LZW

Lempel-Ziv-Welch, **LZW** – metoda strumieniowej bezstratnej kompresji słownikowej, będąca modyfikacją metody LZ78.

Pomysłodawcą algorytmu jest Terry A. Welch. Metodę opisał w 1984 roku, w artykule *A technique for high-performance data compression* opublikowanym w numerze 6. *Computer* (str. 8-19).

Metoda LZW jest względnie łatwa do zaprogramowania, daje bardzo dobre rezultaty. Wykorzystywana jest m.in. w programach ARC, PAK i UNIX-owym compress, w formacie zapisu grafiki GIF, w formatach PDF i PostScript (filtry kodujące fragmenty dokumentu) oraz w modemach (V.42bis). LZW było przez pewien czas algorytmem objętym patentem, co było przyczyną podjęcia prac nad nowym algorytmem kompresji obrazów, które zaowocowały powstaniem formatu PNG.

LZW - to także rozszerzenie do programu **LHA** i algorytmu bezstratnej kompresji danych stworzony przez Haruyasu Yoshizakiego. Inne rozszerzenia: .LHW .LZH .LZS

Spis treści

Zmiany w stosunku do LZ78

W pojedynczym kroku kompresji LZ78 wyszukiwane jest w słowniku najdłuższe słowo będące prefiksem niezakodowanych jeszcze danych. Na wyjście wypisywany jest indeks tego słowa oraz pierwszy symbol z wejścia znajdujący się za dopasowaniem. Np. jeśli w słowniku pod indeksem 2 zapisane jest słowo "wiki", a kodowany jest ciąg "wikipedia", na wyjście zostanie wypisana para (2, 'p'); do zakodowania pozostanie jeszcze ciąg "edia". Jeśliby nie udało się zlokalizować niczego w słowniku, na wyjście wypisywana jest para (0, pierwsza litera) – oznacza to, że w słowniku nie ma jeszcze słowa jednoliterowego równego tej pierwszej literze.

Przewaga LZW nad LZ78 to krótsze wyjście kodera – wypisywany jest wyłącznie indeks słowa. Uzyskano to dzięki pierwszemu etapowi algorytmu, tj. wstępnemu wypełnieniu słownika alfabetem (wszystkimi symbolami, jakie mogą pojawić się w danych), gwarantując w ten sposób, że zawsze uda się znaleźć dopasowanie, przynajmniej jednoliterowe.

Algorytm kompresji (kodowania)

W pojedynczym kroku algorytmu wyszukiwany jest w słowniku najdłuższy prefiks niezakodowanych jeszcze danych. Na wyjście wypisywany jest wówczas kod związany z tym słowem, zaś do słownika dodawana nowa pozycja: konkatenacja słowa i pierwszej niedopasowanej litery.

Algorytm przebiega następująco:

1. Wypełnij słownik alfabetem źródła informacji.
2. **c** := *pierwszy symbol wejściowy*
3. Dopóki są dane na wejściu:
   - Wczytaj znak **s**.
   - Jeżeli ciąg **c + s** znajduje się w słowniku, przedłuż ciąg **c**, tj. **c** := **c** + **s**
   - Jeśli ciągu **c + s** nie ma w słowniku, wówczas:
     - wypisz kod dla **c** (**c** znajduje się w słowniku)
     - dodaj ciąg **c + s** do słownika
     - przypisz **c** := **s**.
4. Na końcu wypisz na wyjście kod związany **c**.

O efektywności kompresji w dość dużym stopniu decyduje sposób zapisu kodów (liczb).

Algorytm dekompresji

Algorytm dekompresji jest nieco bardziej złożony, bowiem dekoder musi wykryć przypadki ciągów **scscs** (które nie znajdują się w słowniku), gdzie ciąg **sc** jest już w słowniku, a podciąg **c** jest dowolny, być może także pusty.

1. Wypełnij słownik alfabetem źródła informacji.
2. **pk** := *pierwszy kod skompresowanych danych*
3. Wypisz na wyjście ciąg związany z kodem **pk**, tj. słownik[**pk**]
4. Dopóki są jeszcze jakieś słowa kodu:
   - Wczytaj kod **k**
   - **pc** := słownik[**pk**] – ciąg skojarzony z poprzednim kodem
   - Jeśli słowo **k** jest w słowniku, dodaj do słownika ciąg (**pc** + pierwszy symbol ciągu słownik[**k**]), a na wyjście wypisz cały ciąg słownik[**k**].
   - W przeciwnym razie (przypadek **scscs**) dodaj do słownika ciąg (**pc** + pierwszy symbol **pc**) i tenże ciąg wypisz na wyjście.
   - **pk** := **k**

Modyfikacje algorytmu

- LZMW (V. Miller, M. Wegman, 1985) – zamiast dodawać do słownika słowa przedłużone o jedną literę, dodaje się połączenie poprzedniego i bieżącego słowa. Tzn. jeśli we wcześniejszej iteracji np. dopasowano słowo "wiki", natomiast w bieżącej znaleziono w słowniku prefiks "pedia", od razu dodawane jest słowo "wikipedia". W klasycznym LZW najprawdopodobniej (zależy to od danych) w kilku krokach algorytmu dodane do słownika zostałyby "wikip", następnie "wikipe", itd.

