bzip2. Kodowanie uniwersalne

Kodowanie i kompresja danych - Wykład 5

Maciek Gębala

25 marca 2020

Notatki

bzip2

- Algorytm kompresji popularny w systemach Linux-owych.
- Standardowa implementacja kompresuje bloki danych o rozmiarach do 100 do 900 kb.
- Blok jest przekształcany transformatą Burrowsa-Wheelera, następnie kodowany algorytmem Move-To-Front i kompresowany algorytmem Huffmana.
- Algorytm osiąga o 10%-20% lepsze wyniki niż najczęściej stosowane algorytmy strumieniowe ale nie może być używany do kodowania strumieni.

Notatki

Transformata Burrowsa-Wheelera

Na początku mamy blok danych o rozmiarze N, np.

a	l	a	-	m	a	-	k	o	t	a

Generujemy wszystkie N rotacji kompresowanego bloku

0	a	l	a	-	m	a	-	k	o	t	a
1	l	a	-	m	a	-	k	o	t	a	a
2	a	-	m	a	-	k	o	t	a	a	l
3	-	m	a	-	k	o	t	a	a	l	a
4	m	a	-	k	o	t	a	a	l	a	-
5	a	-	k	o	t	a	a	l	a	-	m
6	-	k	o	t	a	a	l	a	-	m	a
7	k	o	t	a	a	l	a	-	m	a	-
8	o	t	a	a	l	a	-	m	a	-	k
9	t	a	a	l	a	-	m	a	-	k	o
10	a	a	l	a	-	m	a	-	k	o	t

Notatki

Transformata Burrowsa-Wheelera

Sortujemy powstałe łańcuchy leksykograficznie

0	-	k	o	t	a	a	l	a	-	m	a
1	-	m	a	-	k	o	t	a	a	l	a
2	a	-	k	o	t	a	a	l	a	-	m
3	a	-	m	a	-	k	o	t	a	a	l
4	a	a	l	a	-	m	a	-	k	o	t
5	a	l	a	-	m	a	-	k	o	t	a
6	k	o	t	a	a	l	a	-	m	a	-
7	l	a	-	m	a	-	k	o	t	a	a
8	m	a	-	k	o	t	a	a	l	a	-
9	o	t	a	a	l	a	-	m	a	-	k
10	t	a	a	l	a	-	m	a	-	k	o

Wynik transformacji: numer wiersza z pierwotnym tekstem i ostatnia kolumna tabeli.

5	a	a	m	l	t	a	-	a	-	k	o

Notatki

Odwracanie transformacji

Mamy numer wiersza 5 i tekst

a	a	m	l	t	a	-	a	-	k	o

Dopisujemy do tekstu indeksy

0	1	2	3	4	5	6	7	8	9	10
a	a	m	l	t	a	-	a	-	k	o
0	1	2	3	4	5	6	7	8	9	10

Sortujemy tekst (stabilnie)

0	1	2	3	4	5	6	7	8	9	10
-	-	a	a	a	a	k	l	m	o	t
6	8	0	1	5	7	9	3	2	10	4

Zaczynając od pozycji 5 wypisujemy kolejno znaki zgodnie z indeksami $5 \to 7 \to 3 \to 1 \to 8 \to 2 \to 0 \to 6 \to 9 \to 10 \to 4$ i otrzymujemy

a	l	a	-	m	a	-	k	o	t	a

Notatki

Move-To-Front

- Transformacja strumienia danych mogąca spowodować zmniejszenie entropii.
- Zyski gdy symbole wykazują tendencję do lokalnego grupowania się.
- Dane o takiej charakterystyce są często wynikiem transformaty Burrowsa-Wheelera.

Transformacja

- Na początku mamy tablicę N posortowanych symboli. Kodem symbolu jest numer pozycji w tablicy (numery od 0 do $N-1$).
- W każdym kroku bierzemy pojedynczy symbol, wypisujemy jego numer z tablicy i modyfikujemy tablicę przesuwając rozpatrywany symbol na początek tablicy.
- Jeśli symbole są lokalnie pogrupowane to są przesuwane na początek tablicy i w kodzie wynikowym pojawiają się częściej małe liczby.
- Transformacja odwrotna jest praktycznie identyczna.

