Enajste vaje APS2: Tabela najdaljših skupnih predpon

Tabela najdaljših skupnih predpon (angl. longest common prefix array, LCP) je tesno povezana s priponsko tabelo (angl. suffix array, SA) podanega niza T dolžine n, ki smo jo spoznali na predavanjih. Spomnimo se, da je i-ti element priponske tabele za niz T enak indeksu začetka pripone, ki se v leksikografsko urejenem zaporedju vseh pripon niza T nahaja na i-tem mestu. To pomeni, da je $\mathcal{U}(T) = \langle T[SA[1]:], T[SA[2]:], \ldots, T[SA[n]:]\rangle$ leksikografsko urejeno zaporedje vseh pripon.

Tabela 1 prikazuje priponsko tabelo (in še nekaj drugih reči) za niz `abrakadabra`. Leksikografsko urejeno zaporedje pripon za ta niz je $\mathcal{U} = \langle$`a`, `abra`, `abrakadabra`, `adabra`, `akadabra`, `bra`, `brakadabra`, `dabra`, `kadabra`, `ra`, `rakadabra`$\rangle$.

Najdaljša skupna predpona nizov A in B je najdaljši niz C, ki je tako predpona niza A kot predpona niza B; dolžino takšnega niza C bomo označili z $L(A, B)$. Tabela najdaljših skupnih predpon pa je definirana takole:

- $LCP[1] = 0$.
- Pri $i \geqslant 2$ je $LCP[i]$ dolžina najdaljše skupne predpone pripon, ki v zaporedju $\mathcal{U}$ nastopata na mestih i in $i-1$. Velja torej

$$LCP[i] = L(T[SA[i-1]:], T[SA[i]:]).$$

Tabela 1: Priponska tabela in njene sorodnice za niz `abrakadabra`.

i	$SA[i]$	$T[SA[i]:]$	$R[i]$	$bp(i)$	$LCP[i]$
1	11	`a`	3	–	0
2	8	`abra`	7	6	1
3	1	`abrakadabra`	11	7	4
4	6	`adabra`	5	8	1
5	4	`akadabra`	9	9	1
6	9	`bra`	4	10	0
7	2	`brakadabra`	8	11	3
8	7	`dabra`	2	2	0
9	5	`kadabra`	6	4	0
10	10	`ra`	10	1	0
11	3	`rakadabra`	1	5	2

Izračun

Da si olajšamo notacijo, definirajmo tabelo R kot inverz tabele SA. Vrednost $R[i]$ je torej mesto pripone $T[i:]$ v zaporedju $\mathcal{U}$. Definirajmo še funkcijo

$$bp(i) = R[SA[i] + 1].$$

Vrednost $bp(i)$ je torej indeks tiste pripone v zaporedju $\mathcal{U}$, ki je enaka priponi $T[SA[i]:]$ brez prvega znaka. V našemu primeru je $bp(2) = 6$: v zaporedju $\mathcal{U}$ se na drugem mestu nahaja pripona `abra`, pripona `bra` (`abra` brez prvega znaka) pa se v zaporedju $\mathcal{U}$ nahaja na šestem mestu. Vrednost $bp(i)$ je definirana za vse $i \in \{1, \ldots, n\}$ razen za $i = R[n]$ ($R[n]$ je položaj pripone $T[n:]$ v zaporedju $\mathcal{U}$, ta pripona pa je en sam znak).

Tabelo LCP lahko izdelamo neposredno po definiciji, a za to potrebujemo $O(n^2)$ časa. K sreči pa obstaja tudi algoritem, ki tabelo zgradi v času $O(n)$. Algoritem temelji na razmeroma preprostem opažanju:

Trditev 1. *Za vsak $i \neq R[n]$ velja $LCP[bp(i)] \geqslant LCP[i] - 1$.*

Dokaz. Če je $LCP[i] \leqslant 1$, lastnost iz trditve trivialno velja, zato lahko predpostavimo $LCP[i] \geqslant 2$.

