Kompresní metoda LZ78

Kompresní metoda LZ78 patří do třídy slovníkových metod komprimujících pomocí indexů do postupně *rostoucího slovníku* prefixů. Autory kompresního algoritmu jsou A. Lempel a J. Ziv. Během komprese je organisován slovník, to jest struktura sdružující *prefix vstupních znaků* s jeho *nenulovým celočíselným indexem.* Následující popis stručně popisuje kompresní algoritmus a organisaci slovníku. Další popis je k nalezení v [DS99].

Označení. *Vstupní abeceda* je konečná množina symbolů. Nechť A je vstupní abeceda, pak libovolná konečná posloupnost znaků z A, to jest posloupnost tvaru $\alpha = a_1 a_2 \cdots a_k$, se nazývá *řetězec nad abecedou* A; číslo $k = |\alpha|$ označuje délku řetězce α. Je-li $|\alpha| = 0$, řetězec α se nazývá *prázdný řetězec* a je značen ε. Množina všech řetězců nad abecedou A se nazývá *uzávěr* A jenž je značen A^*. Řetězec π je *prefixem řetězce* α, pokud lze psát $\alpha = \pi\beta$ pro nějaké $\beta \in A^*$. Speciálně π je *triviálním prefixem* pokud $\pi = \varepsilon$.

Kompresní slovník nad vstupní abecedou A je dvojice $\mathcal{S} = \langle S, \iota \rangle$, kde S je konečná $S \subseteq A^*$ a ι je bijektivní zobrazení $\iota \colon S \to \{1, \ldots, |S|\}$. Číslo $\iota(\alpha)$, $\alpha \in S$ se nazývá *index fráze* α. Slovník $\mathcal{T}$ vznikne ze slovníku $\mathcal{S} = \langle S, \iota \rangle$ *přidáním fráze* $\alpha \in A^*$, pokud $\alpha \notin S$ a $\mathcal{T}$ je ve tvaru $\mathcal{T} = \langle S \cup \{\alpha\}, \iota_{\mathcal{T}} \rangle$, kde $\iota_{\mathcal{T}} \colon S \cup \{\alpha\} \to \{1, \ldots, |S| + 1\}$ je rozšířením zobrazení ι a platí $\iota_{\mathcal{T}}(\alpha) = |S| + 1$, nebo $\alpha \in S$ a $\mathcal{T} = \mathcal{S}$. Fakt, že $\mathcal{T}$ vznikne z $\mathcal{S}$ *postupným přidáním jistých frází*, je označován $\mathcal{S} \trianglelefteq_A \mathcal{T}$. Snadno nahlédneme, že relace $\trianglelefteq_A$ je kvaziuspořádání na množině všech slovníků nad abecedou A.

Algoritmus. Kompresní algoritmus je definován induktivně.

1. *Indukční předpoklad.* Na počátku komprese je kompresní slovník $\mathcal{S}_0$ prázdný, to jest $\mathcal{S}_0 = \langle \emptyset, \iota \rangle$. Vstupem kompresního algoritmu je řetězec $\alpha_0 \in A^*$, $\alpha = a_1 \cdots a_n$. Výstupem kompresního algoritmu je posloupnost dvojic $\langle n, a \rangle$, kde n je *index některé fráze* nebo $n = 0$. Dále $a \in A$ je znak vstupní abecedy. Tato posloupnost je na počátku prázdná.

2. $(k+1)$*-ní krok indukce.* Na vstupu je dosud nezpracovaná část vstupního řetězec, označme ji $\alpha_k = a_j a_{j+1} \cdots a_n$. Slovník $\mathcal{S}_k = \langle S_k, \iota_k \rangle$ sestává z $|S_k| = k$ neprázdných frází. Označme

   $$P_k = \{\beta;\, \beta \in S_k,\, \alpha_k = \beta\gamma\},$$

   to jest P_k je množinou frází z $\mathcal{S}_k$ jenž jsou *netriviálními prefixy* α_k. Nastávají dva případy.

   - Pokud je $P_k = \emptyset$, pak je k výstupní posloupnosti připojena dvojice $\langle 0, a_j \rangle$. Dále $\alpha_{k+1} = a_{j+1} \cdots a_n$ a slovník $\mathcal{S}_{k+1}$ vznikne ze slovníku $\mathcal{S}_k$ *přidáním fráze* a_j. Algoritmus pokračuje bodem číslo 3.
   - Pokud platí $P_k \neq \emptyset$, pak existuje $\pi \in P_k$ tak, že $|\beta| \leq |\pi|$ pro libovolný $\beta \in P_k$. K výstupní posloupnosti je připojena dvojice $\langle \iota(\pi), a_{j+|\pi|} \rangle$. Dále $\alpha_{k+1} = a_{j+|\pi|+1} \cdots a_n$ a slovník $\mathcal{S}_{k+1}$ vznikne ze slovníku $\mathcal{S}_k$ *přidáním fráze* π. Algoritmus pokračuje bodem číslo 3.

