

Kompresa dat

Jan Outrata



KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLMOUCI

přednášky



Slovníkové metody

- výskyt symbolu na vstupu není nezávislý na výskytu ostatních symbolů – symboly se často vyskytují (nebo naopak nevyskytují) v opakujících se vzorech (patterns), např. slova nebo části vět v textu
- nepoužívají statistický model, např. (podmíněné) pravděpodobnosti výskytu vzorů
- = pro kompresi slov na vstupu využívání často se vyskytujících vzorů symbolů = posloupnosti předchozích symbolů aktuálního slova na vstupu
- vyhledávání vzorů na vstupu ve a jejich ukládání do slovníku a kódování odkazu na vzor ve slovníku při/místo kódování slov na vstupu → vyšší míra komprese
- slovník = (implicitně) posloupnost předchozích symbolů aktuálního slova na vstupu nebo (explicitně) datová struktura vzorů symbolů
- často se vyskytujících vzorů (ve slovníku) by mělo být relativně málo, vzhledem ke všem vzorům → výběr nejčastěji se vyskytujících vzorů
- pro specifické aplikace se známými často se vyskytujícími vzory statický, příp. semi-adaptivní, model/slovník (neukládání vzorů), jinak adaptivní – počáteční prázdný nebo malý výchozí a při naplnění nepřidávání, vymazání celého nebo nejdéle nepoužitých vzorů (least recently used, LRU)
- jednoduchá a rychlá dekomprese – oproti statistickým metodám

- použití statistického statického, příp. semi-adaptivního, modelu
 - slovník = všechny symboly vstupní abecedy A + nejčastěji se vyskytující 2- až n -tice symbolů (n -gramy) na vstupu do velikosti slovníku, seřazené podle pravděpodobnosti výskytu sestupně
- = kódování nejdelších slov na vstupu indexem stejného slova ve slovníku

$x \leftarrow$ prázdný řetězec;

while načti ze vstupu symbol $a \in A$ **do**

if xa je ve slovníku **then**

$x \leftarrow xa$;

else

 zapiš na výstup index x ve slovníku;

$x \leftarrow a$;

if $x \neq$ prázdný řetězec **then**

 zapiš na výstup index x ve slovníku;

- kódování indexů statickým kódem nebo statistickým kódováním (Huffmanovým nebo aritmetickým)

Příklad

vstup *barbaraabarboraubaru*, $A = \{a, b, r, u, o\}$, $n = 3$, velikost slovníku 11

$$\begin{aligned}
 f(a) &= \frac{7}{20}, f(b) = \frac{5}{20}, f(r) = \frac{5}{20}, f(u) = \frac{2}{20}, f(o) = \frac{1}{20}, \\
 f(ba) &= \frac{4}{19}, f(ar) = \frac{4}{19}, f(rb) = \frac{2}{19}, f(ra) = \frac{2}{19}, f(aa) = \frac{1}{19}, f(ab) = \frac{1}{19}, f(bo) = \frac{1}{19}, \\
 f(or) &= \frac{1}{19}, f(au) = \frac{1}{19}, f(ub) = \frac{1}{19}, f(ru) = \frac{1}{19}, \text{ostatní } f(xx) = 0 \\
 f(bar) &= \frac{4}{18}, f(arb) = \frac{2}{18}, f(rba) = \frac{1}{18}, f(ara) = \frac{1}{18}, f(raa) = \frac{1}{18}, f(aab) = \frac{1}{18}, \\
 f(aba) &= \frac{1}{18}, f(rbo) = \frac{1}{18}, f(bor) = \frac{1}{18}, f(ora) = \frac{1}{18}, f(rau) = \frac{1}{18}, f(aub) = \frac{1}{18}, \\
 f(uba) &= \frac{1}{18}, f(aru) = \frac{1}{18}, \text{ostatní } f(xxx) = 0
 \end{aligned}$$

slovník:	1	<i>a</i>	7	<i>arb</i>	výstup: 4 4 1 1 4 2 11 9 10 4 10
	2	<i>b</i>	8	<i>rb</i>	
	3	<i>r</i>	9	<i>ra</i>	
	4	<i>bar</i>	10	<i>u</i>	
	5	<i>ba</i>	11	<i>o</i>	
	6	<i>ar</i>			

