

# Chapter 6

## Data Compression

- Data compression: is the process of reducing the size of a data file. it is a reduction in the number of bits needed to represent data. Compressing data can save storage capacity, speed file transfer, and decrease costs for storage hardware and network bandwidth.
- it is also named as **source coding**(In the context of data transmission), or **bit-rate reduction** involves encoding information using fewer bits than the original representation.

- Compression is performed by a **program** that uses a formula or **algorithm** to determine how to shrink the size of the data. Text compression can be as simple as removing all unneeded characters, inserting a single repeat character to indicate a string of repeated characters, and substituting a smaller bit string for a frequently occurring bit string. Compression can reduce a text file to 50% or a significantly higher percentage of its original size.

# Lossless and lossy compression

- $\text{compressedData} = \text{compress}(\text{originalData})$
- $\text{decompressedData} = \text{decompress}(\text{compressedData})$
- When  $\text{originalData} = \text{decompressedData}$ , the compression is lossless.
- When  $\text{originalData} \neq \text{decompressedData}$ , the compression is lossy.

- Lossy compression:-** permanently eliminates bits of data that are redundant, unimportant or imperceptible.
- It is useful with graphics, audio, video and images, where the removal of some data bits has little or no visible effect on the representation of the content.
  - Graphics image compression can be lossy or lossless. Graphic image file formats are typically designed to compress information since the files tend to be large.

- JPEG is an image file format that supports lossy image compression. Formats such as GIF and PNG use lossless compression.

**Lossless compression:-** is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data.

- In lossless methods, original data and the data after compression and decompression are exactly the same.
- Redundant data is removed in compression and added during decompression. ***example***

**Uncompressed Data=Hello Sandra**

**Compressed Data=HllSndr**

**Decompressed Data=Hello Sandra**

- Lossless methods are used when we can't afford to lose any data: legal and medical documents, computer programs.
- Typical examples are executable programs, text documents, and source code

# Pros and cons of data compression

- ➡ Make optimal use of limited storage space, data transmission time and communication bandwidth, and the resulting cost savings.
- ➡ A compressed file requires less storage capacity than an uncompressed file, and the use of compression can lead to a significant decrease in expenses for disk.
- ➡ A compressed file also requires less time for transfer, and it consumes less network bandwidth than an uncompressed file.

***The main disadvantage of compression is :***

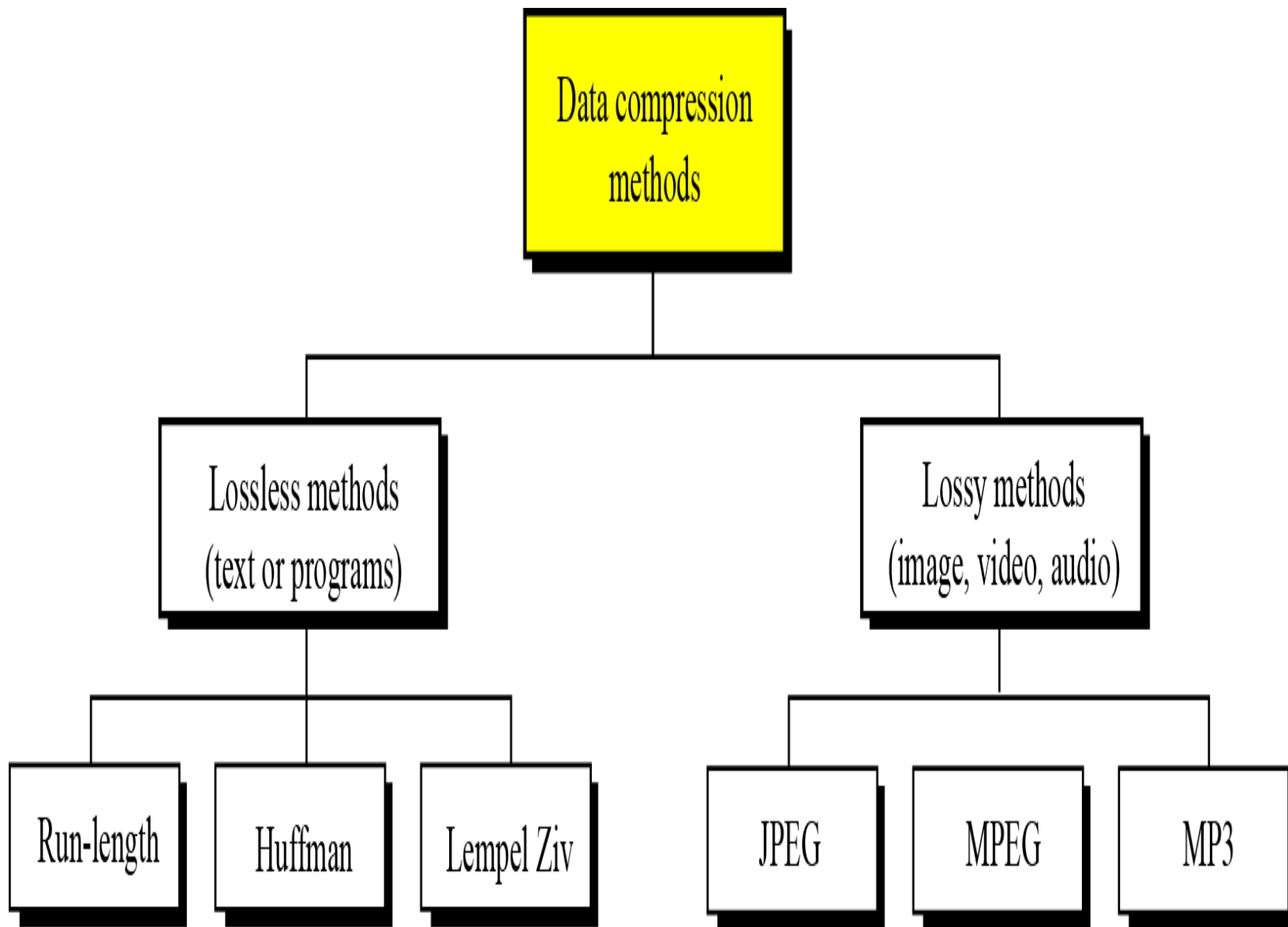
- ➡ the performance impact resulting from the use of CPU and memory resources to compress and decompress the data.

