Merkkijonojoukkojen louhinta kattavuusrajoitteilla

Niko Välimäki

Helsinki 7.2.2008

Seminaarityö

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Johdanto

Merkkijonojoukkojen louhinnassa [RJL+02, FHK06] ollaan kiinnostuneita löytämään muun muassa merkkijonojoukkojen välisiä eroja. Tässä kirjoitelmassa tarkastellaan louhintaongelmaa, jossa halutaan luetella *osajonot*, jotka ovat yleisiä toisessa merkkijonojoukossa, mutta samalla harvinaisia toisessa. Osajonon yleisyyttä merkkijonojoukossa mitataan sen esiintymiä sisältävien merkkijonojen lukumäärällä.

Tarkastellaan esimerkkitapausta [Fis07], jossa geneettisen sairauden arvellaan johtuvan X-kromosomin virheestä. On kuitenkin epäselvää missä ja miten tämä virhe kromosomissa esiintyy. Ongelmaa voitaisiin yrittää selvittää keräämällä 1000 sairastuneen potilaan ja 1000 terveen potilaan X-kromosomit positiiviseksi ja negatiiviseksi merkkijonojoukoksi. Kaikki osajonot, jotka olisivat yleisiä positiivisessa joukossa ja samalla harvinaisia negatiivisessa joukossa, olisivat potentiaalisia merkkejä geneettisen ongelman lähteestä.

Tässä kirjoitelmassa näytetään, kuinka merkkijonojoukkojen louhinta voidaan ratkaista optimaalisessa ajassa syötteen pituuden suhteen [FHK06]. Ratkaisu pohjautuu tunnettuihin merkkijonoalgoritmeihin ja -tietorakenteisiin. Louhinta-algoritmia voidaan käyttää esimerkiksi transkriptiotekijöiden sitovien motiivien (binding motifs of transcription factors) tai sekvenssejä luokittelevien erikoispiirteiden etsintään. Bioinformatiikan ulkopuolella louhinta-algoritmi soveltuu muun muassa kielen tunnistukseen, roskapostin suodatukseen sähköposteista sekä MIDI-sekvenssien analysointiin. Algoritmissa käytettävät menetelmät ovat sovellettavissa myös tiedonhaun ongelmiin [FMV08].

Seuraavassa luvussa käydään tarkemmin läpi louhintaongelman kuvaus sekä määritellään muutamia merkkijonomenetelmistä tuttuja tietorakenteita. Kolmannessa luvussa esitellään itse louhinta-algoritmi. Neljännessä luvussa tarkastellaan, kuinka louhinta-algoritmi suoriutuu käytännön testeissä. Yhteenveto ja johtopäätökset ovat viimeisessä luvussa.

2 Määritelmiä

Merkkijonon $T = t_1 t_2 \cdots t_n$ pituutta merkitään $|T| = n$. Merkkijonon T *osajonoa* $t_i t_{i+1} \cdots t_j$ merkitään $T[i \ldots j]$ kaikille $1 \leq i \leq j \leq n$. Jos merkkijono ϕ on merkkijonon T osajono, eli ϕ:llä on vähintään yksi *esiintymä* merkkijonossa T, merkitään $\phi \preceq T$. Merkkijonon *alku-* ja *loppuosilla* tarkoitetaan osajonoja $T[1 \ldots i]$ ja $T[i \ldots n]$,

missä $1 \leq i \leq n$.

Merkkijonojoukko $\mathcal{D}$ on kokoelma merkkijonoja $T_i \in \mathcal{D}$ samasta äärellisestä aakkostosta Σ, jonka kokoa merkitään σ. Merkkijonojen lukumäärää joukossa $\mathcal{D}$ merkitään $|\mathcal{D}|$ ja merkkijonojen yhteispituutta $\|\mathcal{D}\| = \sum_{T_i \in \mathcal{D}} |T_i|$.

