Contents

1 Data Compression
2 Lossless Compression Methods
3 Lossy Compression Methods

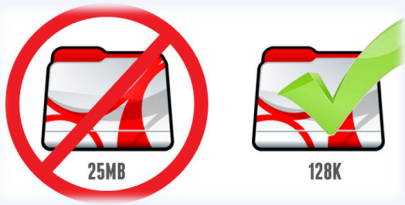
7

Data Compression

8

File Compression

- Reasons for file compression:
  - Less storage
  - Transmitting faster, decreasing access time
  - Processing faster sequentially



9

9

Why Data Compression?

- Make optimal use of limited storage space
- Save time and help to optimize resources
  - If compression and decompression are done in I/O processor, less time is required to move data to or from storage subsystem, freeing I/O bus for other work
  - In sending data over communication line: less time to transmit and less storage to host

10

10

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:

$$H(x) = - p(x) \bullet \log_2 p(x)$$

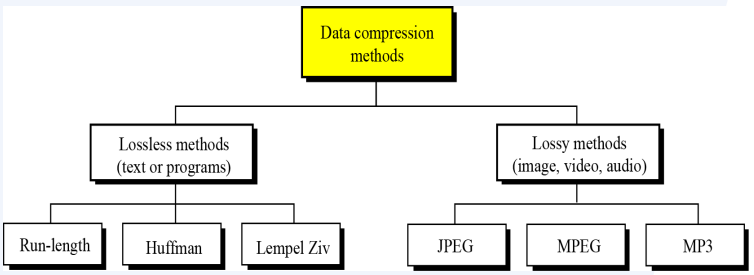
- The average entropy over the entire message is the sum of the entropy of all n symbols in the message

11

11

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



12

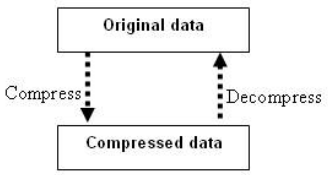
12

Lossless Compression Methods

13

Lossless Compression Methods

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



14

Run-length Encoding

15

Run-length Encoding

- Simplest method of compression.
- Replace consecutive repeating occurrences of a symbol by one occurrence of the symbol itself, then followed by the number of occurrences.

a. Original data: BBBBBBBBBAAAAAAAAAAAAAAANMMMMMMMMMM

b. Compressed data: B09A16N01M10

16

Run-length Encoding

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 255 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 255 \end{bmatrix}$$

17

17

Run-length Encoding

- **Run-length encoding algorithm:**
  - Read through pixels, copying pixel values to file in sequence, except the same pixel value occurs more than once in succession
  - When the same value occurs more than once in succession, substitute the following three bytes:
    - special run-length code indicator(e.g. 0xFF)
    - pixel value repeated
    - the number of times that value is repeated

- **Example:**

  22 23 24 24 24 24 24 24 24 25 26 26 26 26 26 26 25 24

  RL-coded stream: 22 23 ff 24 07 25 ff 26 06 25 24

18

18

Run-length Encoding

- **Run-length encoding algorithm:**
  - It is an example of redundancy reduction
  - **Drawbacks:**
    - not guarantee any particular amount of space savings
    - under some circumstances, compressed image is larger than original image
    - Why? Can you prevent this?

19

19

Huffman Coding

20

20

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.

21

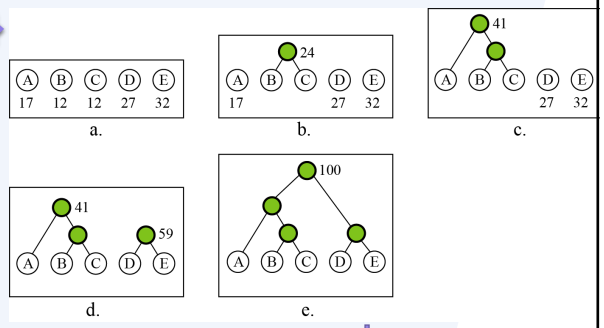
Huffman Coding

- **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 or 0 to each 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)

