\{x_1, x_2, x_3, ..\}$	27	28	29	28	26	27	29	28	30	32	34	36	38

  - If the data are represented as binary, code rate = $\lceil \log_2(38) \rceil$ = 6 bits per sample
  - The whole sequence requires 13 x 6 = 78 bits
  - If using the modeling, $x'_n = n + 26$

$\{x'_1, x'_2, x'_3, ..\}$	27	28	29	30	31	32	33	34	35	36	37	38	39
$\{e_1, e_2, e_3, ..\}$	0	0	0	-2	-5	-5	-4	-6	-5	-4	-3	-2	-1

  - There are 7 distinct values in the residual sequence

Modeling: Example 2

- The sequence is not close to a linear function
  - However, the value in this sequence is close to the previous value

$\{x_1, x_2, x_3, ..\}$	27	28	29	28	26	27	29	28	30	32	34	36	38

  - Define the model $x'_n = \begin{cases} x_n & , n = 1 \\ x_{n-1}, & n \geq 2 \end{cases}$

$\{x'_1, x'_2, x'_3, ..\}$	27	27	28	29	28	26	27	29	28	30	32	34	36
$\{e_1, e_2, e_3, ..\}$	0	1	1	-1	-2	1	2	-1	2	2	2	2	2

  - There are 5 distinct values in the residual sequence
    - Can use fewer bits to represent the residuals now!
    - With the description of the model, the first value of x_n and the residual sequence, it is possible to reconstruct the x_n sequence

Modeling: Example 3

- Consider the following sequence of numbers

$\{x_1, x_2, x_3, ..\}$

1	1	1	1	2	2	2	3	3	3	3	3	3

  - It can be represented as {(1,4),(2,3),(3,6)}
- This method is called Run Length Encoding (RLE)
  - Map a sequence of n symbols: $\{x_1, x_1, \cdots x_n\}$ to m pairs: $\{(v_1, c_1), (v_2, c_2), \cdots (v_m, c_m)\}$
  - where $m \leq n$, v_i represents the value of x_k
  - c_i represents the no. of consecutive repeated x_k

How Good is RLE?

- Dependent on the input data
  - If the data are consecutive repeated frequently
    - e.g. input = {1,1,1,1,1,1,1,1,1,1,1,1,1,1,3,3,3,3,3,3,3}
    - RLE = {(1,14),(3,7)}
    - Often happened in images
  - If the data is non-consecutively repeated, the encoded sequence could be more heavy than the input
    - e.g. input = {1,2,3,4,5,1,2,3,4,5}
    - RLE = {(1,1),(2,1),(3,1),(4,1),(5,1),(1,1),(2,1),(3,1),(4,1),(5,1)}
    - The worst case: twice the data rate than the input
    - Often happened in text data
    - How about {({1,2,3,4,5},2)}?

Modeling: Summary

- Example 1 uses the difference between the data and the model to predict the values of a sequence
- Example 2 uses the past values of a sequence to predict the current value
- Example 3 maps the data to pairs of values, group a sequences of data as a symbol
- How to represent (encode) each of the distinct values/symbols of the residual sequence/mapped sequence? See the below pages

Information Theory

Self-Information

- Shannon defined a quantity called self-information
  - For an event A, which is a set of outcomes of some random experiment
  - If $P(A)$ is the probability that the event A will occur
  - The self-information associated with A is given by
    - $i(A) = \log_b \frac{1}{P(A)} = -\log_b P(A)$
  - If the probability of an event is low, the amount of self-information associated with it is high
  - In contrast, if $P(A)$ is high, $i(A)$ is low

Self-Information: Example 1

- What does it mean by self-information?
  - Event A = my dog barks at night
  - $P(A)$ = the probability that my dog barks at night
  - If my dog barks, there is some information $i(A)$ (e.g. a thief entered my home, there is a fire, etc)
  - However, if my dog often barks (i.e. $P(A)$ is high), the information associated with it is low (i.e. $i(A)$ is low)
  - In contrast, if my dog seldom barks (i.e. $P(A)$ is low) and when it barks, it contains a lot of information (i.e. $i(A)$ is high)

Self-Information: Example 2

- Let *H* and *T* be the outcomes of flipping a coin
  - If the coin is fair, $P(H) = P(T) = \frac{1}{2}$
    - $i(H) = i(T) = -\log_b P(H)$
    - If we use base 2 for the log, the unit is bits (binary digits)
    - $-\log_b P(H) = -\log_2 \frac{1}{2} = 1\ bit$
  - If the coin is not fair, we would expect the information associated with each event to be different
    - Suppose $P(H) = \frac{1}{8}, P(T) = \frac{7}{8}$
    - $i(H) = -\log_b P(H) = -\log_2 \frac{1}{8} = 3\ bits$
    - $i(T) = -\log_b P(T) = -\log_2 \frac{7}{8} = 0.193\ bits$

Entropy

- If we have a set of independent events A_i, which are sets of outcomes of some experiment $\mathcal{S}$ such that $\bigcup A_i = \mathcal{S}$
  - The expected value of the self-information associated with the random experiment is given by
  - $H = \sum P(A_i) i(A_i) = -\sum P(A_i) \log_b P(A_i)$
  - H is the entropy associated with the experiment

