10. Suffixové stromy

V této kapitole popíšeme jednu pozoruhodnou datovou strukturu, pomocí níž dokážeme problémy týkající se řetězců převádět na grafové problémy a řešit je tak v lineárním čase.

Řetězce, trie a suffixové stromy

Definice:

Σ	konečná abeceda – množina znaků (znaky budeme značit latinskými písmeny)
Σ^*	množina všech slov nad Σ (slova budeme značit řeckými písmeny)
ε	prázdné slovo
$\|\alpha\|$	délka slova α
$\alpha\beta$	zřetězení slov α a β ($\alpha\varepsilon = \varepsilon\alpha = \alpha$)
α^R	slovo α napsané pozpátku
α je *prefixem* β	$\exists\gamma : \beta = \alpha\gamma$ (β začíná na α)
α je *suffixem* β	$\exists\gamma : \beta = \gamma\alpha$ (β končí na α)
α je *podslovem* β	$\exists\gamma, \delta : \beta = \gamma\alpha\delta$ (značíme $\alpha \subset \beta$)
α je *vlastním prefixem* β	je prefixem a $\alpha \neq \beta$ (analogicky vlastní suffix a podslovo)

Pozorování: Prázdné slovo je prefixem, suffixem i podslovem každého slova včetně sebe sama. Podslova jsou právě prefixy suffixů a také suffixy prefixů.

Definice: *Trie (Σ-strom)* pro konečnou množinu slov $X \subset \Sigma^*$ je orientovaný graf $G = (V, E)$, kde:

$V = \{\alpha : \alpha \text{ je prefixem nějakého } \beta \in X\}$,
$(\alpha, \beta) \in E \equiv \exists x \in \Sigma : \beta = \alpha x$.

Pozorování: Trie je strom s kořenem ε. Jeho listy jsou slova z X, která nejsou vlastními prefixy jiných slov z X. Hrany si můžeme představit popsané písmeny, o něž prefix rozšiřují, popisky hran na cestě z kořene do vrcholu α dávají právě slovo α.

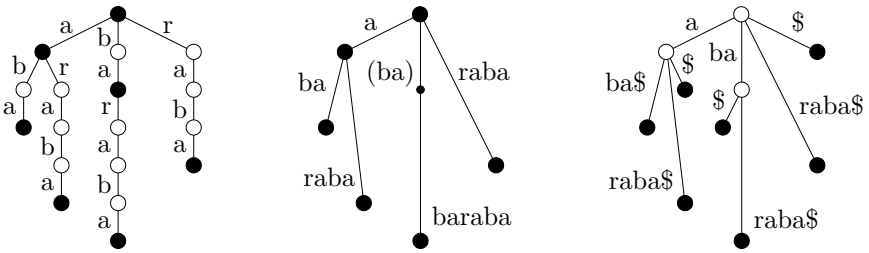
Definice: *Komprimovaná trie (Σ^+-strom)* vznikne z trie nahrazením maximálních nevětvících se cest hranami. Hrany jsou tentokrát popsané řetězci místo jednotlivými písmeny, přičemž popisky všech hran vycházejících z jednoho vrcholu se liší v prvním znaku. Vrcholům „uvnitř hran“ (které padly za oběť kompresi) budeme říkat *skryté vrcholy.*

Definice: *Suffixový strom (ST)* pro slovo $\sigma \in \Sigma^*$ je komprimovaná trie pro $X = \{\alpha : \alpha \text{ je suffixem } \sigma\}$.

Pozorování: Vrcholy suffixového stromu (včetně skrytých) odpovídají prefixům suffixů slova σ, tedy všem jeho podslovům. Listy stromu jsou suffixy, které se v σ již nikde jinde nevyskytují (takovým suffixům budeme říkat *nevnořené*). Vnitřní vrcholy odpovídají *větvícím podslovům,* tedy podslovům $\alpha \subset \sigma$ takovým, že $\alpha a \subset \sigma$ i $\alpha b \subset \sigma$ pro nějaké dva různé znaky a, b.

Někdy může být nepraktické, že některé suffixy neodpovídají listům (protože jsou vnořené), ale s tím se můžeme snadno vypořádat: přidáme na konec slova σ nějaký znak \$, který se nikde jinde nevyskytuje. Neprázdné suffixy slova $\sigma\$$ odpovídají suffixům slova σ a žádný z nich nemůže být vnořený. Takový suffixový strom budeme značit ST\$.

