Kompaktna priponska drevesa

AVTOR: JANI SUBAN, 89222015

MENTOR: PROF. DR. ANDREJ BRODNIK

Kazalo

Opis problema

Uporaba v procesiranju Naravnih jezikov ter v Bioinformatiki

Problemi, ki jih rešujemo z priponskimi drevesi:

- Iskanje vzorcev v besedilu

Brez priponskega drevesa lahko vsak vzorec najdemo v času O(n+m)

- Knuth–Morris–Pratt algoritem
- Algoritem z končnim avtomatom

Definicije

Besedilo je polje črk T[1..n]

- $T[i] \in \Sigma$
- Σ je poljubna abeceda

Vsako besedilo se konča z posebnim znakom $

- Ni del abecede
- Se ne pojavi nikjer drugje v besedilu

Priponska dreva

Definicija: Priponsko drevo podpira naslednje operacije:

1. koren()
2. jeList(v)
3. otrok(v,c)
4. sorojenec(v)
5. starš(v)
6. povezava(v,i)
7. SVišina(v)
8. lca(v,w): najnižji skupni predhodnik
9. sl(v): prionska povezava ali *suffix link*

Definicije

Priponska povezava vozlišča v z nizom $y=\alpha x$

- α je črka v $\sum$
- Kaže na vozlišče w z nizom x
- x je lahko prazen niz

Uporablja:

- Izgradnja priponskega drevesa
- Iskanje v drevesu

Definicije

Kompaktna podatkovna struktura potrebuje manj kot O(n) bitov

Struktura podpira iste operacije kot originalna predstavitev

Priponska dreva

Implementacija abstraktne podatkovne strukture

- Vsak list predstavlja pripono
- Dodatne povezave med notranjimi vozlišči za hitrejšo izgradnjo drevesa

Priponska dreva

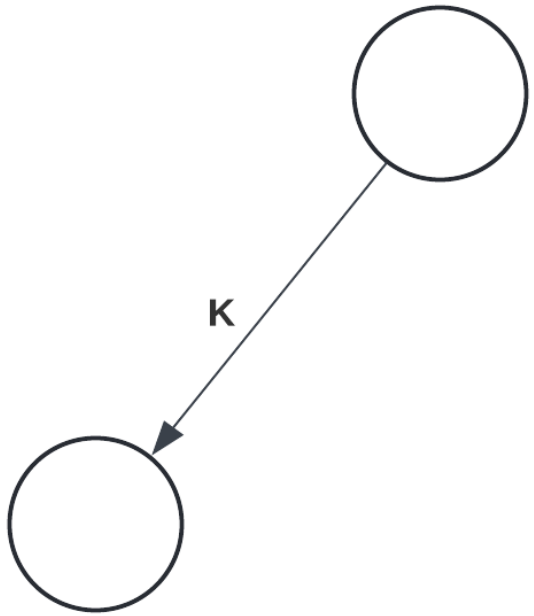
Čas izgradnje priponskega drevesa je O(n)

Dva načina izgradnje drevesa:

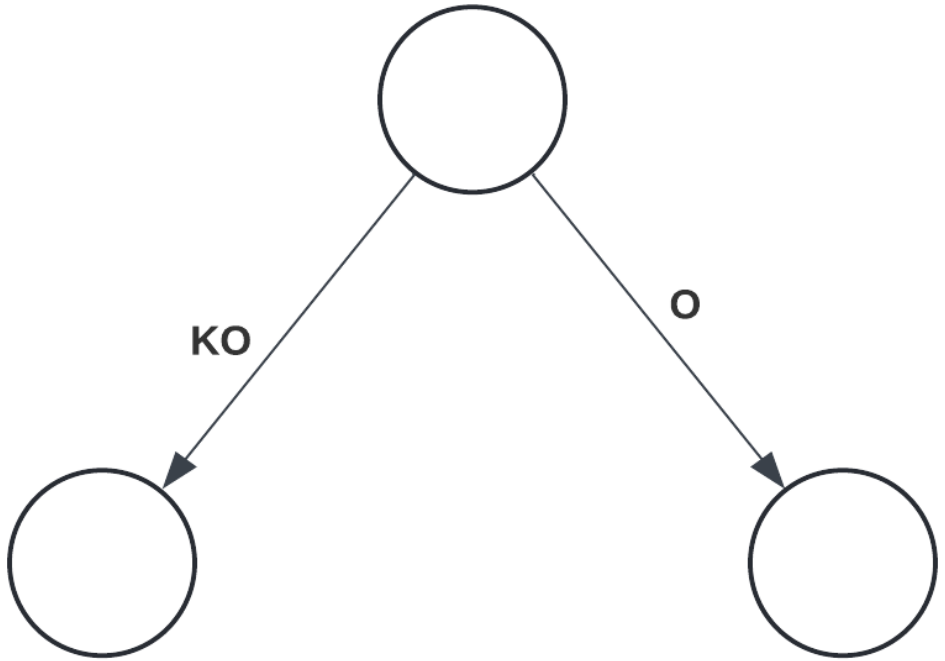
- Vzvratna izgradnja drevesa – McCreightov algoritem
- On-line konstrukcija drevesa – Ukkonenov algoritem

