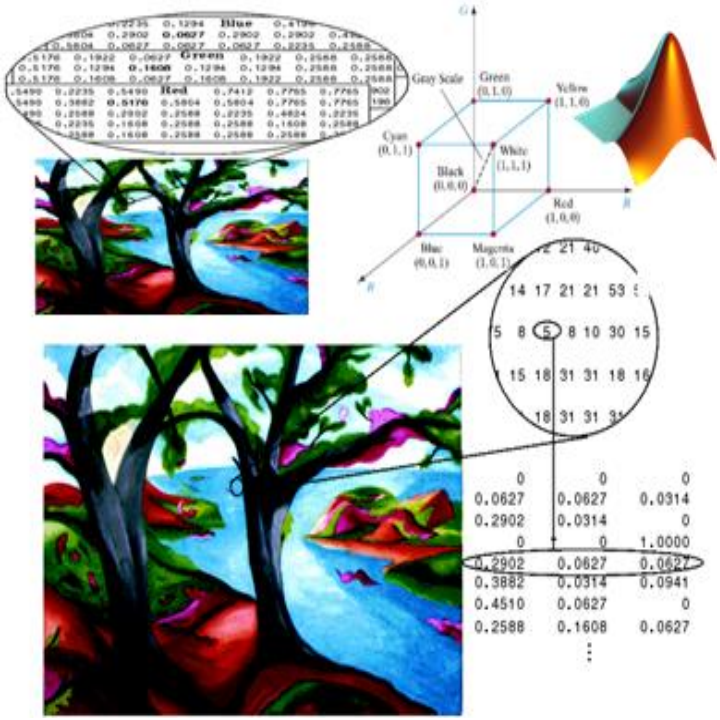
Advanced Image Processing
Part 6



Lecturer: Dr. Taban Fouad Majeed

E-mail : taban.majeed@su.edu.krd

Master Students / Computer Science

2023-2024

Image Compression

- The field of image compression continues to grow at a rapid pace
- As we look to the future, the need to store and transmit images will only continue to increase faster than the available capability to process all the data
- Applications that require image compression are many and varied such as:

1. Internet,
2. Businesses,
3. Multimedia,
4. Satellite imaging,
5. Medical imaging

Image Compression (Continued)

- *Image compression* involves reducing the size of image data files, while *retaining necessary information*
- Retaining necessary information depends upon the application
- Image segmentation methods, which are primarily a data reduction process, can be used for compression
- The reduced file created by the compression process is called the *compressed file* and is used to reconstruct the image, resulting in the *decompressed image*
- The original image, before any compression is performed, is called the *uncompressed image file*

Image Compression Ratio

- The ratio of the original, uncompressed image file and the compressed file is referred to as the *compression ratio*

$$Compression\ Ratio = \frac{Uncompressed\ file\ size}{Compressed\ file\ size} = \frac{SIZE_U}{SIZE_C}\ ;\ Often\ written\ as \rightarrow SIZE_U : SIZE_C$$

EXAMPLE 10.1.1: The original image is 256x256 pixels, single-band (grayscale), 8-bits per pixel. This file is 65,536 bytes (64k). After compression the image file is 6,554 bytes. The compression ratio is: $SIZE_U/SIZE_C = 65536/6554 = 9.999 \approx 10$. This can also be written as 10:1

This is called a "10 to 1 compression", a "10 times compression", or can be stated as "compressing the image to 1/10 its original size". Another way to state the compression is to use the terminology of *bits per pixel*. For an NxN image:

$$Bits\ per\ pixel = \frac{Number\ of\ bits}{Number\ of\ pixels} = \frac{(8)(Number\ of\ bytes)}{NxN}$$

Image Compression Ratio

EXAMPLE 10.1.2: Using the preceding example, with a compression ratio of 65,536/6,554 bytes, we want to express this as bits per pixel. This is done by first finding the number of pixels in the image: 256x256 = 65,536 pixels. We then find the number of bits in the compressed image file: (6,554 bytes)(8 bits/byte) = 52,432 bits. Now we can find the bits per pixel by taking the ratio: 52,432/65,536 = 0.8 bits/pixel