# Data Compression- Entropy

- Entropy is the measure of information content in a message.
  - Messages with higher entropy carry more information than messages with lower entropy.
- How to determine the entropy
  - Find the probability  **$p(x)$**  of symbol  **$x$**  in the message
  - The entropy  **$H(x)$**  of the symbol  $x$  is:
$$H(x) = - p(x) \cdot \log_2 p(x)$$
- The average entropy over the entire message is the sum of the entropy of all  $n$  symbols in the message

# Data Compression Methods

- Data compression is about storing and sending a smaller number of bits.
- There're two major categories for methods to compress data: lossless and lossy methods



# Run-length encoding

- Simplest method of compression.
- How: replace consecutive repeating occurrences of a symbol by 1 occurrence of the symbol itself, then followed by the number of occurrences.

a. Original data

BBBBBBBBBAAAAAAAAAAAAAAAAANMMMMMMMMMM

b. Compressed data

B09A16N01M10

The method can be more efficient if the data uses only 2 symbols (0s and 1s) in bit patterns and 1 symbol is

a. Original data

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0

14

4

0

12

1110 0100 0000 1100

b. Compressed data

# Huffman Coding

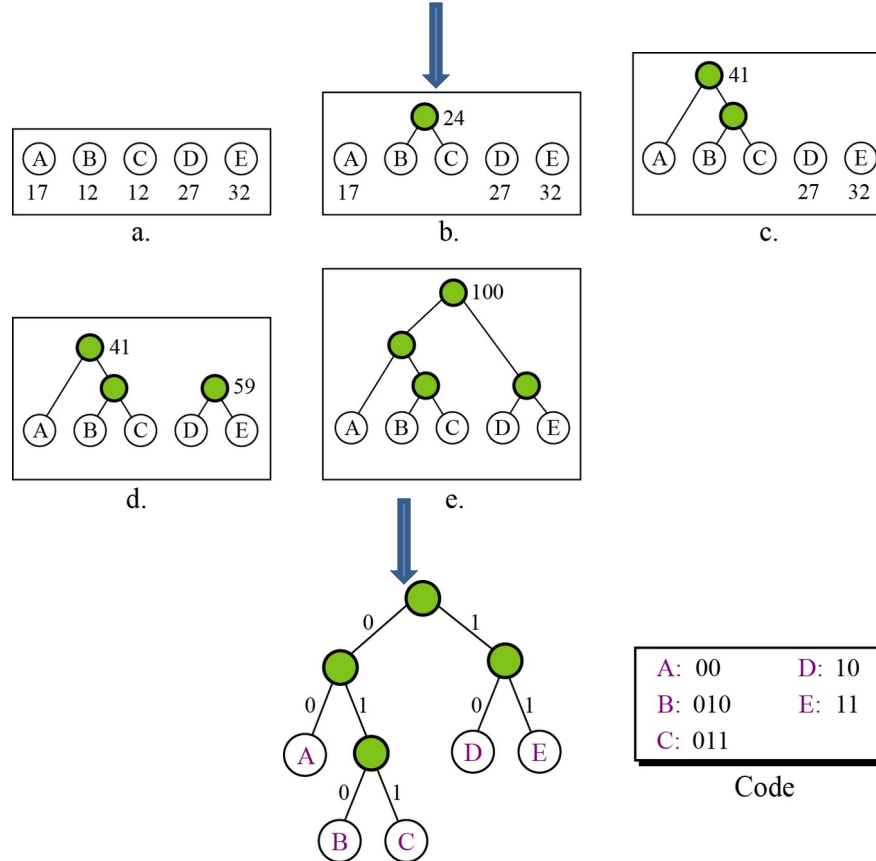
- Assign fewer bits to symbols that occur more frequently and more bits to symbols appear less often.
- There's no unique Huffman code and every Huffman code has the same average code length.
- Algorithm:
  - ① Make a leaf node for each code symbol
    - ◆ Add the generation probability of each symbol to the leaf node
  - ② Take the two leaf nodes with the smallest probability and connect them into a new node
    - ◆ Add 1 to the right or 0 to left node of the two branches
    - ◆ The probability of the new node is the sum of the probabilities of the two connecting nodes
  - ③ If there is only one node left, the code construction is completed. If not, go back to (2)

