Суфиксни дървета част 2

11.12.2020 г.

(Алгоритъм на Уконен (Ukkonen))

$$V_w = \{\varepsilon\} \cup \underbrace{\{v \mid \exists a,b \in \Sigma(a \neq b) \ \& \ va, vb \in Inf(w)\}}_{V_w^{\geq 2}} \cup \underbrace{\{v \in Suff(w) \mid v \notin Inf(w) \backslash Suff(w)\}}_{L_w}$$

$\tilde{\delta}_w : \left(V_w^{\geq 2} \cup \varepsilon \times \Sigma\right) \longrightarrow V_w$

$\tilde{\delta}(v,a) = \overrightarrow{va}$ - най-късия инфикс във V_w, които започва с va.

Видяхме няколко неща до сега:

1. Интуиция за това как може да поддържаме V_w:

- Важни са суфиксите на w, които се срещат поне два пъти;
- Листата не се променят (те стават по-дълги), те винаги си остават листа и никога нр се срещат като ссщински инфикс;
- Евентуално част от инфиксите на w могат да преминат във $V_{wb}^{\geq 2}$ при добавянена нова буква b.

2. Другата посока, по която пое интуицията е чисто имплементационна:

- Представяне на $V_w^{\geq 2} \cup \{\varepsilon\}$:
  int $start$ - начална позиция на срещане;
  int len - дължина на съответния представител;
  $list\ \tilde{\delta}$ - списък съответстващ на делта вълна;
  int s - указател или връзка към т.нар. суфикс линк, който всъщност е думата, която попада във V_w с отрязаната буква.
- представяне на L_w :
  int $start$ - единственото, което ни трябва е началната позиция.

Друго съображение, което направихме е, че фиксирахме представяне на произволен инфикс α - инфикс на w : наредена тройка ($\underset{\text{връх}}{v}$, $\underset{\text{буква}}{a}$, $\underset{\text{дължина}}{l}$), като смисъла е следния:

или $\alpha = v$, $a = \perp$ и $l = 0$ или α е префикс с дължина l на представителя $\overrightarrow{va}$ с дължина l.

Използвайки гореописаното представяне се научихме да правим две неща:

– да проверяваме дали даден инфикс на думата w се следва от определена буква;

– като имаме произволен инфикс α на w - да намираме $s(\alpha)$, т.е. да намираме представянето на инфикса на w като отрежем първата буква на α.

Необходимо е да покажем по-формално как се променя w при добавянето на буква, за да подкрепим интуицията, да преведем тази интуиция в реализация и накрая да анализираме времето, което ще похарчим за тази реализация.

Характеризация на обновяването:

Имаме w, V_w, s_w, $\tilde{\delta}_w$ и също така имаме $b \in \Sigma$.
Тогава:

1. За върховете:

- (листата) $L_w = \{vb \mid v \in L_w\} \cup \{vb \mid v \in Suff(w) \ \& \ vb \notin Inf(w)\}$

- (разклоненията) $V_{wb}^{\geq 2} = V_w^{\geq 2} \cup \{v \in Suff(w)\}$, v се среща поне 2 пъти в w и $vb \notin Inf(w)\}$
- $s_{wb}(v) = \begin{cases} s_w(v) \text{ не се променя}, & \text{ако } v \in V_w^{\geq 2} \\ v', & \text{ако } v = av' \in V_{wb}^{\geq 2} \backslash V_w^{\geq 2} \end{cases}$

Това беше интуицията, която си изградихме миналия път и сега само ще я потвърдим.

Доказателство:

1) $v \in L_{wb} \backslash \{\varepsilon\}$. Тогава $v = v'b \Rightarrow v' \in Suff(w)$.
Тъй като $v \in L_{wb}$, т.е. $v \notin Inf(wb) \backslash Suff(wb)$, то $v \notin Inf(w)$.
Следователно или v' не принадлежи на $Inf(w)$ или $v'b$ не принадлежи на $Inf(w)$ и в двата случая $v'b \notin Inf(w)$ & $v'b \in Suff(w) \overset{def.}{\Rightarrow} v \in$ дясната страна на равенството (листата).

