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 [8]

Definicije

Besedilo je polje črk T[1..n]

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

Vsako besedilo se konča z posebnim znakom $

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); suffix link [2]

Priponska dreva

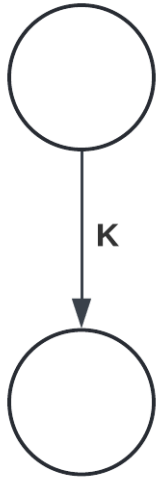
Implementacija abstraktne podatkovne strukture

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

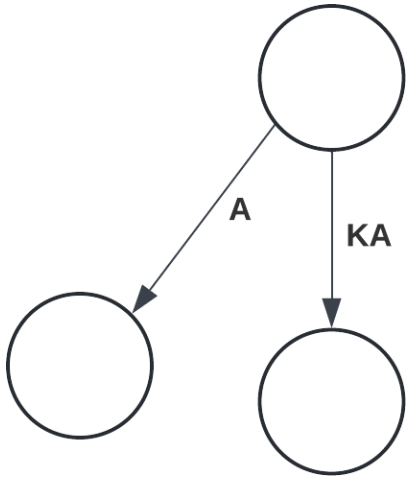
Izgradnja priponskega drevesa je O(n) [1]

- On-line konstrukcija drevesa
- Primer konstrukcije drevesa za besedo: kakav

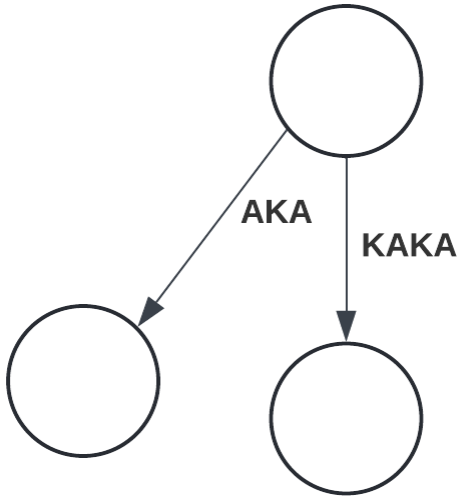
Primer konstrukcije priponskega drevesa



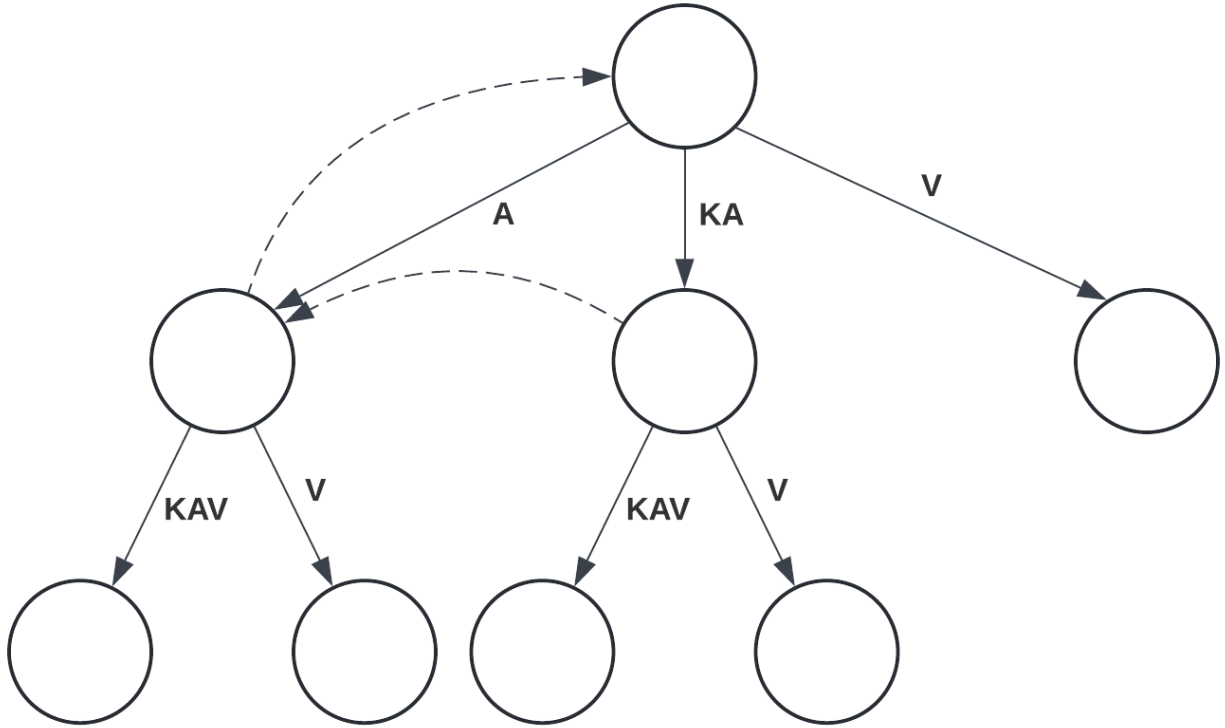
Primer konstrukcije priponskega drevesa