- The reduction in file size is necessary to meet the bandwidth requirements for many transmission systems, and for the storage requirements in computer databases

EXAMPLE 10.1.3:

To transmit an RGB (color) 512x512, 24-bit (8-bit per pixel per color) image via modem at 56 kbaud (kilo-bits per second), it would take about:

$$\frac{(512 \times 512 \text{ pixels})(24 \text{ bits/pixel})}{(56 \times 1024 \text{ bits/second})} \approx 109 \text{ seconds} \approx 1.8 \text{ minutes}$$

- This number is based on the actual transmission rate being the maximum, which is typically not the case due to Internet traffic, overhead bits and transmission errors
- Additionally, considering that a web page might contain more than one of these images, the time it takes is simply too long
- For high quality images the required resolution can be much higher than the previous example

EXAMPLE 10.1.4:

To transmit a digitized color 35mm slide scanned at 4000x3000 pixels, and 24-bits, at 56 kbaud would take about:

$$\frac{(4000 \times 3000 pixels)(24 bits/pixel)}{(56 \times 1024 bits/\text{sec})} \approx 5022 \text{sec} \approx 83 \text{min}$$ **, too long to wait!**

EXAMPLE 10.1.5:

To transmit a digitized color 35mm slide scanned at 4000x3000 pixels, and 24-bits, at 3 Mbps would take about:

$$\frac{(4000 \text{ x } 3000 \text{ pixels})(24 \text{ bits/pixel})}{(3 \text{ x } 1024 \text{ x } 1024 \text{ bits/second})} \approx 91 \text{ seconds} \approx 1.5 \text{ minutes}$$

Video images transmission time

- Now, consider the transmission of video images, where we need multiple frames per second
- If we consider just one second of video data that has been digitized at 640x480 pixels per frame, and requiring 15 frames per second for interlaced video, then:

EXAMPLE 10.1.6: To transmit one second of interlaced video that has been digitized at 640x480 pixels:

$$\frac{(640 \times 480 \times 15\, frames/\mathbf{sec})(24 bits/pixel)}{3 \times 1024 \times 1024 bits/\mathbf{sec})} \approx 35\, \mathbf{sec}\, onds$$

Video images compression

- Waiting 35 seconds for one second's worth of video is not exactly real time!
- Even attempting to transmit uncompressed video over the highest speed Internet connection is impractical
- Applications requiring high speed connections such as high definition television, real-time teleconferencing, and transmission of multiband high resolution satellite images, leads us to the conclusion that *image compression is not only desirable but necessessary*

Data Redundancy

- Data that provide no relevant information=*redundant data* or *redundancy*.
- Image coding or compression has a goal to reduce the amount of data by reducing the amount of redundancy.
- The compression is achieved by taking advantage of the redundancy that exists in images

Compression

- Data compression implies sending or storing a smaller number of bits. Although many methods are used for this purpose, in general these methods can be divided into two broad categories: lossless and lossy methods.

