Heidi Jauhiainen
011512192
Ohjaaja: Kristiina Paloheimo
Tietorakenteiden harjoitustyö, loppukesä 2012
Helsingin yliopisto, Tietojenkäsittelytieteen laitos
31.8.2012

Toteutusdokumentti ja käyttöohjeet

WordI - Sanaindeksi Trie-puuta käyttäen

Johdanto

WordI on ohjelma, joka saa syötteenä yhden tai useamman tekstitiedoston ja josta pystyy hakemaan annettuja sanoja tai sananalkuja sisältävät rivit. Toteutettu sanaindeksi tukee useamman sanan tai sen osan hakemista kaikista tai yhdestä tiedostosta kerrallaan. Trie-puu on tehokas tietorakenne tällaisten String-muotoisten avainten hakemiseen. Triessä sanan jokainen kirjain talletetaan eri solmuun. Sanan ensimmäinen kirjain on tyhjän juurisolmun lapsi ja toinen kirjain tämän ensimmäisen solmun lapsi ja niin edelleen. Jokaisella solmulla voi olla lapsia niin paljon kuin käytetyssä aakkostossa on kirjaimia (plus mahdollisesti joitain muita merkkejä). Samanalkuiset sanat jakavat alkupään solmut. Trie-puun aikavaativuus on riippuvainen siitä, miten tieto solmujen lapsisolmuista on tallennettu ja etsittävissä.

WordI toteutettiin trie-puulla, johon eri tiedostojen sanat tallennetaan. Tässä puussa tieto yksittäisen solmun lapsisolmuista tallennetaan järjestettyyn taulukkoon, josta tietyn kirjaimen sisältämä solmu haetaan binäärihaulla. Jokaiseen solmuun tallennetaan tieto riveistä, joilla kyseinen kirjain ja sen prefiksi sijaitsevat. Rivinumerot tallennetaan dynaamiseen taulukkoon. Koska sanojen tallennus tapahtuu tiedostosta järjestyksessä, solmussa olevat rivinumerot ovat myös järjestyksessä. Jokaisen sanan viimeisen kirjaimen kohdalle tallennetaan rivinumero myös toiseen dynaamiseen taulukkoon. Jälkimmäistä taulukkoa käytetään kokonaisten sanojen hakemiseen ja ensimmäistä sanan osien hakuun. Tekstipohjaisessa käyttöliittymässä sanan osan hakua merkitään sanan perään lisättävällä *-merkillä).

Toteutus

WordI:stä voi hakea yhtä tai useampaa sanaa tai sanaosaa yhdistämällä niitä &-merkillä (= ja), /-merkillä (= tai) ja sulkeilla. Kun käyttäjä syöttää hakulausekkeen, sen purkamisessa käytetään apuna Shunting yard-algoritmin versiota. Shunting yard on Edsker W. Dijkstran kehittämä alkoritmi matemaattisten lausekkeiden muuttamiseen sellaiseen muotoon, että niitä on helppo käsitellä algoritmeilla [Wikipedia, Shunting-yard algorithm]. Shunting Yard poistaa lausekkeista sulut ja järjestää lausekkeen niin, että operandit tulevat ennen niitä koskevaa operaatio-symbolia eli niin sanotussa postfix-järjestyksessä. Dijkstran algoritmi käyttää pinoa operaattoreille ja jonoa postfix-lausekkeen kokoamiseen. [Wikipedia, Shunting-yard algorithm.] WordI:ssä käytetään kuitenkin kahta pinoa: toinen operaattoreille ja toinen hakusanoille. Hakulauseke käydään läpi merkki kerrallaan ja jokainen operaattori lisätään hakutermit-pinoon. Kun vastaan tulee kirjain, otetaan talteen kirjaimia kunnes tulee vastaan jokin merkeistä ‘välilyönti’, ‘&’, ‘/’ tai ‘)’. Näin saadulle sanalle kutsutaan etsi-metodia, joka palauttaa taulukon riveistä, joilla sana esiintyy. Taulu tallennetaan sanat-pinoon. Kun vastaan tulee loppu-sulku ‘)’, sen sijaan että laitettaisiin merkki pinoon, kutsutaan haeRivit-metodia, joka ottaa hakutermit-pinosta päällimmäisen ja kutsuu sen mukaan joko yhdistaOr- tai yhdistaAnd-metodia. Kumpikin näistä metodeista poistaa sanat-pinosta kaksi taulukkoa rivejä ja yhdistää ne. YhdistaOr-metodi lisää kolmanteen taulukkoon järjestyksessä pienimmästä suurimpaa kaikki rivit, jotka esiintyvät jommassa kummassa taulussa, kuitenkin kukin rivi vain kerran. YhdistaAnd tekee taulukon, jossa on vain kummankin taulun yhteiset rivit. Lopuksi haeRivit-metodi laittaa uuden yhdistetyn taulukon sanat-pinoon. Tätä toistetaan kunnes hakutermit pinossa tulee alkusulku ‘(‘ vastaa. Sulku poistetaan pinosta ja palataan käymään hakulauseketta läpi. Kun hakulausekkeen merkit on käyty loppuun, kutsutaan haeRivit-metodia niin kauan kunnes hakutermit-pino on tyhjä. Tämän jälkeen sanat-pinossa on yksi taulukko, joka sisältää rivit, joilla haetut sanat esiintyvät hakulausekkeen määrittelemällä tavalla, ja joka palautetaan printtaa-metodille.

