Proste wyznaczenie maksymalnego sufiksu słowa

Na podstawie "A note on a simple computation of the maximal suffix of a string" – Adamczyk, Rytter

Krzysztof Pióro

Maj 2022

1 Wstęp

Autorzy pracy pokazują prosty w opisie algorytm wyznaczający maksymalny sufiks słowa działający w czasie liniowym i stałej dodatkowej pamięci.

2 Definicje

Będziemy rozważać słowo w długości n indeksowane od 1 do n. Dodatkowo definiujemy:

- MaxSuf(w) – maksymalny sufiks słowa w
- period(w) – najkrótszy okres słowa w
- słowo nazywamy *samo-maksymalnym* $\iff$ MaxSuf(w) $= w$.
- MaxSufPos(w) – pozycja od której zaczyna się maksymalny sufiks słowa w
- $w_1 < w_2$ – w_1 jest mniejsze leksykograficznie od w_2

3 Algorytm

Algorithm Compute-MaxSufPos(w)

```
i := 1; j := 2;
while j ⩽ n do
    k := 0
    while j + k < n and w[i + k] = w[j + k] do
        k := k + 1;
    if w[i + k] < w[j + k] then
        i := i + k + 1;
    else
        j := j + k + 1;
    if i = j then
        j := j + 1;
return i
```

Algorytm oczywiście działa w stałej (dodatkowej) pamięci i w liniowej liczbie operacji.

4 Dowód poprawności

Niech $(i, j) \rightarrow (i', j')$ oznacza, że z konfiguracji (i, j) w jednej iteracji przechodzimy do (i', j') i niech $\rightarrow^*$ będzie domknięciem przechodnim relacji $\rightarrow$.

Pokażemy, że po każdej głównej iteracji będą zachodziły poniższe **niezmienniki** (gdzie oznaczamy $u := w[i \dots j-1]$):

1. $(i < j < n) \Rightarrow u$ jest *samo-maksymalne* oraz $period(u) = u$
2. maksymalny sufiks słowa w nie zaczyna się przed i

Dowód niezmienników. Początkowo $i = 1, j = 2$ i niezmiennik jest spełniony

Rozważmy iterację, w której i jest zmienione po raz pierwszy. Pokażemy, że niezmienniki są spełnione przed tą iteracją (autorzy pracy piszą tutaj tylko, że jest to "easy to see", ale my to uzasadnimy). Oczywiście drugi niezmiennik jest spełniony, zatem skupimy się na pierwszym.

Rozważamy zatem przejście $(i = 1, j) \rightarrow (i = 1, j' = j + k + 1)$, gdzie niezmienniki zachodzą dla słowa $u = w[i \dots j-1]$.

Wtedy $u' := w[i \dots j+k] = u^t va$, gdzie v jest ścisłym prefiksem u (być może pustym), a a jest literką taką, że $va < u$.

Z *samo-maksymalności* słowa u możemy wywnioskować, że słowo u' również jest *samo-maksymalne*.

Aby udowodnić, że $period(u') = u'$ pokażemy, że u' nie ma żadneo właściwego prefikso-sufiksu. Każdy prefikso-sufiks u' musi być przedłużeniem prefikso-sufiksu $w[i \dots j+k-1]$. Najdłuższy prefikso-sufiks słowa $w[i \dots j+k-1]$ to prefiks

$p = w[i \dots i+k-1]$ (odpowiada mu sufiks $s = w[j \dots j+k-1]$). Z działania algorytmu wynika, że po prefiksie p występuje litera b taka, że $b > a$. Zauważmy dodatkowo, że wszystkie literki występujące po prefikso-sufiksach słowa p są większe bądź równe b (wpp. mielibyśmy sprzeczność z *samo-maksymalnością u)*. Zatem nie istnieje prefikso-sufiks słowa $w[i \dots j+k-1]$, którego dałoby się rozszerzyć do prefikso-sufiksu słowa u', czyli $period(u') = u'$.

Pierwsza modyfikacja zmiennej i: Rozpatrzmy zatem pierwszą zmianę zmiennej i z $i = 0$ na $i' = i + k + 1$.

Wtedy $w[i \dots j+k] = u^t vb$, gdzie v jest ścisłym prefiksem u oraz $u < vb$. Niech m będzie pierwszą pozycją za u^t. Pokażemy, że (częściowa) historia algorytmu wygląda jak poniżej:

$$(i, j) \to (i', j) \to^* (m, j) \to^(m, m+1)$$

Z niezmiennika dla słowa u możemy wywnioskować, że u jest najmniejszym okresem słowa $u^t v$ (u oczywiście jest okresem, a gdyby istniał jakiś mniejszy, to byłby on też okresem u) oraz to, że słowo $u^t v$ jest *samo-maksymalne*.

Zauważmy, że jeśli zmienna i zostanie przeniesiona na dowolną pozycję z zakresu $[i', m-1]$, to następna pozycja dla i nie może być większa niż m. Wynika to z tego, że w takim przypadku znalezlibyśmy prefikso-sufiks słowa u, co byłoby sprzeczne z $period(u) = u$. Z *samo-maksymalności* słowa u możemy dodatkowo wywnioskować, że dla takiej zmiennej i następna iteracja zwiększy tą zmienną i. Zatem wartość zmiennej i będzie zwiększana z i', aż osiągnie m.

Następnie zwiększana będzie wartość zmiennej j, aż do momentu, gdy osiągnie ona wartość $m + 1$ (możemy to uzasadnić podobnymi argmentami jak dla zmiennej i).

Prześliśmy więc z (i, j) do $(m, m+1)$. Z faktu, że $u < vb$, wiemy, że maksymalny sufiks nie może zacząć się przed pozycją m. Zatem możemy odciąć prefiks u^t słowa w i rozpocząć całe obliczenia od nowa dla mniejszego słowa(traktujemy m jako 1). Ostatecznie dowód poprawności niezmiennika otrzymujemy z poprawności niezmiennika dla mniejszych słów. □

Dowód poprawności algorytmu. Rozważmy ostatnią wartość zmiennej i oraz przedostatnią wartość zmiennej j. Zgodnie z niezmiennikiem otrzymujemy, że słowo $u = w[i..j-1]$ jest *samo-maksymalne*. Dodatkowo sufiks $w[i \dots n]$ jest postaci $u^t va$, gdzie v jest prefiksem u, a a jest taką literką, że $va \leqslant u$. Z tej postaci sufiksu możemy już łatwo wywnioskować, że $w[i \dots n]$ jest maksymalnym sufiksem, co kończy dowód poprawności algorytmu.

Warto tutaj zwrócić uwagę, że autorzy błędnie stwierdzili, że u jest okresem sufiksu $w[i \dots n]$. Możemy mieć bowiem taką ostatnią literkę a, że $va < u'$, gdzie u' to prefiks u długości $|va|$ (ostatnie porównanie w algorytmie zwraca, że ostatnia litera jest mniejsza od tej, z którą ją porównywaliśmy). □