22

Huffman Coding

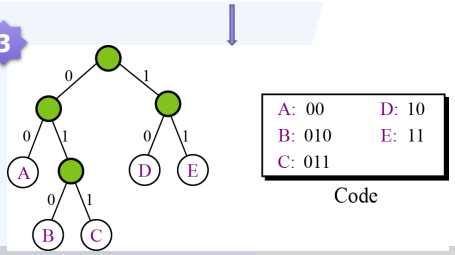
- Example:



1

Table 15.1 Frequency of characters

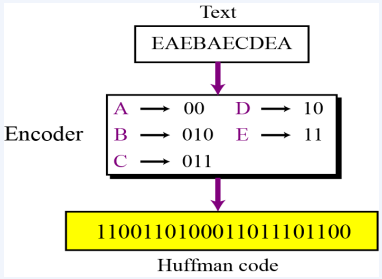
Character	A	B	C	D	E
Frequency	17	12	12	27	32



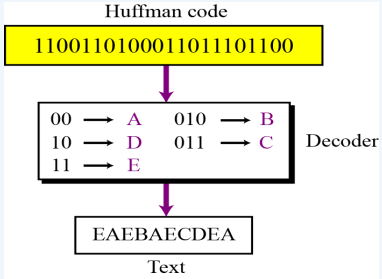
23

Huffman Coding

- Encoding



- Decoding



24

Huffman Coding

- Example:

File :

b	p	`	m	j	o	d	a	i	r	u	l	s	e	
1	1	2	2	3	3	3	4	4	5	5	6	6	8	12

25

25

Lempel-Ziv Codes

26

26

Lempel-Ziv Codes

- There are several variations of Lempel-Ziv Codes.
- We will look at LZ78
- Commands zip and unzip and Unix compress and uncompress use Lempel-Ziv codes

27

Lempel-Ziv Codes

- Let us look at an example for an alphabet having only two letters:

  a a a b a b b b a a a b a a a a a a a b a a b b

- Rule
  - Separate this stream of characters into pieces of text so that each piece is the shortest string of characters that we have not seen yet.

a | a a | b | a b | b b | a a a | b a| a a a a | a a b | a a b b

28

Lempel-Ziv Codes

a a a b a b b b a a a b a a a a a a a b a a b b

a | a a | b | a b | b b | a a a | b a| a a a a | a a b | a a b b

1. We see “a”
2. “a” has been seen, we now see “aa”
3. We see “b”
4. “a” has been seen, we now see “ab”
5. “b” has been seen, we now see “bb”
6. “aa” has been seen, we now see “aaa”
7. “b” has been seen, we now see “ba”
8. “aaa” has been seen, we now see “aaaa”
9. “aa” has been seen, we now see “aab”
10. “aab” has been seen, we now see “aabb”

29

29

Lempel-Ziv Codes

- **Index:**
  - We have index values from 1 to n
  - For the previous example

1 2 3 4 5 6 7 8 9 10

a | a a | b | a b | b b | a a a | b a| a a a a | a a b | a a b b

30

30

Lempel-Ziv Codes

- **Encoding:**
  - Since each piece is the concatenation of a piece already seen with a new character, the message can be encoded by a previous index plus a new character.

1 2 3 4 5 6 7 8 9 10
a | a a | b | a b | b b | a a a | b a| a a a a | a a b | a a b b

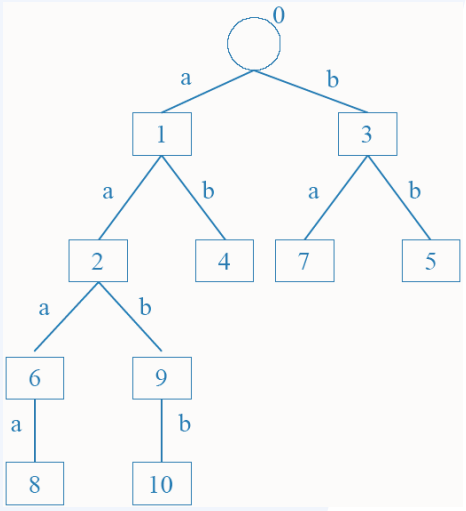
1 2 3 4 5 6 7 8 9 10
0 a | 1 a | 0 b | 1 b | 3 b | 2 a | 3 a | 6 a | 2 b | 9 b

31

31

Lempel-Ziv Codes

- **Encoding tree:**
  - A tree can be built when encoding



1 2 3 4 5 6 7 8 9 10
a | a a | b | a b | b b | a a a | b a| a a a a | a a b | a a b b

32

Lempel-Ziv Codes

- **Exercise No. 1:**
  - Encode the file containing the following characters, drawing the corresponding digital tree.