Primer On-line konstrukcije drevesa za besedo: kokoš

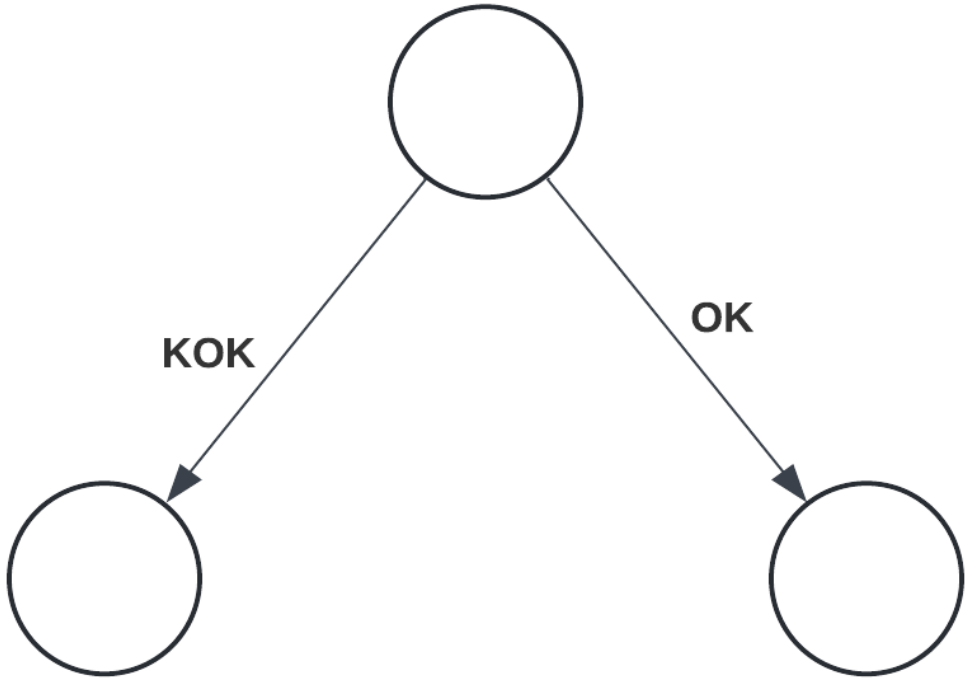
Primer konstrukcije priponskega drevesa



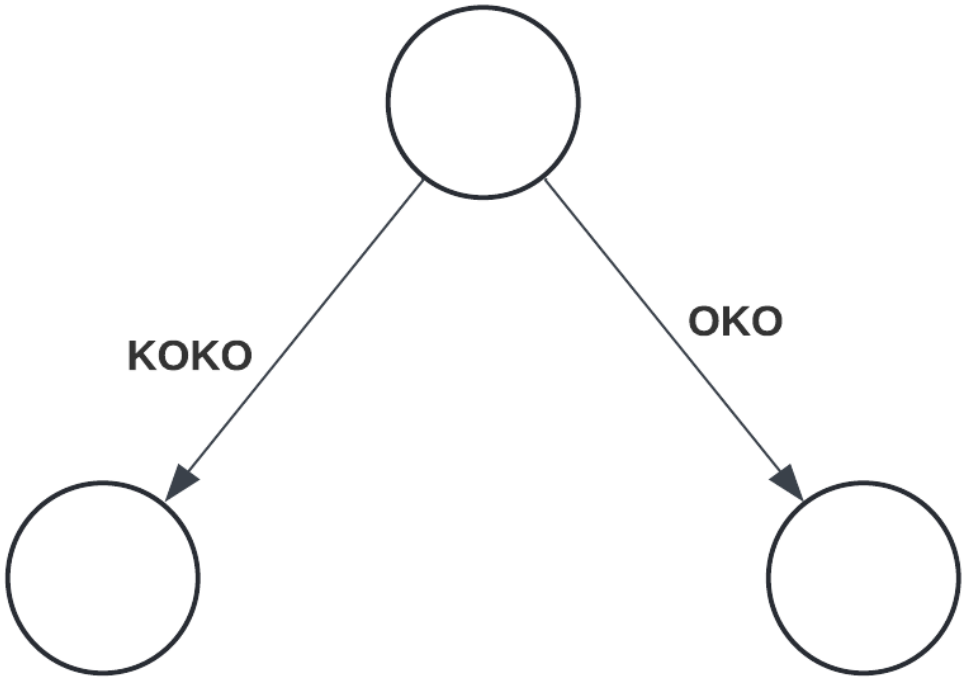
Primer konstrukcije priponskega drevesa



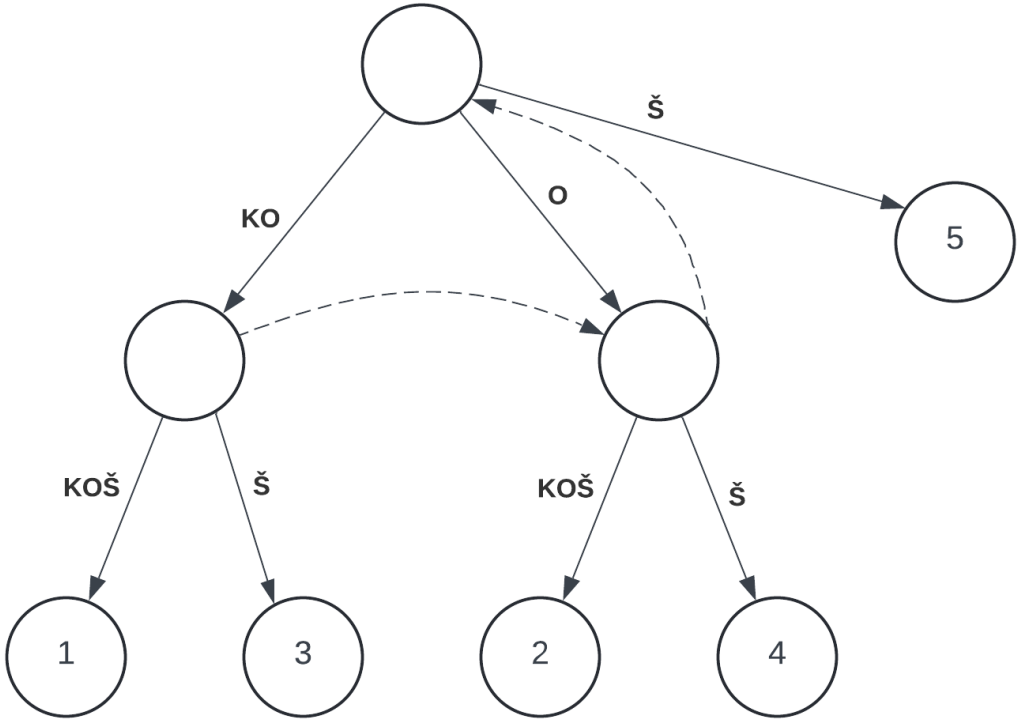
Primer konstrukcije priponskega drevesa



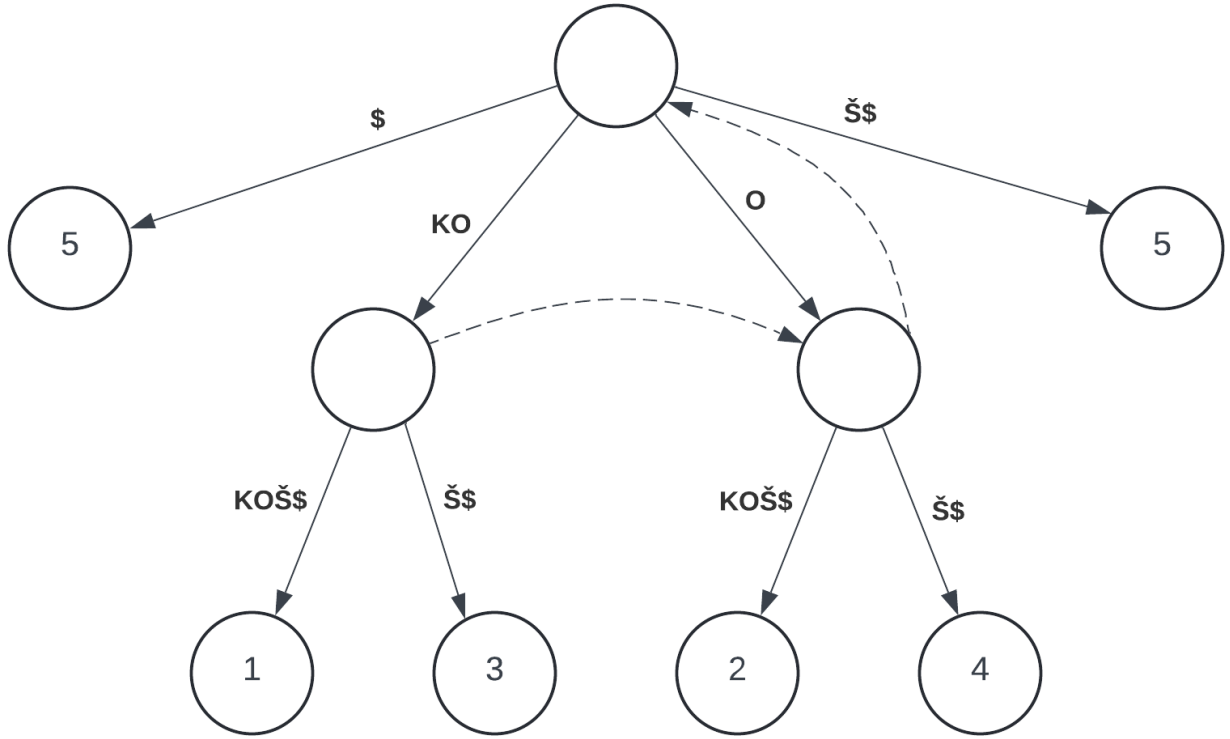
Primer konstrukcije priponskega drevesa



Primer konstrukcije priponskega drevesa



Primer konstrukcije priponskega drevesa



Priponska dreva

Priponsko drevo zniža čas iskanja problema na račun prostorske zahtevnosti.