Za lažjo predstavo se zopet obrnimo na niz `abrakadabra`. Vzemimo $i = 3$. Ker sta druga in tretja pripona v zaporedju $\mathcal{U}$ enaki $T[SA[i-1]:] = \texttt{abra}$ in $T[SA[i]:] = \texttt{abrakadabra}$, je $LCP[i] = 4$. Ker velja $bp(i) = 7$, $T[SA[bp(i)]:] = \texttt{brakadabra}$ in $T[SA[bp(i)-1]:] = \texttt{bra}$, je $LCP[bp(i)] = L(\texttt{brakadabra}, \texttt{bra}) = 3$. Vsaj v tem primeru je res $LCP[bp(i)] \geqslant LCP[i] - 1$.

Po definiciji funkcije bp je $T[SA[bp(i)]:] = T[SA[i]+1:]$. Če je $LCP[i] = L(T[SA[i]:], T[SA[i-1]:]) = \ell$, potem je torej $L(T[SA[bp(i)]:], T[SA[i-1]+1:]) = \ell - 1$; če nizoma z dolžino skupne predpone $\ell \geqslant 2$ odbijemo prvi znak, bo dolžina skupne predpone seveda $\ell - 1$. Žal pa ne iščemo

$$L(T[SA[bp(i)]:], T[SA[i-1]+1:]) = \ell - 1,$$

pač pa

$$LCP[bp(i)] = L(T[SA[bp(i)]:], T[SA[bp(i)-1]:]).$$

Niz $T[SA[i-1]+1:]$ je pripona na mestu $(i-1)$ v zaporedju $\mathcal{U}$, ki ji odbijemo prvi znak, niz $T[SA[bp(i)-1]:]$ pa je pripona, ki jo dobimo tako, da priponi na i-tem mestu odbijemo prvi znak in poiščemo njeno predhodnico v zaporedju $\mathcal{U}$. V primeru $i = 3$ gre sicer za eno in isto pripono (`bra`), pri $i = 11$ pa to ni res: pripona `rakadabra` je neposredna naslednica pripone `ra` v zaporedju $\mathcal{U}$, pripona `akadabra` pa ni neposredna naslednica pripone `a`. V vsakem primeru pa velja sledeče:

- Ker je $T[SA[i-1]:] \prec T[SA[i]:]$, je tudi $T[SA[i-1]+1:] \prec T[SA[bp(i)]:]$. Jasno: če je niz A leksikografsko pred nizom B, bo niz $A[2:]$ prav tako leksikografsko pred nizom $B[2:]$ (ob predpostavki, da sta vsaj prva znaka nizov A in B med seboj enaka).
- Pripona $T[SA[i-1]:]$ (npr. `ra`) je po definiciji priponske tabele neposredna predhodnica pripone $T[SA[i]:]$ (npr. `rakadabra`), pripona $T[SA[i-1]+1:]$ (npr. `a`) pa v splošnem ni nujno *neposredna* predhodnica pripone $T[SA[bp(i)]:]$ (npr. `akadabra`); lahko bi v zaporedju $\mathcal{U}$ obstajale še kakšne vmesne pripone (npr. `abra`, `abrakadabra` in `adabra`). Vendar pa bo za vsako tako vmesno pripono P dolžina najdaljše skupne predpone niza P in niza $T[SA[bp(i)]:]$ enaka najmanj $L(T[SA[i-1]+1:], T[SA[bp(i)]:]) = \ell - 1$. (Na primer, niz `post` ni nujno neposredni predhodnik niza `potnik`, vendar pa za vse morebitne vmesne nize S (`pošta`, `pot`, `potnica`, ...) velja $L(\texttt{potnik}, S) \geqslant L(\texttt{potnik}, \texttt{post})$.)

Lastnost iz trditve je tako dokazana. □

Kako nam lahko trditev 1 pomaga pri računanju najdaljše skupne predpone med zaporednimi priponami v zaporedju $\mathcal{U}$? Če za pripono $A = T[SA[i]:]$ (in njeno predhodnico $B = T[SA[i-1]:]$ v zaporedju $\mathcal{U}$) ugotovimo, da je $L(A, B) = \ell$ (torej $LCP[i] = \ell$), potem se bomo v naslednjem koraku ukvarjali s pripono $A' = A[2:] = T[SA[bp(i)]:]$, saj bomo vedeli, da je dolžina najdaljše skupne predpone med A' in njeno predhodnico $B' = T[SA[bp(i)-1]:]$ v zaporedju $\mathcal{U}$ najmanj $\ell - 1$, kar pomeni, da bomo med seboj primerjali le znake pripon A' in B' od ℓ-tega naprej. Na ta način bomo učinkovito določili $LCP[bp(i)]$.