“ a a a b b c b c d d d e a b ”

Lempel-Ziv Codes

- **Exercise No. 1:**

1 2 3 4 5 6 7 8

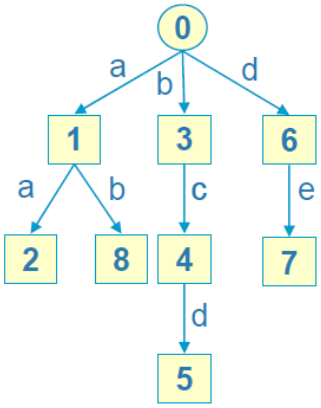
a | a a | b | b c | b c d | d | d e | a b

0 a | 1 a | 0 b | 3 c | 4 d | 0 d | 6 e | 1 b

Lempel-Ziv Codes

- **Exercise No. 1:**

1	2	3	4	5	6	7	8
a	a a	b	b c	b c d	d	d e	a b
0 a	1 a	0 b	2 c	4 d	0 d	6 e	1 b



35

35

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.

36

36

Lempel Ziv Compression

- **Have 2 phases:**
  1. Building an indexed dictionary
  2. Compressing a string of symbols

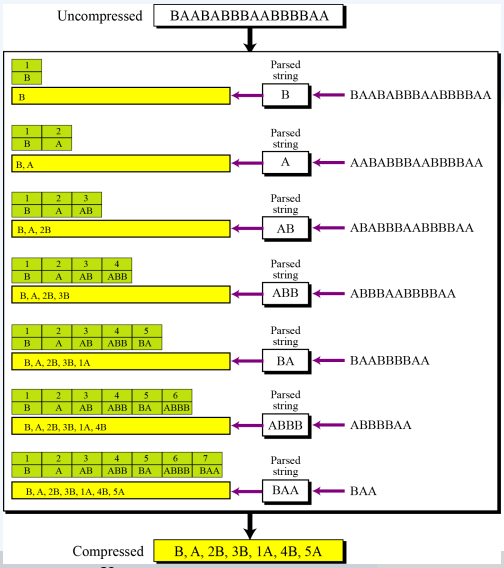
- **Algorithm:**
  1. Extract the smallest substring that cannot be found in the remaining uncompressed string.
  2. Store that substring in the dictionary as a new entry and assign it an index value
  3. Substring is replaced with the index found in the dictionary
  4. Insert the index and the last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

37

37

Lempel Ziv Compression

- Compression example:

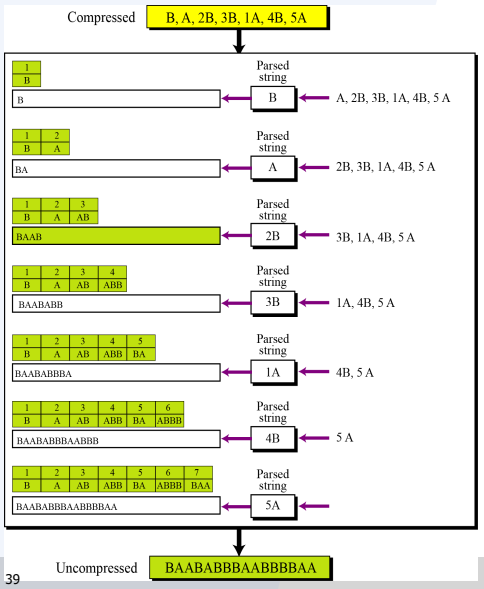


38

38

Lempel Ziv Decompression

- It's just the inverse of compression process



39

39

40

Lossy Compression Methods

- 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

41

41

Lossy Compression Methods

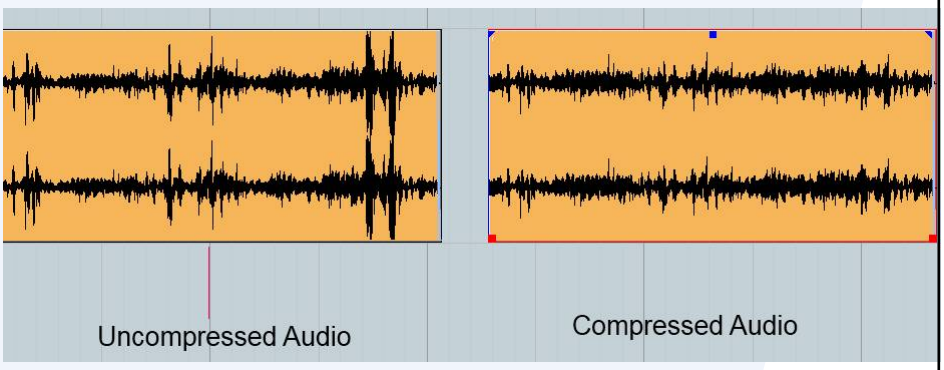
Compressed by 90%- 5.93 KB

Uncompressed- 57.5 KB

42

42

Lossy Compression Methods



43

43

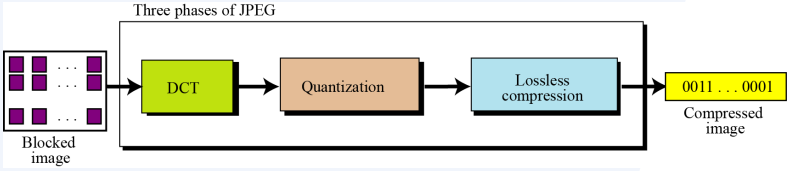
JPEG Encoding

44

44

JPEG Encoding

- Used to compress pictures and graphics.
- In JPEG, a grayscale picture is divided into 8x8 pixel blocks to decrease the number of calculations.
- **Basic idea:**
  1. Change the picture into a linear (vector) sets of numbers that reveals the redundancies.
  2. The redundancies is then removed by one of lossless compression methods.



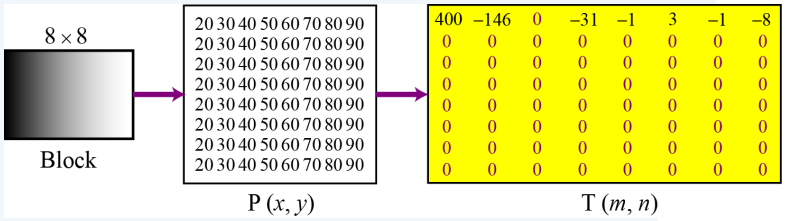
45

45

JPEG Encoding- DCT

- **DCT**: Discrete Concise Transform
- DCT transforms the 64 values in 8x8 pixel block in a way that the relative relationships between pixels are kept but the redundancies are revealed.
- Example:

A gradient grayscale



46

46

Quantization & Compression

- **Quantization:**
  - After T table is created, the values are quantized to reduce the number of bits needed for encoding.
  - Quantization divides the number of bits by a constant, then drops the fraction. This is done to optimize the number of bits and the number of 0s for each particular application.

47

47

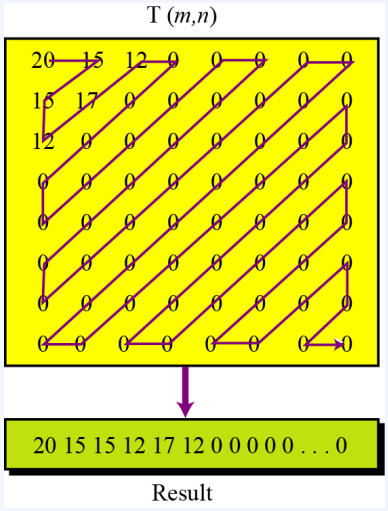
Quantization & Compression

- **Compression:**
  - Quantized values are read from the table and redundant 0s are removed.
  - To cluster the 0s together, the table is read diagonally in an zigzag fashion. The reason is if the table doesn't have fine changes, the bottom right corner of the table is all 0s.
  - JPEG usually uses lossless run-length encoding at the compression phase.

