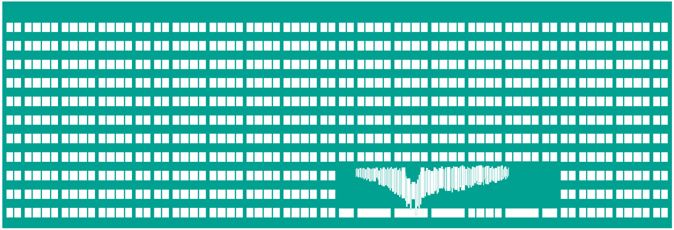
VŠB TECHNICKÁ
UNIVERZITA
OSTRAVA

VSB TECHNICAL
UNIVERSITY
OF OSTRAVA



www.vsb.cz

Dictionary Compression

LZ algorithms

Michal Vasinek

VSB – Technical University of Ostrava

name.surname@vsb.cz

May 23, 2019

VSB TECHNICAL UNIVERSITY OF OSTRAVA

Overview

- Dictionary coding
- LZ77
- LZSS
- LZ78
- LZW

Dictionary Coding

Principle

Encode the next phrase as a pointer to an already processed part of a message.

$$processed \mid unprocessed$$

`abacdddd|abaceeee`

- Different methods differ in a construction of processed part and encoding of pointer to a processed part.
- Processed part becomes a dictionary.

- Input string: $s = abbabbabbbaab$.

Processed	Unprocessed	Pointer	Match Length
	abbabbabbbaab	0	0
a	bbabbabbbaab	0	0
ab	babbabbbaab	1	1
abb	abbabbbaab	3	6
abbabbabb	baab	4	2
abbabbabbbba	ab	6	2

LZ77

- Each step of LZ77 algorithm forms a triplet (Pointer, Match Length, Next symbol) → (P, L, N).

Processed	Unprocessed	P	L	N
	abbabbabbbaab	0	0	a
a	bbabbabbbaab	0	0	b
ab	babbabbbaab	1	1	a
abba	bbabbbaab	3	5	b
abbabbabbb	aab	4	1	a
abbabbabbbbaa	b	0	0	b

Encoded as: (0,0,a), (0,0,b), (1,1,a),(3,5,b),(4,1,a),(0,0,b)

LZ77

- Decoding triplets: (0,0,a), (0,0,b), (1,1,a),(3,5,b),(4,1,a),(0,0,b)

Processed	Triplet
	(0,0,a)
a	(0,0,b)
ab	(1,1,a)
abba	(3,5,b)
abbabbabba	(4,1,a)
abbabbabbbaa	(0,0,b)
abbabbabbbaab	

LZ77 - Implementation details

- No need to store double zeros: (0,0,x) $\rightarrow$ (0,x).
- Typical Pointer size 10-12 bits: 1024, 4096 last symbols stored in the processed part
- Match length up to 32 symbols, i.e. 5 bits.
- The next symbol is usually encoded by 8 bits.
- Totally: 25 bits per triplet.
- Further improved by statistical coding of fields P, L and N separately.
- Processed part implemented using circular buffer.

LZ77 - remarks

- Described in 1977 by Lempel and Ziv.
- Processed and unprocessed part together called sliding window.
- Approaches k-order entropy.
- Key method in PKZIP v1.

- Modification of LZ77 by Szymanski and Storer published in 1982.
- Encodes a phrase by (Flag, Pointer, Length) or (Flag, Next).

Processed	Unprocessed	Output
	abbabbabbbaab	(0,a)
a	bbabbabbbaab	(0,b)
ab	babbabbbaab	(1,1,1)
abb	abbabbbaab	(1,3,6)
abbabbabb	baab	(1,4,2)
abbabbabbbba	ab	(1,6,2)

LZSS - remarks

- Unprocessed part stored in circular queue.
- Processed part stored in binary search tree - more efficient localization of matches.
- LZSS followed by Huffman coding used in Deflate(PKZIP v2), GZIP, RAR.
- Deflate - main algorithm in HTTP compression.

- Published in 1978 by Lempel and Ziv.
- Builds a dictionary of observed phrases and outputs tupple (Pointer, Next Symbol).

