Datové struktury I

10. přednáška: Sufixová pole

Jirka Fink

https://ktiml.mff.cuni.cz/~fink/

Katedra teoretické informatiky a matematické logiky
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Zimní semestr 2024/25

Hledání jehly v kupce sena

- Máme daný velmi dlouhý text (seno) a krátký text (jehla)
- Úkolem je najít všechny výskyty jehly v kupce sena
- Příklad: v seně bananas se jehla ana vyskytuje hned dvakrát
- V seně anna se tatáž jehla nevyskytuje vůbec

Algoritmy na hledání jedné jehly ve velmi velkém senu

- Knuth, Morris, Pratt
- Aho, Corasick: Více předem daných jehel
- Robin, Karp: Randomizovaný algoritmus

Opačný přístup

Máme pevně dané seno a chceme hledat různé jehly.

Značení

Abeceda

- Σ je konečná množina znaků (písmen)
- $\Sigma^\star$ je množina konečných posloupností (slov, řetězců) nad Σ
- ε je speciální prázdné slovo
- \$ je speciální znak značící konec slova

Pro slovo $\alpha \in \Sigma$ značíme ①

- $|\alpha|$ délku α
- $\alpha[k]$ je k-tý znak slova α, počítáno od 0 do $|\alpha| - 1$
- $\alpha[k : l]$ je podslovo začínající k-tým znakem a končící těsně před l-tým
- $\alpha[: l]$ je prefix prvních l znaků
- $\alpha[k :]$ je sufix začínající k-tým znakem

Motivace práce se všemi sufixy

Výskyt jehly znamená, že nějaký sufix sena začíná jehlou

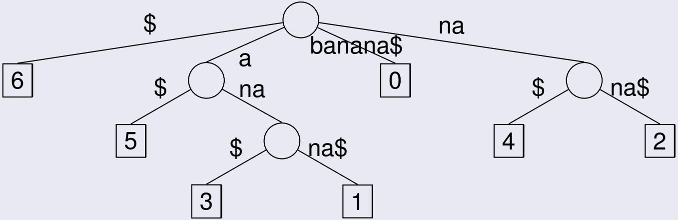
1. Značení je podobné jako v Python.

Sufixový strom

Definice

(Komprimovaný) sufixový strom je (komprimovaná) trie obsahující všechny sufixy, kde synové vrcholů jsou lexikografickém pořadí. ①

Komprimovaný sufixový strom pro slovo banana$ ②



Čísla v listech značí počáteční pozici sufixu.

Dotazy

- Nalezení všech výskytů jehly v kupce sena ③
- Nalezení nejdelšího podslova s dvěma (více) výskyty ④
- Nalezení nejdelšího společného podslova dvou slov ⑤

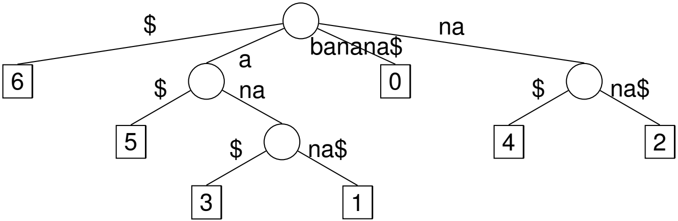
1. Hloubkou vrcholu v komprimované trii rozumíme hloubkou odpovídajícího vrcholu v nekomprimované trii, tj. počet písmen na cestě z kořene do vrcholu.
2. K uložení komprimovaného sufixového stromu potřebujeme $\mathcal{O}(|\alpha|)$ paměti, protože každému listu odpovídá sufix a vnitřních vrcholů je méně než listů.
3. Vyhledáme jehlu v sufixovém stromu a listy v podstromu udávají všechny výskyty. Chceme-li znát jen četnost, tak si stačí navíc ve všech vrcholech pamatovat počet listů v podstromu.
4. Vyhledáme nejhlubší vnitřní vrchol, kde hloubkou rozumíme počet písmen na cestě z kořene do vrcholu.
5. Pro slova $\alpha, \beta \in \Sigma^\star$ vytvoříme sufixový strom pro slovo $\alpha\#\beta\$$, kde hledáme nejhlubší vrchol obsahující v podstromu jak sufix obsahující #, tak i neobsahující.

Sufixové pole

- Sufixové pole X udává lexikografické pořadí sufixů daného slova α.
  $X[i]$ říká, kde v řetězci začíná i-tý sufix v lexikografickém pořadí.
- Rankové pole R je inverzní k X, takže platí $X[R[i]] = i$.
  Hodnota $R[i]$ říká, kolikátý v lexikografickém pořadí je sufix $\alpha[i:]$.
- $LCP(\alpha, \beta)$ udává délku nejdelšího společného prefixu α a β. ①
- Pole společných prefixů (LCP) L udává délku společného prefixu mají sufixy sousedící v lexikografickém pořadí.
  Tedy $L[i] = LCP(\alpha[X[i]:], \alpha[X[i+1]:])$.
- Lexikografický následník sufixu začínajícího na pozici i začíná na pozici $X[R[i]+1]$.