- také LZ1, Abraham Lempel, Jakob Ziv, 1977 – základ, mnoho variant → rodina metod LZ77
- adaptivní slovník = posloupnost bezprostředně předchozích K symbolů aktuálního symbolu na vstupu (kontext délky K) – search buffer (délky K)
- = kódování nejdelšího (i prázdného) slova a dalšího symbolu na vstupu kódy pozice stejného slova začínajícího v search bufferu (nebo pozice 0), jeho délky a symbolu
- aktuální posloupnost/slovo L symbolů na vstupu – look-ahead buffer (délky L)
- search + look-ahead buffer = posuvné okno (délky $K + L$)

$x \leftarrow$ prázdný řetězec, $o \leftarrow 0$;

while načti ze vstupu symbol $a \in A$ **do**

if xa začíná v search bufferu a délka $x < L$ **then**

$o \leftarrow$ nejmenší vzdálenost (v počtu symbolů) prvního symbolu xa v search bufferu od konce bufferu;

$x \leftarrow xa$;

else

zapiš na výstup kódy o , délky x a a ;

$x \leftarrow$ prázdný řetězec, $o \leftarrow 0$;

if $x \neq$ prázdný řetězec **then**

zapiš na výstup kódy o a délky x ;

- kódování vzdáleností, délek a symbolů statickým kódem nebo (adaptivním) statistickým kódováním (Huffmanovým = LZH nebo aritmetickým)

Příklad

vstup *barbaraabarboraubaru*, $A = \{a, b, r, u, o\}$, $K = 10$, $L = 3$

výstup: 0, 0, *b*, 0, 0, *a*, 0, 0, *r*, 3, 3, *a*, 1, 1, *b*, 8, 3, *o*, 8, 2, *u*, 8, 3, *u*



- kódování celých trojic vzdálenost-délka-symbol místo jednotlivě
- proměnlivé délky K a L bufferů, delší K a L častější a delší nález v search bufferu, ale delší hledání a větší vzdálenosti a délky

LZR (Rodeh)

- délky K a L bufferů neomezené
- = kompresní metoda z algoritmu LZ76 pro měření „složitosti“ textu hledáním předchozích výskytů slov



LZSS (Storer, Szymanski)

- (bitový) příznak pro kódování dvojic vzdálenost-délka anebo symbolu – ukládány skupiny příznaků pro skupiny kódů, např. 8
- kódování samostatných symbolů, pokud by kód dvojice byl stejně dlouhý nebo delší (nebo i mírně kratší, kvůli náročnějšímu kódování dvojice)
- délky K a L tak, aby dvojice byla kódována do pevného dvojnásobného počtu bitů než symbol, např. $K = 2^{11}$ a $L = 2^5$ pro $|A| = 2^8$
- varianta LZB: kódování vzdáleností do postupně se zvyšujícího počtu bitů podle aktuální velikosti search bufferu a délek Eliasovým Gamma kódem
- varianta SLH: kódování vzdáleností a symbolů semi-adaptivním Huffmanovým kódem

Deflate (Philip W. Katz)

- nejúspěšnější varianta LZ77 a LZSS – dvojice délka-vzdálenost
- volitelně (příp. redukované) vyhledání i delšího vzoru po kódování samostatných symbolů na vstupu
- komprese vstupních dat po blocích různé délky
- mód 1 = bez komprese – 5 B záhlaví navíc (3 b v 1 B kódy ne/posledního bloku a módu 1, 2 B délka bloku a 2 B jedničkový komplement délky)
- mód 2: kódování dvojic a symbolů podle statických modelů
 - symboly a délky čísla 0 – 285: 0 – 255 symboly (byty), 256 konec bloku, 257 – 285 s dalšími 0 – 5 bity (pro určení hodnoty) délky 3 – 258 = L , kódovanými statickým Huffmanovým kódem (7 – 9 bitů/číslo)
 - vzdálenosti 1 – $2^{15} = K$ čísla 0 – 29 s dalšími 0 – 13 bity (pro určení hodnoty v rozsahu), kódovanými binární reprezentací čísla (5 bitů/číslo)