Příklad:



Suffixy slova „baraba“: trie, suffixový strom, ST s dolarem

Nyní jak je to s konstrukcí suffixových stromů:

Lemma: Suffixový strom pro slovo σ délky n je reprezentovatelný v prostoru $\mathcal{O}(n)$.

Důkaz: Strom má $\mathcal{O}(n)$ listů a každý vnitřní vrchol má alespoň 2 syny, takže vnitřních vrcholů je také $\mathcal{O}(n)$. Hran je rovněž lineárně. Nálepky na hranách stačí popsat počáteční a koncovou pozicí v σ. ♡

Věta: Suffixový strom pro slovo σ délky n lze sestrojit v čase $\mathcal{O}(n)$.

Důkaz: Ve zbytku této kapitoly předvedeme dvě různé konstrukce v lineárním čase. ♡

Aplikace – co vše dokážeme v lineárním čase, když umíme lineárně konstruovat ST:

1. *Inverzní vyhledávání* (tj. předzpracujeme si v lineárním čase text a pak umíme pro libovolné slovo α v čase $\mathcal{O}(|\alpha|)$ rozhodnout, zda se v textu vyskytuje)$^{\langle 1\rangle}$ – stačí sestrojit ST a pak jej procházet od kořene. Také umíme najít všechny výskyty (odpovídají suffixům, které mají jako prefix hledané slovo, takže stačí vytvořit ST\$ a vypsat všechny listy pod nalezeným vrcholem) nebo přímo vrátit jejich počet (předpočítáme si pomocí DFS pro každý vrchol, kolik pod ním leží listů).
2. *Nejdelší opakující se podslovo* – takové podslovo je v ST\$ nutně větvící, takže stačí najít vnitřní vrchol s největší *písmenkovou hloubkou* (tj. hloubkou měřenou ve znacích místo ve hranách).
3. *Histogram četností podslov délky k* – rozřízneme ST\$ v písmenkové hloubce k a spočítáme, kolik původních listů je pod každým novým.
4. *Nejdelší společné podslovo* slov α a β – postavíme ST pro slovo $\alpha\$_1\beta\$_2$, jeho listy odpovídají suffixům slov α a β. Takže stačí pomocí DFS najít

$^{\langle 1\rangle}$ Čili přesný opak toho, co umí vyhledávací automat – ten si předzpracovává dotaz.

nejhlubší vnitřní vrchol, pod kterým se vyskytují listy pro α i β. Podobně můžeme sestrojit ST\$ pro libovolnou množinu slov.⟨2⟩

5. *Nejdelší palindromické podslovo* (tj. takové $\beta \subset \alpha$, pro něž je $\beta^R = \beta$) – postavíme společný ST\$ pro slova α a α^R. Postupně procházíme přes všechny možné středy palindromického podslova a všimneme si, že takové slovo je pro každý střed nejdelším společným prefixem podslova od tohoto bodu do konce a podslova od tohoto bodu pozpátku k začátku, čili nějakého suffixu α a nějakého suffixu α^R. Tyto suffixy ovšem odpovídají listům sestrojeného ST a jejich nejdelší společný prefix je nejbližším společným předchůdcem ve stromu, takže stačí pro strom vybudovat datovou strukturu pro společné předchůdce a s její pomocí dokážeme jeden střed prozkoumat v konstantním čase.
6. *Burrows-Wheelerova Transformace* [1] – jejím základem je lexikografické setřídění všech rotací slova σ, což zvládneme sestrojením ST pro slovo $\sigma\sigma$, jeho uříznutím v písmenkové hloubce $|\sigma|$ a vypsáním nově vzniklých listů v pořadí „zleva doprava“.

Cvičení: Zkuste vymyslet co nejlepší algoritmy pro tyto problémy bez použití ST.

Suffixová pole

V některých případech se hodí místo suffixového stromu používat kompaktnější datové struktury.

Notace: Pro slovo σ bude $\sigma[i]$ značit jeho i-tý znak (číslujeme od nuly), $\sigma[i : j]$ pak podslovo $\sigma[i]\sigma[i+1]\ldots\sigma[j-1]$. Libovolnou z mezí můžeme vynechat, proto $\sigma[i :]$ bude suffix od i do konce a $\sigma[: j]$ prefix od začátku do $j-1$. Pokud $j \leq i$, definujeme $\sigma[i : j]$ jako prázdné slovo, takže prázdný suffix můžeme například zapsat jako $\sigma[|\sigma| :]$.