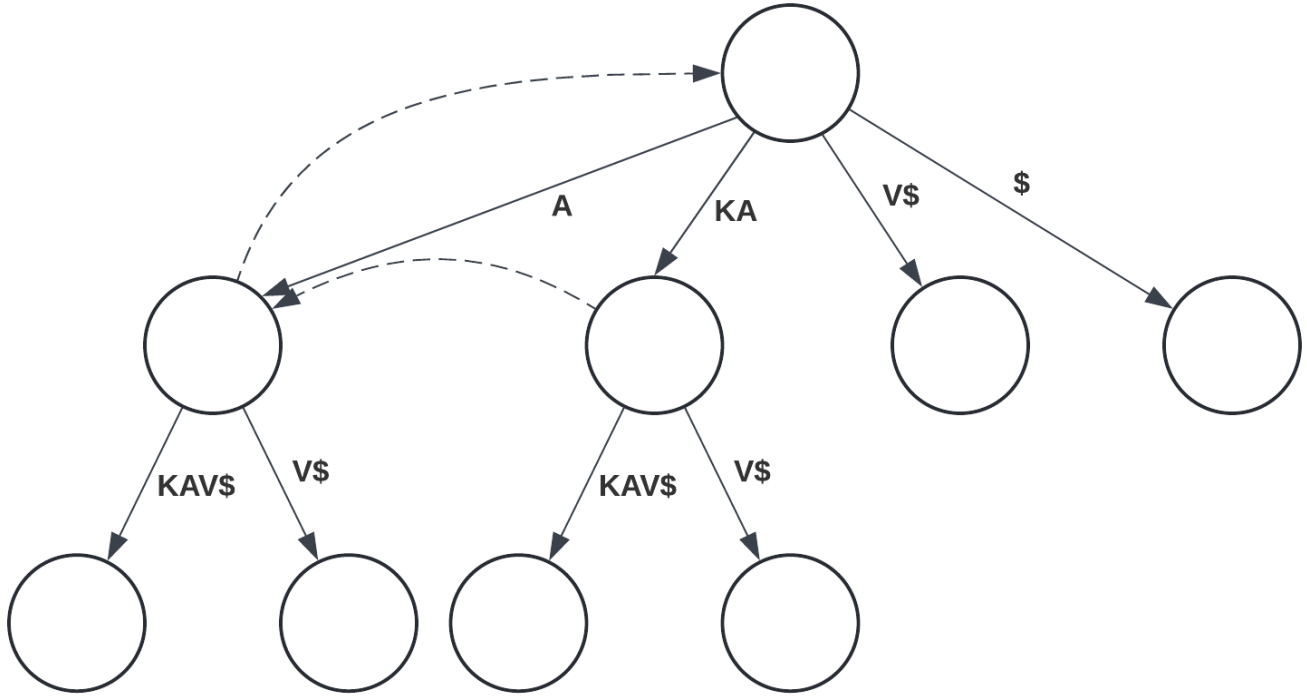
Primer konstrukcije priponskega drevesa



Primer konstrukcije priponskega drevesa



Primer konstrukcije priponskega drevesa



Priponska dreva

Priponsko drevo zniža čas iskanja problema na račun prostorske zahtevnosti.

Za shranit 10 milijonov znakov dolg genom 347,5MB (priponsko drevo), namesto 2,4 MB (2 bita za znak) [3]

Za izračunat LCSS na 5 milijonov znakov dolg genomu potrebujemo 2s (priponsko drevo), namesto 13,6h (2 bita za znak) [3]

Kompaktna priponska drevesa

Problem potrebujemo preveč spomina

- Drevo lahko potrebuje več spomina kot ga ima na voljo [3]

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

- Kompaktna predstavitev drevesa
- Kompaktna priponsko polje (CSA)
- Polje višin / LCP polje [2]