Ni težko ugotoviti, da nas to opažanje privede do časovne zahtevnosti $O(n)$ (skupno število medsebojnih primerjav znakov je kvečjemu $2n$). Algoritem za izračun tabele LCP je prikazan kot algoritem 1.

V primeru niza `abrakadabra` bi se prve štiri iteracije algoritma odvile takole:

- Pričnemo s pripono `abrakadabra`. Njena predhodnica v zaporedju $\mathcal{U}$ je `abra`. Velja $L(\texttt{abrakadabra}, \texttt{abra}) = 4$, zato je $\ell \leftarrow LCP[R[1]] = LCP[3] = 4$.
- Nadaljujemo s pripono `brakadabra`. Njena predhodnica v zaporedju $\mathcal{U}$ je `bra`. Po trditvi 1 vemo, da je $L(\texttt{brakadabra}, \texttt{bra}) \geqslant \ell - 1 = 3$, zato se lahko omejimo na primerjave istoležnih znakov od četrtega naprej. Vidimo, da ujemajočih znakov ni več (saj je pripona `bra` dolga zgolj 3), zato nastavimo $\ell \leftarrow LCP[R[2]] = LCP[7] = 3$.
- Pri priponi `rakadabra` si spet pomagamo s trditvijo 1 in dobimo $\ell \leftarrow LCP[R[3]] = LCP[11] = L(\texttt{rakadabra}, \texttt{ra}) = 2$.
- Naslednja pripona je `akadabra`. Njena predhodnica v zaporedju $\mathcal{U}$ je `adabra`, po trditvi 1 pa vemo, da je $L(\texttt{akadabra}, \texttt{adabra}) \geqslant L(\texttt{akadabra}, \texttt{a}) = \ell - 1 = 1$. Primerjamo istoležne znake od drugega naprej in ugotovimo, da se nič ne ujema. Zato nastavimo $\ell \leftarrow LCP[R[4]] = LCP[5] = 1$.

Algoritem 1 Izračun tabele LCP.

```
function SKUPNA-PREDPONA(S, i, j)
    ▷ Vrne L(S[i :], S[j :]). ◁
    n ← |S|
    d ← 0
    while i + d ⩽ n ∧ j + d ⩽ n ∧ S[i + d] = S[j + d] do
        d ← d + 1
    return d

function LCP(S)
    n ← |S|
    ustvari tabelo LCP[1 : n]
    ℓ ← 0
    LCP[1] ← 0
    for i ← 1 to n do                          ▷ i teče po indeksih v nizu S
        if R[i] ⩾ 2 then
            j ← SA[R[i] − 1]                   ▷ T[j :] je predhodnica T[i :] v zaporedju U
            ℓ ← max(0, ℓ − 1)
            d ← SKUPNA-PREDPONA(S, i + ℓ, j + ℓ)
            ℓ ← ℓ + d
            LCP[R[i]] ← ℓ
    return LCP
```

Uporaba

S tabelo LCP lahko neposredno pridobimo dolžino najdaljšega ponovljenega podniza podanega niza: to je preprosto maksimalna vrednost v tabeli. Na primer, pri nizu `abrakadabra` je dolžina najdaljšega ponovljenega podniza enaka 4 (to je podniz `abra`). V okviru domače naloge boste ugotovili, da je razmeroma enostavno mogoče pridobiti tudi dolžino najdaljšega skupnega podniza dveh ločenih nizov (namig: lahko dva niza združimo v en niz?). Z nekoliko več truda lahko tabelo LCP uporabimo tudi za to, da časovno zahtevnost iskanja

vseh k pojavitev vzorca dolžine m v besedilu dolžine n zmanjšamo z $O(m \log n + km)$, kot potrebujemo zgolj z uporabo priponske tabele, na $O(m + \log n + k)$.