Bezstratová kompresia údajov
slovníkové kódovanie

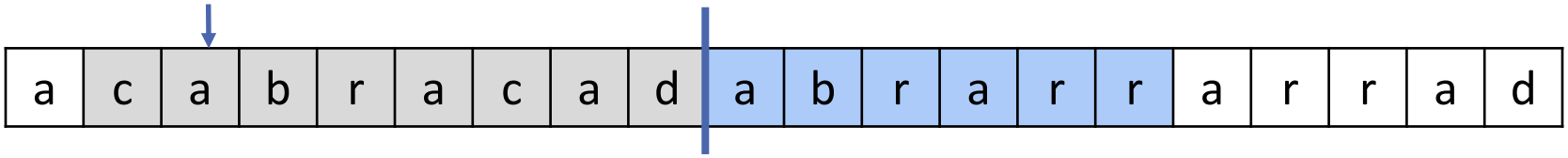
ÚINF/KMÚ ZS 2019/20
doc. RNDr. Jozef Jirásek, PhD.

Slovníkové techniky kompresie

- kniha s veľkým množstvom slov – kód = strana/poradové číslo slova
- pre často sa vyskytujúce slová – kódy pevnej dĺžky (napr. zdrojové texty programu, špecializované texty ...) začiatok – 1-bit – príznak kódu ...
- digramy, trigramy – kódy podľa frekvencie v texte
- adaptívne techniky – prispôsobia sa charakteristikám textu

LZ77

(Lempel-Ziv 1977) ako slovník používa práve odoslané údaje metódou posuvného okna (sliding window), rozdeleného na dve časti



prehľadávací buffer doprednýý (look-ahead) buffer

neodoslané znaky začínajú v doprednom buffri

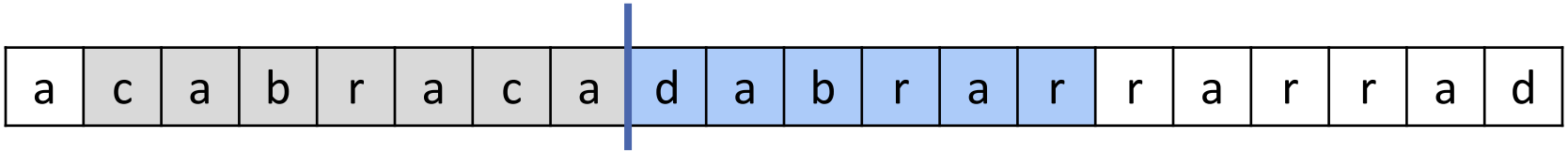
hľadáme v prehľadávacom buffri najdlhší reťazec, ktorý sa zhoduje s reťazcom, začínajúcim v doprednom buffri

kódujeme trojicou (offset, length, char) offset – pozícia začiatku nájdeného reťazca v prehľadávacom buffri od jeho konca (7), length – dĺžka reťazca (4), char – kód nasledujúceho znaku (C(r))

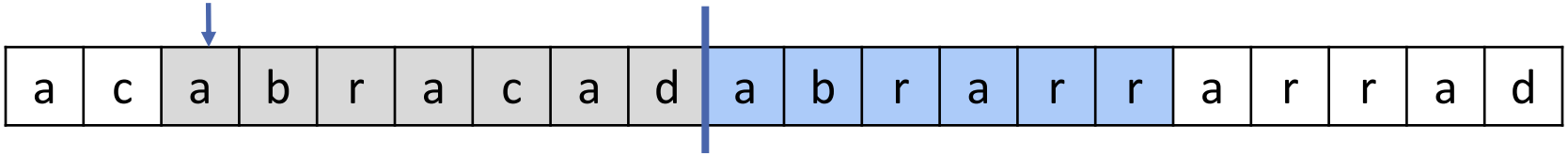
fixná dĺžka kódového slova $\lceil \log S \rceil + \lceil \log W \rceil + \lceil \log N \rceil$

LZ77

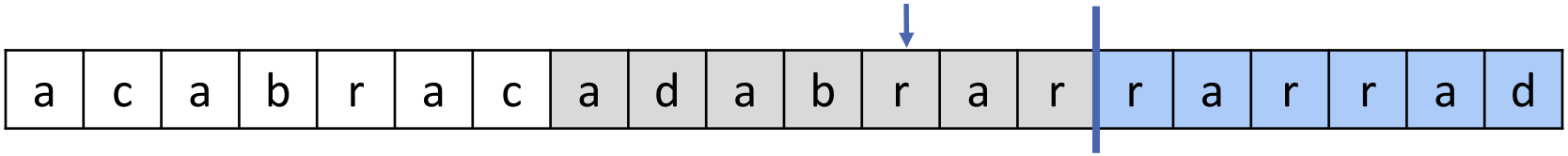
S = 7, W = 13



(0,0,C(d))



(7,4,C(r))



(3,5,C(d))

dekódovanie ...

