Faktoryzacja Lyndona

Na podstawie "Factorizing Words over an Ordered Alphabet " – Duval

Krzysztof Pióro

Maj 2022

1 Wstęp

Słowo będziemy nazywać *prostym* (lub słowem Lyndona) jeśli jest ściśle mniejsze od wszystkich swoich nietrywialnych przesunięć cyklicznych.

Faktoryzacją Lyndona słowa w nazwiemy podział słowa $w = w_1 w_2 \dots w_k$ taki, że wszystkie słowa w_i są proste oraz zachodzi $w_1 \geqslant w_2 \geqslant \dots \geqslant w_k$. Taka faktoryzacja zawsze istnieje i jest unikalna.

Algorytm Duval'a konstruuje faktoryzację Lyndona w czasie liniowym i stałej dodatkowej pamięci.

2 Własności faktoryzacji Lyndona

Obserwacja 1. *Słowo s jest proste wtedy i tylko wtedy, gdy jest ściśle mniejsze niż wszystkie swoje nietrywialne sufiksy*

Obserwacja 2. *Dla faktoryzacji Lyndona $w = w_1 w_2 \dots w_k$ słowo w_k jest minimalnym sufiksem słowa w.*

Twierdzenie 1. *Dla każdego słowa w istnieje faktoryzacja Lyndona.*

Dowód. Pojedyncza litera jest słowem *prostym*, zatem możemy zacząć od podziału słowa w na pojedyncze literki. Łatwo zauważyć, że dla dwóch słów prostych u, v takich, że $u < v$ słowo uv również jest słowem prostym. Zatem dopóki będą istniały dwa sąsiednie słowa u, v w naszej faktoryzacji takie, że $u < v$ to możemy je łączyć w jedno słowo uv. Powtarzając tą procedurę otrzymamy faktoryzację Lyndona. □

Twierdzenie 2. *Dla każdego słowa w istnieje **unikalna** faktoryzacja Lyndona.*

Dowód. Z poprzedniego twierdzenia wiemy, że faktoryzacja Lyndona zawsze istnieje. Pozostało nam pokazać jej unikalność. Wykorzystamy do tego celu fakt, że ostatnie słowo z faktoryzacji Lyndona słowa w jest minimalnym sufiksem słowa w. Możemy zatem odciąć minimalny sufiks słowa w i wywołać się indukcyjnie na krótszym słowie. □

3 Algorytm Duval'a

Zacznijmy od wprowadzenia dodatkowej definicji. Słowo nazwiemy *prawie prostym* jeśli jest postaci $w = u^t\bar{u}$, gdzie $\bar{u}$ jest prefiksem słowa u (być może pustym).

Algorytm Duval'a będzie utrzymywał podział słowa wejściowego w na 3 słowa $w = v_1v_2v_3$ takie, że dla fragmentu v_1 faktoryzacja Lyndona jest już znana, a fragment v_2 jest słowem *prawie prostym*.

Algorytm będzie utrzymywał ten podział za pomocą dwóch zmiennych i, j. Zmienna i będzie wskazywała na początek słowa v_2, a zmienna j będzie wskazywała na początek słowa v_3. W każdym kroku algorytmu będziemy próbowali doczepić literę $w[j]$ to słowa v_2. W tym celu będziemy porównywali ją z literą słowa v_2 wyznaczoną przez zmienną k. Dokładniej będziemy mieli trzy przypadki:

- $w[j] = w[k]$: dodanie $w[j]$ do v_2 nie narusza założenia, że v_2 jest *prawie proste*. Także w tym przypadku po prostu zwiększamy zmienne j oraz k.
- $w[j] > w[k]$: słowo $v_2+w[j]$ staje się proste. W tym przypadku zwiększamy j i ustawiamy k na początek słowa v_2.
- $w[j] < w[k]$: słowo $v_2+w[j]$ przestaje być *prawie proste*. W tym przypadku dla naszego słowa $v_2 = u^t\bar{u}$ dzielimy u^t na t słów *prostych*, dodajemy je do v_1 (czyli wynikowej faktoryzacji Lyndona), ustawiamy zmienne i oraz k na początek pozostałej części słowa v_2 ($\bar{u}$), a zmienną j na jedną pozycję dalej.

Algorithm 1 Duval(w)

```
i := 1
factorization := empty list
while i ⩽ n do
    j := i + 1; k := i;
    while j ⩽ n and w[k] ⩽ w[j] do
        if w[k] < w[j] then
            k := i
        else
            k := k + 1;
        j := j + 1;
    while i ⩽ k do
        factorization append w[i ... i + j − k − 1]
        i := i + j − k;
return factorization
```

Dowód poprawności. Z powyższych przypadków możemy od razu wywnioskować, że algorytm Duval'a rozkłada słowo w na $w = w_1w_2 \dots w_k$ takie, że wszystkie w_i są słowami *prostymi*. Pozostało pokazać, że $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_k$. W tym

celu zastanówmy się co dzieje się w trakcie kroku z trzeciego przypadku. Niech $a := w[j]$. Zauważmy, że $\bar{u}av < u$ dla dowolnego słowa v. Zatem wszystkie prefiksy słowa v_2v_3, które otrzymamy po kroku z trzeciego przypadku, będą ściśle mniejsze od słowa u, czyli od ostatniego słowa z faktoryzacji słowa v_1. Z tego wynika, że słowa, które później będą dodane do faktoryzacji zachowają warunek monotoniczności. □

Dowód złożoności. Pierwsza wewnętrzna pętla w każdym kroku zwiększa zmienną j. Zmienna ta może być jednak zmniejszana w wyniku trzeciego przypadku. Zauważmy jednak, to o ile cofniemy tą zmienną wynosi co najwyżej tyle, jak długie jest poprzednio dodane słowo Lyndona. Dodatkowo za każdym razem kiedy cofamy zmienną j, ostatnie słowo w faktoryzacji jest inne, zatem możemy oszacować sumę cofnięć przez $O(n)$. Otrzymujemy więc, że ta pętla działa w czasie $O(n)$.

Druga wewnętrzna pętla w algorytmie działa sumarycznie w czasie $O(n)$, bo wypisuje faktoryzację.

Ponadto zewnętrzna pętla algorytmu nie przekroczy n iteracji, bo jej każda iteracja zwiększa zmienną i, co kończy dowód złożoności. □