$\mathrm{LCP}(\alpha, \beta)$ bude značit délku nejdelšího společného prefixu slov α a β, čili největší $i \leq |\alpha|, |\beta|$ takové, že $\alpha[: i] = \beta[: i]$.

Definice: *Suffixové pole (Suffix Array)* A_σ pro slovo σ délky n je posloupnost všech suffixů slova σ v lexikografickém pořadí. Můžeme ho reprezentovat například jako permutaci A čísel $0, \ldots, n$, pro níž $\sigma[A[0] :] < \sigma[A[1] :] < \ldots < \sigma[A[n] :]$.

Definice: *Pole nejdelších společných prefixů (Longest Common Prefix Array)* L_σ pro slovo σ je posloupnost, v níž $L_\sigma[i] := \mathrm{LCP}(A_\sigma[i], A_\sigma[i+1])$.

Věta: Suffixový strom pro slovo $\sigma\$$ a dvojici (A_σ, L_σ) na sebe lze v lineárním čase převádět.

Důkaz: Když projdeme ST($\sigma\$$) do hloubky, pořadí listů odpovídá posloupnosti A_σ a písmenkové hloubky vnitřních vrcholů v inorderu odpovídají L_σ. Naopak ST($\sigma\$$) získáme tak, že sestrojíme kartézský strom pro L_σ (získáme vnitřní vrcholy ST), doplníme do něj listy, přiřadíme jim suffixy podle A_σ a nakonec podle listů rekonstruujeme nálepky hran. ♡

⟨2⟩ Jen si musíme dát pozor, abychom si moc nezvětšili abecedu; jak moc si ji můžeme dovolit zvětšit, vyplyne z konkrétních konstrukcí.

Rekurzivní konstrukce

Ukážeme algoritmus, který pro slovo $\sigma \in \Sigma^*$ délky n sestrojí jeho suffixové pole a LCP pole v čase $\mathcal{O}(n + \mathrm{Sort}(n, \Sigma))$, kde $\mathrm{Sort}(\ldots)$ je čas potřebný pro setřídění n symbolů z abecedy Σ. V kombinaci s předchozími výsledky tedy dostaneme lineární konstrukci $\mathrm{ST}(\sigma)$ pro libovolnou fixní abecedu.

Algoritmus: (Konstrukce A a L podle Kärkkäinena a Sanderse [2])

1. Redukujeme abecedu na $1 \ldots n$: ve vstupním slovu je nejvýše n různých znaků, takže je stačí setřídit a přečíslovat.
2. Definujeme slova σ_0, σ_1, σ_2 následovně:

$$\begin{aligned}\sigma_0[i] &:= \langle \sigma[3i], \sigma[3i+1], \sigma[3i+2] \rangle \\ \sigma_1[i] &:= \langle \sigma[3i+1], \sigma[3i+2], \sigma[3i+3] \rangle \\ \sigma_2[i] &:= \langle \sigma[3i+2], \sigma[3i+3], \sigma[3i+4] \rangle\end{aligned}$$

   Všechna σ_k jsou slova délky $\approx n/3$ nad abecedou velikosti n^3. Dovolíme si mírně zneužívat notaci a používat symbol σ_k i jejich přepis do abecedy původní.
3. Zavoláme algoritmus rekurzivně na slovo $\sigma_0\sigma_1$, čímž získáme A_{01} a L_{01}. (Suffixy slova $\sigma_0\sigma_1$ odpovídají suffixům slov σ_0 a σ_1.)
4. Spočítáme pole P_0 a P_1, která nám budou říkat, kde se v A_{01} vyskytuje který suffix slov σ_0 a σ_1. Tedy $A_{01}[P_0[i]] = i$, $A_{01}[P_1[i]] = i + |\sigma_0|$. Jinými slovy, P_0 a P_1 budou části inverzní permutace k A_{01}. Všimněte si, že platí $P_i[x] < P_j[y]$ právě tehdy, když $\sigma_i[x :] < \sigma_j[y :]$, takže suffixy slov σ_0 a σ_1 od této chvíle umíme porovnávat v čase $\mathcal{O}(1)$.
5. Vytvoříme A_2 (suffixové pole pro σ_2): Jelikož $\sigma_2[i :] = \sigma[3i+2 :] = \sigma[3i+2]\sigma[3i+3 :] = \sigma[3i+2]\sigma_0[i+1 :]$, odpovídá lexikografické pořadí suffixů $\sigma_2[i :]$ pořadí dvojic $(\sigma[3i+2], P_0[i+1])$. Tyto dvojice ovšem můžeme setřídit dvěma průchody přihrádkového třídění.
6. Slijeme A_{01} a A_2 do A: sléváme dvě setříděné posloupnosti, takže stačí umět jejich prvky v konstantním čase porovnat:

$$\begin{aligned}\sigma_0[i :] < \sigma_2[j :] &\Leftrightarrow \sigma[3i :] < \sigma[3j+2 :] \\ &\Leftrightarrow \sigma[3i]\,\sigma_1[i :] < \sigma[3j+2]\,\sigma_0[j+1 :], \\ \sigma_1[i :] < \sigma_2[j :] &\Leftrightarrow \sigma[3i+1 :] < \sigma[3j+2 :] \\ &\Leftrightarrow \sigma[3i+1]\,\sigma[3i+2]\,\sigma_0[i+1 :] < \\ &\quad \sigma[3j+2]\,\sigma[3j+3]\,\sigma_1[j+1 :].\end{aligned}$$

   Pokaždé tedy porovnáme nejvýše dvě dvojice znaků a pak dvojici suffixů slov σ_0 a σ_1, k čemuž nám pomohou pole P_0 a P_1.
7. Dopočítáme L:
8. Pokud v A sousedí suffix slova $\sigma_{0,1}$ se suffixem slova $\sigma_{0,1}$, sousedí tyto dva suffixy i v A_{01}, takže jejich LCP najdeme přímo v L.

9. Setkají-li se dva suffixy slova σ_2, všimneme si, že $\sigma_2[i:] = \sigma[3i+2:] = \sigma[3i+2]\,\sigma_0[i+1:]$. $\mathrm{LCP}(\sigma_2[i:], \sigma_2[j:])$ je tedy buďto 0 (pokud $\sigma[3i+2] \neq \sigma[3j+2]$), nebo $1+3\cdot\mathrm{LCP}(\sigma_0[i+1:], \sigma_0[j+1:])$, případně totéž zvýšené o 1 nebo 2, pokud se trojznaky v σ_0 následující po LCP zčásti shodují. Přitom $\mathrm{LCP}(\sigma_0[p:], \sigma_0[q:])$ spočítáme pomocí L. Je to totiž minimum intervalu v L mezi indexy $P_0[p]$ a $P_0[q]$. To zjistíme v konstantním čase pomocí struktury pro intervalová minima.
10. Pokud se setká suffix slova $\sigma_{0,1}$ se suffixem slova σ_2, stačí tyto suffixy přepsat podobně jako v 6. kroku a problém tím opět převést na výpočet LCP dvou suffixů slov $\sigma_{0,1}$.

Analýza časové složitosti: Třídění napoprvé trvá $\mathrm{Sort}(n, \Sigma)$, ve všech rekurzivních voláních už je lineární (trojice čísel velikosti $\mathcal{O}(n)$ můžeme třídit tříprůchodovým přihrádkovým tříděním s $\mathcal{O}(n)$ přihrádkami). Z toho dostáváme:

$$T(n) = T(2/3 \cdot n) + \mathcal{O}(n), \text{ a tedy } T(n) = \mathcal{O}(n).$$

♡

Ukkonenova inkrementální konstrukce

Ukkonen popsal algoritmus [3] pro konstrukci suffixového stromu bez dolarů, pracující inkrementálně: Začne se stromem pro prázdné slovo a postupně na konec slova přidává další znaky a přepočítává strom. Každý znak přitom přidá v amortizovaně konstantním čase. Pro slovo σ tedy dokáže sestrojit ST v čase $\mathcal{O}(|\sigma|)$.

Budeme předpokládat, že hrany vedoucí z jednoho vrcholu je možné indexovat jejich prvními písmeny – to bezpečně platí, pokud je abeceda pevná; není-li, můžeme si pomoci hešováním.