LZSS

- LZ77 + Storer-Szymanski (gzip)

kódovanie s 1-bitovým príznakom (1,offset, lenght) (0,char)

- prakticky v PKZip, Zip, LHarc, PNG, ARJ sa tento kód ďalej komprimuje VLC kódom (Huffman, AC) – zvlášť offsety, dĺžky
- search buffer ~2^{15} znakov, bloky 64 kB, kvôli rýchlosti sa nemusí hľadať najdlhší podreťazec ... do dĺžky 255

- problém s opakujúcim sa cyklom dlhším ako search buffer

LZ78

(Lempel-Ziv 1978) - buduje sa slovník použitých reťazcov

počiatočný slovník 0: no match

kód je dvojica (index, char) index najdlhšieho reťazca, ktorý sa zhoduje od kódovacej pozície, kód nasledujúceho znaku vstupu

nájdený zhodný reťazec aj s nasledujúcim vstupným znakom sa uložia ako nové slovo do slovníka (v stromovej štruktúre)

kód pre abacababac			dekódovanie		
a	(0,a)	1: a	(0,a)	a	1: a
b	(0,b)	2: b	(0,b)	b	2: b
ac	(1,c)	3: ac	(1,c)	ac	3: ac
ab	(1,b)	4: ab	(1,b)	ab	4: ab
aba	(4,a)	5: aba	(4,a)	aba	5: aba

LZ78

S – max. veľkosť slovníka N – veľkosť abecedy

dĺžka kódového slova $\lceil \log S \rceil + \lceil \log N \rceil$

ak sa dosiahne veľkosť slovníka S

- vymazať celý slovník a začať odznova (GIF)
- mazať druhú polovicu slovníka
- kódovať len so slovami v slovníku (nepridávať ďalšie)
- mazať (čiastočne) keď je zlý kompresný pomer – oznámiť v kóde špeciálnym znakom (UNIX compress)

LZW

(Lempel-Ziv-Welch 1984)

počiatočný známy slovník 0: ... N-1: znaky abecedy

kód je index najdlhšieho spoločného reťazca v slovníku

nájdený reťazec aj s nasledujúcim vstupným znakom sa uložia ako nové slovo do slovníka (v stromovej štruktúre)

abacababac (0: a, 1: b, 2: c)

a	(0)	3: ab
b	(1)	4: ba
a	(0)	5: ac
c	(2)	6: ca
ab	(3)	7: aba
aba	(7)	8: abac

dekódovanie

(0)	a	3: a_
(1)	b	3: ab, 4: b_
(0)	a	4: ba, 5: a_
(2)	c	5: ac , 6: c_
(3)	ab	6: ca, 7: ab_
(7)	ab_	7: aba aba 8:aba_

LZW

- UNIX compress – začína so slovníkom veľkosti 512 (9 bitov)
- postupne sa zväčšuje slovník na 16 bitový, potom sa už nezväčšuje, ale sa sleduje kompresný pomer – ak sa nedosiahne požadovaný, slovník sa vymaže

BWT transformácia

(Burrows-Wheeler 1994) transformácia – cieľom je preskupiť znaky vstupu do tvaru, ktorý je možné efektívne komprimovať

uvažujeme všetky cyklické posuny bloku zdrojového textu, ktoré usporiadame lexikograficky

kódujeme poslednými symbolmi týchto textov aj s pozíciou pôvodného textu v zozname

cyklické posuny	usporiadané	
bacabba	0: abacab**b**	
acabbab	1: abbaba**c**	výsledkom je postupnosť
cabbaba	2: acabba**b**	bcbbaaa a pozícia 4
abbabac	3: babaca**b**	
bbabaca	4: bacabb**a**	
babacab	5: bbabac**a**	
abacabb	6: cabbab**a**	