Tässä kirjoituksessa oletetaan RAM-laskennan (Random Access Memory) malli, jossa jokainen korkeintaan $\Theta(\log n)$ bittiä pitkän muistialueen luku tai kirjoitus onnistuu vakioajassa. Lisäksi oletetaan, että tavanomaiset aritmeettiset laskutoimitukset sekä vertailut onnistuvat vakioajassa kahdelle korkeintaan $\Theta(\log n)$ bittiä pitkälle luvulle. Merkinnän log kantaluku on 2.

2.1 Louhintaongelmat

Osajonon *kattavuudella* (frequency) tarkoitetaan sen esiintymiä sisältävien merkkijonojen lukumäärää annetussa merkkijonojoukossa. Tarkemmin sanottuna osajonon ϕ kattavuus joukossa $\mathcal{D}$ on

$$freq(\phi, \mathcal{D}) = |\{T_i \in \mathcal{D} \mid \phi \preceq T_i\}|.$$

Vastaavasti osajonon ϕ *tuki* (support) joukossa $\mathcal{D}$ on

$$supp(\phi, \mathcal{D}) = \frac{freq(\phi, \mathcal{D})}{|\mathcal{D}|}.$$

Merkkijonojoukkojen louhinta *kattavuusrajoitteilla* [RJL+02, FHK06] on ongelma, jossa halutaan löytää joukkojen $\mathcal{D}_1, \ldots, \mathcal{D}_r$ merkkijonojen kaikki osajonot, joilla annetut kattavuusrajoitteet ovat voimassa. Kattavuusrajoite (frequency constraint) on jokaiselle merkkijonojoukolle $\mathcal{D}_i$ erikseen määritelty pari (min_i, max_i), jossa $1 \leq i \leq r$ ja $0 \leq min_i \leq max_i \leq |\mathcal{D}_i|$. Tulokseen kuuluvat siis kaikki osajonot ϕ, joilla pätee $min_i \leq freq(\phi, \mathcal{D}_i) \leq max_i$ kaikille $1 \leq i \leq r$.

Esimerkki 1 *Olkoon* $\mathcal{D}_1 = \{\texttt{bbabab}, \texttt{abacac}, \texttt{bbaaa}\}$ *ja* $\mathcal{D}_2 = \{\texttt{aba}, \texttt{babbc}, \texttt{cba}\}$. *Lisäksi on annettu kattavuusrajoitteet* $min_1 = 2$, $max_1 = 3$, $min_2 = 0$ *sekä* $max_2 = 2$. *Kattavuusrajoitteet ovat voimassa vain osajonoilla* $\{\texttt{ab}, \texttt{aba}, \texttt{bb}, \texttt{bba}\}$. *Esimerkiksi osajonon* `aba` *kattavuudet ovat* $freq(\texttt{aba}, \mathcal{D}_1) = 2$ *ja* $freq(\texttt{aba}, \mathcal{D}_2) = 1$.

Nousevien osajonojen (emerging substrings) louhinnassa [DL99, CKYT03] on annettu positiivinen ja negatiivinen merkkijonojoukko, joista halutaan löytää osajonot, joiden *kasvunopeus* (growth-rate) ylittää annetun kynnysarvon ρ. Osajonon ϕ

kasvunopeus negatiivisesta joukosta D_2 positiiviseen joukkoon D_1 on

$$growth_{\mathcal{D}_2 \to \mathcal{D}_1}(\phi) = \frac{supp(\phi, \mathcal{D}_1)}{supp(\phi, \mathcal{D}_2)},$$

jos $supp(\phi, \mathcal{D}_2) \neq 0$, ja muutoin $growth_{\mathcal{D}_2 \to \mathcal{D}_1}(\phi) = \infty$. Kasvunopeuden raja-arvon lisäksi annetaan yleensä vielä minimikattavuus min_1. Näin ollen nousevia osajonoja ovat kaikki osajonot ϕ, joilla $growth_{\mathcal{D}_2 \to \mathcal{D}_1}(\phi) \geq \rho$ ja $freq(\phi, \mathcal{D}_1) \geq min_1$.