Obrázek: Kódování symbolů/délek a vzdáleností v módu 2

Extra			Extra			Extra		
Code	bits	Lengths	Code	bits	Lengths	Code	bits	Lengths
257	0	3	267	1	15,16	277	4	67–82
258	0	4	268	1	17,18	278	4	83–98
259	0	5	269	2	19–22	279	4	99–114
260	0	6	270	2	23–26	280	4	115–130
261	0	7	271	2	27–30	281	5	131–162
262	0	8	272	2	31–34	282	5	163–194
263	0	9	273	3	35–42	283	5	195–226
264	0	10	274	3	43–50	284	5	227–257
265	1	11,12	275	3	51–58	285	0	258
266	1	13,14	276	3	59–66			

edoc	Bits	Prefix codes
0–143	8	00110000–10111111
144–255	9	110010000–111111111
256–279	7	0000000–0010111
280–287	8	11000000–11000111

Extra			Extra			Extra		
Code	bits	Distance	Code	bits	Distance	Code	bits	Distance
0	0	1	10	4	33–48	20	9	1025–1536
1	0	2	11	4	49–64	21	9	1537–2048
2	0	3	12	5	65–96	22	10	2049–3072
3	0	4	13	5	97–128	23	10	3073–4096
4	1	5,6	14	6	129–192	24	11	4097–6144
5	1	7,8	15	6	193–256	25	11	6145–8192
6	2	9–12	16	7	257–384	26	12	8193–12288
7	2	13–16	17	7	385–512	27	12	12289–16384
8	3	17–24	18	8	513–768	28	13	16385–24576
9	3	25–32	19	8	769–1024	29	13	24577–32768

Deflate (Philip W. Katz)

- mód 3: kódování dvojic a symbolů semi-adaptivními Huffmanovými kódy (pro symboly/délky a vzdálenosti s čísly podle módu 2, max. 15 bitů) napříč bloky, kódovanými jako posloupnosti délek kódů RLE (s min. počtem 4 stejných symbolů za sebou) a (semi-adaptivním) Huffmanovým kódem uloženým jako modifikovaná posloupnost délek kódů (3 bity/délka)
 - (ekvivalentní) Huffmanův kód C' z posloupnosti $l(a_1) \leq \dots \leq l(a_n)$ délek kódových slov $C(a_i)$ Huffmanova kódu C : $C'(a_i) =$ binární reprezentace čísla $B_j + k$ délkou $l(a_i) = j$, kde $B_j = 2(B_{j-1} + \text{počet } a \text{ s } l(a) = j - 1)$, $B_1 = 0$ a a_i je k -tý (od 0) a s $l(a) = j$

Příklad

$C(a) = 00$, $C(b) = 01$, $C(r) = 10$, $C(u) = 110$, $C(o) = 111$

$l(a) = 2$, $l(b) = 2$, $l(r) = 2$, $l(u) = 3$, $l(o) = 3$

$C'(a)$: binární reprezentace $B_j + k = 0 + 0$

$C'(b)$: $0 + 1$

$C'(r)$: $0 + 2$

$C'(u)$: $6 + 0$

$C'(o)$: $6 + 1$

LZPP (Pylak, 2003)

