Introduction to Multimedia Technology (MM301)

By Dr. Heba Hamdy and Dr. Ahmed Anter

Assistant Professor, Multimedia Dep.

CODING AND COMPRESSION

- 5 m
- 24 frame x 300 x 1280 x 720 x 24= 24 x 300 x 921600x24=159252480000 bit
- 18.5 GB

Data Compression

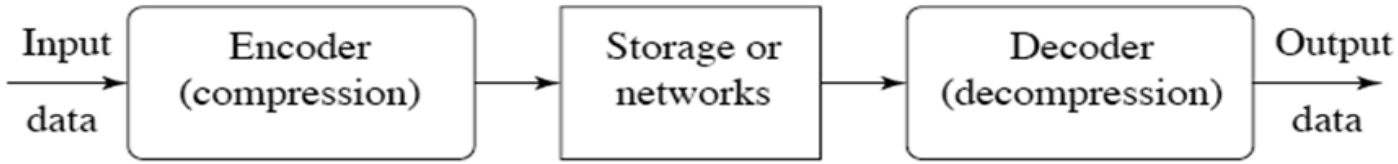
- A key problem with multimedia is the huge quantities of data that result from raw digitized data of audio, image or video source, data compression is usually desirable.
- The main goal for coding and compression is to improve the storage, processing and transmission costs for these data.
- A compression Algorithm is considered to be Universal If it can compress data efficiantly despite possesing no prior stasistical knlowladge of the process that generated the data.

Compression

- **Compression:**

  the process of coding that will effectively reduce the total number of bits needed to represent certain information.

- **Benefits**
  - Reduce storage needed
  - Reduce transmission cost / latency / bandwidth



A General Data Compression Scheme.

Data Compression Terms

The Compressor (Encoder):

the program that compress the input raw data into an output compressed stream with low redundancy.

The De-compressor (Decoder):

the program that convert in opposite direction

Stream: This term may used instead of file because the compressed stream may be transmitted directly to the decoder.

Compression principles

- **There are many known methods for data compression based on different ideas to be suitable for different types of data. But, all are based on the same principle .**

- **This principle :**

namely compressing data by removing the Redundancy from the original data.

- **The idea of compression by reducing Redundancy suggests the general law of data compression by "assign short codes to common events (event may be symbols, phrase, digital audio samples, image, and video pixels) and long code to rare events"**

