Aplikovaná informatika

Podklady předmětu
Aplikovaná informatika
pro akademický rok 2013/2014
Radim Farana

12

Obsah

- Komprese dat, základní principy a použití.
- Bezeztrátová komprese,
  - Slovníkové metody.
  - Aritmetické metody.

Komprese dat

- Bezeztrátová komprese - odstranění redundance:
  - Huffmanův kód,
  - Slovníkové metody,
    - opakování znaků (RLE),
    - Lempel-Ziv 1977, 1978, Lempel-Ziv-Welch 1984,
  - Aritmetická komprese
- Ztrátová komprese:
  - zvuku, obrazu, …

Huffmanův kód

Triviální případ

Zpráva	*P*(*i*)	kód
1	>	0
2	<	1

Redukovaná abeceda

Zpráva	*P*(*i*)	redukce	kód	expanze
1	»	1	0	0
2	>	2,3	1	10
3	<			11

Příklad

Zpráva	A	B	C	D	E
P(*i*)	0,4	0,3	0,1	0,1	0,1
1.redukce	0,4	0,3	0,2	0,1	
2.redukce	0,4	0,3	0,3		
3.redukce	0,6	0,4			
znaky	0	1			
kód	1	00	011	0100	0101

Postup:
- seřazení podle pravděpodobnosti,
- postupná redukce a oprava pořadí,
- přiřazení znaků 0, 1 a zpětná expanze.

Problémy:
- definice pořadí zpráv pro stejnou *P*(*i*),
- zařazení skupin se stejnou *P*(*i*),
- pořadí přiřazení znaků 0, 1.

Varianty komprese

- Statická komprese – dvouprůchodová,
  - zjištění četnosti jednotlivých znaků,
  - sestavení optimálního kódu,
  - komprese souboru.
- Dynamická komprese – jednoprůchodová,
- průběžné upřesňování stromu kódu.

Opakování znaků

- Sekvence znaků kódována jako:
  zarážka, opakovaný znak, počet
- Výskyt zarážky vede na tvar:
  zarážka, zarážka, 1
- RLE – Run Length Encoding (součást formátu PCX) – sloučení zarážky a počtu:

LZ

Abraham Lempel Jacob Ziv Terry A. Welch

http://www.hpl.hp.com/about/bios/abraham_lempel.html

* 27. 11. 1931 Tiberias, Izrael

http://www.marconifoundation.org/pages/fellows/Fellows_details/Ziv.htm

- 1977 LZ, vylepšena 1978
  - vyhledávání opakujícího se řetězce v kruhovém zásobníku, kódování s využitím devátého bitu:
    0xxxxxxxx – nekomprimovaný znak
    1xxxxxxxx – komprese: odkaz na začátek řetězce a jeho délka

LZ

- využívá se kruhový zásobník velikosti 32 768 Byte.
  - Do bufferu načteme několik znaků a vyhledáme nejdelší řetězec shodující se s takto načtenou posloupností. Pokud je jeho délka menší než určená minimální hodnota (threshold), zapíšeme první znak posloupnosti na výstup a pokračujeme dalším znakem.
  - Jinak zapíšeme na výstup odkaz na nalezený řetězec a pokračujeme znakem za řetězcem. Odkaz se skládá z délky a pozice dřívějšího výskytu řetězce. Pro rozlišení mezi odkazem a běžným znakem se přidává devátý bit.