- varianta LZSS – minimální délka 3 nalezeného slova
- snaha odstranit veškeré redundance ve výstupu vzdáleností, délek, symbolů i příznaků
- většina vzdáleností a délek je malých, pravděpodobnost příznaku pro symbol klesá \Rightarrow velká entropie \rightarrow modifikované adaptivní aritmetické kódování (range encoding), i pro symboly:
 - po bytech, pravděpodobnosti výskytu bez kontextu a v kontextu délky 1 – u symbolů s vyloučením symbolů za vyhledaným slovem v search bufferu (exclusion principle)
 - u symbolů speciální (escape) symbol abecedy pro neexistující/první výskyt symbolu v search bufferu (místo inicializace počtu výskytu každého symbolu na 1) – pravděpodobnost pro každý symbol abecedy (z počtu kódování symbolu speciálním symbolem)
- další příznaky pro pouze vzdálenosti s nejčastější délkou 3 a kódování symbolu v kontextu délky 1

LZMA (Igor Pavlov)



Další

- LZX
- LZP

Implementace

- délky K a L bufferů až tisíce a desítky až stovky symbolů (bytů)
- pro posuvné okno kruhová fronta (LZSS, LZPP), pro search buffer např. suffix stromy (LZR), (vyvážené) binární vyhledávací stromy (s lexikografickým uspořádáním slov v uzlech, LZSS), hešovací tabulky (SLH, Deflate – volitelně tři symbolů, LZPP – CRC-32)
- ARJ, (PK)Arc, LHArc, LHA (LZSS + Huffman), (PK)Zip, gzip, zlib (Deflate), RAR, ACE (LZSS + Huffman) aj.

Aplikace

- nárůst kvůli poplatkům z patentu na LZW
- v síťových protokolech, např. HTTP, PPP – Deflate
- v kompresi obrazu a dokumentů, např. PNG, PDF – Deflate

- LZ77 předpokládá, že opakující se vzory se vyskytují blízko sebe (do vzdálenosti délky K search bufferu), ale stejné slovo na vstupu může začínat před search bufferem
 - také LZ2, Abraham Lempel, Jakob Ziv, 1978 – základ, mnoho variant → rodina metod LZ78
 - adaptivní slovník = slova na vstupu uložená ve slovníku (nebo prázdné slovo) zřetěžená s dalším symbolem na vstupu za slovem, počáteční prázdný
- = kódování nejdelšího (i prázdného) slova a dalšího symbolu na vstupu kódy indexu stejného slova ve slovníku (nebo indexu 0) a symbolu, a uložení do slovníku zřetězení slova a symbolu
- vytváření stejného slovníku při kódování i dekódování
 - ze slovníku se nemaže – na rozdíl od search bufferu LZ77, vymazání slovníku při jeho naplnění = předpoklad LZ77 (že opakující se vzory se vyskytují blízko sebe – do vzdálenosti odpovídající velikosti slovníku)
 - pomalá adaptace na vstup – do slovníku se přidávají slova jen o 1 symbol delší než slovo již ve slovníku

$x \leftarrow$ prázdný řetězec;

while načti ze vstupu symbol $a \in A$ **do**

if xa je ve slovníku **then**

$x \leftarrow xa$;

else

 ulož xa jako další položku do slovníku;

 zapiš na výstup kódy indexu x ve slovníku (počínaje 1) nebo 0 (pro x prázdný řetězec) a a ;

$x \leftarrow$ prázdný řetězec;

if $x \neq$ prázdný řetězec **then**

 zapiš na výstup kód indexu x ve slovníku;

- kódování indexů a symbolů statickým kódem nebo (adaptivním) statistickým kódováním (Huffmanovým nebo aritmetickým)

Příklad

vstup *barbaraabarboraubaru*, $A = \{a, b, r, u, o\}$

slovník:	1		<i>b</i>	7		<i>ar</i>	výstup: <i>0 b 0 a 0 r 1 a 3 a 2 b 2 r 1 o 5 u 4 r 0 u</i>
	2		<i>a</i>	8		<i>bo</i>	
	3		<i>r</i>	9		<i>rau</i>	
	4		<i>ba</i>	10		<i>bar</i>	
	5		<i>ra</i>	11		<i>u</i>	
	6		<i>ab</i>				

while načti ze vstupu a dekoduj kód indexu i ve slovníku **do**
 $x \leftarrow$ prázdný řetězec;
if $i \neq 0$ **then**
 zapiš na výstup slovo $x \in A^+$ na indexu i ve slovníku;
if načti ze vstupu a dekoduj kód symbolu $a \in A$ **then**
 ulož xa jako další položku do slovníku;
 zapiš na výstup symbol $a \in A$;