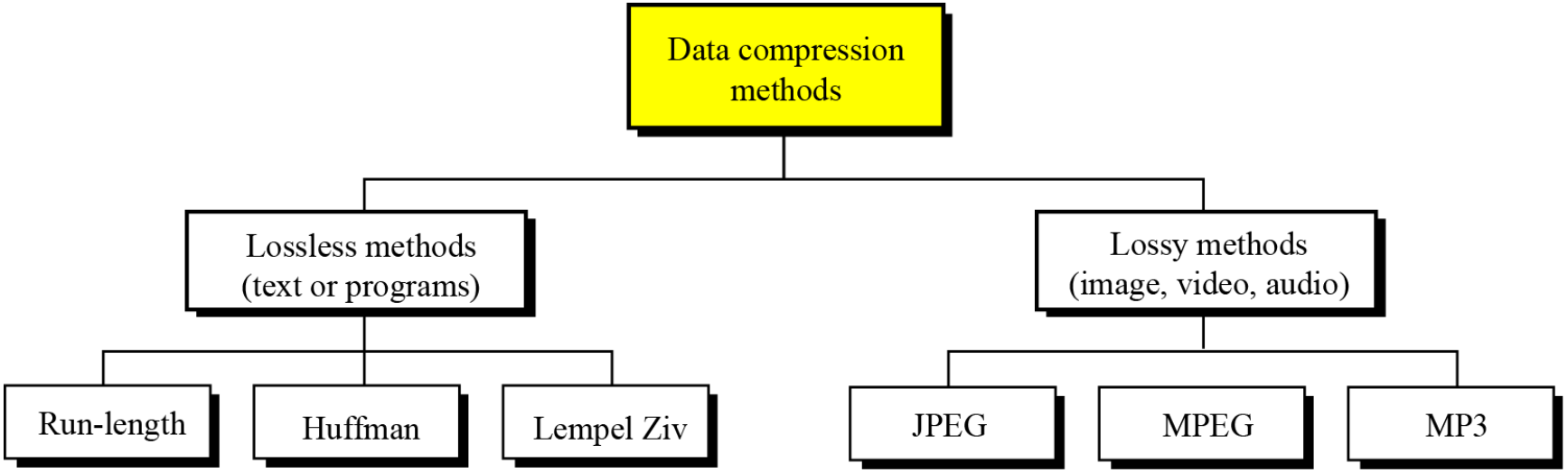


Image Compression Methods

- Image data compression methods fall into two common categories:

I. Information preserving compression (*Lossless compression)*

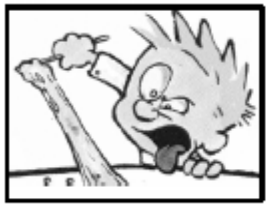
- Allows for the exact recreation of the original image data, and can compress complex images to a maximum 1/2 to 1/3 the original size – 2:1 to 3:1 compression ratios
- Preserves the data *exactly (*Compress and decompress images without losing information
- Redundant data is removed in compression and added during decompression.

Image Compression Methods

II. Lossy image compression

- Data loss, original image cannot be re-created exactly
- Can compress complex images 10:1 to 50:1 and retain high quality, and 100 to 200 times for lower quality, but acceptable, images
- Provide higher levels of data reduction
- Result in a less than perfect reproduction of the original image
- Applications: –*broadcast television, videoconferencing*

Image Compression Methods



compression

102900910-23155
70291-787418029
809578759187582
745187598475198
751878758457109
-585079057509O5

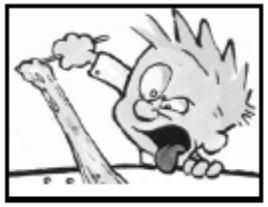
original image

"compact information"
(for storage or transmission)

(loss less compression)

decompression

(lossy compression)



approximation of the original image

Run-length encoding (RLE)

- It is probably the simplest method of compression.
- It can be used to compress data made of any combination of symbols.
- It does not need to know the frequency of occurrence of symbols.
- The general idea behind this method is to replace consecutive repeating occurrences of a symbol by one occurrence of the symbol followed by the number of occurrences.

Example 1

- Given the following 4*4 4_bit image find the RLC of the image:

$$\begin{bmatrix} 10 & 10 & 10 & 10 \\ 10 & 12 & 12 & 12 \\ 5 & 5 & 5 & 10 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- The corresponding gray and length (G, L) pair of each row of the image are:
- First row: (10, 4)
- Second row: (10, 1), (12, 3)
- Third row: (5, 3), (10, 1)
- Fourth row: (0, 4)
- These numbers are stored or transmitted in RLC compressed form as: 10, 4, 10, 1, 12, 3, 5, 3, 10, 1, 0, 4
- The compression ratio CR= (4-bit × 4 × 4 /4-bit × 12) = 1.1428