Обратно, ако $v \in L_w \Rightarrow v$ се среща веднъж в w и единствено като суфикс
$\Rightarrow vb \in Suff(wb)$ и $vb \notin Inf(w)$ (иначе v се среща поне 2 пъти в w)

Накрая, ако $v \in Suff(w)$ & $vb \notin Inf(w) \Rightarrow vb \in Suff(wb)$ (от първото) и vb се среща точно веднъж в wb (от второто) $\overset{def.}{\Rightarrow}$ по дефиниция $vb \in L_{wb}$.

2) $v \in V_w^{\geq 2}$. Тогава може да кажем, че $\exists a, c (a \neq c)$ и va и vc са инфикси на w (*точно както беше в структурата на Blumer), т.е. на минималния суфиксен автомат - там беше с леви, а тук с десни контексти*). Щом $va, vc \in Inf(w)$, значи те ще продължат да бъдат инфикси на wb $\left(v \in V_w^{\geq 2} \Rightarrow v \in V_{wb}^{\geq 2}\right)$.
Сега остана да докажем, че разликата на тези две множества $V_{wb}^{\geq 2} \backslash V_w^{\geq 2}$ е
$\underbrace{\{v \in Suff(w) \backslash V_w^{\geq 2} \mid v \text{ се среща поне 2 пъти в } w \text{ и } vb \notin w\}}_{M}$.

Да разгледаме един елемент от множеството M. Трябва да докажем, че той е от множеството $V_{wb}^{\geq 2}$, защото и от двете страни сме махнали $V_w^{\geq 2}$.

Ако $v \in M \Rightarrow v$ се среща поне два пъти в $w \Rightarrow \exists$ буква $\underline{a}$, за която $va \in Inf(w)$ (т.е. не се среща само като суфикс), но ние знаем, че $v\underline{b} \notin Inf(w)$ и следователно $a \neq b$. Тогава va и vb са инфикси на wb, откъдето директно от дефиницията следва, че $v \in V_{wb}^{\geq 2}$.

Обратно. Нека $v \in V_{wb}^{\geq 2} \backslash V_w^{\geq 2}$. Тогава $\exists a, c (a \neq c) : va, vc \in Inf(wb)$. Да допуснем, че va и vb едновременно са инфикси на w. Обаче тогава v ще се среща в различни десни контексти в w и ще попадне в $V_w^{\geq 2}$, което е противоречие с допускането. Следователно остава някоя от тези две думи да не бъде инфикс на w. Това показва, че както и да изберем различни букви $a \neq c$, така че va и vc да бъдат инфикси на wb, една от двете думи не е инфикс на w. Б.о.о. нека това е vc и $vc \notin Inf(w)$. Тъй като $vc \in Inf(wb)$, то $vc \in Suff(wb) \Rightarrow c = b$ и $v \in Suff(w)$. (*Нещо повече: всички останали срещания на думата v вътре като инфикс на w се следват от буквата a*) $\Rightarrow v \in M$!

Относно суфикс линковете - нещата би трябвало да са ясни. Дефиницията дава отговор на тези въпроси с актуализацията. Нека я припомним:

$s_w(v) = v' \Leftrightarrow v = av'$ за някоя буква $a \in \Sigma$. Предишния път стана ясно, че би било скъпо и неефективно да поддържаме суфикс линковете за всички върхове в дървото и ще ни трябва такава функционална стойност на s само за вътрешните върхове $V_w^{\geq 2}$ и се концентрираме върху нея.

Всъщност, ако $v \in V_{wb}^{\geq 2}$, то има различни букви $a \neq c$ такива, че $va, vc \in Inf(wb)$ и ако $v = dv'$, то $v'a$ и $v'c \in Inf(wb) \Rightarrow v' \in V_{wb}^{\geq 2}$.
(*това, което човек трябва да забележи е може би следното: v' може да не е връх във $V_w^{\geq 2}$, но задължително ще се появи във $V_{wb}^{\geq 2}$*)

$v = dv'$ - той е нов връх, т.е. $v \in V_{wb}^{\geq 2} \backslash V_w^{\geq 2}$, $v' \notin V_{wb}^{\geq 2} \Rightarrow v'$, но тогава v' ще бъде добавено във $V_{wb}^{\geq 2}$.