Datová reprezentace slovníku = n -ární strom $T = \langle V, E \rangle$ (n velikost A)

- velikost slovníku jednotky až desítky tisíc (2^{16}) symbolů
- trie jako u PPM: $a|c^k \approx$ slovo $a_{-k} \dots a_{-1}a$, pro $a \in A$ s prefixem $a_{-k} \dots a_{-1} = x \in A^+$ index slova xa ve slovníku
- kódování:
 - modifikace: pro uzly $v(xa), v(xab), v(xac), v(xad), \dots \in V$ pro symboly a s prefixem x a b, c, d, \dots s prefixem xa hrany $\langle v(xab), v(xac) \rangle, \langle v(xac), v(xad) \rangle, \dots \in E$ místo hran $\langle v(xa), v(xac) \rangle, \langle v(xa), v(xad) \rangle, \dots$
 - tabulka se sloupci index slova xa , symbol a , index slova xab a pro slova xab, xac, \dots index slova xac, xad, \dots
- dekódování: tabulka se sloupci index slova xa , symbol a a pro slova xab index slova xa

Příklad

slovník kódování:

1	b	4		5	a	9		9	u
2	a	6		6	b		7	10	r
3	r	5		7	r			11	u
4	a	10	8	8	o				

slovník dekódování:

1	b		5	a	3	9	u	5
2	a		6	b	2	10	r	4
3	r		7	r	2	11	u	
4	a	1	8	o	1			

LZFG (Fiala, Greene)

- kombinace LZ77 (search buffer) a LZ78 (dvojice kódů)
- = kódování nejdelšího (neprázdného) slova na vstupu kódy délky a pozice stejného slova začínajícího v search bufferu anebo kódy délky slova a všech jeho symbolů
- varianta A1: 4 bity pro binární reprezentaci délky $2 - 16 = L$ slova následované pozicí (vzdáleností) $1 - 4096 = K$ kódovanou do 12 bitů anebo délky $1 - 16$ slova s prefixem **0000** následované symboly
- varianta A2: zobecněné unární kódy (start-step-stop kódy) pro délku $2 - 2044$ ((2, 1, 10) kód, 3 - 18 bitů) následovanou pozicí až $K = 2^{10} + 2^{12} + 2^{14}$ ((10, 2, 14) kód, 11 - 16 bitů) anebo délku $1 - 63$ ((0, 1, 5) kód, 1 - 10 bitů) následovanou symboly, další vylepšení s postupně narůstajícím K od 21 a pozicí kódovanou do postupně $1 - 16$ bitů ((10 - d , 2, 14 - d) kód, d postupně 10, ..., 0) aj.
- další varianty B1, B2, C1 a C2

Další

- LZRW1
- LZRW4

= nejpopulárnější varianta LZ78, Terry Welch, 1984

- počáteční slovník (komprese i dekomprese) = všechny symboly vstupní abecedy A
- = kódování nejdelšího (neprázdného) slova na vstupu kódem indexu stejného slova ve slovníku, a uložení do slovníku zřetězení slova a dalšího symbolu na vstupu

$x \leftarrow$ prázdný řetězec;

while načti ze vstupu symbol $a \in A$ **do**

if xa je ve slovníku **then**

$x \leftarrow xa$;

else

 ulož xa jako další položku do slovníku;

 zapiš na výstup kód indexu x ve slovníku;

$x \leftarrow a$;

if $x \neq$ prázdný řetězec **then**

 zapiš na výstup kód indexu x ve slovníku;

