Wstęp do Programowania potok funkcyjny

Marcin Kubica

2018/2019

Outline

Czym jest tablica sufiksowa

Definition (Tablica sufiksowa)

- **Dane:** Napis x złożony z n liter.
  Zakładamy, że alfabet jest rozmiaru $O(n)$.
- Rozpatrujemy wszystkie sufiksy y.
  Chcemy je uporządkować leksykograficznie.
- **Wynik:**
  - tablica SUF zawierająca początkowe pozycje sufiksów w porządku leksykograficznym.
  - Tablica Rank odwrotna do tablicy SUF.
  - Tablica LCP określająca na ilu znakach są zgodne dwa kolejne sufiksy w SUF.

Czym jest tablica sufiksowa

Example

$x =$ yabbadabbado

i	0	1	2	3	4	5	6	7	8	9	10	11	12
$x.(i)$	y	a	b	b	a	d	a	b	b	a	d	o	$\emptyset$
Rank.(i)	12	1	7	5	3	9	2	8	6	4	10	11	0
SUF.(i)	12	1	6	4	9	3	8	2	7	5	10	11	0
LCP.(i)	0	5	1	2	0	3	1	4	0	1	0	0	

Sortowanie pozycyjne

- Sortowanie kubełkowe $O(n)$.
- Sortowanie stabilne.
- Sortowanie pozycyjne krotek.
- Sortowanie pozycyjne słów.

Algorytm Sandersa-Kärkkäinena-Burkhardta

- Dzielimy pozycje w tekście ze względu na reszy $\mod 3$.
  $B_k = \{i \in \{0, \ldots, n\} : i \mod 3 = k\}$
- Ograniczymy się do obliczenia tablic SUF i Rank dla sufiksów zaczynających się na pozycjach $B_1 \cup B_2$.
  Potem rozszerzymy wyniki o sufiksy zaczynające się na pozycjach B_0.
- r_k to napis zredukowany do kolejnych trójek znaków zaczynających się w B_k.
  $r_k = [|(x.(k), x.(k+1), x.(k+2)); (x.(k+3), x.(k+4), x.(k+5)); \ldots |]$
  Ostatnią trójkę uzupełniamy znakiem $\emptyset$ (obowiązkowo).

Algorytm Sandersa-Kärkkäinena-Burkhardta

- Na przykład, dla
  $x = [|'y';' a';' b';' b';' a';' d';' a';' b';' b';' a';' d';' o'|]$:
  $r_0 = [|('y',' a',' b'); ('b',' a',' d'); ('a',' b',' b'); ('a',' d',' o')|; (\emptyset,\emptyset,\emptyset)]$
  $r_1 = [|('a',' b',' b'); ('a',' d',' a'); ('b',' b',' a'); ('d',' o',\emptyset)|]$
  $r_2 = [|('b',' b',' a'); ('d',' a',' b'); ('b',' a',' d'); ('o',\emptyset,\emptyset)|]$
- Fakt: Kolejność sufiksów o początkach w $B_1 \cup B_2$ jest taka sama, jak odpowiednich sufiksów $r_{1,2} = r_1 \hat{} r_2$.
- Trójki tworzące $r_{...}$ sortujemy i zamieniamy na liczby całkowite odpowiadające ich kolejności, tworząc $r'_{...}$.
  Przykład:
  $r_{1,2} = [|''\mathrm{abb}'';''\mathrm{ada}'';''\mathrm{bba}'';''\mathrm{do}'';\ \ ''\mathrm{bba}'';''\mathrm{dab}'';''\mathrm{bad}'';''\mathrm{o}'']$
  $r'_{1,2} = [|1;2;4;6;\ \ 4;5;3;7|]$

Algorytm Sandersa-Kärkkäinena-Burkhardta

- Rekurencyjnie wyznaczymy $\mathrm{SUF}'_{1,2}$ i $\mathrm{Rank}'_{1,2}$ dls $r'_{1,2}$.

$$
\begin{aligned}
r'_{1,2} &= [|1;2;4;6;\quad 4;5;3;7|] \\
\mathrm{Rank}'_{1,2} &= [|1;2;5;7;\quad 4;6;3;8|] \\
\mathrm{SUF}'_{1,2} &= [|0;1;6;4;2;5;3;7|]
\end{aligned}
$$

- Przenumerowujemy i permutujemy pozycje zgodnie z ich położeniem w x, otrzymując $\mathrm{SUF}_{1,2}$ i $\mathrm{Rank}_{1,2}$:

i	0	1	2	3	4	5	6	7	8	9	10	11	12
$x.(i)$	y	a	b	b	a	d	a	b	b	a	d	o	$\emptyset$
$\mathrm{Rank}_{1,2}$	_	1	4	_	2	6	_	5	3	_	7	8	_
$\mathrm{SUF}_{1,2}$	1	4	8	2	7	5	10	11					