Остана да покажем какво се случва с преходите.

<u>За преходите</u>:

$$\tilde{\delta}_{wb}(v,x) = \begin{cases} vb, & vb \in L_{wb}, b = x \\ u, & u \in V_{wb}^{\geq 2} \backslash V_w^{\geq 2} \text{ и } u \in Pref(\overrightarrow{vx}) \\ \tilde{\delta}_w(v,x) & \end{cases}$$

$\alpha = (v,a,l)$ - най-дългия суфикс в w, който се среща поне 2 пъти в w.

procedure $ProcessNextChar(v,a,l,b,$ $\underset{\text{дълж. на } w}{n}$ $)\{$

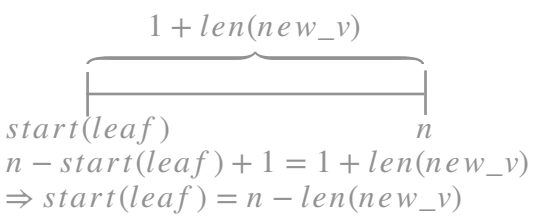
$wb = n+1$
$b = w[n]$

$\alpha \leftarrow (v,a,l)$
$last\_new\_v \leftarrow \perp$
while $v \neq \perp$ **and** $!followedBy(v,a,l,b)$ **do**{
if $l \neq 0$ **then**

$O(1)$ създава се ново състояние $NV^{\geq 2}$:
$new\_v \leftarrow new\ state(V_w^{\geq 2})$
$start(new\_v) \leftarrow start\left(\tilde{\delta}(v,a)\right)$
$len(new\_v) \leftarrow len(v) + l$
$c \leftarrow start(new\_v) + len(new\_v)$
$\tilde{\delta}(new\_v, c) \leftarrow \tilde{\delta}(v,a)$
$\tilde{\delta}(v,a) \leftarrow new\_v$

else
$new\_v \leftarrow v$
if $last\_new\_v \neq \perp$ **then**
$s(last\_new\_v) \leftarrow new\_v$
if $new\_v \neq v$ **then**
$last\_new\_v \leftarrow new\_v$

ново състояние NL:
$leaf \leftarrow new\ state(L_w)$
$start(leaf) \rightarrow n - len(new, v)$
$\tilde{\delta}(new\_v, b) \leftarrow leaf$



```
            if v = ε and l = 0 then
                v ← ⊥
            else
                (v, a, l) ← FindNextStem(v, a, l)
        }
O(1)    { if last_new_v ≠ ⊥
            s(last_new_v) ← v
        { if v = ⊥ then
            return (ε, ⊥, 0) // ε е най-дългия суфикс, който се среща поне 2 пъти в wb
O(1)    { return followInfix(v, a, l, (n, 1))
                                      w[n]=b
}
```

```
procedure BuildSuffixTree(w){
    ε ← new state(V_w^{≥2})
    start(ε) ← 0
    len(ε) ← 0
    leaf ← new state(L_w)
    start(leaf) ← 0
    δ̃(ε, w[0]) ← leaf
    (v, a, l) ← (ε, ⊥, 0)
    for i = 1 to |w| − 1 do
        (v, a, l) ← ProcessNextChar(v, a, l, w[i])
    done
}
```

Забележка: Обърнете внимание, че в суфиксното дърво не всички суфикси са експлицитно представени. Има суфикси, които могат да се окажат вътрешни върхове и да имат единствено представяне. Това малко противоречи с концепцията на суфиксното дърво да представи всички суфикси на w. Поради тази причина, за да се избегне този проблем, обикновено към w се добавя специален маркер за край на дума ($). Благодарение на него, всички суфикси на оригиналната дума w ще съответстват на суфикс, който ще бъде листо в $w\$$.

$w \longmapsto w\$$

$\alpha \in Suff(w) \Leftrightarrow \underbrace{w\$ \in Suff(w\$)}_{\alpha \text{ се среща точно веднъж в } w\$, \text{ заради особеноста на символа } \$ \text{ (технически хак)}}$

Знаем, че:

1) $NV^{\geq 2} + NL \leq V_w \leq 2|w| + 1$

Остава да пресметнем времето загубено от рекурсивната функция $FindNextStem$.