1. LCP = longest common prefix

Souvislost sufixového stromu a pole



i	X[i]	R[i]	L[i]	sufix
0	6	4	0	ε
1	5	3	1	a
2	3	6	3	ana
3	1	2	0	anana
4	0	5	0	banana
5	4	1	2	na
6	2	0	–	nana

Pozorování

- Sufixové pole udává DFS pořadí listů sufixového stromu
- LCP pole $L[i]$ udává hloubku nejbližšího společného předka listů $X[i]$ a $X[i+1]$

Sufixové pole: Příklad pro slovo barokoarokoko

i	X[i]	R[i]	L[i]	sufix
0	13	3	0	ε
1	1	1	5	arokoarokoko
2	6	12	0	arokoko
3	0	10	0	barokoarokoko
4	11	5	2	ko
5	4	8	2	koarokoko
6	9	2	0	koko
7	12	13	1	o
8	5	11	1	oarokoko
9	10	6	3	oko
10	3	9	3	okoarokoko
11	8	4	0	okoko
12	2	7	4	rokoarokoko
13	7	0	–	rokoko

- Sufixové pole $X[i]$ říká, kde v řetězci začíná i-tý sufix v lexikografickém pořadí.
- Rankové pole $R[i]$ říká, kolikátý v lexikografickém pořadí je sufix $\alpha[i:]$.
- Pole společných prefixů LCP $L[i] = LCP(\alpha[X[i]:], \alpha[X[i+1]:])$.

Konstrukce sufixového stromu a pole

Motivace sufixového pole

- Sufixové pole potřebuje méně paměti

Cíle

- Vytvořit pole R z X a opačně ①
- Vytvořit sufixový strom z polí X a L a opačně
- Vytvořit pole X a L
- Upravit algoritmy hledání v textu, aby používali pole místo stromu

1. Tento krok je triviální, protože jen vytváříme inverzní permutaci.

Převod sufixového pole a stromu

Pozorování

- Sufixové pole udává DFS pořadí listů sufixového stromu
- LCP pole $L[i]$ udává hloubku nejbližšího společného předka listů $X[i]$ a $X[i+1]$

Konstrukce sufixového pole a LCP ze stromu

Obě pole vytváříme při průchodu stromu do hloubky

Konstrukce sufixového stromu z pole a LCP

Strom vytváříme průchodem do hloubky

- Nejprve vytvoříme list pro slovo ε
- Po vytvoření listu pro sufix $X[i]$
  - Vynoříme se do vrcholu v hloubce $L[i]$
  - Přidáme list pro sufix $X[i]$ v hloubce $|\text{seno}| - X[i]$

Časová složitost

Složitost těchto převodů je lineární pro komprimovaný sufixový strom

Vytvoření LCP pole ze sufixového pole: Triviální postup

Algoritmus

for $i = 0, \ldots, |\alpha| - 1$ **do**
  $j = X[R[i] + 1]$
  $l = 0$
  while $i + l < |\alpha|$ *&&* $j + l < |\alpha|$ *&&* $\alpha[i + l] == \alpha[j + l]$ **do**
    $l = l + 1$
  $L[R[i]] = l$

Časová složitost je $\mathcal{O}(|\alpha|^2)$

Algoritmus zrychlíme tak, že si všimneme, že mnoho porovnání provádíme opakovaně.

Vytvoření LCP pole ze sufixového pole (Kasai, 2001)

Pozorování

Pro každé slovo α a pro všechna $i = 0, \dots, |\alpha| - 1$ platí $L[R[i+1]] \geq L[R[i]] - 1$.

Důkaz

- Zřejmé pro $L[R[i]] \leq 1$
- Zřejmé pro $X[R[i+1]+1] = X[R[i]+1]+1$ ①
- Jinak platí $\alpha[i+1:] < \alpha[X[R[i+1]+1]:] < \alpha[X[R[i]+1]:]$ ②
- Platí

$$\begin{aligned}
L[R[i+1]] &= LCP(\alpha[i+1:], \alpha[X[R[i+1]+1]:]) \\
&\geq LCP(\alpha[i+1:], \alpha[X[R[i]+1]:]) \\
&= LCP(\alpha[i:], \alpha[X[R[i]]:]) - 1 \\
&= L[R[i]] - 1
\end{aligned}$$

1. i je pozice ve slovu α, $R[i]$ udává pořadí sufixu na pozici i, $X[R[i]+1]$ udává pozici následníka v lexikografickém uspořádání
2. Sufix začínající na pozici $X[R[[i+1]+1]]$ je za sufixem začínající na pozici $i+1$, ale před sufixem na pozici $X[R[[i]+1]]$

Vytvoření LCP pole ze sufixového pole (Kasai, 2001)

Pozorování

Pro každé slovo α a pro všechna $i = 0, \ldots, |\alpha| - 1$ platí $L[R[i+1]] \geq L[R[i]] - 1$.

