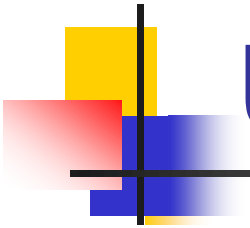
Source Coding & Data Compression

- Useful Compression Terms
- Compression Performance
- Statistical Compression Methods (RLE)
- Dictionary Compression Methods (LZ77)



Useful Compression Terms

- Adaptive / Non-adaptive Compression
- Lossy/Lossless Compression
- Perceptive Compression
- Symmetric/ Asymmetric Compression
- Universal Compression

Compression Performance

$$\text{Compression ratio} = \frac{\text{size of the output stream}}{\text{size of the input stream}}.$$

$$\text{Compression factor} = \frac{\text{size of the input stream}}{\text{size of the output stream}}.$$

For Images and Moving Pictures (film):

- Bit per pixels (bpp)
- Peak Signal to Noise Power Ratio or PSNR (in dB)

Statistical Compression/ Run Length Encoding

Run Length Encoding (RLE) make use of the runs in the source output to perform a source coding method in which only the length of character run is sent rather than the actual repeated characters. This is usually need a special character to be declared to indicate the presence of run. This special character is known as escape character. The following represent the development of RLE which are introduced first for FAX transmission and later developed for standard image and video compression systems.

1- Simple RLE (# is the escape character)

. . . . A A B B B C C C C A A A A A A C C C C B A A A . . .

#A2, #B3, #C4, #A6, #C4, B, #A3

2- Simple RLE (Binary source for FAX)

11111000000011100001111100000001111111. . . . . .

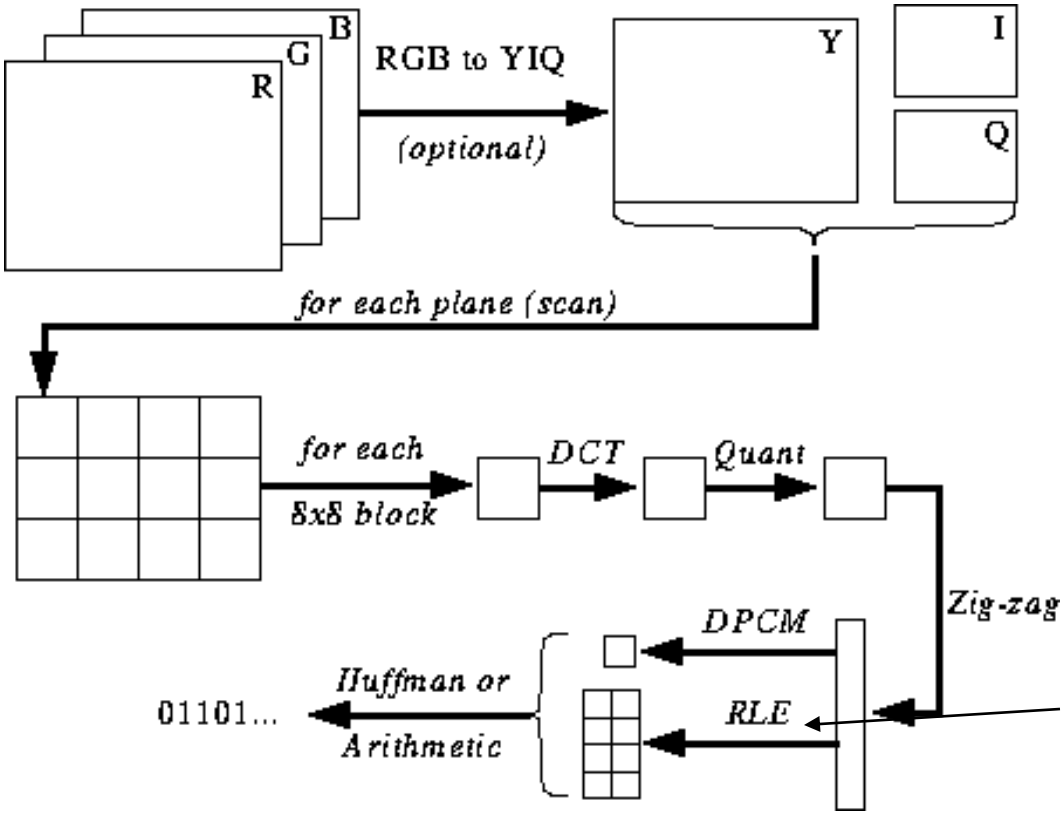
v1 (no scape char.) 15, 07, 13, 04, 15, 07, 17

v2(alternate values 1,0,1,0..) 5, 7, 3, 4, 5, 7, 7

V3(differential or relative) 5, 2, -4, 1, 1, 2, 0, ...

Q- Find the compression factor of above examples.

RLE Image Compression – JPEG



Run Length Encoder!

Dictionary Compression Methods

1- LZ77 or Sliding Window;

In general, an LZ77 token has three parts: offset, length, and next symbol in the look-ahead buffer (which, in our case, is the **second** e of the word teases). This token is written on the output stream, and the window is shifted to the right (or, alternatively, the input stream is moved to the left) four positions: three positions for the matched string and one position for the next symbol.

...sir␣sid␣eastman␣easily␣tease|s␣sea␣sick␣seals...|...

If the backward search yields no match, an LZ77 token with zero offset and length and with the unmatched symbol is written. This is also the reason a token has to have a third component. Tokens with zero offset and length are common at the beginning of any compression job, when the search buffer is empty or almost empty. The first five steps in encoding our example are the following:

	sir␣sid␣eastman␣	⇒	(0,0,"s")
s	ir␣sid␣eastman␣e	⇒	(0,0,"i")
si	r␣sid␣eastman␣ea	⇒	(0,0,"r")
sir	␣sid␣eastman␣eas	⇒	(0,0,"␣")
sir␣	sid␣eastman␣easi	⇒	(4,2,"d")

The next step matches the space and encodes the string "␣e".

sir␣sid	␣eastman␣easily␣	⇒	(4,1,"e")
sir␣sid␣e	astman␣easily␣te	⇒	(0,0,"a")

and the next one matches nothing and encodes the "a".

Example for LZ77:

Use LZ77 to compress the following string:

sir^sid^eastman^easily^teases^sea^sick^seals……

Search Buffer	Look Ahead Buffer	Token		
		D	M	Next
	sir^sid^eastman^easily^teases^sea^sick^seal…	0	0	s
s	ir^sid^eastman^easily^teases^sea^sick^seal…	0	0	i
si	r^sid^eastman^easily^teases^sea^sick^seal…	0	0	r
sir	^sid^eastman^easily^teases^sea^sick^seals	0	0	^
sir^	sid^eastman^easily^teases^sea^sick^seals……	4	2	d
sir^sid	^eastman^easily^teases^sea^sick^seals……	4	1	e
sir^sid^e	astman^easily^teases^sea^sick^seals……	0	0	a
sir^sid^ea	stman^easily^teases^sea^sick^seals……	6	1	t
sir^sid^east	man^easily^teases^sea^sick^seals……	0	0	m
sir^sid^eastm	an^easily^teases^sea^sick^seals……	4	1	n
sir^sid^eastman	^easily^teases^sea^sick^seals……	8	4	i
sir^sid^eastman^easi	ly^teases^sea^sick^seals……	0	0	l
sir^sid^eastman^easil	y^teases^sea^sick^seals……	…..etc		