Ohjelma säilyttää kerran sinne syötettyjen tekstitiedostojen datan, mutta sille voi myös syöttää uusia tiedostoja. Uutta tiedostoa lisättäessa ohjelma tarkistaa löytyykö senniminen tiedosto annetusta paikasta ja onko samanniminen tiedosto jo tallennettuna ohjelmaan. Tiedostojen rivit tallennetaan dynaamiseen taulukkoon. Solmuihin tallennettavat rivinumerot ovat tämän taulun indeksejä, joten rivien printtaaminen on nopeaa sen jälkeen kun sanahaulla on saatu lista riveistä, joilla haettu sana tai sen osa sijaitsee. Tieto siitä, millä tekstit sisältävän taulun riveillä kunkin tiedoston rivit löytyvät,

tallennettaan tiedoston nimen kanssa tiedosto-olioon. Tiedosto-oliot puolestaan löytyvät dynaamisesta taulukosta.

Aikavaativuus

Yhden sanan haku käytetystä trie-puusta on aikavaatimukseltaan O(m log n), jossa m on hakusanan kirjainten määrä ja n kunkin kirjaimen solmun lasten määrä. Tämä aikavaativuus koostuu seuraavista tapahtumista: Riippuen siitä onko hakusanan perässä * vai ei kutsutaan joko Puu-olennon haeOsa- tai haeSana-metodia. Kumpikin näistä metodeista käy läpi haetun sanan jokaisen kirjaimen (O(m)). Kumpikin metodi myös kutsuu jokaisen kirjaimen kohdalla PuuSolmu-olennon getLapsi-metodia, joka hakee binäärihaulla lapsen, jolla on haettu kirjain. Binäärihaun aikavaativuus on O(log n), jossa n on solmun lasten määrä. Koska lasten määrä on rajallinen, korkeintaan käytössä olevien kirjainten määrä (muitakin merkkejä voi olla, toisaalta harvoin sanassa kirjainta voi seurata mikä tahansa kirjain), binäärihaun aikavaativuus on useimmilla kielillä hyvin pieni. Koska sanojen sisältämien kirjaintenkin määrä on rajallinen, harvemmin yli 50 kirjainta [Wikipedia, Longest words], ei tuo O(m log n) yleensä kasva kovin suureksi pahimmassakaan tapauksessa.