Příklad zápisu

- opakující se řetězec znaků délky 40 a pozice 200 znaků (od počátku právě zpracovávaného řetězce zpět).
  - komprimační algoritmus nalezl duplicitní řetězec délky 40 znaků a v tabulce LZ1 najde této délce přiřazený kód (pro délku 35 až 42 znaků kód 273) a počet upřesňujících bitů (3 bity pro určení přesné délky v příslušném intervalu),
  - tento kód 273 je odeslán na výstup, čímž je také nastaven indikátor komprimace označující řetězec komprimovaných dat (273_{10} = **1** 0001 0001_2),
  - vypočítají a odešlou se tři bity pro upřesnění délky: zjištěná délka minus spodní hranice výše určeného intervalu: $40 - 35 = 5_{10}$ = **101**$_2$,
  - dále komprimační algoritmus stejným způsobem zpracuje a odešle kód pozice duplicity za použití tabulky LZ2. Najde v ní příslušný kód (pro pozici v intervalu 193 až 253 je to kód 15) a počet upřesňujících bitů (6 bitů pro určení přesné pozice v příslušném intervalu),
  - tento kód 15 je odeslán na výstup,
  - vypočítá a odešle se na výstup šest bitů pro upřesnění pozice: zjištěná pozice minus spodní hranice výše určeného intervalu:
    $200 - 193 = 7_{10}$ = **000111**$_2$ (musí být dodržen počet upřesňujících bitů zjištěný v tabulce).

Tabulky

tabulka LZ1: délka duplicity				tabulka LZ2: pozice duplicity		
kód	počet bitů	délka		kód	počet bitů	pozice
257	0	3		0	0	1
…				…		
272	2	31-34		14	6	129-192
273	*3*	*35-42*		*15*	*6*	*193-256*
274	3	43-50		16	7	257-384
…				…		
284	5	227-257		28	13	16385-24576
285	0	258		29	13	24577-32768

Dekomprimace LZ

- Při načítání se kontroluje nastavení indikátoru komprimace (první bit devítibitové sekvence je roven jedné).
- Po jeho nalezení je následující kód (257 až 285) určující pro nalezení délky opakujícího se řetězce v tabulce LZ1 spolu s počtem následujících upřesňujících bitů.
- Další kód (0 až 29) je stejným způsobem považován za ukazatel pozice v tabulce LZ2 spolu s počtem očekávaných upřesňujících bitů.
- Výpočet přesné délky a pozice duplicitního řetězce proběhne opačným způsobem než při komprimaci.
- Pak je již tento délkou a pozicí určený shodný řetězec znaků zkopírován z již dekomprimované části souboru na aktuální pozici.

LZW

- 1984 LZW
  - Využití hašování (hešování) – moderní metody rychlého vyhledávání.

Algoritmus LZW

- Využívá dva buffery – PREFIX (dvoubytové pole) a TAIL (jednobytové pole), hašovací tabulku pro rychlé vyhledávání a několik proměnných:
- X – aktuální znak,
- KX – poslední prefix,
- D – hodnota hašovací funkce,
- Z – obsah položky hašovací tabulky.

Algoritmus LZW

1. Inicializujeme hašovací tabulku – naplníme ji jinak nepoužívanými hodnotami (NULL).
2. Do proměnné KX načteme znak ze vstupu, do X následující znak.
3. Spočítáme hodnotu hašovací funkce, např: D = (KX 2H) XOR X AND (Q – 1)
   - kde je H – hašovací konstanta (hašování je algoritmus rychlého vyhledávání),
   - Q – velikost hašovací tabulky.

Algoritmus LZW

4. Proměnné Z přiřadíme obsah hašovací tabulky s indexem D.
   - Má-li Z prázdnou hodnotu, pak sekvenci (KX, X) nelze zkomprimovat, obsah KX zapíšeme na výstup (vyšleme *n* bitů, kde *n* je dvojkový logaritmus z délky tabulek, viz dále). Do tabulky PREFIX zapíšeme hodnotu KX na první volné místo, na stejné místo v tabulce TAIL zapíšeme X. Do hašovací tabulky na pozici D uložíme index tohoto prvního volného místa. Do KX přesuneme obsah X, do X načteme další znak, jdeme na bod 3.
   - Má-li Z hodnotu, pak je ukazatelem do tabulky PREFIX a TAIL a byla nalezena shoda s hledanou posloupností (KX, X). Vyzkoušíme, zda se neshodují i další znaky do KX umístíme hodnotu Z (odkaz na již nalezenou hodnotu), do X další znak ze vstupu a jdeme na bod 3. (na výstup bude vyslána poslední hodnota Z – pozice v tabulce PREFIX a TAIL poslední shody, tedy konce shodného řetězce.).