Algoritmus

$l = 0$
for $i = 0, \ldots, |\alpha| - 1$ **do**
  $l = \max(0, l - 1)$
  $j = X[R[i] + 1]$
  while $i + l < |\alpha|$ && $j + l < |\alpha|$ && $\alpha[i + l] == \alpha[j + l]$ **do**
    $l = l + 1$
  $L[R[i]] = l$

Časová složitost je $\mathcal{O}(|\alpha|)$

- Vnější cyklus má složitost $\mathcal{O}(|\alpha|)$
- Ve vnitřním cyklu se l vždy zvyšuje o 1
  - Hodnata l začíná na 0
  - Hodnota l na konci je nejvýše $\mathcal{O}(|\alpha|)$
  - Vnější cyklus hodnotu l sníží dohromady o nejvýše $\mathcal{O}(|\alpha|)$

Vytvoření pole R zdvojováním

Popis algoritmu

1. $R_k[i]$ je počet j takových, že $\alpha[j:j+k] < \alpha[i:i+k]$ ① ②
2. Cílem je spočítat $R_k[i] = R[i]$ pro nějaké $k \geq |\alpha|$
3. K výpočtu R_1 jen počítáme počet výskytů jednotlivých písmen
4. $\alpha[i:i+2k] < \alpha[j:j+2k]$ právě tehdy, když
   $\alpha[i:i+k] < \alpha[j:j+k] \vee (\alpha[i:i+k] = \alpha[j:j+k] \,\&\, \alpha[i+k:i+2k] < \alpha[j+k:j+2k])$
5. $R_{2k}[i] < R_{2k}[j]$ právě tehdy, když
   $R_k[i] < R_k[j] \vee (R_k[i] = R_k[j] \,\&\, R_k[i+k] < R_k[j+k])$
6. K získání R_{2k} třídíme dvojice $(R_k[i], R_k[i+k])$ pro $i = 0, \ldots, |\alpha| - 1$

Časová složitost

1. Třídění pro získání R_0 máme v čase $\mathcal{O}(|\alpha| \log |\alpha|)$ ③
2. Následuje $\mathcal{O}(\log n)$ přihrádkového třídění
3. Celková složitost je $\mathcal{O}(|\alpha| \log |\alpha|)$
4. Kärkkäinen, Sanders, 2003: Konstrukce v lineárním čase

1. Porovnáváme tedy jen prvních k znaků sufixů.
2. Pokud v zápisu $\alpha[i : j]$ horní index přesahuje délku slova, pak $\alpha[i : j]$ značí jen $\alpha[i :]$.
3. Pokud je $|\Sigma| \leq |\alpha|$, pak můžeme použít přihrádkové třídění a docílit tím času $\mathcal{O}(|\alpha|)$.

Vytvoření sufixového pole zdvojováním

Vytvoříme pole $X[0 \dots n]$ a $R[0 \dots n]$.
`# Báze: vytvoření` R_1
$D = \{(\alpha[i], i);\ i = 0, \dots, n\}$ setříděné lexikograficky
for $j = 0, \dots, n$ **do**
    $X[j] = D[j][1]$
    if $j = 0$ *nebo* $D[j][0] \neq D[j-1][0]$ **then**
        $R[X[j]] = j$
    else
        $R[X[j]] = R[X[j-1]]$

`# Indukční krok: vytvoření` R_{2k} `z` R_k
for $(k = 1; k < n; k = 2k)$ **do**
    $D = \{(R[i], R[i+k], i);\ i = 0, \dots, n\}$ setříděné lexikograficky
    for $j = 0, \dots, n$ **do**
        $X[j] = D[j][2]$
        if $j = 0$ *nebo* $(D[j][0], D[j][1]) \neq (D[j-1][0], D[j-1][1])$ **then**
            $R[X[j]] = j$
        else
            $R[X[j]] = R[X[j-1]]$

Použití sufixového pole

- Jak najít nejdelší opakující se podřetězec?
- Jak určit počet různých podřetězců délky k?
- Jak najít jehlu v kupce sena?

Příští týden

Hledání bodů v rovině i více-dimenzionálním prostoru.