Example 2

a. Original data BBBBBBBBBAAAAAAAAAAAAAAAANMMMMMMMMMM

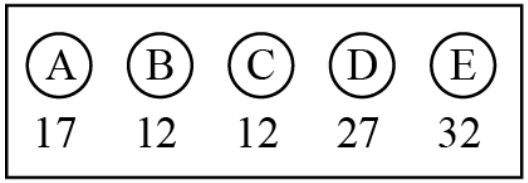
b. Compressed data B09A16N01M10

Huffman coding

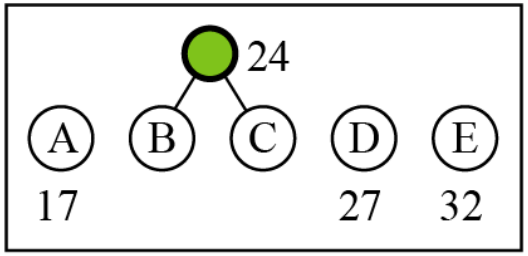
- The algorithm assigns shorter codes to symbols that occur more frequently and longer codes to those that occur less frequently.
- For example, imagine we have a text file that uses only five characters (A, B, C, D, E). Before we can assign bit patterns to each character, we assign each character a weight based on its frequency of use.
- In this example, assume that the frequency of the characters is as shown:

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

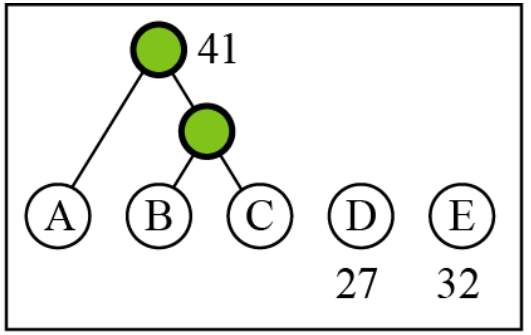
Huffman coding



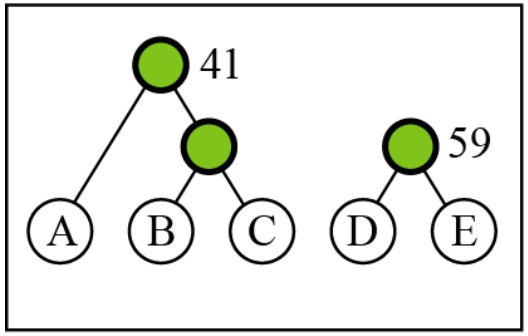
a.



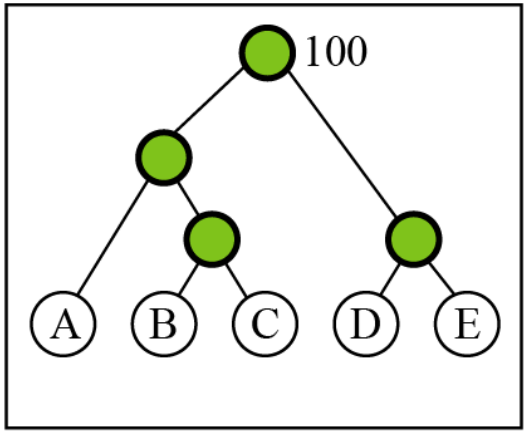
b.



c.



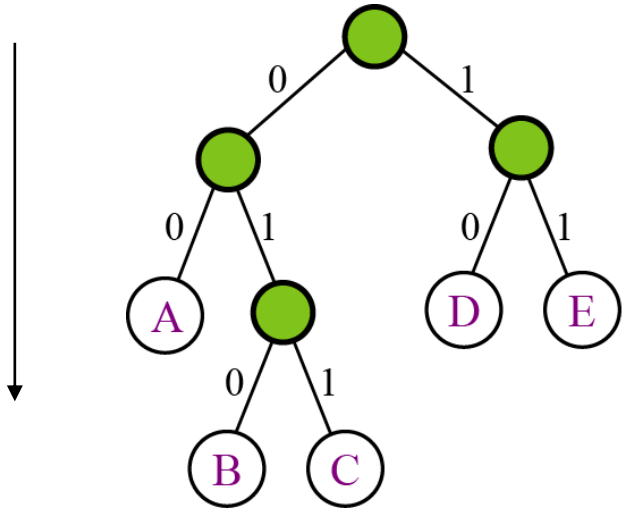
d.



e.

Huffman coding

- A character's code is found by starting at the root and following the branches that lead to that character. The code itself is the bit value of each branch on the path, taken in sequence.