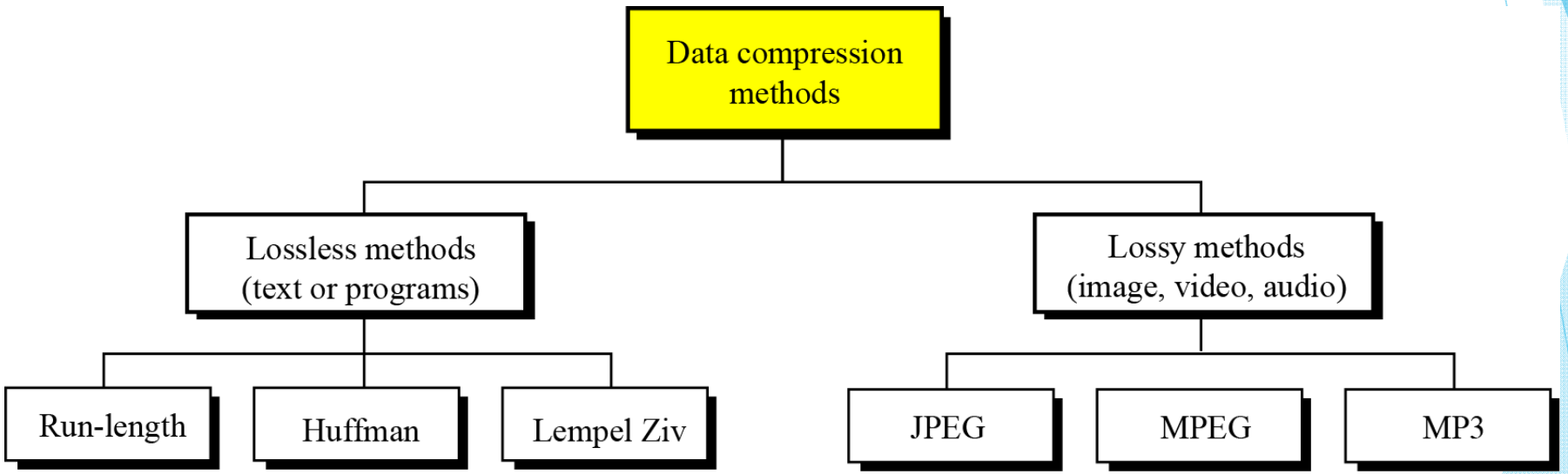
Lossless vs. Lossy

- Lossless compression of digitized data such as video, digitized film, and audio **preserves all the information**, but it does not generally achieve compression ratio much better than 2:1 because of the key entropy of the data. Compression algorithms which provide higher ratios either incur very large overheads or work only for specific data sequences (e.g. compressing a file with mostly zeros).
- In contrast, lossy compression (e.g. JPEG for images, or MP3 for audio) can achieve much higher compression ratios at the cost of a decrease in quality, such as Bluetooth audio streaming, as visual or audio compression artifacts from loss of important information.

Types of Compression Reduction

- Lossless
  - recovery of the original non compressed data.
  - Exploits redundancy in data
  - Applied to general data
  - Information in a spreadsheet or text document would be useless if any information was removed.
  - Zip files are one of the many examples that compression doesn't removes data in the original file.
  - Lossless formats RAW, MOV, GIF
  - Compression doesn't sacrifice quality
  - Allow complete
- Lossy
  - Exploits redundancy.
  - Removes some data that's not noticeable to the human eye or ear.
  - Lower quality to decrease file size.
  - Applied to audio, image, video
  - Some lossy formats are JPEG, mp4, and mp3.
  - does not allow recovery but is designed to be perceived as sufficient by the user.

Data Compression Methods



Effectiveness of Compression

- In ASCII codes, every character takes 8 bits.
- To compress data, the most common approach is to reduce the number of bits that represent the most used characters.
- Compression ratio:

$$compression\ ratio = \frac{B_0}{B_1}$$

B_0 – number of bits before compression
B_1 – number of bits after compression

Effectiveness of Compression

- **A good metric for compression is the *compression factor* given by:**

$$\text{Compression factor} = 1 - \left[\frac{\text{compressed size}}{\text{uncompressed size}}\right] \times 100\%$$

- **If we have a 100KB file that we compress to 40KB, we have a compression factor of:**

$$1 - \left[\frac{40\text{KB}}{100\text{KB}}\right] \times 100\% = 60\%$$

Keyword Encoding

Replace frequently used words with a single character

Word	Symbol
as	^
the	~
and	+
that	$
must	&
well	%
these	#

Instantaneous Variable-Length Codes

Instantaneous Variable-Length Codes

variable-length code

- a code which maps source symbols to a variable number of bits
- can allow sources to be compressed and decompressed with zero error (lossless data compression) and still be read back symbol by symbol.
- Some examples of well-known variable-length coding strategies are Huffman coding, Lempel-Ziv coding and arithmetic coding.

Instantaneous Variable-Length Codes

Example

Code word	Symbol
0	A
10	B
110	C
111	D

The source is one of four symbols: a, b, c or d. The binary codeword for each symbol is given. The encoding and decoding of a portion of an example source sequence is given below:

aabacda → 001001101110 → **0|0|10|0|110|111|0**

Instantaneous Variable-Length Codes

Samuel Morse principle

In the 1800s Samuel F. B. Morse made an efficient telegraph transmission method by assigning short code strings to frequently occurring letters and longer code strings to letters that occur less often.

When applying Morse's method , we simply arrange the source letters in order of decreasing probability and let p_j be shorthand for $P(a_j)$

- for example lets assume we have the source letters ($a_1, a_2, \ldots a_m$) where $p_1 = 0.4$, $p_2 = p_3 = 0.15$, $p_4 = p_5 = 0.1$, $p_6 = 0.05$, $p_7 = 0.04$, $p_8 = 0.01$
- Notice that these letter probabilities have to satisfy the requirement $(p_1 + p_2 + \ldots + p_8) = 1$

Entropy

Information Theory

- **Information theory:** an entropy encoding is a lossless data compression scheme.
- **Entropy encoders:** compress data by replacing each fixed-length input symbol with the corresponding variable-length prefix-Code-free output code-word.
- **The length of each code-word is approximately** proportional to **negative logarithm probability**
- So, the length for a symbol is : $(-\log_b P) = 1 / \log_b P$ where:

b: the number of symbols used to make output codes and

P: is the probability of the input symbol.