Algorytm Sandersa-Kärkkäinena-Burkhardta

- Znamy względną kolejność leksykograficzną sufiksów zaczynających się na pozycjach z $B_1 \cup B_2$.
- Sufiksy x zaczynające się na pozycjach i i j są w takim samym porządku leksykograficznym, jak:
  $(x.(i), \text{Rank}(i+1))$ i $(x.(j), \text{Rank}(j+1))$, oraz
  $(x.(i), x.(i+1), \text{Rank}(i+2))$ i $(x.(j), x.(j+1), \text{Rank}(j+2))$.
- Sufiksy zaczynające się na pozycjach z B_0 sortujemy porównując $(x.(i), \text{Rank}_{1,2}(i+1))$.
  Przykład: $r'_0 = [|('y', 1); ('b', 2); ('a', 5); ('a', 7)|]$
  $\text{Rank}'_0 = [|4; 3; 1; 2; 0|]$
  $\text{SUF}'_0 = [|4; 2; 3; 1; 0|]$

Algorytm Sandersa-Kärkkäinena-Burkhardta

- $\text{Rank}'_0 = [|4; 3; 1; 2; 0|]$
  $\text{SUF}'_0 = [|4; 2; 3; 1; 0|]$
- Przenumerowujemy pozycje zgodnie z ich położeniem w x, otrzymując SUF_0 i Rank_0:

i	0	1	2	3	4	5	6	7	8	9	10	11	12
$x.(i)$	y	a	b	b	a	d	a	b	b	a	d	o	$\emptyset$
Rank_0	4	_	_	3	_	_	1	_	_	2	_	_	0
SUF_0	12	6	9	3	0								

Algorytm Sandersa-Kärkkäinena-Burkhardta

Scalanie:

Mamy dwa posortowane leksykograficznie ciągi sfiksów do scalenia.
Ciąg zaczynający się na pozycji $i \in B_0$ porównujemy z:

- $j \in B_1$ — porównując:
  $(x.(i), \text{Rank}(i+1))$ i $(x.(j), \text{Rank}(j+1))$,
- $j \in B_2$ — porówmując:
  $(x.(i), x.(i+1), \text{Rank}(i+2))$ i $(x.(j), x.(j+1), \text{Rank}(j+2))$.

Algorytm Sandersa-Kärkkäinena-Burkhardta

Example

i	0	1	2	3	4	5	6	7	8	9	10	11	12
$x.(i)$	y	a	b	b	a	d	a	b	b	a	d	o	$\emptyset$
$\text{Rank}_{1,2}$	_	1	4	_	2	6	_	5	3	_	7	8	_
$\text{SUF}_{1,2}$	1	4	8	2	7	5	10	11					
Rank_0	4	_	_	3	_	_	1	_	_	2	_	_	0
SUF_0	12	6	9	3	0								
Rank	12	1	7	5	3	9	2	8	6	4	10	11	0
SUF	12	1	6	4	9	3	8	2	7	5	10	11	0

Algorytm Sandersa-Kärkkäinena-Burkhardta

- Złożoność czasowa:
  $T(n) = O(n) + T(\frac{2}{3}n) = O(n)$

Wyznaczenie tablicy LCP

- Oznaczenie $\mathrm{LCP}'.(i) = \mathrm{LCP}.(\mathrm{Rank}.(i))$.
- Fakt: $\mathrm{LCP}'.(i+1) \geq \mathrm{LCP}'(i) - 1$.
i	0	1	2	3	4	5	6	7	8	9	10	11	12
$x.(i)$	y	a	b	b	a	d	a	b	b	a	d	o	$\emptyset$
$\mathrm{Rank}.(i)$	12	1	7	5	3	9	2	8	6	4	10	11	0
$\mathrm{LCP}'.(i)$		5	4	3	2	1	1	0	1	0	0	0	0
$\mathrm{SUF}.(i)$	12	1	6	4	9	3	8	2	7	5	10	11	0
$\mathrm{LCP}.(i)$	0	5	1	2	0	3	1	4	0	1	0	0	

Wyszukiwanie wzorców

- Przez bisekcję wyszukujemy wystąpienia pierwszego znaku — spójny fragment SUF.
- Drugi znak — zawężamy fragment; itd.
- Złożoność czasowa $O(m \log m)$.

Kompresja tekstu

- Idea kompresji LZ77.
- Tablica LPF.
- Jak obliczyć LPF na podstawie SUF i LCP.
- Fakt: LPF jest permutacją LCP.