

Zadanie 1 z listy 6 - “Kompresja Danych”

Łukasz Klasieński

11 maja 2020

Zadanie 1

Pokaż, że optymalna gramatyka dla słowa

$$w_k = a^{k(k+1)/2} (ba^k)^{(k+1)^2}$$

ma wielkość $O(\log k)$, a kodowanie LZ78 daje w wyniku kod o rozmiarze (liczbie elementów) $\Omega(k^2)$

Rozwiązanie

Zobaczmy najpierw jak wygląda konstrukcja gramatyki dla w_k , która ma wielkość $O(\log k)$. Najpierw mamy produkcję słowa startowego: $S \rightarrow LR$ Gdzie L będzie prawymi a , natomiast R prawymi.

Jak wygląda tworzenie słowa złożonego z a^k elementów: Rozkładamy wpierw k na postać binarną $k = 2^{i_1} + 2^{i_2} + 2^{i_3} \dots$ Wtedy produkcja a^k wygląda następująco:

$$A_K \rightarrow A_{i_1} A_{i_2} A_{i_3} \dots$$

Natomiast kolejne produkcje A_j tworzymy następująco:

$$A_j \rightarrow A_{j/2} A_{j/2}$$

$$A_1 \rightarrow a$$

Otrzymujemy wtedy $\lceil \log k \rceil = O(\log k)$ produkcji.

Teraz chcemy znaleźć produkcję tworzącą słowo składające się z $(k+1)/2$ elementów x . Robimy to tak samo jak wcześniej - szukamy rozkładu binarnego i robimy dodatkowe produkcje:

$$A_{\frac{K+1}{2}} \rightarrow A_{\frac{K+1}{2}i} A_{\frac{K+1}{2}i+1} \dots$$

$$A_{\frac{K+1}{2}1} \rightarrow x$$

Możemy teraz stworzyć produkcję konstruującą słowo $a^{k(k+1)/2}$ podstawiając pod x z poprzedniej produkcji A_K

$$L \rightarrow A_{\frac{K+1}{2}}$$

Mamy już zatem lewe wyrażenie szukanego słowa. Zostało stworzenie produkcji dla $(ba^k)^{(k+1)^2}$. Zauważmy, że

$$(k+1)^2 = k^2 + 2k + 1$$

Zatem wystarczy znaleźć produkcje dające kolejne długości tej sumy. Jedynek mamy za darmo bo $(ba^k)^1 = bA_K$ $(ba^k)^{2k} = 2 \times \text{prod}(ba^k)^k$ natomiast $k^2 = k \cdot k = k \times \text{prod}(ba^k)^k$ Wystarczy zatem zrobić produkcję, które aplikuje elementy k razy oraz produkcję $(ba^k)^k$ Najpierw $(ba^k)^k$ - podobnie jak wcześniej korzystamy z rozkładu k i robimy produkcję

$$BA_K \rightarrow BA_{i_1} BA_{i_2} \dots$$

$$BA_j \rightarrow BA_{j/2} BA_{j/2}$$

$$BA_1 \rightarrow bA_K$$

Oraz identycznie produkcję która zaaplikuje produkcję BA_K - k razy:

$$K_K \rightarrow K_K i_1 K_K i_2 \dots$$

$$K_1 \rightarrow BA_K$$

Ostatecznie mamy produkcję dla prawej strony:

$$R \rightarrow bA_K BA_K BA_K K_K$$

Wystarczy teraz zlepić całość w

$$S \rightarrow LR$$

Jaką mamy wielkość gramatyki - $4 \cdot O(\log k) = O(\log k)$

Dlaczego to jest optymalna wielkość gramatyki? Wiemy z wykładu, że gramatyka konstruowana przez algorytm LZ77 jest większa od optymalnej gramatyki o jakąś stałą $\leq \log n$ gdzie $n = |w|$. Algorytm na początku dopasuje długi ciąg złożony z samych a i zapisze to jako jedną krotkę. Następnie zmatchuje ba^k jednokrotnie, następnie dwukrotnie, czterokrotnie - łącznie $\log(k+1)^2 = 2\log(k+1)$ razy ostatecznie mamy zatem kod długości $2\log(k+1) \cdot \log(|w|) = O(\log k) \cdot \log(|w|)$. Zatem $O(\log k)$ musi być optimum (na mocy tw. z wykładu).

Jak wygląda kodowanie LZ78 - zakładamy nieograniczony słownik. Wpierw zobaczymy L . Na początku wstawi do pustego słownika a i wypłuje $(0, a)$. W następnym kroku zmatchuje a i doda do słownika aa (wypłuje $(1, a)$). I tak dalej - łącznie zamieni a^1 na krotkę, a^2 na krotkę $\dots a^k$ na krotkę, ponieważ $\sum_1^k = k(k+1)/2$.

Prawa część będzie wyglądać następująco - algorytm zobaczy b na początku wyrazu, zatem nie znajdzie tego w słowniku i doda to jako nową krotkę. Następnie zobaczy że a^k jest w słowniku i doda do słownika $a^k b$. W kolejnej iteracji dodamy $a^k b a$. Potem $a^{k-1} b$. Ogólnie otrzymamy w słowniku wszystkie możliwe kombinacje $a^i b a^j$, gdzie $i, j = 0..k$. Zatem ustawiamy i, j na k^2 sposobów. Na wyjściu dostaniemy więc co najmniej $O(k^2)$ krotek. Ostatecznie mamy $O(k) + O(k^2) \geq \Omega(k^2)$