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 Duvala konstruuje faktoryzację Lyndona w czasie liniowym i stałej dodatkowej pamięci.

2 Własności faktoryzacji Lyndona

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

Dowód. Załóżmy, że słowo w jest ściśle mniejsze niż wszystkie swoje nietrywialne sufiksy oraz, że nie jest *proste*. Wtedy mamy takie dwa słowa u, v, że $w = uv$, $vu \leqslant uv$ (vu jest świadkiem dla faktu, że w nie jest *proste*) oraz $uv < v$ (uv jest ściśle mniejsze niż wszystkie inne sufiksy). Ale z $uv < v$ możemy wywnioskować $uv < vu$, czyli mamy sprzeczność.

W drugą stronę załóżmy, że słowo w jest *proste* oraz, że nie jest ściśle mniejsze od wszystkich swoich nietrywialnych sufiksów. Wtedy mamy takie słowa u, v, że $w = uv$, $uv < vu$ (w jest słowem *prostym*) oraz $v \leqslant uv$. Rozważmy dwa przypadki:

- v nie jest prefiksem uv – wtedy $vu < uv$, czyli sprzeczność
- v jest prefiksem uv – tutaj zauważamy, że u jest najmniejsze leksykograficznie spośród prefiksów długości $|u|$ przesunięć cyklicznych słowa w. Zatem $vu \leqslant uv$, czyli sprzeczność.

□

Obserwacja 2. *Słowa proste nie posiadają właściwych prefikso-sufiksów.*

Obserwacja 3. *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 Duvala

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).

Dodatkowo udowodnimy teraz, że słowo *prawie proste* ma tylko jeden okres u, który jest słowem *prostym*. W przeciwnym wypadku mielibyśmy dwa *proste* okresy u_1, u_2. Załóżmy, że $|u_1| < |u_2|$. Wtedy u_1 jest okresem u_2. Ale z tego wynikałoby, że u_2 ma właściwy prefikso-sufiks, co jest sprzeczne z tym, że u_2 jest słowem *prostym*.

Algorytm Duvala będzie utrzymywał podział słowa wejściowego w na 3 słowa $w = v_1 v_2 v_3$ 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 $w[j]$ do słowa v_2. W tym celu będziemy porównywali ją z literą słowa v_2 wyznaczoną przez zmienną k (taką, że $j - k$ to długość słowa u występującego w *prawie prostym* słowie $v_2 = u^t \bar{u}$). Dokładniej będziemy mieli trzy przypadki:

- $w[j] = w[k]$: dodanie $w[j]$ do v_2 nie narusza założenia, że v_2 jest *prawie proste*.

  W tym przypadku zwiększamy zmienne j oraz k.

- $w[j] > w[k]$: słowo $v_2 + w[j]$ staje się proste. Aby to udowodnić zauważmy, że minimalny sufiks słowa $u^t \bar{u} w[j]$ musi zacząć się w pierwszym słowie u.

W przeciwnym przypadku moglibyśmy rozszerczyć go o długość $|u|$ i z faktu, że $w[j] > w[k]$ otrzymalibyśmy, że ten dłuższy sufiks jest mniejszy. Wiemy zatem, że minimalny sufiks zaczyna się w pierwszym fragmencie u, natomiast z faktu, że u jest proste otrzymujemy, że minimalny sufiks musi zacząć się od pierwszej litery u, czyli 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*. Aby to udowodnić załóżmy, że słowo $v_2 + w[j]$ pozostaje *prawie proste*. Wtedy $v_2 + w[j] = s^t\bar{s}$ dla jakiegoś słowa prostego s. Oczywiście $s \neq u$. Ponadto s jest okresem słowa v_2, co daje nam sprzeczność z tym, że *prawie proste* słowo ma tylko jeden *prosty* okres.

  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 Duvala rozkłada słowo w na $w = w_1 w_2 \dots w_k$ 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 $v_2 v_3$, 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. □

Złożoność algorytmu. Pierwsza wewnętrzna pętla w każdym kroku zwiększa zmienną j. Zmienna ta może być jednak zmniejszana w wyniku trzeciego przypadku. Zauważmy jednak że, 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 pokazuje, że cały algorytm działa w czasie $O(n)$.

□