Notatki

Przykład transformacji

- a a m l t a - a - k o

0	1	2	3	4	5	6
-	a	k	l	m	o	t

- 1 a m l t a - a - k o

0	1	2	3	4	5	6
a	-	k	l	m	o	t

- 1 0 m l t a - a - k o

0	1	2	3	4	5	6
a	-	k	l	m	o	t

- 1 0 4 l t a - a - k o

0	1	2	3	4	5	6
m	a	-	k	l	o	t

- 1 0 4 4 t a - a - k o

0	1	2	3	4	5	6
l	m	a	-	k	o	t

- 1 0 4 4 6 a - a - k o

0	1	2	3	4	5	6
t	l	m	a	-	k	o

- 1 0 4 4 6 3 - a - k o

0	1	2	3	4	5	6
a	t	l	m	-	k	o

- 1 0 4 4 6 3 4 a - k o

0	1	2	3	4	5	6
-	a	t	l	m	k	o

- 1 0 4 4 6 3 4 1 - k o

0	1	2	3	4	5	6
a	-	t	l	m	k	o

- 1 0 4 4 6 3 4 1 1 k o

0	1	2	3	4	5	6
-	a	t	l	m	k	o

- 1 0 4 4 6 3 4 1 1 5 o

0	1	2	3	4	5	6
k	-	a	t	l	m	o

- 1 0 4 4 6 3 4 1 1 5 6

0	1	2	3	4	5	6
o	k	-	a	t	l	m

Notatki

Przykład transformacji

- Dla ciągu a a m l t a - a - k o entropia wynosi 2, 550.
- Dla ciągu 1 0 4 4 6 3 4 1 1 5 6 entropia wynosi 2, 413.
- Dla długich ciągów zyski mogą być dużo większe i kodowanie Huffmana efektywniejsze.

Notatki

Kodowanie uniwersalne - Definicja problemu

Jak kodować liczby naturalne?

Szukamy sposobu kodowania dowolnej liczby naturalnej (całkowitej, większej od zera) za pomocą kodu o zmiennej długości. Ilość wykorzystanych bitów powinna być proporcjonalna do kodowanej wartości (asymptotycznie $\Theta(\log_2 n)$).

Zastosowanie

Czasami konieczne jest przesłanie kilku liczb, których wielkość ciężko oszacować (wykorzystywane np. przez niektóre odmiany LZ77 do kodowania przesunięć).

Notatki

Kodowanie Eliasa

Kodowanie Eliasa (lata '70)

Wyróżniamy trzy rodzaje kodów Eliasa (co najmniej trzy), różniące się konstrukcją: γ, δ, ω. Ogólna idea jest następująca:

- kodujemy zadaną liczbę binarnie;
- „doklejamy" informację o długości tak, by powstał kod jednoznacznie dekodowalny.

Notatki

Kodowanie γ

Kodowanie

- kodowana liczba x jest zapisywana w kodzie binarnym,
- niech n oznacza liczbę cyfr w zapisie binarnym,
- słowo składa się z:
  1. $(n-1)$ bitów o wartości 0,
  2. liczby x zapisanej binarnie.

Przykład

Niech $x = 137_{10}$. Zatem $x = 10001001_2$, $n = 8$ a wynikowy kod to:

$$\underbrace{0000000}_{7\times 0}\underbrace{10001001}_{x \text{ binarnie}}.$$

Słowo kodowe ma w tym przypadku 15 bitów.

Notatki

Kodowanie δ

Kodowanie

- liczba x jest zapisywana w kodzie binarnym,
- niech n oznacza liczbę bitów binarnej reprezentacji x,
- liczbę n przedstawiamy w kodzie γ,
- z binarnej reprezentacji liczby x usuwamy najbardziej znaczącą cyfrę,
- niech k oznacza liczbę bitów n,
- wynikowym kodem jest:
  - $k-1$ zer,
  - binarna reprezentacja liczby n,
  - binarna reprezentacja liczby x bez pierwszej jedynki.