Pozorování: Když slovo σ rozšíříme na σa, ST se změní následovně:

1. Všechny stávající vrcholy stromu (včetně skrytých) odpovídají podslovům slova σ. Ta jsou i podslovy σa, takže se budou nacházet i v novém stromu.
2. Pokud β bylo větvící slovo, zůstane nadále větvící – tedy vnitřní vrcholy ve stromu zůstanou.
3. Každý nový suffix βa vznikne prodloužením nějakého původního suffixu β. Přitom:
   - Pokud byl β nevnořený suffix (čili byl reprezentovaný listem), ani βa nebude vnořený. Z toho víme, že listy zůstanou listy, pouze jim potřebujeme prodloužit nálepky. Aby to netrvalo příliš dlouho, zavedeme *otevřené hrany,* jejichž nálepka říká „od pozice i do konce“. Listy se tak o sebe postarají samy.
   - Pokud β byl vnořený suffix (tj. vnitřní či skrytý vrchol):
     - Buď se βa vyskytuje v σ, a tím pádem je to vnořený suffix nového slova a strom není nutné upravovat;

- nebo se βa v σ nevyskytuje – tehdy pro něj musíme založit nový list s otevřenou hranou a případně i nový vnitřní vrchol, pod nímž bude tento list připojen.

Víme tedy, co všechno je při rozšíření slova potřeba ve stromu upravit. Zbývá vyřešit, jak to udělat efektivně.

Vnořené suffixy: Především potřebujeme umět rozpoznat, které suffixy jsou vnořené a které nikoliv. K tomu se hodí všimnout si, že vnořené suffixy tvoří souvislý úsek:

> **Lemma:** Je-li α vnořený suffix slova σ a β je suffix slova α, pak β je v σ také vnořený.
>
> *Důkaz:* Ve slově sigma se vyskytuje αx a αy pro nějaké dva různé znaky x a y. Každý z těchto výskytů přitom končí výskytem slova β, jednou následovaným x, podruhé y. ♡

Stačí si tedy zapamatovat nejdelší vnořený suffix slova σ. Tomu budeme říkat *aktivní suffix* a budeme ho značit $\alpha(\sigma)$. Libovolný suffix $\beta \subseteq \sigma$ pak bude vnořený právě tehdy, když $|\beta| \leq |\alpha(\sigma)|$.

Aktivní suffix tedy tvoří hranici mezi nevnořenými a vnořenými suffixy. Jak se tato hranice posune, když slovo σ rozšíříme? Na to je odpověď snadná:

> **Lemma:** Pro každé σ, a platí: $\alpha(\sigma a)$ je suffixem $\alpha(\sigma)a$.
>
> *Důkaz:* $\alpha(\sigma a)$ i $\alpha(\sigma)a$ jsou suffixy slova σa, a proto stačí porovnat jejich délky. Slovo $\beta :=$ „$\alpha(\sigma a)$ bez koncového a“ je vnořeným suffixem v σ, takže $|\beta| \leq |\alpha(\sigma)|$, a tedy také $|\alpha(\sigma a)| = |\beta a| \leq |\alpha(\sigma)a|$. ♡

Hranice se tedy může posouvat pouze doprava, případně zůstat na místě. Toho lze snadno využít.

Idea algoritmu: Udržujeme si $\alpha = \alpha(\sigma)$ a při přidání znaku a zkontrolujeme, zda αa je stále vnořený suffix. Pokud ano, nic se nemění, pokud ne, přidáme nový list a případně také vnitřní vrchol, α zkrátíme zleva o znak a testujeme dál.

Analýza: Po přidání jednoho znaku na konec slova σ provedeme amortizovaně konstantní počet úprav stromu (každá úprava slovo α zkrátí, po všech úpravách přidáme k α jediný znak). Tudíž stačí ukázat, jak provést každou úpravu v (amortizovaně) konstantním čase. K tomu potřebujeme šikovnou reprezentaci slova α, která bude umět efektivně prodlužovat zprava, zkracovat zleva a testovat existenci vrcholu ve stromu.

Definice: *Referenční pár* pro slovo $\alpha \subseteq \sigma$ je dvojice (π, τ), v níž π je vrchol stromu, τ libovolné slovo a $\pi\tau = \alpha$. Navíc víme, že $\tau \subseteq \sigma$, takže si τ stačí pamatovat jako dvojici indexů ve slově σ.

Referenční pár je *kanonický,* pokud neexistuje hrana vedoucí z vrcholu π s nálepkou, která by byla prefixem slova τ. (Všimněte si, že taková hrana se pozná podle toho, že první znak nálepky se shoduje s prvním znakem slova τ a nálepka není delší než slovo τ. Shodu ostatních znaků není nutné kontrolovat.)

Pozorování: Ke každému slovu $\alpha \subseteq \sigma$ existuje právě jeden kanonický referenční pár, který ho popisuje. To je ze všech referenčních párů pro toto slovo ten s nejdelším π (nejhlubším vrcholem).