Esimerkki 2 *Olkoon* $\mathcal{D}_1 = \{\texttt{aaba}, \texttt{abaaab}\}$ *ja* $\mathcal{D}_2 = \{\texttt{bbabb}, \texttt{abba}\}$. *Kun on lisäksi annettu kasvunopeuden raja-arvo* $\rho = 2$ *ja minimikattavuus* $min_1 = 1$, *nousevat osajonot* $\mathcal{D}_2$*:sta* $\mathcal{D}_1$*:een ovat* $\{\texttt{aa}, \texttt{aab}, \texttt{aba}\}$.

2.2 Tietorakenteet

Tarkastellaan seuraavaksi merkkijonon T loppuosista $T[i \ldots n]$, $1 \leq i \leq n$, muodostettavia tietorakenteita, joita käytetään myöhemmin luvussa 3. Oletetaan, että merkkijono päättyy erityiseen loppumerkkiin $T[n] = \$$, joka ei esiinny muualla merkkijonossa T. Loppumerkki estää sen, ettei mikään loppuosista voi olla toisen loppuosan alkuosa. Lisäksi oletetaan loppumerkin olevan aakkosjärjestyksessä kaikkia muita aakkosia $c \in \Sigma \setminus \{\$\}$ pienempi, $\$ < c$.

Loppuosataulukko. *Loppuosataulukko* SA sisältää merkkijonon järjestetyt loppuosat [MM93]. Arvo $SA[i]$ vastaa merkkijonon loppuosaa $T[SA[i] \ldots n]$ siten, että merkkijonojen aakkosjärjestys

$$T[SA[i] \ldots n] \ < \ T[SA[i+1] \ldots n]$$

pätee kaikilla $1 \leq i < n$. Käänteistaulukolle $SA^{-1}[j]$ pätee $SA^{-1}[j] = i$, jos ja vain jos $SA[i] = j$. Kuvassa 1 on esimerkki loppuosataulukosta merkkijonolle $\texttt{ababac}\$$. Loppuosataulukko sekä sen käänteistaulukko on mahdollista muodostaa $O(n)$ ajassa [KSB06].

LCP-taulukko. Kahden merkkijonon T ja T' *pisimmän yhteisen alkuosan* (longest common prefix) pituutta merkitään $lcp(T, T')$. Jos $lcp(T, T') = \ell$, niin täytyy olla $t_i = t'_i$ kaikille $1 \leq i \leq \ell$ sekä $t_{\ell+1} \neq t'_{\ell+1}$. Taulukko $LCP[i]$ tallentaa kahden peräkkäisen loppuosan pisimmän yhteisen alkuosan pituuden siten, että $LCP[1] = 0$ ja kaikille $1 < i \leq n$ pätee

$$LCP[i] = lcp(\ T[SA[i] \ldots n],\ T[SA[i-1] \ldots n]\ ).$$

Loppuosat
ababac$
babac$
abac$
bac$
ac$
c$
$

Aakkos-järjestys $\Longrightarrow$

i	$SA[i]$	$LCP[i]$	$T[SA[i] \ldots n]$
1	7	0	$
2	1	0	ababac$
3	3	3	abac$
4	5	1	ac$
5	2	0	babac$
6	4	2	bac$
7	6	0	c$

Kuva 1: Vasemmalla kaikki merkkijonon $T =$ ababac$ loppuosat. Oikealla loppuosa- ja LCP-taulukko samalle merkkijonolle T.

Kuvassa 1 on esimerkki LCP-taulukosta merkkijonolle ababac$. Taulukko LCP voidaan muodostaa loppuosataulukosta $O(n)$ ajassa [KLA+01].

RMQ-rakenne. *Osavälin minimi* (range minimum query) -kysely $RMQ_C(l, r)$ taulukolle $C[1 \ldots n]$ palauttaa *pienimmän arvon sijainnin* osavälillä $C[l \ldots r]$. Jos sijainti ei ole yksikäsitteinen, valitaan vasemman puoleisin. Kysely $RMQ_C(l, r)$ ratkeaa vakioajassa mille tahansa $l, r \in [1, n]$ sen jälkeen, kun taulukko C on esikäsitelty $O(n)$ ajassa [BFC00].

