7. Vyhledávání v textu *(zapsali J. Kunčar, M. Demin a J. Chludil)*

Na minulých prednáškách jsme si ukázali, jak se v textu (seně) vyhledává slovo (jehlu). Teď si ovšem úlohu zobecníme a ukážeme si, jak v kupce sena hledat současně více než jednu jehlu.

Hledání výskytu všech slov

- $\iota_1, \ldots, \iota_k$ – vyhledávaná slova (jehly) délek $J_i = |\iota_i|$
- σ – text (seno) délky $S = |\sigma|$

Nejprve si řekneme, jak chceme, aby vypadal výstup. Výstupem pro nás budou všechny uspořádané dvojice (i, j) (i je index jehly, kterou jsme nalezli, a j je počáteční pozice v seně, kde se jehla nachází) takové, že

$$\iota_i = \sigma[j : j + J_i].$$

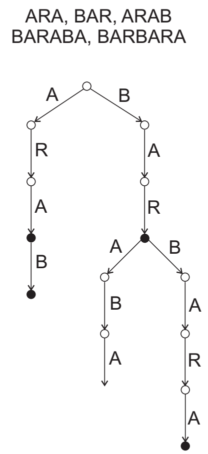
Postavme si proto vyhledávací automat podobný tomu, který jsme viděli na minulé přednášce. Tento automat nám všechny takové uspořádané dvojice najde.

Vyhledávací automat

Vyhledávací automat je strom[1], kde každý vrchol může mít stupeň až do velikosti abecedy a kde jednotlivé hrany odpovídají písmenům této abecedy. Vrcholy, ve kterých končí slovo, jsou označené (na obrazcích černě). Dále si časem do tohoto vyhledávacího stromu přidáme zpětné hrany a „zkratky“.

Stavy automatu jsou určeny vrcholy stromu, pro které platí rovněž stejný *invariant* z předchozí přednášky.

Zpětná hrana $z(\alpha)$:= nejdelší vlastní suffix[2] slova α, který je stavem.



Vyhledávací automat

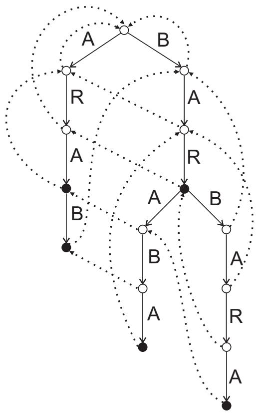
[1] http://en.wikipedia.org/wiki/Trie
[2] definováno na minulé přednášce

Výstup z automatu

Při vypisování výsledků mužeme narazit na určité problémy, které jsou dobře vidět na následujícím obrázku. První problém určitě nastane, protože v automatu není přesně řečeno, které slovo končí v jakém vrcholu. Například ve stavu, kde končí slovo BARBARA, končí také slovo ARA, ale o tom nevíme. Druhý problém nastává, když v automatu není informace o konci slova. Příkladem je seno BARAB (jednoduché k nahlédnutí, viz obrázek). Teď nám nezbývá nic jiného, než najít řešení těchto záludných problémů. Řešeší se nám naskýtá hned několik:

- Projdeme všechy zpětné hrany a vypíšeme slova, jež v daných stavech končí. Toto řešení funguje, ale je pomalé, protože pokaždé procházíme všechny zpětné hrany.
- Pro následující řešení, jenž spočívá v nalezení zkratek ve stromě, si zavedeme toto značení:
  $slovo(s)$ = index slova ι, které končí ve stavu s, nebo $\emptyset$
  $out(s)$ = nejbližší vrchol, do kterého se lze z s dostat po zpětných hranách a *slovo(v)* $\neq 0$ (končí tam slovo)

ARA, BAR, ARAB
BARABA, BARBARA



Vyhledávací automat se zpětnými hranami

Podle posledního bodu vytvoříme algoritmus na vyhledávání „jehel v seně“.

1. $s \leftarrow kořen$ (s bude aktuální stav vyhledávacího automatu).
2. Procházíme všechny písmena c v seně σ:
3. $\quad s \leftarrow krok(s, c)$.
4. $\quad$ Je-li $slovo(s) \neq 0$, vypíšeme $slovo(s)$.
5. $\quad v \leftarrow out(s)$.
6. $\quad$ Dokud $v \neq 0$:
7. $\quad\quad$ Vypíšeme $slovo(v)$.

8. $v \leftarrow out(v)$.

krok(s,c):= jeden *krok* vyhledávacího automatu:

1. Dokud $\nexists f(s,c) \wedge s \neq \mathit{kořen}$: $s \leftarrow z(s)$.
2. Pokud $\exists f(s,c)$: $s \leftarrow f(s,c)$.
3. Vrátíme s.

Reprezentace v paměti

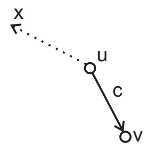
První možnost, jak reprezentovat vyhledávací automat, je jednorozměrné pole vrcholů stromu, v němž ukládáme seznam synů pro každý vrchol. Je to jednoduchá varianta, ale má nevýhodu pro velké abecedy, protože procházení seznamu synů může trvat neúměrně dlouho. Proto se nabízí druhá možnost a to hashovací tabulka $(stav, znak) \rightarrow f(stav, znak)$, kde se „ztratí" používání hashovací funkce.

Složitost

- Kroky 2.–5. mají časovou složitost $\mathcal{O}(|\sigma|)$, kterou jednoduše dokážeme pomocí potenciálu – počet kroků nahoru je menší nebo roven počtu kroků dolů. A to je maximálně S.
- Kroky 6.–8. mají časovou složitost $\mathcal{O}(\textit{počet výskytů})$, protože rychleji doopravdy nelze všechny výskyty vypsat.