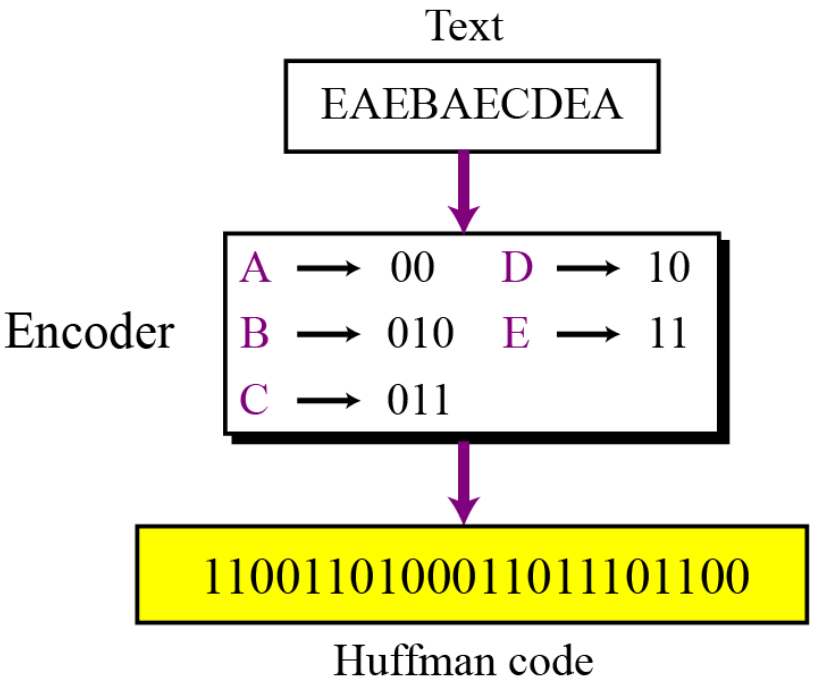


A: 00	D: 10
B: 010	E: 11
C: 011	

Code

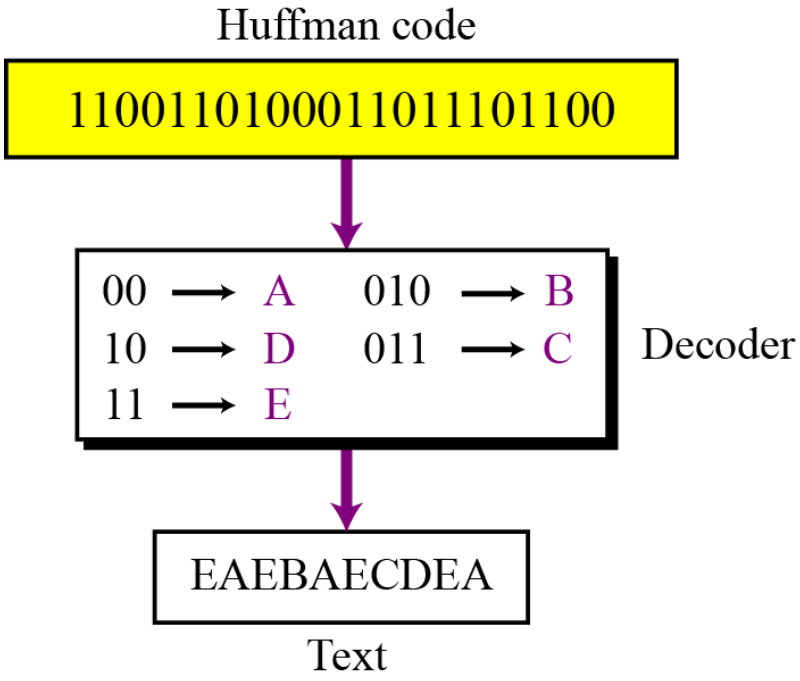
Encoding

- Let us see how to encode text using the code for our five characters.



Decoding

- The recipient has a very easy job in decoding the data it receives.



References for these slides

Digital Image Processing, 4th edition by Rafael C. Gonzalez and Richard E. Woods.

Chapters 8