Dictionary methods

- encode „phrases", not single symbols into code words
- output words / symbols are equipotable

Ex: Encoding English text

'␣the␣' → 1
'␣a␣' → 2
'␣is␣' → 3
⋮

- based on „transformation table" → DICTIONARY
- table is dynamic → coding improves with the number of symbols processed (usually!)

Historical remark: RLE (Run-Length Encoding)

PCX (i.i.r.c. max 16 colors)

Ex: binary B&W bitmap

0... black
1... white

111100000010111
4,1,6,0,1,1,1,0,1,4,1

× can be handled if binary

8-bit packed

3bits => 8 colours
5bits => max. 32 repetitions

can be implemented as a dictionary, BUT it is computationally ineffect.

Sliding window

- given by its width in 'elements' (source symbols)
- moves through the data set

Ex: sliding window width 8

abbadbabaabbaadbba

position 0
position 10

LZ77 (Lempel-Ziv)

- use a sliding window to search for repetitions of a phrase; if found, the current phrase is replaced with a reference

(n, d, c)

- c — next character
- d — length of phrase
- n — the number of chars to go backwards

Kdyby␣byly␣ryby␣nebyly␣by␣chyby.

K	d	y	b	y␣	byl	y␣r	yby␣n
(0,0,K)	(0,0,d)	(0,0,y)	(0,0,b)	(2,1,␣)	(3,2,l)	(5,2,r)	(10,4,n)

(0,0,K) — no previous reference found

e	byly␣b	y␣c	h	yby.
(0,0,e)	(12,5,b)	(3,2,c)	(0,0,h)	(16,3,.)

32 characters → 13 triplets

× 8bit → 256 bits
× 7bit → 224 bits
(uncompressed, but "optimized" sequence) → ASCII

(5bit, 4bit, 7bit) → 16 bits/triplet

- 4bit: max. length of phrase 15 char
- 5bit: length of the window is 32 chars (i.e. the max. backwards step is 32)

13 triplets → 208 bits if 16 bits/triplet

LZW (Lempel-Ziv-Welch)

- search in a long sliding window is slow

→ use a tree-based dictionary

a) dictionary is adaptive
(can be discarded after compression and re-created when decompressing)

b) pre-defined basis

Kdyby␣byli␣by␣bylyby␣chyby.

K	d	y	b	y␣	by	l	yu	r	yb	yu	n	e	byl	yu	by
0	1	2	3	2	4	15	5	16	6	14	16	7	8	18	16	15

+ + + +

Kd dy yb by ....

this goes to dictionary

␣	c	h	yby	.	EOF
4	9	10	22	11	

32 characters → 256 bits
224 bits

22 symbols à 6 bits = 132 bits

Dictionary:

k→0	kd→12	byly→26
d→1	dy→13	yub→27
y→2	yb→14	by␣→28
b→3	by→15	␣c→29
␣→4	yu→16	ch→30
l→5	ub→17	by→31
r→6	byl→18	yby.→32
n→7	ly→19	
e→8	yur→20	
c→9	ry→21	
h→10	yby→22	
.→11	yun→23	
(basis)	ne→24	
	eb→25	