Datové struktury 1

9. Přednáška 2.12.

Pavel Veselý

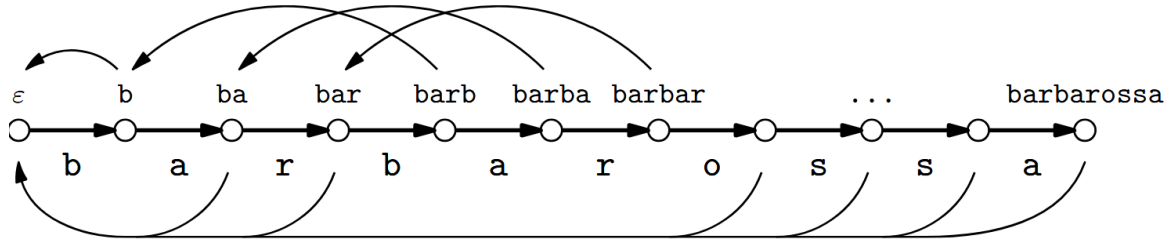
Plán: řetězcové DS

Sufixové stromy a pole

- … aneb seřadíme sufixy a vyřešíme téměř vše v lineárním čase ☺

Vyhledávání v textu (jehly v kupce sena)

- Dán text S („seno“) a jehla J
- Cíl: najít všechny výskyty J v textu = sufixy S začínající na J
- Algoritmy:
  - Knuth–Morris–Pratt (KMP):
    - pomocí automatu
  - Rabinův-Karpův
    - Pomocí „rolling hash“
  - Aho-Corasick: rozšíření KMP na více jehel



Obrázek 13.1: Vyhledávací automat pro slovo barbarossa

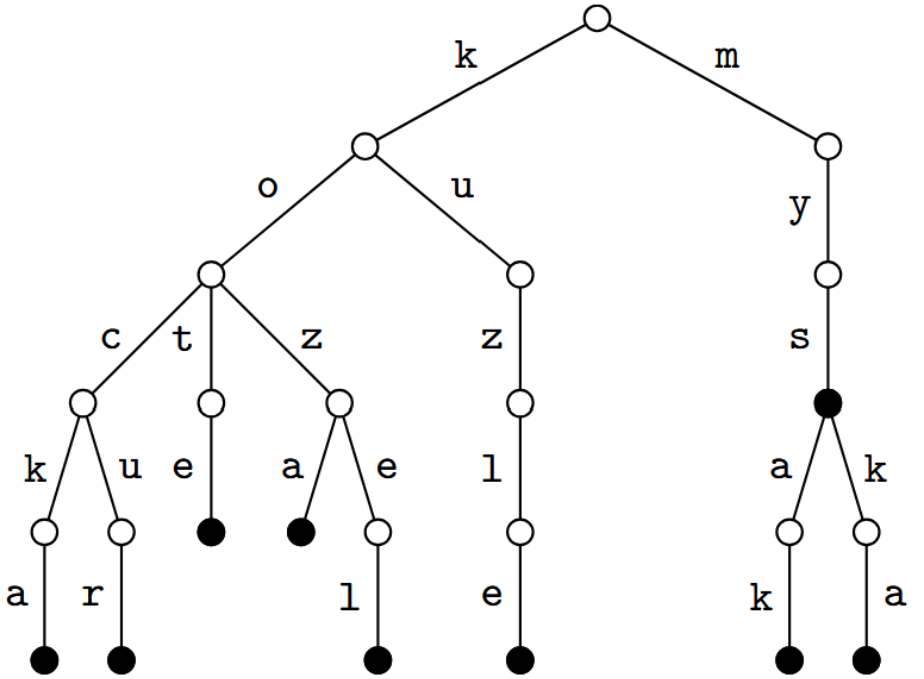
- Co když nám chodí jehly jako dotazy?
  - Z textu tedy chceme vytvořit DS, která pro zadanou jehlu najde její výskyty

Značení pro řetězce

- $\sum$ = abeceda znaků, $\sum^*$ jsou konečné řetězce nad $\sum$
- S, α, β = text, řetězec, slovo, ...; $|S|$ = délka S (počet znaků)
- ε = prázdné slovo délky 0
- $S[i]$ = znak na pozici $i = 0, \dots, |S| - 1$
- $S[i : j]$ = podřetězec (podslovo) znaků na pozicích $i, i + 1, \dots, j$-1
- $S[: j]$ = prefix z prvních j znaků
- $S[i :]$ = sufix z posledních $|S| - i$ znaků, tedy od znaku i