- The most common symbols use the shortest codes.

First-order Information

- Suppose we are given symbols {a1, a2, ... , am}.
- P(ai) = probability of symbol ai occurring in the absence of any other information.
- P(a1) + P(a2) + ... + P(am) = 1
- inf (ai) = $\log_2$ (1/P(ai)) is the information of ai in bits.
- Example
- {a, b, c} with P(a) = 1/8, P(b) = 1/4, P(c) = 5/8
- – inf(a) = $\log_2(8) = 3$
- – inf(b) = $\log_2(4) = 2$
- – inf(c) = $\log_2(8/5) = .678$

Receiving an "a" has more information than "b" or "c".

First Order Entropy

The average amount of information **needed** to represent symbol is called *Entropy*

Messages with **higher entropy** carry more information than messages with lower entropy.

How to determine the entropy

The first order entropy is defined for a probability distribution over symbols *{a1, a2, … , am}*.

$$H=\sum_1^m p(ai)\ \log_2(1/p(ai))$$

***H*: the average number of bits required to code a symbol.**

- **For all symbols we know, H is the probability distribution of the symbols.**
- ***H* is the Shannon lower bound on the average number of bits to code a symbol in this "source model".**

Entropy Examples

{a, b, c} with p(a)=1/8 , p(b)=1/4 , p(c)=5/8

- inf (ai) = $\log_2 (1/P(ai))$

- Inf (a)=$\log_2 (8) = 3$
- Inf ()=$\log_2 (4) = 2$
- Inf (c)=$\log_2 (8/5) = .678$

$H=\sum_1^m p(ai) \log_2 (1/p(ai))$

H= 1/8 * 3 + ¼* 2 + 5/8 * 0.678 = 1.3 bits/symbol

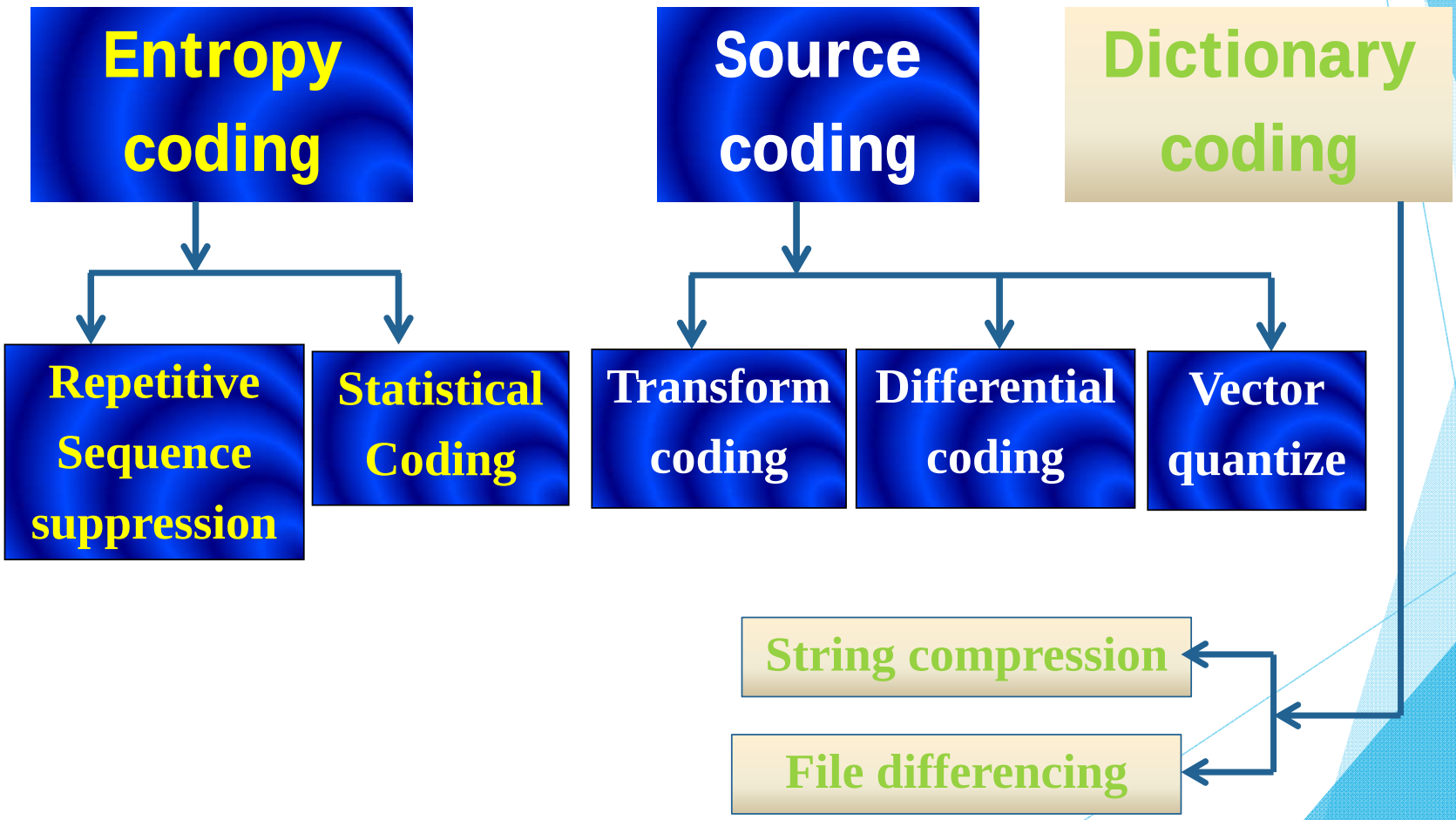
Example:

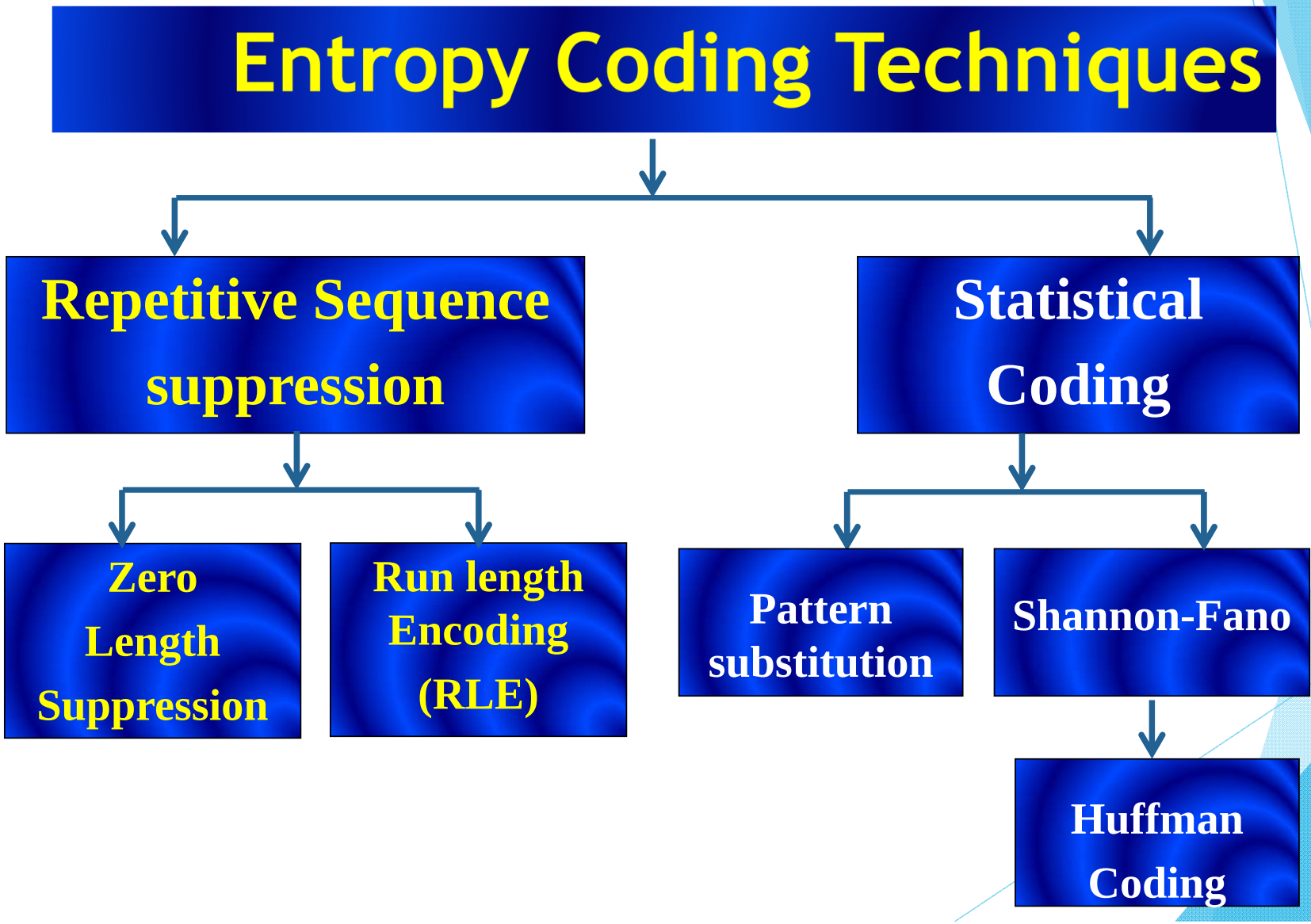
{a, b, c} with p(a)=1/3 , p(b)=1/3 , p(c)=1/3