BWT – inverzná transformácia

postupnosť bcbbaaa je posledným stĺpcom L usporiadanej tabuľky, a keď usporiadame znaky podľa abecedy, dostaneme F - prvý stĺpec usporiadanej tabuľky – zvyšok už môžeme dopočítať

po znaku na pozícii i v L nasleduje (kruhovo) znak na pozícii i v F, ktorý ale odpovedá najvyššiemu zatiaľ nepoužitému výskytu tohto znaku v stĺpci L – pokračovacie pozície uložíme do poľa T

	F	L		T = (4,5,6,0,2,3,1) (T[i] - pozícia i-teho znaku z F v L)
0:	abacab	**b**		F[4] = L[T[4]] = L[2] = **b**
1:	abbaba	**c**		L[T[T[4]]] = L[T[2]] = L[6] = **a**
2:	acabba	**b**		L[T[T[T[4]]]] = L[T[6]] = L[1] = **c**
3:	babaca	**b**		$L[T^4[4]] = L[T[1]] = L[5] =$ **a**
4:	bacabb	**a**	*	$L[T^5[4]] = L[T[5]] = L[3] =$ **b**
5:	bbabac	**a**		$L[T^6[4]] = L[T[3]] = L[0] =$ **b**
6:	cabbab	**a**		$L[T^7[4]] = L[T[0]] = L[4] =$ **a**

BWT transformácia

- lexikografické usporiadanie posunov sa dá jednoducho realizovať kruhovým buffrom, dá sa využiť aj rýchle indexovanie podľa prefixov (dynamické programovanie)
- F nie je potrebné usporadúvať, na výpočet T stačí počítať priebežne početnosť výskytov jednotlivých znakov z L (rank)
- transformovaná postupnosť odzrkadľuje pravdepodobnosť výskytu dvojíc v texte (opakované dvojice znakov sa po transformácii vo výsledku zhlukujú – najskôr všetky znaky predchádzajúce znak a v abecednom poradí, potom všetci predchodcovia b ...)
- transformovanú postupnosť ďalej môžeme zakódovať MTF (Move-to-Front) kódovaním, ktoré kóduje malými číslami znaky, ktoré sú na vstupe v zhlukoch

MTF kódovanie

začíname základným kódovaním (podľa abecedy) všetkých znakov

znak kódujeme jeho pozíciou v abecede a následne predradíme tento znak na prvú pozíciu abecedy

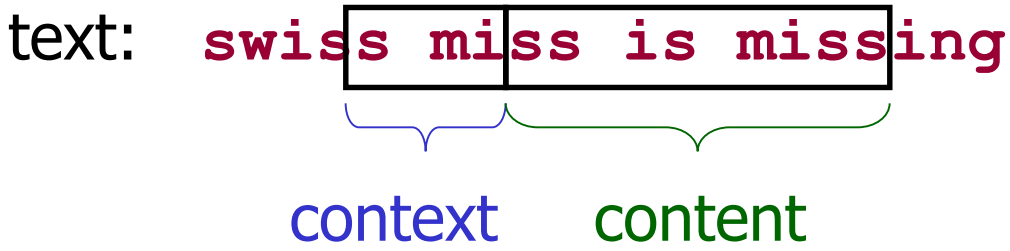
základné kódovanie (0: a, 1: b, 2: c)

znak	kód	úprava kódovania abecedy	dekódovanie		úprava
b	1	(b,a,c)	1	b	(b,a,c)
c	2	(c,b,a)	2	c	(c,b,a)
b	1	(b,c,a)	1	b	(b,c,a)
b	0	(b,c,a)	0	b	(b,c,a)
a	2	(a,b,c)	2	a	(a,b,c)
a	0	(a,b,c)	0	a	(a,b,c)
a	0	(a,b,c)	0	a	(a,b,c)

dostaneme postupnosť malých čísel – ďalej komprimovať RG kódmi, HC resp. AC, sekvencie RLE pre 0 ...

ACB – Associative Coder of Buyanovsky

- efektívna kompresia textu
- podobne ako v LZ77 – uchovávajú sa dvojice



- každý posun okna o jeden znak vytvorí novú položku

```
ss m|iss is mis
s mi|ss is miss
```

- slovník dvojíc context-content sa utriedi podľa kontextov sprava
- kóduje sa pozícia najviac podobného obsahu od pozície najviac podobného kontextu (plus nový znak)

DMC – Dynamic Markov Compression

- na základe stavových automatov – markovovské reťazce

Ďakujem za pozornosť.

jozef.jirasek@upjs.sk