Phrase ID	Dictionary	Unprocessed	Token
0	null	abbabbabbbaab	
1	a	bbabbabbbaab	(0,a)
2	b	babbabbbaab	(0,b)
3	ba	bbabbbaab	(2,a)
4	bb	abbbaab	(2,b)
5	ab	bbaab	(1,b)
6	bba	ab	(4,a)
7			(1,b)

Encoded as: (0,a), (0,b), (2,a), (2,b), (1,b), (4,a), (1,b).

LZ78

Decoding: (0,a), (0,b), (2,a), (2,b), (1,b), (4,a), (1,b).

Token	Output	Phrase ID	Dictionary
		0	null
(0,a)	a	1	a
(0,b)	b	2	b
(2,a)	ba	3	ba
(2,b)	bb	4	bb
(1,b)	ab	5	ab
(4,a)	bba	6	bba
(1,b)	ab		

- The size of dictionary is either fixed or uses all available memory.
- As dictionary grows it may fill all memory. Possible solutions:
  - Freeze the dictionary (no new entries will be added) and use it as a static dictionary.
  - Delete the entire dictionary and start building from scretch.
  - Delete some of the most recently added entries. No good heuristics known.
- Dictionary stored in LZ78 dictionary tree:

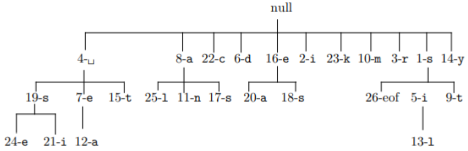


Figure: Handbook of Data Compression, Salomon, p. 356

LZW

- Variant of LZ78 published in 1984 by Terry Welsch.
- Eliminates the symbol field in the token.
- It starts with initialization of the dictionary with all symbols of the alphabet (usually all 8 bit values).
- The principle of LZW is that the encoder inputs symbols one by one and accumulates them in a string I.
- After each symbol is input and is concatenated to I, the dictionary is searched for string I.
- As long as I is found in the dictionary, the process continues. At a certain point, adding the next symbol x causes the search to fail; string I is in the dictionary but string Ix (symbol x concatenated to I) is not.
- At this point the encoder (1) outputs the dictionary pointer that points to string I, (2) saves string Ix (which is now called a phrase) in the next available dictionary entry, and (3) initializes string I to symbol x.

LZW

Let s = abbabb

I	in Dict?	Dict ID	New entry	Output
		0	a	
		1	b	
a	Y			
ab	N	2	ab	0
b	Y			
bb	N	3	bb	1
b	Y			
ba	N	4	ba	1
a	Y			
ab	Y			
abb	N	5	abb	2
b	Y			1

- Decoder should reinitialize dictionary with input alphabets symbols.
- In the first decoding step, the decoder inputs the first pointer and uses it to retrieve a dictionary item I. This is a string of symbols, and it is written on the decoder's output. String Ix needs to be saved in the dictionary, but symbol x is still unknown; it will be the first symbol in the next string retrieved from the dictionary.
- In each decoding step after the first, the decoder inputs the next pointer, retrieves the next string J from the dictionary, writes it on the output, isolates its first symbol x, and saves string Ix in the next available dictionary entry (after checking to make sure string Ix is not already in the dictionary). The decoder then moves J to I and is ready for the next step.

LZW - decoding

Decoding $s = 0, 1, 1, 2, 1$, given the alphabet is $\Sigma = \{a, b\}$.

Pointer	I	J	in Dict?	Dict ID	New entry	Output
				0	a	
				1	b	
0		a	Y			a
1	ab	b	N	2	ab	b
1	bb	b	N	3	bb	b
2	ba	ab	N	4	ba	ab
1	abb	b	N	5	abb	b

LZW - remarks

- Unix - compress utility.
- GIF image format.
- Optionally used in TIFF and PDF files. Adobe Acrobat prefers DEFLATE for text.

Additional Reading

- LZ77 and LZ78 optimality proof - Elements of Information Theory - pp 440-456
- Technical discussion of LZ algorithm family - Handbook of Data Compression, Salomon, pp 329-375.

Thank you for your attention

Michal Vasinek

VSB – Technical University of Ostrava

name.surname@vsb.cz

May 23, 2019

VSB TECHNICAL UNIVERSITY OF OSTRAVA