20SK – Signály a kódy

Přednáška 11 – Slovníkové metody komprese dat (26.11.2018)

Probíraná témata:

- RLE (Run-Length Encoding)
- Slovníkové metody komprese – princip, teoretické vlastnosti
- Algoritmus LZ77
- Algoritmus LZW

Relevantní literatura je [1,2] pro LZ77 a [3] pro LZW. Širší úvod do metod zdrojového kódování (komprese) obsahují knihy Davida Salomona [4,5]. Některé relevantní informace lze nalézt i na Wikipedii nebo jiných online zdrojích.

Seznam literatury

[1] Shor, P.: *Lempel-Zip notes* [online]. MIT, 2005. Available from: http://www-math.mit.edu/~shor/PAM/lempel_ziv_notes.pdf

[2] Ziv, J., Lempel, A.: *A Universal Algorithm for Sequential Data Compression*. IEEE Transactions on Information Theory 23 (3), 1977, pp. 337–343. doi:10.1109/TIT.1977.1055714.

[3] Welch, T.: *A Technique for High-Performance Data Compression*. Computer 17 (6), 1984, pp. 8–19. doi: 10.1109/MC.1984.1659158.

[4] Salomon, D.: *Concise Introduction to Data Compression*. London: Springer Verlag, 2008, 311 pp.

[5] Salomon D., Motta G.: *Handbook of Data Compression*. London: Springer Verlag, 2010, 1360 pp. doi: 10.1007/10.1007/978-1-84882-903-9

Slovníkové metody komprese

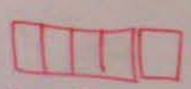
Princip: kódujeme "fráze" z nějaké tabulky na kódová slova (tabulka ≡ slovník, vzniká dynamicky)

Výskyt: kódová slova / symboly se stejnou pravděpodobností výskytu

Předchůdce je **RLE (Run-Length Encoding)**

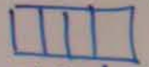
Př: 000000111110000001100000001111

6,0,5,1,6,0,2,1,8,0,4,1



1 bit na hodnotu

4 bity na čítač



4 bity na čítač

⊕ pravidlo: začínáme v 0

6,5,6,2,8,4,...

Ex: 1000...00111

16x

0,1,15,0,1,3

↑

0x nula na zač.

16x nula → 15x nula
0x jedna
1x nula

nemusí jít o binární formát

→ třeba 16 barev (nebo 256)

→ PCX

→ nevhodné pro krátké sekvence!

LZ77 (Lempel-Ziv)

- textová komprese
- používají posuvné okno jako slovník

Ex: abbcbabcbbbab...

posuvné okno délky 8 znaků

pozice 0: abbcbabc 8 znaků

pozice 4: babcbbba 8 znaků

Princip: text na výstupu nahrazují odkazy na již zpracovaný text v posuvném okně

Příklad: Kdyby_byly_vyby_nebyly_by_chyby.

K d y b y_u byl y_r yby_n

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

počet znaků na vykopírování
posun zpět o tolik znaků

l bylyb yuc h yby.

(0,0,l) (12,5,b) (3,2,c) (0,0,h) (16,3,.)

→ není žádný odkaz na předchozí výskyt

32 znaků ⟹ 13 trojic, které „nějak" reprezentují

odkaz (8 bitů, 8 bitů, 8 bitů) ⇒ 13 × 24 bitů = 312

(5 bitů, 3 bity, 7 bitů) ⇒ 13 × 15 bitů = 195

32 × 8 bitů = 256

32 × 7 bitů = 224

Nedostatky: je třeba dlouhé okno ⇒ pomalé
dlouhé okno ⇒ delší reprezentace trojic

LZW (Lempel-Ziv-Welch)

- používá adaptivní slovník, který rozšiřuje nějaký předdefinovaný slovník základní
(slovník lze po kompresi zahodit a při rozbalování zprávy vytvořit znovu)

Princip:
1. Začni se základním slovníkem
2. opakuj:
   a) přidej zakódovanou frázi na výstup
   b) rozšiř slovník o frázi, zahrnující další písmeno

K na výstup a 'Kd' jako novou frázi

Příklad: Kdyby_byly_ryby_nebyly_by_chyby.

K	d	y	b	y	_	by	l	y_	r	yb
0	1	2	3	2	4	15	5	16	6	14
y_	n	e	byl	y_	by	_	c	h		
16	7	8	18	16	15	4	9	10		
yby	.	EOT								
22	11									

Slovník:

základní:
K→0, d→1, y→2, b→3, _→4, e→5, r→6, n→7, l→8, c→9, h→10, .→11

Kd→12, dy→13, yb→14, by→15, y_→16, _b→17, byl→18, ly→19, y_r→20, ry→21, yby→22, y_n→23, ne→24, eb→25, byly→26, y_b→27, by_→28, _c→29, ch→30, hy→31, yby.→32

1. K→0 add('Kd',12)
2. d→1 add('dy',13)
⋮

původně: 32×8 bitů ... 256
7 bitů ... 224

LZW: 22×6 bitů ... 132