Yhden sanan haku voitaisiin WordI:ssä toteuttaa helposti aikavaativuudella O(m log n) eriyttämällä yhden sanan haku pidemmistä jo käyttäjältä hakusanaa kysyttäessä ja kutsumalla sitten suoraan Puu-olennon haeOsa- tai haeSana-metodia. WordI:ssä on kuitenkin valittu toteuttaa yhden ja useamman hakusanan haku yhteisellä metodilla, jonka aikavaativuus on O(k + m log n), jossa k on riippuvainen haun tuottamien rivien määrästä. Aina kun yhdistetään kahden hakusanan tulokset (joko yhdistaOr- tai yhdistaAnd-metodilla), käydään läpi molempien hakusanojen tuottamat rivit. Näin ollen kyseisten metodien aikavaativuus on O(max p), jossa p on hakujen tuottama rivimäärä. K on tällöin kaikkien yhdistämistapahtumien aikavaativuudet yhteensä. Hae-metodin aikavaativuus on siis O(k log n) mutta käytännössä, jos haetaan monia sanoja, joita löytyy usealta riviltä, aikaa hakuun kuluu enemmän kuin yhden sanan hakuun. Kuitenkin testit osoittavat (ks. Testaus.doc), että ohjelman hae-metodin aikavaativuus yhden sanan haulla lähentelee O(n):ää, jossa n on sanan pituus, sillä mitä pidempi sana sitä vähemmän loppupään kirjaimilla on lapsisolmuja. Lisäksi pitkiä sanoja on kielissä vähemmän, joten niiden tuottamat rivimäärät ovat pienempiä kuin lyhyillä sanoilla.

Käyttöohjeet

WordI-ohjelma löytyy WordI-kansiosta, jonka voi sijoittaa haluamaansa paikkaan. Ohjelma käynnistetään terminaalista menemällä kyseiseen kansioon ja komentamalla "java -jar WordI.jar" tai suoraan komennolla "java -jar [polku/WordI/]WordI.jar". Kansiossa on myös muutamia ohjelman kokeiluun soveltuvia tekstitiedostoja. Ohjelman päävalikosta voi valita numeroilla seuraavat toiminnot: 1. *Hae kaikista tiedostoista,* 2. *Hae tietystä tiedostosta,* 3. *Lisää tiedosto* ja 4. *Listaa Tiedostot*. Numero 0 lopettaa ohjelman. Tietystä tiedostosta haettaessa ohjelma kysyy ensin tiedoston nimeä, josta haluaa löytää rivejä. Molemmilla hakutavoilla voi hakea kokonaista sanaa, sanoja jotka alkavat haetulla tavalla (sanan voi katkaista *-merkillä), tai sanojen yhdistelmiä. Sanoja voi yhdistää &-merkillä, jolloin tuloksena on rivit, jotka sisältävät molemmat sanat, tai /-merkillä, joka hakee kaikki rivit, joilla esiintyy jompikumpi sanoista. Erilaisia hakuja voi lisäksi yhdistää niin paljon kuin haluaa, mutta yli kahden sanan halussa kannattaa käyttää sulkeita ryhmittämässä hakuja, muuten tulos ei ehkä vastaa haluttua.

Ohjelmaan voi koska tahansa lisätä uusia tiedostoja. Tiedostot täytyy sijoittaa samaan kansioon ohjelman jar:in kanssa. Tiedostot voivat olla joko txt- tai rtf-päätteisiä. Samannimistä tiedostoa ei voi uudelleen tallentaa ohjelmaan.

Ohjelman puutteet

Ohjelmaan voi periaatteessa lisätä minkä tahansa kielisiä tiedostoja, sillä lapsisolmujen määrää saa kasvaa. Käytännössä ohjelma ei kuitenkaan tällaisenaan tunnista esimerkiksi kiinan kielen merkkejä eikä osaa erotella niitä toisistaan. Ohjelma ei myöskään osaa käsitellä doc-, odt- tai pages-päätteisiä tiedostoja.

Monen sanan haku toimii ohjelmassa ilman sulkeitakin, mutta sen tuottama tulos voi olla eri kuin odottaisi. Käytetty Shunting yard -toteutus nimittäin aloittaa hakusanojen käsittelyn lopusta, mikäli suluilla ei ole muuta ohjeistettu. Näin ollen haku brother/sun&him* tuottaa saman tuloksen kuin brother/(sun&him*), vaikka voisi ehkä odottaa tulosta joka vastaa hakua (brother/sun)&him*.

Lähteet:
Wikipedia, Longest words. http://en.wikipedia.org/wiki/Longest_words. Haettu 31.8.2012.
Wikipedia, Shunting-yard algorithm. http://en.wikipedia.org/wiki/Shunting-yard_algorithm. Haettu 31.8.2012.