Liczba bitów reprezentacji

$$\underbrace{2\lceil \log_2(\lceil \log_2 x \rceil)\rceil - 1}_{n \text{ i } k-1} + \underbrace{\lceil \log_2 x \rceil - 1}_{x}.$$

Notatki

Przykład kodowania δ

Niech $x = 137_{10}$. Zatem $x = 10001001_2$, $n = 8 = 1000_2$ a $k = 4$. Wynikowy kod (niech $x_{(-1)}$ to x po usunięciu najbardziej znaczącej jedynki) to:

$$\underbrace{000}_{3\times 0}\,\underbrace{1000}_{n}\,\underbrace{0001001}_{x_{(-1)}}.$$

Notatki

Kodowanie ω

Kodowanie

Algorytm wygląda następująco (kolejne podsłowa binarne dołączamy na początek powstającego kodu):

1. zapisz bit 0 (znacznik końca),
2. $k \leftarrow x$,
3. while $k > 1$:
   1. zapisz dwójkowo liczbę k,
   2. nowe k, to liczba bitów k z poprzedniego kroku, pomniejszona o 1.

Liczba bitów kodu wynosi (w każdym kroku zapisujemy $k_i = \lceil \log_2 k_{i-1} \rceil$ bitów): $1 + \sum_{i=1}^{n} k_i$, gdzie n jest liczbą kroków, czyli

$$\mathcal{O}\left(\sum_{i=1}^{\log^* x} \log^{(i)} x\right)$$

Notatki

Dekodowanie ω

Dekodowanie

1. $n \leftarrow 1$,
2. while „kolejny bit nie ma wartości 0”:
   1. $n \leftarrow$ wartość zapisana na kolejnych $n+1$ bitach,
3. $x \leftarrow n$.

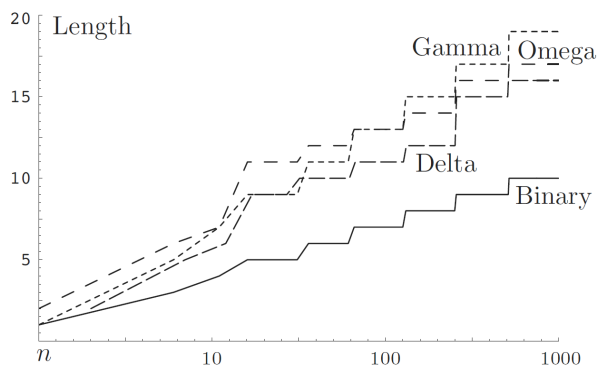
Przykład kodowania

Niech $x = 137_{10}$.

krok	k	słowo binarne	l. bitów
0		0	1
1	137	10001001	8
2	7	111	3
3	2	10	2
4	1		
Kod końcowy		10 111 10001001 0	14

Notatki

Porównanie kodowań: γ, δ i ω



Notatki

Kody Fibonacciego

Liczby Fibonacciego

Liczby postaci: $f_0 = f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$ dla $n \geqslant 2$.

Jedna z własności liczb Fibonacciego

Każda liczba całkowita dodatnia k może być przedstawiona jednoznacznie jako: $k = \sum_{i=1}^{\infty} a_i f_i$, gdzie a_i jest równe 0 lub 1.

Obserwacja

Zawsze możemy zapisać ten ciąg w taki sposób, że nie wystąpią obok siebie dwie jedynki!

Algorytm kodujący

- znajdujemy współczynniki a_i,
- zapisujemy w kolejności od najmniej znaczącego,
- na końcu dopisujemy jedynkę (na końcu zawsze wystąpią dwie jedynki!).

Notatki

Kodowanie Fibonacciego

Przykładowe liczby

liczba	kod binarny		liczba	kod binarny
1	11		7	01011
2	011		8	000011
3	0011		9	100011
4	1011		10	010011
5	00011		11	001011
6	10011		12	101011

Algorytm dekodowania i oszacowanie długości kodu – do przemyślenia.

Notatki

Notatki

Notatki