# Huffman Coding

- Example

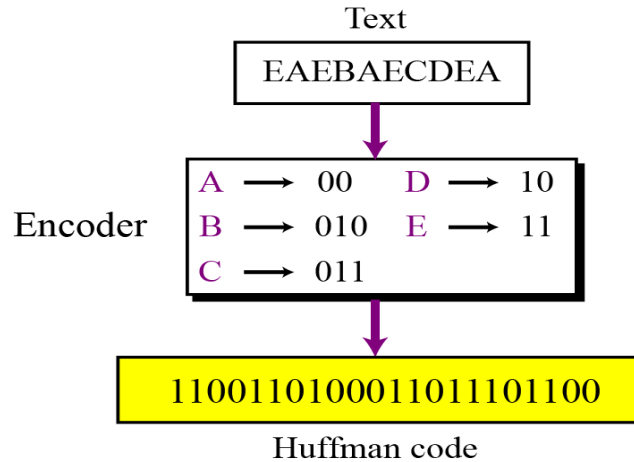
Table 15.1 Frequency of characters

| Character | A  | B  | C  | D  | E  |
|-----------|----|----|----|----|----|
| Frequency | 17 | 12 | 12 | 27 | 32 |

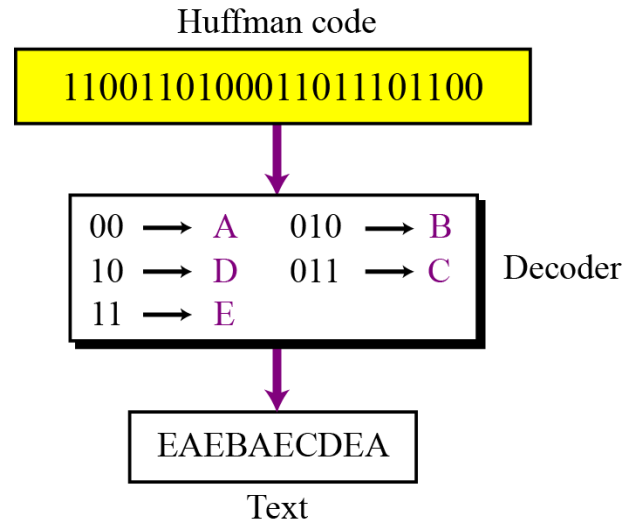


# Huffman Coding

- Encoding



- Decoding



# Lempel Ziv Encoding

➡ It is *dictionary-based* encoding

➡ Basic idea:

- Create a dictionary(a table) of strings used during communication.
- If both sender and receiver have a copy of the dictionary, then previously-encountered strings can be substituted by their index in the dictionary.