Entropy: Example 1

- Consider the outcome of a fair coin:
  - H, T, T, H, H, T, H, T
  - The probability of occurrence of each symbol:
  - $P(H) = P(T) = \frac{1}{2}$
  - Since the sequence is independent and identically distributed (i.i.d.), the entropy for this sequence
  - $H = -P(H)\log_2 P(H) - P(T)\log_2 P(T) = 1\, bit$
  - It means that the best scheme we could find for coding this sequence could only code it at 1 bit/sample

Entropy: Example 2

- Consider the outcome of an unfair coin:
  - H, T, T, T, T, T, T, T
  - The probability of occurrence of each symbol:
  - $P(H) = \frac{1}{8}, P(T) = \frac{7}{8}$
  - Since the sequence is iid, the entropy for this sequence
  - $H = -P(H)\log_2 P(H) - P(T)\log_2 P(T)$
  - $= 0.544\ bits$
  - It means that the best scheme we could find for coding this sequence could only code it at 0.544 bits/sample
  - Why can use fewer bits per sample?

Entropy: Example 3

- Consider the following sequence:
  - 1, 2, 3, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10
  - The probability of occurrence of each symbol:
  - $P(1) = P(6) = P(7) = P(10) = \frac{1}{16}$,
  - $P(2) = P(3) = P(4) = P(5) = P(8) = P(9) = \frac{2}{16}$
  - Assume the sequence is iid, the entropy for this sequence
  - $H = -\sum P(i) \log_2 P(i) = 3.25\ bits$
  - It means that the best scheme we could find for coding this sequence could only code it at 3.25 bits/sample

Entropy: Example 3

- There is sample-to-sample correlation between the samples
  - If we remove the correlation by taking difference of neighboring sample values
  - x_n = {1, 2, 3, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10}
  - e_n = {1, 1, 1, -1, 1, 1, 1, -1, 1, 1, 1, 1, 1, -1, 1, 1}
  - $P(1) = \frac{13}{16}, P(-1) = \frac{3}{16}$
  - $H = -P(1)\log_2 P(1) - P(-1)\log_2 P(-1) = 0.70\ bits$
- Conclusion: knowing the structure of the sequence and choosing proper modeling can help to "reduce the entropy"

Entropy: Example 4

- Consider the following sequence:
  - 1, 2, 1, 2, 3, 3, 3, 3, 1, 2, 3, 3, 3, 3, 1, 2, 3, 3, 1, 2
  - The probability of occurrence of each symbol:
  - $P(1) = P(2) = \frac{1}{4}, P(3) = \frac{1}{2}$
  - $H = -\sum P(i) \log_2 P(i) = 1.5\ bits$
  - Total number of bits required to represent the sequence = 1.5 x 20 = 30 bits

Entropy: Example 4

- How about picking larger blocks of data to calculate the probability over?
  - The sequence becomes {{1, 2}, {1, 2}, {3, 3}, {3, 3}, {1, 2}, {3, 3}, {3, 3}, {1, 2}, {3, 3} , {1, 2}}
  - The probability of occurrence of each "symbol":
  - $P(\{1,2\}) = P(\{3,3\}) = \frac{1}{2}$
  - $H = -\sum P(i) \log_2 P(i) = 1\ bit$
  - Total number of bits required to represent the sequence = 1 x 10 = 10 bits (a reduction of a factor of 3)
- Conclusion: extract the structure of the data by taking larger block sizes could help to reduce the entropy

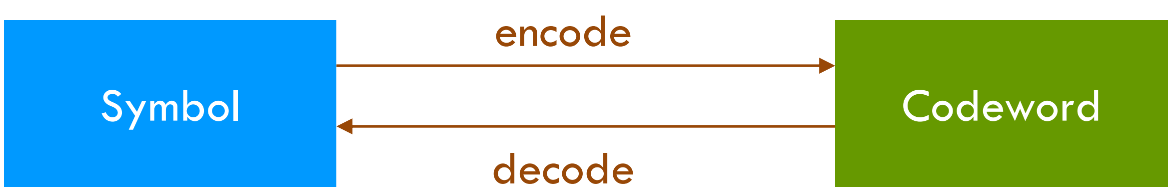
Coding

Coding

- Coding is the assignment of binary sequences to symbols
  - The set of binary sequences is called a code
  - The individual binary sequences are called codewords
- e.g. for ASCII code
  - 'A' coded as 1000001, 'B' coded as 1000010, etc
  - 1000001 is the codeword of symbol 'A'
  - The set {1000001, 1000010, ..} is called the code
- Of course, we do not arbitrary assign the binary values
  - We should aim to achieve Shannon's limit (i.e. entropy)
  - What are the strategies? (see the coming pages)

Encode and Decode

- Encode
  - The process of mapping symbols to codewords
- Decode
  - The process of mapping codewords to symbols



Example: 'A' '1000001'

Classification of Codes

- Fixed-length codes vs. variable-length codes
- Uniquely decodable codes vs. non-uniquely decodable codes
- Instantaneous codes vs. non-instantaneous codes