Příklad

vstup *barbaraabarboraubaru*, $A = \{a, b, r, u, o\}$

slovník:	1	<i>a</i>	7	<i>ar</i>	13	<i>barb</i>
	2	<i>b</i>	8	<i>rb</i>	14	<i>bo</i>
	3	<i>r</i>	9	<i>bar</i>	15	<i>or</i>
	4	<i>u</i>	10	<i>ra</i>	16	<i>rau</i>
	5	<i>o</i>	11	<i>aa</i>	17	<i>ub</i>
	6	<i>ba</i>	12	<i>ab</i>	18	<i>baru</i>

výstup: 2 1 3 6 3 1 1 9 2 5 10 4 9 4

Kódování

- 1 další položka xa ve slovníku: x slovo ve slovníku na aktuálně kódovaném indexu i , a první symbol slova na dalším kódovaném indexu
- 2 slovo na dalším kódovaném indexu může být posledně uložená položka ve slovníku

Dekódování

- 1 další položka xa ve slovníku: x slovo na předchozím dekodovaném indexu, a první symbol slova na aktuálně dekodovaném indexu i
- 2 slovo na aktuálně dekodovaném indexu i se má právě vložit jako další položka do slovníku $\Rightarrow a =$ první symbol slova x na předchozím dekodovaném indexu

$x_p \leftarrow$ prázdný řetězec;

while načti ze vstupu a dekoduj kód indexu i ve slovníku **do**

if i je index další položky ve slovníku **then**

$x \leftarrow x_p$;

else

$x \leftarrow$ slovo na indexu i ve slovníku;

if $x_p \neq$ prázdný řetězec **then**

$a \leftarrow$ první symbol x ;

ulož $x_p a$ jako další položku do slovníku;

$x_p \leftarrow x$;

zapiš na výstup slovo $x \in A^+$ na indexu i ve slovníku;



LZC

- postupně zdvojnásobovaná velikost slovníku od 512 (9 bitů/index) do 2^{16} , při naplnění statický a při poklesu kompresního poměru pod mez vymazání
- vylepšení: při kódování indexu slova vyloučení slov ve slovníku začínajících symbolem, kterým předchází slovo pokračovalo v jiném slově ve slovníku (exclusion principle)

LZT (Tischer)

- varianta LZC
- při naplnění slovníku vyřazení slova s nejdéle nekódovaným indexem – kromě slovníku i seznam slov ze slovníku seříděný podle počtu kódování indexu slova, podobné LZ77, ale „posuvné okno“ slov podle počtu kódování indexu, ne pořadí slov na vstupu

LZMW (Miller, Wegman)

- při naplnění slovníku vyřazení slova s nejdéle nekódovaným indexem – nejstarší ze slov bez pokračování v jiném slově (tzn. index slova nebyl kódován), vyžaduje dat. strukturu slov podle jejich „věku“
- uložení do slovníku zřetězení slova na vstupu, které je ve slovníku, a dalšího slova na vstupu, které je ve slovníku (místo pouze jeho prvního symbolu) \Rightarrow rychlejší adaptace na vstup – do slovníku se přidávají slova o víc než jen o 1 symbol delší
- datová reprezentace slovníku nemůže být trie – ve slovníku nemusí být každý prefix slova \rightarrow dodání s příznakem platnosti, ale při vyhledávání vracení se od neplatných

LZAP (All Prefixes)

- varianta LZMW
- zřetězení nejen s dalším slovem, ale se všemi jeho neprázdnými prefixy (včetně celého slova) \rightarrow větší slovník, ale nevracení se při vyhledávání

Další

- LZJ
- LZY

Implementace

- pro slovník hešovací tabulka – řádků tabulky pro dekódování ukládající trie LZ78 slovníku
- compress na UNIXu (LZC)

Aplikace

- narůstající až do poplatků z patentu
- v kompresi obrazu GIF – podobně jako compress
- v kompresním módu modemových přenosů dat po telefonní síti podle standardu V.42bis – podobně jako compress, ale při naplnění slovníku jeho promazávání o dlouho nepoužitá slova, nekódování posledně uloženého slova ve slovníku, ale jeho složek