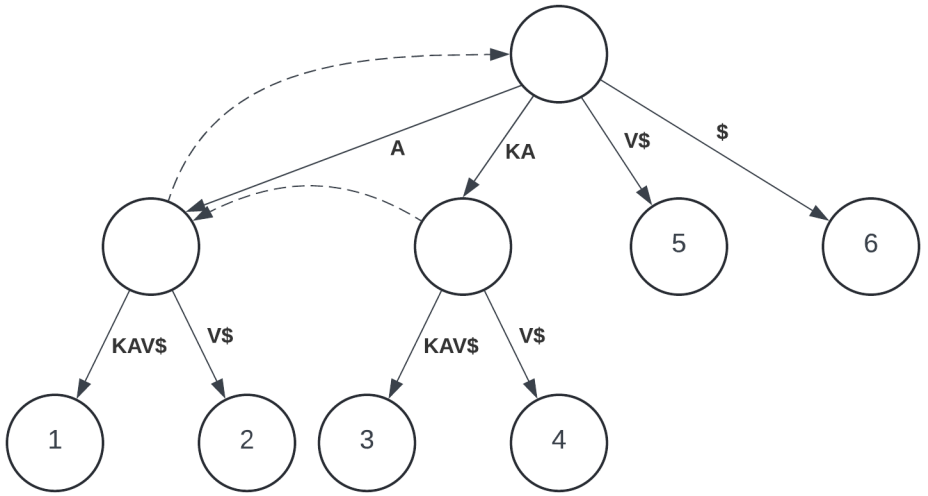
Kompaktna priponska drevesa - primer

BP: ((()())(()())()())

Priponsko polje:

- AKAV$
- AV$
- KAKAV$
- KAV$
- V$
- $

Polje višin:[1,0,2,0,0,0]



Kompaktna priponska drevesa

Za shranit 10 milijonov znakov dolg genom 30,5MB (kompaktno priponsko drevo), namesto 347,5MB (priponsko drevo) [3]

Za izračunat LCSS na 5 milijonov znakov dolg genomu potrebujemo 51s (kompaktno priponsko drevo), namesto 2s (priponsko drevo) [3]

Ampak za izračunat LCSS na 40 milijonov znakov dolg genomu potrebujemo 9,15min (kompaktno priponsko drevo), namesto 50,19h (priponsko drevo) [3]

- Razlog: drevo preraste delovni spomin (RAM)

Dinamična kompaktna priponska drevesa

Nov problem: CST je statična struktura

- Ne podpira operacij dodajanja besedila

Dinamična verzija priponskega drevesa [4]

Časovne zahtevnosti

	Priponsko drevo (reference)	**Kompaktno priponsko drevo (bit)**	**Dinamično kompaktno priponsko drevo (bit)**
Vstavi	O(n+m)	/	$m(\log_\sigma \log n)\,(\log n)^2$
Najnižji skupni predhodnik	O(n)	1	$(\log_\sigma \log n)\,(\log n)^2$
Globina	O(n)	1	/
Dolžina podniza	O(n)	$(\log_\sigma \log n) \log n$	$(\log_\sigma \log n)\,(\log n)^2$
Prostor	O(n)	$\mathrm{n}H_k + 6n + o(n \log \sigma)$	$\mathrm{n}H_k + o(n \log \sigma)$

Zaključek

Predstavljene so bile implementacije Priponskega drevesa

- Priponsko drevo
- Kompaktno priponsko drevo
- Dinamično kompaktno priponsko drevo

Implementacija konkatenacije dveh priponskih dreves

Vprašanja

Hvala za vašo pozornost!

Viri

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

[2] K. Sadakane, Compressed Suffix Trees with Full Functionality. Theory of Computing Systems 41 (2007) 589–607.

[3] N. Valimaki, W. Gerlach, K. Dixit in V. Makinen, Engineering a Compressed Suffix Tree Implementation. V 6th International Workshop on Experimental and Efficient Algorithms, 2007, 217–228.

[4] L. M. S. Russo, G. Navarro in A. L. Oliveira, Dynamic Fully-Compressed Suffix Trees. V 19th Annual Symposium on Combinatorial Pattern Matching, 2008, 191–203.

[5] E. M. McCreight, A Space-Economical Suffix Tree Construction Algorithm. Journal of the Association for Computing Machinery 23 (1976) 262–272.

[6] P. Weiner, Linear pattern matching algorithms. V 14th Annual Symposium on Switching and Automata Theory (swat 1973), 1973, 1–11.

[7] G. Navarro, Compact Data Structures: A Practical Approach, Cambridge University Press, 2016.

Viri

[8] D. E. Knuth, J. H. Morris in V. R. Pratt, Fast Pattern Matching in Strings. SIAM Journal on Computing 6 (1977) 323-350.

[9] K. Sadakane, New text indexing functionalities of the compressed suffix arrays. Journal of Algorithms 48 (2003) 294-313.

[10] I. J. Munros in V. Raman, Succinct representation of balanced parentheses, static trees and planar graphs. Proceedings 38th Annual Symposium on Foundations of Computer Science (1997) 118-126.

[11] L. M. S. Russo, G. Navarro in A. L. Oliveira, Fully-Compressed Suffix Trees. LATIN 2008: Theoretical Informatics (2008) 362–373.