Opakování: trie = písmenkový strom (prefixový strom)

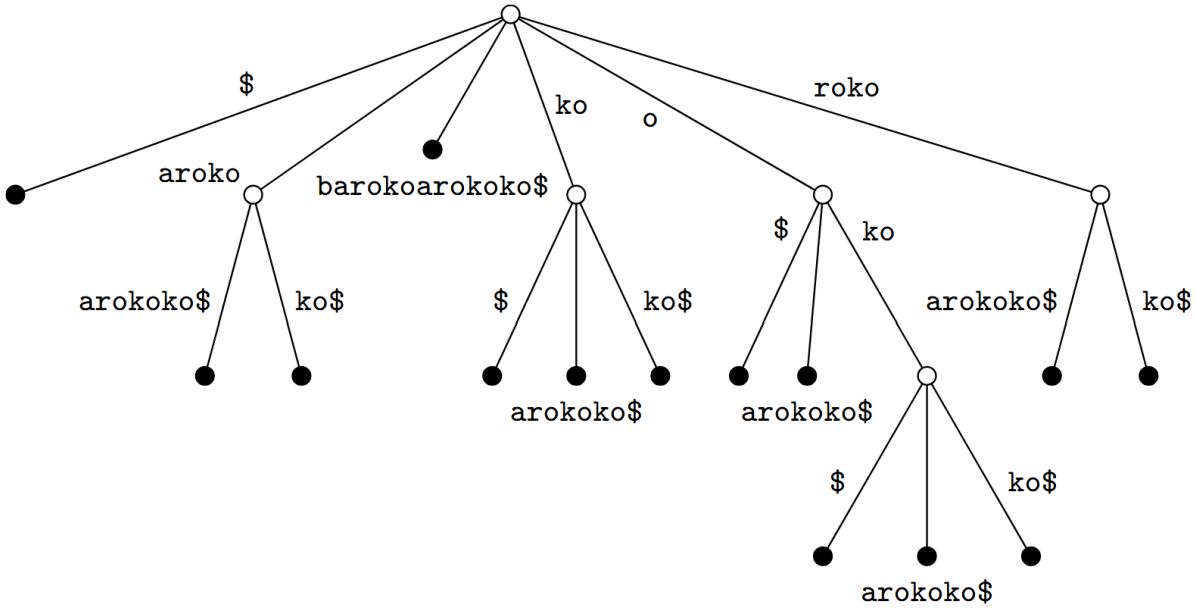
- Dána množina slov (řetězců)
- Každý uzel odpovídá prefixu nějakého slova
- Uzel:
  - Pole indexované písmeny $\sum$
  - Příznak konce slova
- Komprimovaná trie
  - Odstraníme uzly s jedním synem
  - Hrana
    - dva indexy na začátek a konec podslova



Obrázek 4.4: Písmenkový strom pro slova `kocka`, `kocur`, `kote`, `koza`, `kozel`, `kuzle`, `mys`, `mysak`, `myska`

Sufixový strom pro S

= komprimovaná trie všech sufixů $S\$$



Obrázek 13.4: Suffixový strom pro slovo barokoarokoko

Jak spočíst sufixový strom z S?

- Snadno v kvadratickém čase ☹
- V lineárním čase – později

K čemu je dobrý sufixový strom?

- Vyhledání výskytů jehly
- Nejdelší opakující se podslovo
- …

Sufixové pole X (a spol.)