- Vsak vzorec se lahko najde v času $O(m)$ ter drevo se lahko konstruira v času $O(n)$

Za shranit 10 milijonov znakov dolg genom 347,5MB (priponsko drevo), namesto 2,4 MB (2 bita za znak)

Za izračunat LCSS na 5 milijonov znakov dolg genomu potrebujemo 2s (priponsko drevo), namesto 13,6h (2 bita za znak)

Kompaktna priponska drevesa

Problem potrebujemo preveč spomina

- Drevo lahko potrebuje več spomina kot ga ima na voljo

Rešitve je uporaba kompaktnih podatkovnih struktur

- Podatkovna struktura, ki potrebuje manj kot $O(n)$ bitov prostora
- Reprezentacija strukture drevesa s kompaktno predstavitvijo

Kompaktna priponska drevesa

Kompaktno priponsko drevo (CST) je sestavljen iz 3 delov:

- Kompaktna predstavitev drevesa
- Kompaktna priponsko polje (CSA)
- Polje najnižjega skupnega predhodnika

Kompaktna priponska drevesa

Kompaktna predstavitev drevesa:

- 2 bita za vozlišče
- Potrebuje $4n+o(n)$ bitov
- Poznamo več tipov kodiranja
- Uporabljeno: Uravnoteženi oklepaji

Kompaktna priponska drevesa

Kompaktna priponsko polje (CSA):