Definice: Zpětná hrana $back(\pi)$ vede z vrcholu π do vrcholu, který je zkrácením slova π o jeden znak zleva. (Nahlédneme, že takový vrchol musí existovat: pokud je π vnitřní vrchol, pak je slovo π větvící, takže každý jeho suffix musí také být větvící, a tím pádem musí odpovídat nějakého vrcholu.)

Operace s referenčními páry: S referenčním párem (π, τ) popisujícím slovo α potřebujeme provádět následujicí operace:

- *Přidání znaku a na konec:* Připíšeme a na konec slova τ. To je jistě referenční pár pro αa, ale nemusí být kanonický. Přitom můžeme snadno ověřit, zda se αa ve stromu nachází, a případně operaci odmítnout.
- *Odebrání znaku ze začátku:* Pokud π není kořen stromu, položíme $\pi \leftarrow back(\pi)$ a zachováme τ. Pokud naopak je π prázdný řetězec, odebereme z τ jeho první znak (to lze udělat v konstantním čase, protože τ je reprezentované dvojicí indexů do σ).
- *Převedení na kanonický tvar:* Obě předchozí operace mohou vytvořit referenční pár, který není kanonický. Pokaždé proto kanonicitu zkontrolujeme a případně pár upravíme. Ověříme, zda hrana z π indexovaná písmenem a není dost krátká na to, aby byla prefixem slova τ. Pokud je, tak se po této hraně přesuneme dolů, čímž π prodloužíme a τ zkrátíme, a proces opakujeme. Jelikož tím pokaždé τ zkrátíme a kdykoliv jindy se τ prodlouží nejvýše o 1, mají všechny převody na kanonický tvar amortizovaně konstantní složitost.

Nyní již můžeme doplnit detaily, získat celý algoritmus a nahlédnout, že pracuje v amortizovaně konstantním čase.

Algoritmus podrobněji:

1. *Vstup:* $\alpha = \alpha(\sigma)$ reprezentovaný jako kanonický referenční pár (π, τ), T suffixový strom pro σ spolu s hranami *back*, nový znak a.
2. Zjistíme, jestli αa je přítomen ve stromu, a případně ho založíme:
3. Pokud $\tau = \varepsilon$: ($\alpha = \pi$ je vnitřní vrchol)
4. Vede-li z vrcholu π hrana s nálepkou začínající znakem a, pak je přítomen.
5. Nevede-li, není přítomen, a tak přidáme novou otevřenou hranu vedoucí z π do nového listu.
6. Pokud $\tau \neq \varepsilon$: (α je skrytý vrchol)
7. Najdeme hranu, po níž z π pokračuje slovo τ (která to je, poznáme podle prvního znaku slova τ).
8. Pokud v popisce této hrany po τ následuje znak a, pak je αa přítomen.
9. Pokud nenásleduje, tak nebyl přítomen, čili tuto hranu rozdělíme: přidáme na ni nový vnitřní vrchol, do nějž povede

hrana s popiskou τ a z něj zbytek původní hrany a otevřená hrana do nového listu.

10. Pokud αa nebyl přítomen, tak α zkrátíme a vrátíme se na krok 2.
11. Nyní víme, že αa již byl přítomen, takže upravíme referenční pár, aby popisoval αa.
12. Dopočítáme zpětné hrany (viz níže).
13. *Výstup:* $\alpha = \alpha(\sigma a)$ jako kanonický referenční pár (π, τ), T suffixový strom pro σa a jeho zpětné hrany *back*.

Zpětné hrany: Zbývá dodat, jak nastavovat novým vrcholům jejich zpětné hrany. To potřebujeme jen pro vnitřní vrcholy (na zpětné hrany z listů se algoritmus nikdy neodkazuje). Všimneme si, že pokud jsme založili vrchol, odpovídá tento vrchol vždy současnému α a zpětná hrana z něj povede do zkrácení slova α o znak zleva, což je přesně vrchol, který založíme (nebo zjistíme, že už existuje) v příští iteraci hlavního cyklu. V další iteraci ještě určitě nebudeme tuto hranu potřebovat, protože π vždy jen zkracujeme, a tak můžeme vznik zpětné hrany o iteraci zpozdit. Výroba zpětné hrany tedy bude také trvat jen konstantně dlouho.

Literatura

[1] M. Burrows and D. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Systems Research Center, 1994.

[2] J. Kärkkäinen and P. Sanders. Simple linear work suffix array construction. In *Proc. 13th International Conference on Automata, Languages and Programming*. Springer Verlag, 2003.

[3] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.