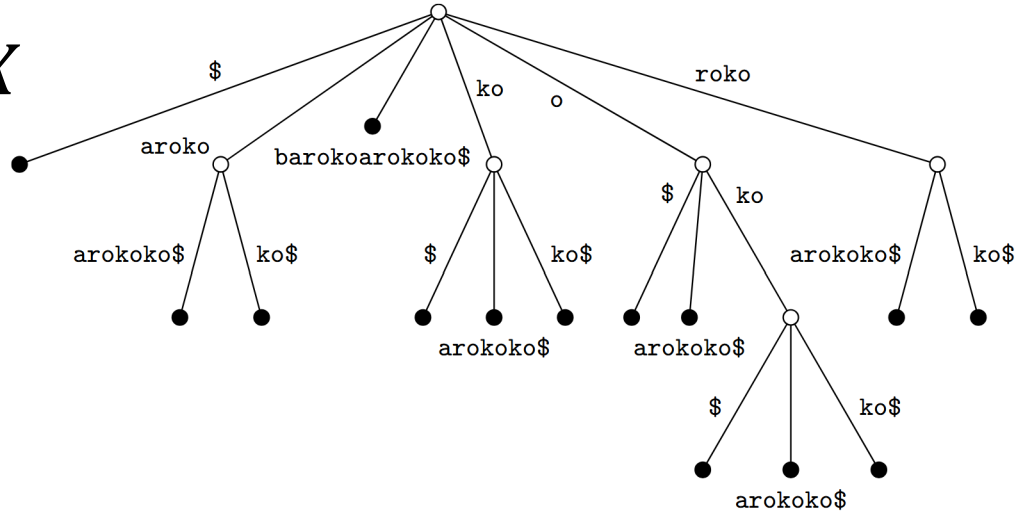
- Udává lexikografické pořadí sufixů S
  - $X[i] =$ pozice začátku i-tého sufixu S v lexikografickém pořadí
- Rankové pole R = inverzní permutace k X
  - $R[i] =$ kolikátý v lexikografickém pořadí je sufix $S[i:]$
- LCP pole L (pole společných prefixů, “longest common prefix”)
  - $\mathrm{LCP}(\alpha, \beta) =$ nejdelší společný prefix slov α a β
  - $L[i] = \mathrm{LCP}(S[X[i]:], S[X[i+1]:])$
- Aplikace:
  - Vyhledávání výskytu jehly
  - Počet k-gramů = počet různých podslov délky k
  - Nejdelší opakující se podslovo
  - Nejdelší společné podslovo
  - …

Jsou lepší sufixové stromy nebo pole?

Algoritmy pro sufixové pole/stromy

i	$X[i]$	$R[i]$	$L[i]$	*suffix*
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

1. Výpočet sufixového pole X:
   - Tedy seřazení všech sufixů S
   a) Zdvojováním v čase $O(n \log n)$
      - Seřadíme dle prvních 2^i písmen pro $i = 0, 1, 2, \ldots, \lceil \log_2 n \rceil$
   b) Algoritmus Skew v čase $O(n)$ – pro abecedy, které lze setřídit v $O(n)$
      - [Kärkkäinen&Sanders '03]
2. Výpočet LCP pole ze sufixového pole X
   - Kasaiův algoritmus
3. Sufixový strom ↔ sufixové +LCP pole
   - Lze v čase $O(n)$ – cvičení



Obrázek 13.4: Suffixový strom pro slovo barokoarokoko

Řazení sufixů zdvojováním – pseudokód

Algoritmus SuffixovéPole
Vstup: Řetězec α délky $n > 0$

1. Vytvoříme pole $X[0 \ldots n]$ a $R[0 \ldots n]$.
2. První fáze:
3. $D \leftarrow \{(\alpha[i], i) \mid i = 0, \ldots, n\}$, kde $\alpha[n]$ považujeme za nejmenší znak.
4. Setřídíme D lexikograficky.
5. Pro $j = 0, \ldots, n$:
6. $X[j] = D[j][1]$
7. Je-li $j = 0$ nebo $D[j][0] \neq D[j-1][0]$:
8. $R[X[j]] = j$
9. Jinak:
10. $R[X[j]] = R[X[j-1]]$
11. $k \leftarrow 1$ ◁ *číslo aktuální fáze*
12. Dokud $k \leq n$: ◁ *z polí* X_k *a* R_k *počítáme* X_{2k} *a* R_{2k}
13. $D \leftarrow \{(R[i], R[i+k], i) \mid i = 0, \ldots, n\}$, kde $R[i+k] = 0$ pro $i+k > n$.
14. Setřídíme D lexikograficky.
15. Pro $j = 0, \ldots, n$:
16. $X[j] = D[j][2]$
17. Je-li $j = 0$ nebo $(D[j][0], D[j][1]) \neq (D[j-1][0], D[j-1][1])$:
18. $R[X[j]] = j$
19. Jinak:
20. $R[X[j]] = R[X[j-1]]$
21. $k \leftarrow 2k$

Příště

- Lineární algoritmus Skew pro konstrukci sufixového pole
  - stručně
- Ještě lepší DS pro řetězce:
  - Burrows-Wheelerova transformace
- Geometrické datové struktury