# Lempel Ziv Compression

## ➡ Have 2 phases:

- Building an indexed dictionary
- Compressing a string of symbols

## ➡ Algorithm:

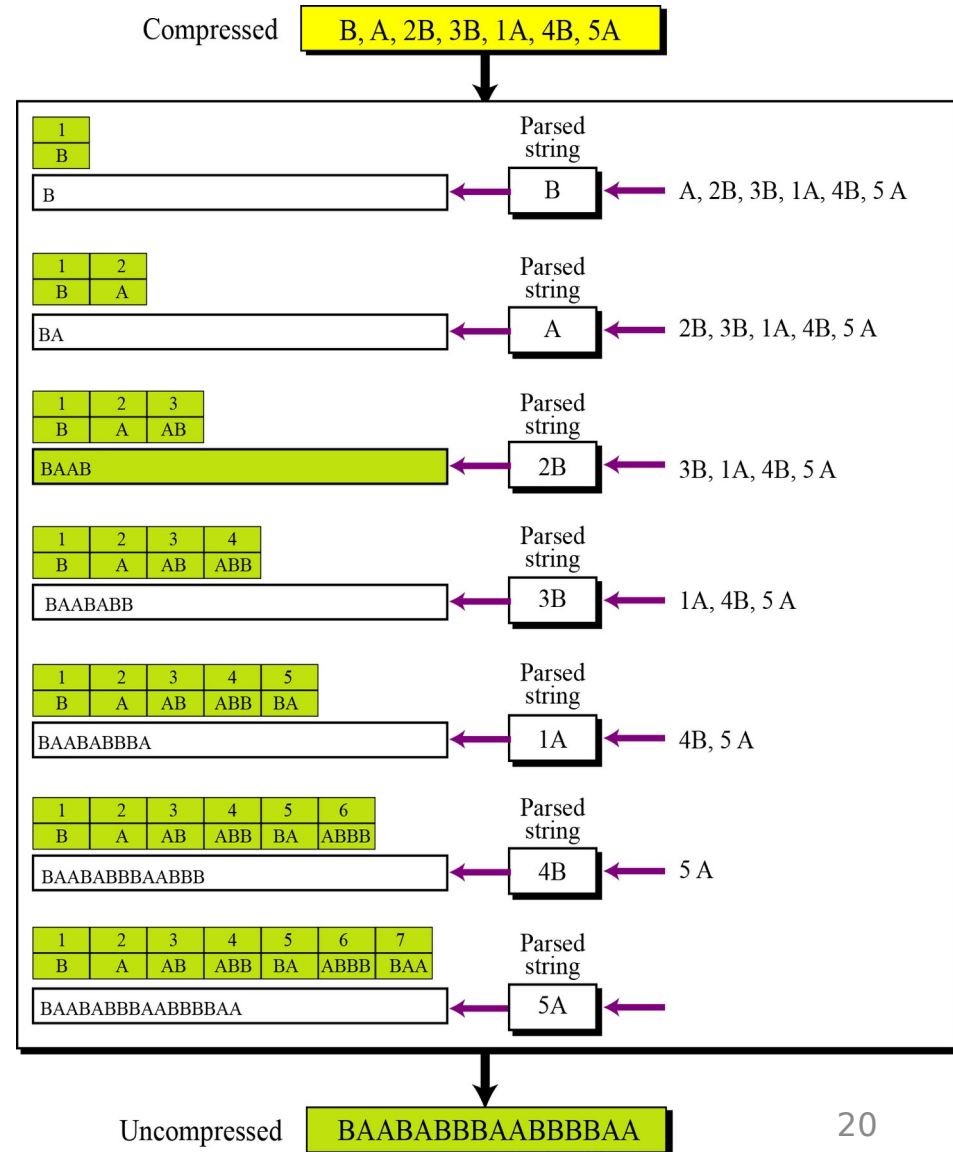
- Extract the smallest substring that cannot be found in the remaining uncompressed string.
- Store that substring in the dictionary as a new entry and assign it an index value
- Substring is replaced with the index found in the dictionary
- Insert the index and the last character of the substring into the compressed string

- Compression example:



# Lempel Ziv Decompression

- It's just the inverse of compression process



# Lossy Compression Methods

- ***lossy data compression*** permits reconstruction only of an approximation of the original data, though this usually improves ***compression rates*** (and therefore reduces file sizes).
- Used for compressing images and video files (our eyes cannot distinguish subtle changes, so lossy data is acceptable).

- These methods are cheaper, less time and space.
- Several methods:
  - JPEG: compress pictures and graphics
  - MPEG: compress video
  - MP3: compress audio

- **MPEG compression:** is achieved using five different compression techniques:
  1. The use of a frequency-based transform called Discrete Cosine Transform (DCT).
  2. Quantization, a technique for losing selective information (sometimes known as lossy compression) that can be acceptably lost from visual information.
  3. Huffman coding, a technique of lossless compression that uses code tables based on statistics about the encoded data.

4. ***Motion compensated predictive coding***, in which the differences in what has changed between an image and its preceding image are calculated and only the differences are encoded.
5. **Bi-directional prediction**, in which some images are predicted from the pictures immediately preceding and following the image.
- The first three techniques are also used in **JPEG** file compression.

# Audio Compression

- Used for speech or music
  - Speech: compress a 64 kHz digitized signal
  - Music: compress a 1.411 MHz signal
- Two categories of techniques:
  - Predictive encoding
  - Perceptual encoding

# Audio Encoding

- Predictive Encoding
  - Only the differences between samples are encoded, not the whole sample values.
- Perceptual Encoding: **MP3**
  - CD-quality audio needs at least 1.411 Mbps and cannot be sent over the Internet without compression.
  - MP3 (MPEG audio layer 3) uses perceptual encoding technique to compress audio.