Да разгледаме следния инвариант. С всеки $\alpha = (v, a, l)$ - *суфикс* на w, асоцииране цена $cost(\alpha) = |\alpha| + l$. Да разгледаме суфиксите $\alpha_0, \alpha_1, \ldots, \alpha_k$ в рамките на $while$ цикъла на едно извикване на $ProcessNextChar$.

$\alpha_i = (v_i, a_i, l_i)$

Времето за преход от i-тата до $i+1$-вата фаза:
$FindNextStem(v_i, a_i, l_i, b) = (v_{i+1}, a_{i+1}, l_{i+1}) \Rightarrow$ времето за това изпълнение е $\leq l_i - l_{i+1}$.
Освен това ние знаем, че дължината на суфикса α намалява с точно 1-ца. Следователно

$$l_i - l_{i+1} \leq 2\left(|\alpha_i| - |\alpha_{i+1}|\right) + l_i - l_{i+1} = cost(\alpha_i) - cost(\alpha_{i+1})$$

$\Rightarrow$ времето на всички изпълнения на $FindNextStem$ в рамките на едно извикване на обработката на следващия символ ще бъде цената на началното α минус цената на следващото α_k, което ще получим от резултата на рекурсивната функция. Т.е. времето за цялото изпълнение на $FindNextStem \leq cost(\alpha_0) - cost(\alpha_k)$.

Ние искаме да направим амортиззиран анализ, за това трябва да вържем цената на α_k със цената на следващото α_0, което ще дойде от следващата итерация.

Нека $\alpha^{(i)}$ е най-дългия повтарящ се суфикс на $w[0...j]$. Тогава, ако $while$ цикъла с $ProcessNextChar(\cdot, \cdot, \cdot, j+1)$ завърши с $\alpha_k^{(j)}$, то $\alpha^{(j+1)}$ ще има цена

$$cost\left(\alpha^{(j+1)}\right) \leq 2\left(|\alpha_k^{(j)}| + 1\right) + l_k^{(j)} + 1 \leq cost\left(\alpha_k^{(j)}\right) + 3$$
$$\Rightarrow cost\left(\alpha_0^{(j)}\right) - cost\left(\alpha_k^{(j)}\right) \leq cost\left(\alpha^{(j)}\right) - cost\left(\alpha^{(j+1)}\right) + 3$$

Следователно времето за

$$ProcessNextChar(\cdot, \cdot, \cdot, \cdot, j+1) \leq cost\left(\alpha^{(j)}\right) - cost\left(\alpha^{(j+1)}\right) + 3$$

$\Rightarrow$ времето за $BuildSuffixTree$ е $\leq$ от

$$\sum_{j=0}^{|w|-2}\left[cost\left(\alpha^{(j)}\right) - cost\left(\alpha^{(j+1)}\right) + 3\right] = 3\left(|w| - 1\right) + cost\left(\alpha^{(0)}\right) - cost\left(\alpha^{(|w|-1)}\right) \leq$$

$$\leq 3\left(|w| - 1\right).$$

Коментари:
Отново имаме две дървета. Едното е очевидното суфиксно дърво, а другото е дървото, което е построено с функцията на бащите (суфиксните линкове). Интерпретацията на тези две дървета е дуална в някакъв смисъл на двете дървета, които имахме при суфиксния автомат.

$\mathcal{S}_w = (V_w, \delta_w, \varepsilon)$ - суфиксно дърво.

1. $u, v \in V_w$
$LCA(u, v) = x$, който връх е префикс на u и v и при това ще бъде най-дългия общ префикс на u и v.

$\mathcal{Y} = (V_w, s_w, \varepsilon)$ - второ дърво

1. $|s_w(u)| = |u| - 1$, за $u \in V_w^{\geq 2} \setminus \{\varepsilon\}$
2. $LA(u, d) = u_d$ ще бъде суфиксът на u с дължина $|u_d| = |u| - d$
3. $LCA(u, v) = y$ - най-дългия суфикс на u и v.