Fixed-Length vs. Variable-Length

- ASCII code is a kind of fixed-length codes
  - All codewords use the same number of bits to represent each symbol
  - e.g. 7 bits per symbol for ASCII code
- In contrast, variable-length codes use non-identical number of bits to represent each symbol
  - e.g. A coded as '011', B coded as '1011', C coded as '110'
- What are their advantages?
  - Fixed-length codes are more resilient to errors and allow parallel encoding/decoding (why?)
  - Variable-length codes generally allow better compression ratio
    - In general, if we use fewer bits to represent symbols that occur more often, on the average we would use fewer bits for the whole sequence

Uniquely vs. Non-Uniquely Decodable

- The assignment of binary sequences to the symbols may introduce ambiguity in decoding
- Example

Symbol	A	B	C	D
Codeword	0	1	00	11

  - If the encoded sequence is 1100,
  - The original sequence may be {BBAA}, {BBC}, {DAA}, {DC}
- Example

Symbol	A	B	C	D
Codeword	0	10	110	111

  - If the encoded sequence is 1100,
  - The original sequence must be {CA}
- If any given sequence of codewords can be decoded in one and only one way, it is an uniquely decodable code

Instantaneous vs. Non-Instantaneous

- Consider the below two codes:

Code X

Symbol	A	B	C
Codeword	0	10	110

Code Y

Symbol	A	B	C
Codeword	0	01	11

  - For the decoder of code X, once it meets a 0, it is the end of a codeword (since all codewords ends with 0)
    - e.g. 011010 = 00010 =
  - But for the decoder of code Y, it has to wait till the beginning of the next codeword or sometimes the end of the sequence before it can decode correctly
    - e.g. 01111 = 01110 =
  - Code X is uniquely decodable and instantaneous
  - While code Y is uniquely decodable but non-instantaneous

Unique Decodability

- An uniquely decodable code is not necessary instantaneous
- Instantaneous is preferred (allow on-the-fly decoding), however is not a must property
- In contrast, uniquely decodable is a must
- Otherwise the original sequence cannot be recovered with certainty
- How to ensure / determine whether a code is uniquely decodable?

Unique Decodability Test

- For 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, a is called a prefix of b
- The last n-k bits of b are called the dangling suffix
  - e.g. a = 010, b = 01011
  - a is a prefix of b, and the dangling suffix is "11"

Unique Decodability Test

- Sardinas-Patterson Algorithm
  - 1) Construct a list $\mathcal{S}$ of all the codewords
  - 2) Examine all pairs of codewords from $\mathcal{S}$ to see if any codeword is a prefix of another codeword
  - 3) When there is such a pair, add the dangling suffix to $\mathcal{S}$
  - 4) Repeat step 2 until (a) there is no more unique dangling suffixes, or (b) the dangling suffix is a codeword
- If the algorithm ends with condition (a),
  - The code is uniquely decodable
- If the algorithm ends with condition (b),
  - The code is non-uniquely decodable

Unique Decodability Test: Example 1

- Determine whether code X is uniquely decodable or not

Symbol	A	B	C
Codeword	0	01	11

- 1st Iteration: $\mathcal{S} = \{0, 01, 11\}$
  - Codeword "0" is a prefix of codeword "01", the dangling suffix is "1"
- 2nd Iteration: $\mathcal{S} = \{0, 01, 11, 1\}$
  - "1" is the prefix of the codeword "11", the dangling suffix is "1" (However, "1" is in the list already)
  - We cannot find other pairs that would generate new dangling suffix
  - The test then ends with condition (a)
- Conclusion: code X is uniquely decodable

Unique Decodability Test: Example 2

- Determine whether code Y is uniquely decodable or not

Symbol	A	B	C
Codeword	0	01	10

- 1st Iteration: $\mathcal{S} = \{0, 01, 10\}$
  - Codeword "0" is a prefix of codeword "01", the dangling suffix is "1"
- 2nd Iteration: $\mathcal{S} = \{0, 01, 10, 1\}$
  - "1" is a prefix of codeword "10", the dangling suffix is "0" ("0" is one of the codeword!)
  - The test then ends with condition (b)
- Conclusion: code Y is not uniquely decodable

Prefix Codes

- Prefix code: a code with no codeword is a prefix to another codeword
  - Always uniquely decodable and instantaneous (why?)
  - Since no codeword is a prefix to another codeword, there will be no dangling suffixes
  - That means that it never have a dangling suffix identical to codeword (the test always ends with (a))
- Example
  - Which of the below codes belong to prefix codes?

Symbol	Codeword
A	01
B	10
C	110

Symbol	Codeword
A	0
B	01
C	11

Symbol	Codeword
A	0
B	01
C	10

Summary

- Lossless compression techniques involve no loss of information
  - Data can be recovered exactly from the compressed data
  - Generally used for applications that cannot tolerate any difference between the original and reconstructed data (e.g. text data)
- Lossy compression techniques involve some loss of information
  - Data generally cannot be recovered exactly
  - Generally obtain higher compression ratio than is possible with lossless compression
  - Usually applied in multimedia data