- Shrani pripone v abecednem vrstnem redu
- Potreben prostor za kompaktno priponsko polje je $O(n \log|\sum|)$

Kompaktna priponska drevesa

Polje najnižjega skupnega predhodnika ali LCP polje:

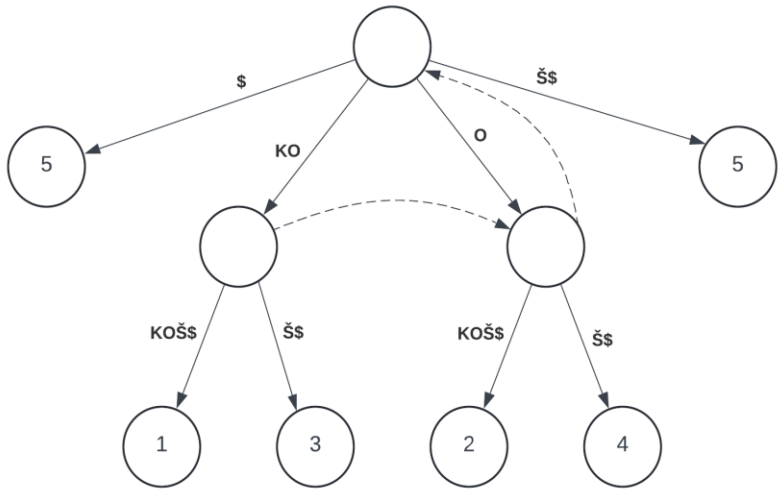
- Shrani globino najnižjega skupnega predhodnika dveh zaporednih pripon
- Prostorska zahtevnost je $2n+o(n)$ bitov
- Vsa element polja je lahko izračunan v konstantnem času

$$lcpArray[i] = \begin{cases} lcp(T[SA[i], n], T[SA[i+1], n]); & 1 \leq i \leq n-1 \\ 0; & i = 0 \end{cases}$$

Kompaktna priponska drevesa - primer

Pripone v priponskem polju:

- $
- KOKOŠ$
- KOŠ$
- OKOŠ$
- OŠ$
- Š$



(()(())(()())())

[6,1,3,2,4,5]

[0,2,0,1,0,0]

Časovne zahtevnosti

	Priponsko drevo	Kompaktno priponsko drevo (bit)
Koren	1	1
Je List	1	1
Otrok	$O(\lvert\sum\rvert)$	$(\log_\sigma \log n)(\log n)^2$
Prvi Otrok	1	1
Brat	1	1
Starš	1	1
Najnižji skupni predhodnik	$O(n)$	1
Globina	$O(n)$	$(\log_\sigma \log n) \log n$
Priponska povezava	1	1
Prostor	$O(n)$ referenc $O(n \log n)$ bitov	$\lvert CSA\rvert + 6n + o(n) =$ $nH_k + 6n + o(n \log \sigma)$ bitov

Kompaktna priponska drevesa

Za shranit 10 milijonov znakov dolg genom 30,5MB (kompaktno priponsko drevo), namesto 347,5MB (priponsko drevo)

Za izračunat LCSS na 5 milijonov znakov dolg genomu potrebujemo 51s (kompaktno priponsko drevo), namesto 2s (priponsko drevo)

Ampak za izračunat LCSS na 40 milijonov znakov dolg genomu potrebujemo 9,15min (kompaktno priponsko drevo), namesto 50,19h (priponsko drevo)

- Razlog: drevo preraste delovni spomin (RAM)

Kompaktna priponska drevesa - izboljšave

Dve izboljšavi kompaktnih priponskih dreves

Popolno stisnjena priponska drevesa:

- Dodatno zniža prostorsko zahtevnost za 6n bitov
- Konstantne operacije postanejo logaritmične

Dinamična kompaktna priponska drevesa:

- Izboljšava popolno stisnjenih priponskih dreves
- Omogoča brisanje in vstavljanje novega besedila v drevo
- Operacije postanejo polilogaritmične

Zaključek

Predstavljene so bile implementacije Priponskega drevesa

- Priponsko drevo
- Kompaktno priponsko drevo
- Izboljšave kompaktnega priponskega drevesa

Nadaljnje raziskave:

- Empirično testiranje predstavljenih dreves

Vprašanja

Hvala za vašo pozornost!