H= 3 * 1/3 * $\log_2 (3)$ = 1.6 bits/symbol (worst case)

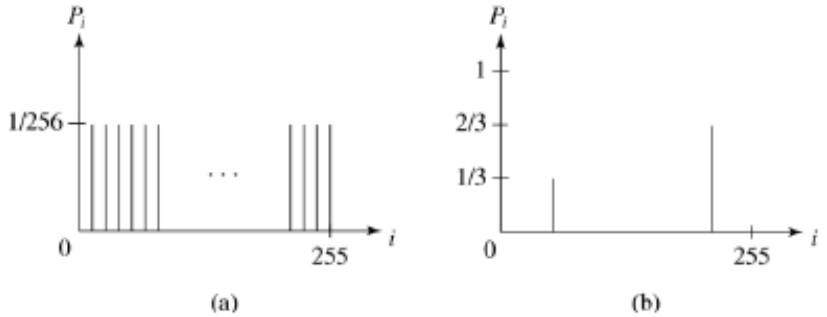
a=00 b=01 c=11

Coding Techniques




Entropy Coding Techniques
Repetitive Sequence suppression
Statistical Coding
Zero Length Suppression
Run length Encoding (RLE)
Pattern substitution
Shannon-Fano
Huffman Coding

Distribution of Gray-Level Intensities



- Histograms for Two Gray-level Images.

- the histogram of an image with uniform distribution of gray -level intensities, i.e., ∀i pi = 1/256. Hence, the entropy of this image is:

Image with 256 gray-level with uniform distribution (equal probability).

Prob. Of each gray-level=1/256

Entropy=$\left[\sum_{i=1}^{256} 1/256 X (\log 1/1/256)\right]$ =256 (1/256 x $\log_2 2^8$)=8 bit

8 bit image Example

- Estimate the information content (entro py) of the following image (8 bit image):

21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243
21	21	21	95	169	243	243	243

Gray □level	*Count*	Pr*obability*
21	12	3/8
95	4	1/8
169	4	1/8
243	12	3/8

- First-order estimate of the entropy (bit s/pixel)

$$\square 3/8 * \log 2(3/8) \square 1/8 * \log 2(1/8) \square 1/8 * \log 2(1/8) \square 3/8 * \log 2(3/8) = 1.81$$

Gray □level □ pair	*Count*	Pr*obability*
(21,21)	8	2/7
(21,95)	4	1/7
(95,169)	4	1/7
(169,243)	4	1/7
(243,243)	8	2/7

Second-order estimate of the entropy (b its every two pixels)

$$-2/7 * \log 2(2/7) * 2 - 1/7 * \log 2(1/7) * 3 = 2.2$$

Entropy: Measure the expected average number of bits to code symbols

Ex. Estimate the Entropy for the following message:

HELLO

Ex. Estimate the entropy of the following image colors where probability for each one is

P(White)=0.7
P(Brown)=0.25
P(Black)=0.025
P(Blue)=0.025

Run Length Encoding

Run-length encoding (**RLE**) is a form of lossless data compression in which *runs* of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.