- **LZAP** (James Storer, 1988) – modyfikacja **LZMW**, w której oprócz konkatenacji poprzedniego i bieżącego słowa dodaje się konkatenację wszystkich prefiksów bieżącego słowa (skrót **AP** pochodzi od *all prefixes* – wszystkie prefiksy). Czyli dla wcześniejszego przykładu zostaną dodane oprócz "wikipedia" także "wikip", "wikipe", "wikiped" oraz "wikipedi". To powoduje szybsze powiększanie słownika, nawet takimi słowami, które mogą nigdy nie pojawić się w kodowanych danych.

Przykład kompresji

Zostanie zakodowany ciąg składający się z 24 znaków: "abccd_abccd_acd_acd_acd_".

poprzedni ciąg (c)	bieżący symbol (s)	c + s	indeks	słownik	komentarz
				1. a 2. b 3. c 4. d 5. _	inicjowanie słownika alfabetem
a	b	ab	1 – indeks 'a'	6. ab	do słownika dodawany jest ciąg 'ab', c = 'b'
b	c	bc	2 – indeks 'b'	7. bc	do słownika dodawany jest ciąg 'bc', c = 'c'
c	c	cc	3 – indeks 'c'	8. cc	do słownika dodawany jest ciąg 'cc', c = 'c'
c	d	cd	3 – indeks 'c'	9. cd	do słownika dodawany jest ciąg 'cd', c = 'd'
d	_	d_	4 – indeks 'd'	10. d_	do słownika dodawany jest ciąg 'd_', c = '_'
_	a	_a	5 – indeks '_'	11. _a	do słownika dodawany jest ciąg '_a', c = 'a'
a	b	ab			'ab' istnieje w słowniku
ab	c	abc	6 – indeks 'ab'	12. abc	do słownika dodawany jest ciąg 'abc', c = 'c'
c	c	cc			'cc' istnieje w słowniku
cc	d	ccd	8 – indeks 'cc'	13. ccd	do słownika dodawany jest ciąg 'ccd', c = 'd'
d	_	d_			'd_' istnieje w słowniku
d_	a	d_a	10 – indeks 'd_'	14. d_a	do słownika dodawany jest ciąg 'd_a', c = 'a'
a	c	ac	1 – indeks 'a'	15. ac	do słownika dodawany jest ciąg 'ac', c = 'c'
c	d	cd			'cd' istnieje w słowniku
cd	_	cd_	9 – indeks 'cd'	16. cd_	do słownika dodawany jest ciąg 'cd_', c = '_'
_	a	_a			'_a' istnieje w słowniku
_a	c	_ac	11 – indeks '_a'	17. _ac	do słownika dodawany jest ciąg '_ac', c = 'c'
c	d	cd			'cd' istnieje w słowniku
cd	_	cd_			'cd_' istnieje w słowniku
cd_	a	cd_a	16 – indeks 'cd_'	18. cd_a	do słownika dodawany jest ciąg 'cd_a', c = 'a'
a	c	ac			'ac' istnieje w słowniku
ac	d	acd	15 – indeks 'ac'	19. acd	do słownika dodawany jest ciąg 'acd', c = 'd'
d	_	d_	10 – indeks		koniec kodowania

			'd_'		

Zakodowane dane składają się z 15 indeksów: 1, 2, 3, 3, 4, 5, 6, 8, 10, 1, 9, 11, 16, 15, 10.

Jeśli przyjąć, że indeksy oraz symbole są zapisane na tej samej liczbie bitów, to współczynnik kompresji wyniesie ok. 37%. Jeśli natomiast przyjąć minimalną liczbę bitów potrzebną do zapisania danych, tj. 3 bity na symbol (w sumie 72 bity), 4 na indeks (w sumie 60 bitów), współczynnik kompresji wyniesie ok. 15%.

Bibliografia

- Adam Drozdek: *Wprowadzenie do kompresji danych*. Warszawa: Wydawnictwa Naukowo-Techniczne, 1999, s. 83-88. ISBN 83-204-2303-1.
- Artur Przelaskowski: *Kompresja danych: podstawy, metody bezstratne, kodery obrazów*. Warszawa: BTC, 2005, s. 164. ISBN 83-60233-05-5.

Zobacz też

- kod programu kompresującego i dekompresującego metodą LZW
- LZC – praktyczna implementacja LZW wykorzystywana w programie compress
- LZ77
- LZ78

Linki zewnętrzne

- http://www.cs.duke.edu/courses/spring03/cps296.5/papers/welch_1984_technique_for.pdf – skan artykułu Terry'ego A. Welcha (ostatni dostęp 14.09.2008)