

Kompaktna priponska drevesa

AVTOR: JANI SUBAN, 89222015

MENTOR: PROF. DR. ANDREJ BRODNIK

Kazalo

Opis problema in definicije

Priponska dreva

Kompaktna priponska drevesa

Nadaljnje raziskave

Opis problema in definicije

Uporaba v procesiranju Naravnih jezikov ter v Bioinformatiki

Problemi, ki jih rešujemo z priponskimi drevesi:

- Iskanje vzorcev v besedilu
- Iskanje najdaljšega podvojenega podniza v besedi

Definicije:

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 predjodnik
9. sl(v); suffix link [2]

Priponska dreva

Izgradnja priponskega drevesa je $O(n)$ [1]

Priponsko drevo zniža čas iskanja problema na račun prostorske zahtevnosti.

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

Za izračunati 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]

Kompaktna predstavitev prionskega drevesa

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

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

Kompaktna priponska drevesa

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

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

Ampak za izračunati 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)

Nadaljnje raziskave

Nov problem: CST je statična struktura

- Ne podpira operacij dodajanja besedila in branja besedila

Pri tem izgubi tudi on-line konstrukcijo strukture

Ali obstaja dinamična verzija strukture?

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. V. Alim-Aki, W. Gerlach, K. Dixit in V. Mäkinen, 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.