This is most useful on data that contains many such runs. Consider, for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size.

Run Length encoding

•This Compression technique represents data using value and run length.

•Run length defined as number of consecutive equal values

•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

Run-Length Coding example

WWWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW

12W1B12W3B24W1B14W

Uncompressed

aaaaabbbbbbbbbbbbccccddddddddddeeeeeeeeeee

Compressed

5a12b4c9d10e

Uncompressed

aabccdeefghijjjklmnopqrrstttuvvwwxyyyz

Compressed *Negative compression*

2a1b2c1d2e1f1g1i3j1k1l1m1n1o1p1q2r1s3t1u2v2w1x3y1z

Negative compression if there is variant of data

Run Length Encoding

15 5

7 5 4 11 4 5

4 5 2 11 1 1 1 15 4 7 1 11 2 5

3 5 2 7 1 11 2 1 1 11 4 7 1 15 1 5

2 5 3 7 1 11 2 1 1 15 4 7 1 11 1 5

1 5 1 11 4 7 3 1 4 7 1 11 1 5

1 5 1 11 4 7 1 15 2 1 1 11 3 7 1 15 1 5

1 5 1 11 4 7 1 11 2 1 1 11 2 7 1 11 2 5

2 5 5 7 2 1 1 11 1 7 1 1 3 5

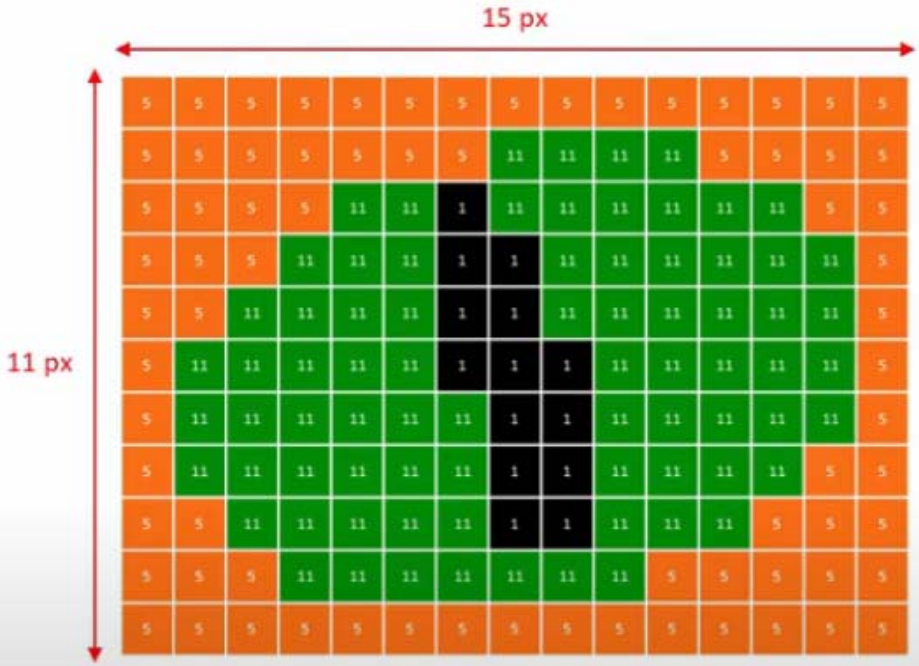
3 5 1 11 3 7 1 11 1 15 1 11 5 5

15 5

Without compression 15 * 11 * 4 = 660 bits

With compression 134 * 4 = 536 bits

Run Length Encoding



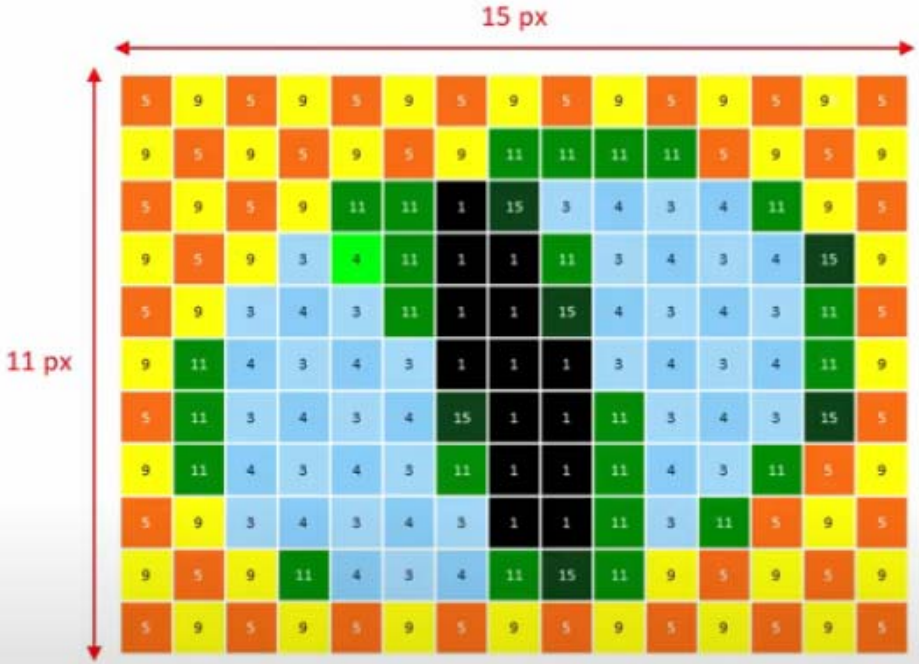
15 5
7 5 4 11 4 5
4 5 2 11 1 1 6 11 2 5
3 5 3 11 2 1 6 11 1 5
2 5 4 11 2 1 6 11 1 5
1 5 5 11 3 1 5 11 1 5
1 5 6 11 2 1 5 11 1 5
1 5 6 11 2 1 4 11 2 5
2 5 5 11 2 1 3 11 3 5
3 5 7 11 5 5
15 5

Without compression 15 * 11 * 4 = 660 bits