48

48

JPEG Encoding



49

49

MPEG Encoding

50

50

MPEG Encoding

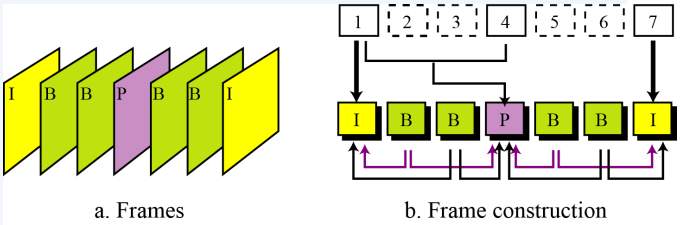
- Used to compress video.
- **Basic idea:**
  - Each video is a rapid sequence of a set of frames.
  - Each frame is a spatial combination of pixels, or a picture.
  - Compressing video =
    spatially compressing each frame
    +
    temporally compressing a set of frames.

51

51

MPEG Encoding

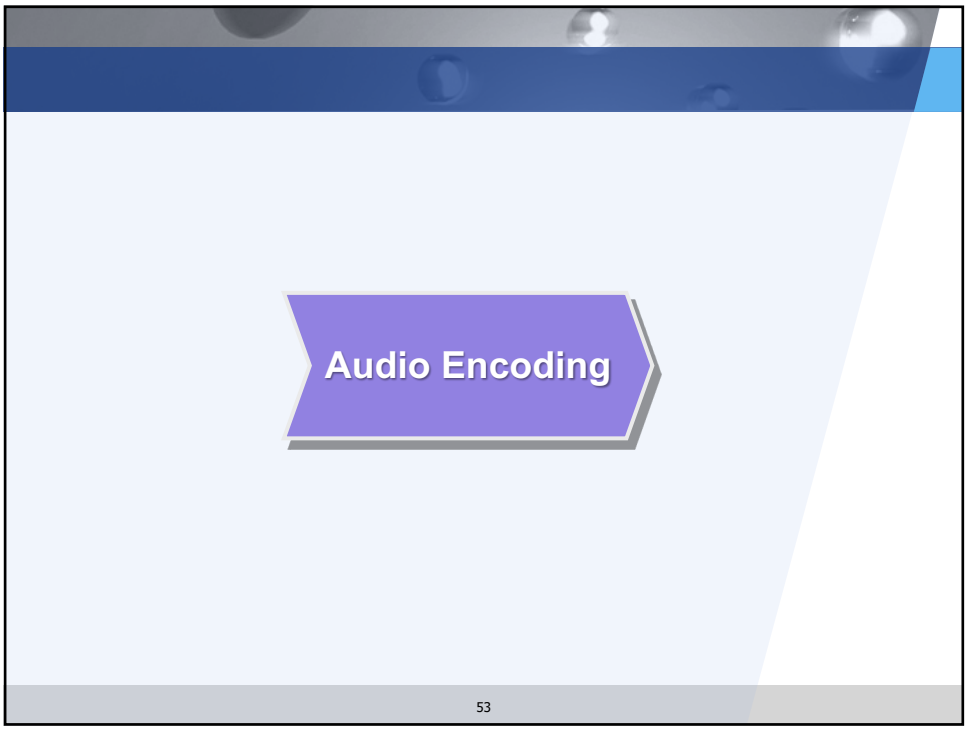
- **Spatial Compression**
  - Each frame is spatially compressed by JPEG.
- **Temporal Compression**
  - Redundant frames are removed.
  - For example, in a static scene in which someone is talking, most frames are the same except for the segment around the speaker's lips, which changes from one frame to the next.



a. Frames b. Frame construction

52

52



53

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

54

54

Audio Encoding

- **Predictive Encoding**
  - Only the differences between samples are encoded, not the whole sample values.
  - Several standards: GSM (13 kbps), G.729 (8 kbps), and G.723.3 (6.4 or 5.3 kbps)

- **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.