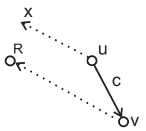
Konstrukce automatu (Aho, Corasicková)

1. Postavíme strom dopředných hran, $r \leftarrow$ kořen stromu.
2. Spočteme $slovo(*)$ – označíme si stavy, kde končí slova.
3. Spočteme $z(*)$: $z(\beta) = \alpha(\beta[1:])$:
    - $z(\beta) = \alpha(\beta[1:])$ – všechny zpětné hrany vedou do vyšších hladin
    - $z(v) = krok(z(u), c)$

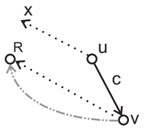


$z(v) = krok(z(u), c)$

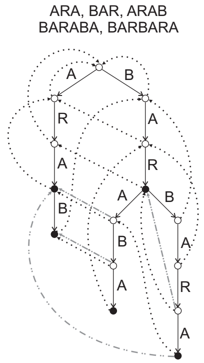
4. $z(r) \leftarrow 0$, do fronty Q přiřadíme všechny syny r, pro všechny v prvky $Q : z(v) \leftarrow r$.
5. Dokud fronta Q není prázdná:
6. $u \leftarrow$ vybereme z Q.
7. Pro syny v vrcholu u:
8. $R \leftarrow krok(z(u))$ [znak na hraně uv].



$z(v) = R$



Nastavení $out(v)$



Vyhledávací automat – kompletní

9. $z(v) \leftarrow R$.
10. Je-li $slovo(R) \neq 0 \Rightarrow out(v) \leftarrow R$, jinak $out(v) \leftarrow out(R)$.

Věta: Algoritmus Aho-Corasicková najde všechny výskyty slov $\iota_1, \ldots, \iota_k$ ve slově σ v čase $\mathcal{O}(\sum_i |\iota_i| + |\sigma| + \textit{počet výskytů})$.

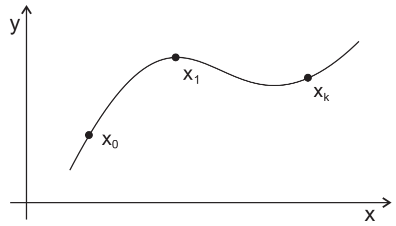
Polynomy a násobení

Mějme dva polynomy definované jako:

$$P(x) = \sum_{j=0}^{n-1} p_j x^j, \quad Q(x) = \sum_{j=0}^{n-1} q_j x^j.$$

Násobení dvou polynomů $R = P \cdot Q$ je ekvivalentní s operací $R = \sum_{j,k} p_j q_k x^{j+k}$. Přičemž na vypočítání členu $r_l = \sum_{j=0}^{l} p_j q_{l-j}$ použijeme $\Theta(n)$ operací, tedy na spočítaní celého polynomu R potřebujeme $\Theta(n^2)$ operací.

Podíváme se na jinou možnost, jak tento problém řešit. Poslouží nám k tomu následující věta o jednoznačné existenci polynomu nejvýše k-tého stupně, pokud známe hodnoty ve více než k bodech.



Polynom

Věta: Jsou-li $x_0, \ldots, x_k \in \mathbb{R}$ navzájem ruzná a $y_0, \ldots, y_k \in \mathbb{R}$, pak $\exists!$ polynom P stupně $\leq k : \forall j : P(x_j) = y_j$.

Plán:

Nechť $k = 2^{n-1}$. Zvolíme čísla $x_0, \ldots, x_k$ libovolná, ale různá, a spočteme $P(x_0)$, $\ldots$, $P(x_k)$ a $Q(x_0), \ldots, Q(x_k)$. Poté $\forall j : y_j = P(x_j)Q(x_j)$ musíme najít polynom R stupně $\leq k : \forall j : R(x_j) = y_j$.

Vyhodnocování polynomů (metodou Rozděl a panuj)

BÚNO $n = 2^m$. Uvažme polynom:

$$P(x) = p_0 x^0 + p_1 x^1 + \ldots + p_{n-1} x^{n-1}.$$

Tento polynom si mužeme rozdělit na dvě časti. V levé budeme mít členy se sudými exponenty a v pravé budou členy s exponenty lichými:

$$P(x) = (p_0 + p_2 x^2 + \ldots + p_{n-2} x^{n-2}) + (p_1 x^1 + p_3 x^3 + \ldots + p_{n-1} x^{n-1}).$$

Z pravé strany můžeme vytknout x a dostaneme:

$$P(x) = (p_0 + p_2 x^2 + \ldots + p_{n-2} x^{n-2}) + x(p_1 + p_3 x^2 + \ldots + p_{n-1} x^{n-2}),$$

$$\vdots$$

$$P(x) = L(x^2) + xN(x^2),$$
$$P(-x) = L(x^2) - xN(x^2),$$

kde $L(x)$ a $N(x)$ jsou polynomy stupně $n/2$. Umocněním x^2 se nám poruší párování x a $-x$, proto musíme počítat v $\mathbb{C}$ místo $\mathbb{R}$. V tomto případě jsme z polynomu s n koeficienty v n bodech dostali 2 polynomy s $n/2$ koeficienty v $n/2$ bodech. Z toho vyplývá časová složitost definována vztahem:

$$T(n) = 2T(n/2) + \mathcal{O}(n).$$

Ten můžeme vyřešit s použitím Master Theoremu z ADS I a dostaneme:

$$T(n) = \mathcal{O}(n \log n).$$