With compression 86 * 4 = 344 bits

Run Length Encoding

Negative Compression



1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5
1 9 1 5 1 9 1 5 1 9 1 5 1 9 4 11 1 5 1 9 1 5 1 9
1 5 1 9 1 5 1 9 2 11 1 1 1 15 1 3 1 4 1 3 1 4 1 11 2 5 1 9 1 5
1 9 1 5 1 9 2 13 1 11 2 1 1 11 1 3 1 4 1 3 1 4 1 15 1 9
1 5 1 9 1 3 1 4 1 3 1 4 1 11 2 1 1 15 1 4 1 3 1 4 1 3 1 11 1 5
1 9 1 11 1 4 1 3 1 4 1 3 3 1 1 3 1 4 1 3 1 4 1 11 1 9
1 5 1 11 1 3 1 4 1 3 1 4 1 15 2 1 1 11 1 3 1 4 1 3 1 4 1 15 1 5
1 9 1 11 1 4 1 3 1 4 1 3 1 11 2 1 1 11 1 4 1 3 1 11 1 5 1 9
1 9 1 5 1 3 1 4 1 3 1 4 1 3 2 1 1 11 13 1 11 1 5 1 9 1
1 9 1 5 1 9 1 11 1 4 1 3 1 4 1 11 1 15 1 11 1 9 1 5 1 9 1 5 1 9
1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5 1 9 1 5

Without compression 15 * 11 * 4 = 660 bits

With compression 311 * 4 = 1244 bits

Exercise

Compress the following runs using RLE and calculate the Compression Ratio.

10,10,10,10,17,10,10,10,20,10,10,10,10,35,10,10,10,40,10,10,10,10,10, 20, 10

Given run length 2bit

Exercise

Compress the following text using RLC and calculate the Compression Ratio.

aaaaaaaa bbbbb cccc dd

Given run length 2bit

This information represent in file header

And symbol is 16 bit

Exercise

Use RLC to compress this string and calculate the Compression Ratio.

00000 11111 00 1 0000 1 0 1

Summary of RLE

- RLE is lossless compression
- Works best when data contain long runs of the same value
- Easy to implement
- Fast execution
- Complex data can result in negative compression

Thanks