Algoritmus LZW

- Problémy jsou spojeny s velikostí tabulek TAIL a PREFIX - rychle se plní.
  - Na počátku nastavíme počáteční délku (512), délka kódu na nejmenší nutný počet znaků ($\log_2$ z počtu položek, tedy 9).
  - Prvních 256 položek se přitom nemůže používat, shodovaly by se s nezakódovanými znaky.
  - Jakmile se tabulky zaplní (a nejsou příliš dlouhé), zdvojnásobíme jejich délku a kód se prodlouží o 1 bit.
  - Jsou-li tabulky příliš dlouhé, vyprázdní se a celý algoritmus začne znovu od začátku.

Aritmetická komprese

- Vychází z myšlenky, že i stejné znaky na různých místech souboru mohou být zapisovány různým počtem bitů.
- Algoritmus je velmi náročný na aritmetické operace (zejména násobení a dělení velkých čísel).
- Pracuje se v něm s intervalem racionálních čísel a výsledek (tedy výstupní řetězec znaků) se nachází vždy v tomto intervalu. Na začátku práce je interval roven polouzavřenému intervalu $<0, 1)$. Hranice intervalu jsou v paměti uloženy jako binární desetinná čísla

Aritmetická komprese

- Interval $<0, 1)$ je rozdělen na části odpovídající výskytu jednotlivých znaků.
- Je vybrán interval odpovídající prvnímu znaku.
- Ten je opět rozdělen na části odpovídající výskytu jednotlivých znaků.
- …
- Odešleme část horní hranice, která začíná nejvyšším bitem hranice (reprezentujícím 2^{-1}) a končí prvním bitem, ve kterém se dolní hranice liší od horní hranice

Aritmetická komprese

1. Znak **R**

$$D_1 = D_0 + \frac{m_R}{n} \cdot L_0 = 0 + \frac{2}{10} \cdot 1 = \frac{2}{10}$$

$$H_1 = D_1 + \frac{n_R}{n} \cdot L_0 = \frac{2}{10} + \frac{4}{10} \cdot 1 = \frac{6}{10}$$

$$L_1 = H_1 - D_1 = \frac{6}{10} - \frac{2}{10} = \frac{4}{10}$$

2. Znak **C**

$$D_2 = D_1 + \frac{m_C}{n} \cdot L_1 = \frac{2}{10} + \frac{0}{10} \cdot \frac{4}{10} = \frac{20}{100}$$

$$H_2 = D_2 + \frac{n_C}{n} \cdot L_1 = \frac{20}{100} + \frac{2}{10} \cdot \frac{4}{10} = \frac{28}{100}$$

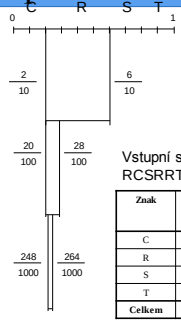
$$L_2 = H_2 - D_2 = \frac{28}{100} - \frac{20}{100} = \frac{8}{100}$$

3. Znak **S**

$$D_3 = D_2 + \frac{m_S}{n} \cdot L_2 = \frac{20}{100} + \frac{6}{10} \cdot \frac{8}{100} = \frac{248}{1000}$$

$$H_3 = D_3 + \frac{n_S}{n} \cdot L_2 = \frac{248}{1000} + \frac{2}{10} \cdot \frac{8}{100} = \frac{264}{1000}$$

$$L_3 = H_3 - D_3 = \frac{264}{1000} - \frac{248}{1000} = \frac{16}{1000}$$



Vstupní soubor:
RCSRRTCSRT

Znak	Počet znaků n_{znak}	Počet menších m_{znak}
C	2	0
R	4	2
S	2	6
T	2	8
Celkem	**10**	

Vícenásobná komprese

- Primární komprese – výchozí,
  - vhodné jsou slovníkové algoritmy.
- Sekundární komprese – následná,
  - vhodný Huffmanův kód, aritmetická komprese.
- Další komprese nepřináší užitek.
- Kvalita komprese,
  - kompresní poměr (komprimovaný soubor oproti původnímu).