3. Pokud je $\alpha_{k+1} = \varepsilon$ algoritmus končí. V opačném případě algoritmus pokračuje bodem číslo 2 pro vstupní řetězec α_{k+1} a kompresní slovník $\mathcal{S}_{k+1}$.

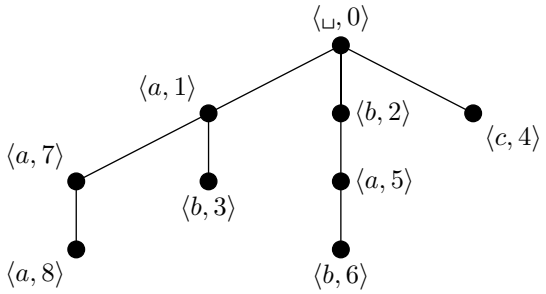
O konečnosti algoritmu se lze snadno přesvědčit matematickou indukcí. V každém kroku algoritmu je konečný vstup vždy zkrácen. Nutně tedy musí po konečně mnoha krocích nastat situace $\alpha_{k+1} = \varepsilon$ a algoritmus končí.

Poznámka. Při technické realisaci algoritmu existuje několik problémů. V prvním případě je to růst slovníku. Při kompresi velkých dat by se mohl slovník velmi rychle zvětšovat. V praxi je problém obvykle řešen „vysypáním slovníku“ v případě, kdy počet frází vzroste nad přípustnou mez. My tento problém velkoryse zanedbáme.

Závažným problémem je rovněž *organisace slovníku.* Slovník v podstatě slouží k vyhledávání *největšího existujícího prefixu* k danému vstupnímu řetězci. Vhodnou representací je například n-ární

strom. Kořen stromu je ohodnocen číslem 0, vnitřní uzly stromu jsou ohodnoceny dvojicí $\langle n, a\rangle$, kde n je číslo indexu a $a \in A$ je znak. Číslo n se vztahuje k frázi vzniklé seřazením znaků obsažených ve všech uzlech na cestě od kořenu do daného vnitřního uzlu.

Příklad. Uvažujme například slovník $\mathcal{S} = \langle S = \{\alpha_1, \ldots, \alpha_8\}, \iota\rangle$ nad tříprvkovou vstupní abecedou $A = \{a, b, c\}$, kde $\alpha_1 = a$, $\alpha_2 = b$, $\alpha_3 = ab$, $\alpha_4 = c$, $\alpha_5 = ba$, $\alpha_6 = bab$, $\alpha_7 = aa$, $\alpha_8 = aaa$. Přitom $\iota(\alpha_i) = i$. Tento slovník vznikl během zakódováním vstupního řetězce $\alpha = ababcbababaaaaa$. Slovník $\mathcal{S}$ může být representován následujícím n-árním stromem.



Předchozí strom lze velmi jednoduše procházet a lze tak najít nejdelší frázi tvořící prefix vstupního řetězce v *lineárním čase.* Přidávání nových frází do slovníku je taktéž jednoduché – jedná se o připojení nového listového uzlu do stromu.

Pro strom frází použijeme v jazyku Scheme jednotnou representaci. Vyjdeme z tradiční representace n-árního stromu. Každý uzel stromu je representován *seznamem.* Tento seznam je vždy neprázdný a jeho první prvek je ohodnocení uzlu. Další prvky jsou *podstromy* representované seznamy. Přijměme úmluvu, že kořen je ohodnocen číslem značícím *nejvyšší použitý index* fráze ve slovníku. Ostatní uzly jsou ohodnoceny tečkovým párem `(`a `.` n`)`, kde a je znak a n je index fráze. Slovník z předcházejícího příkladu je tedy representován seznamem v následujícím tvaru.

```
; strom representující stovník
(#\8
    ((#\a . 1)
            ((#\a . 7)
                    ((#\a . 8)))
            ((#\b . 3)))
    ((#\b . 2)
            ((#\a . 5)
                    ((#\b . 6))))
    ((#\c . 4)))
```

Samozřejmě, že týž n-ární strom je representován obecně více seznamy. Všechny podstromy obsažené v libovolném uzlu je potřeba chápat jako množiny podstromů, to jest na jejich pořadí v representaci nezáleží.

Poznámka. Dekomprese je již o poznání jednodušší. Vstupem dekompresního algoritmu je posloupnost dvojic ve tvaru $\langle n, z\rangle$ kde n je indexem do slovníku a z je bezprostředně následující znak. Slovník je opět postupně budován, na počátku je prázdný. Místo hledání prefixů ale dekompresní algoritmus připojuje na výstup frázi odpovídající nenulovému indexu. Za frázi je na výstup připojen navíc znak z. Dekompresní slovník je vhodné organisovat strukturou s náhodným přístupem, například *dynamickým polem prefixů,* nebo *hashovací tabulkou.*

Reference

[DS99] Dvorský, Jiří – Snášel, Václav. *Algoritmická matematika I.* VUP, Olomouc, 1999. Strany 214–216.

[KS97] Kopka, Martin – Skoupil, David. *Průvodce jazykem Scheme.* VUP, Olomouc, 1997. Strany 70–74.