Esimerkki 3 *Oletetaan, että LCP-taulukon arvot ovat kuten kuvassa 1. Taulukon LCP osavälin minimi -rakenteesta saadaan esimerkiksi arvot $RMQ_{LCP}(2, 5) = 4$ ja $RMQ_{LCP}(5, 7) = 6$. Molemmissa tapauksissa osavälin minimin arvo on $LCP[4] = LCP[6] = 0$.*

3 Louhinta-algoritmi

Optimaalisessa ajassa toimiva merkkijonojoukkojen louhinta-algoritmi [FHK06] on yhdistelmä Kasain et al. [KLA+01] algoritmista *haarautuvien osajonojen läpikäyntiin*, sekä Huin [Hui92] menetelmästä *värijoukon koon* (color set size) laskemiseen. Seuraavissa aliluvuissa esitellään, kuinka näitä menetelmiä käytetään louhintaongelman ratkaisemiseen. Notaation yksinkertaistamiseksi tarkastellaan vain yhden joukon $\mathcal{D}$ louhintaa kattavuusrajoiteella (min, max). Yleistys useamman kuin yhden merkkijonojoukon louhintaan on helposti johdettavissa perusalgoritmista.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
T:	a	a	b	a	$\$_1$	a	b	a	a	a	b	$\$_2$	b	b	a	b	b	$\$_3$	a	b	b	a	$\$_4$
SA:	5	12	18	23	4	22	8	9	1	10	2	6	15	19	11	17	3	21	7	14	16	20	13
LCP:	0	0	0	0	0	1	1	2	3	1	2	3	2	3	0	1	1	2	2	2	1	2	3
	$\$_1$	$\$_2$	$\$_3$	$\$_4$	a $\$_1$	a $\$_4$	a a a b $\$_2$	a a b $\$_2$	a b a $\$_1$	a b $\$_2$	a b a $\$_1$	a b a a a b $\$_2$	a b b $\$_3$	a b b a $\$_4$	b $\$_2$	b $\$_3$	b a $\$_1$	b a $\$_4$	b a a a b $\$_2$	b a b $\$_3$	b b $\$_3$	b b a $\$_4$	b b a b b $\$_3$

Kuva 2: Taulukot SA ja LCP joukon $\mathcal{D} = \{\texttt{aaba}\$_1, \texttt{abaaab}\$_2, \texttt{bbabb}\$_3, \texttt{abba}\$_4\}$ katenoidulle merkkijonolle T. Taulukon alapuolella on jokaista positiota i vastaava loppuosa $T[SA[i] \ldots n]$ ensimmäiseen loppumerkkiin asti.

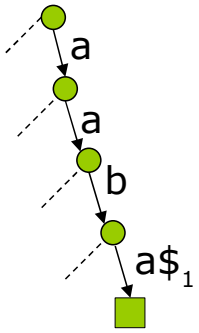
3.1 Haarautuvien osajonojen läpikäynti

Louhintaongelmassa halutaan löytää osajonot, joilla annetut kattavuusrajoitteet ovat voimassa. Kaikkia merkkijonojoukon osajonoja ei kuitenkaan tarvitse käydä läpi — louhintaongelman ratkaisun kannalta riittää laskea vain *haarautuvien osajonojen* kattavuudet [FHK06]. Tarkastellaan tässä aliluvussa haarautuvien osajonojen läpikäyntiä ja käsitellään niiden kattavuuden laskeminen erikseen seuraavassa aliluvussa.

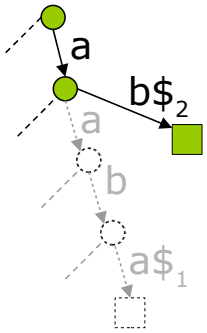
Olkoon $\mathcal{D}$ joukko, joka koostuu merkkijonoista $\{T_1, T_2, \ldots, T_r\}$. Oletetaan, että jokainen merkkijono $T_i \in \mathcal{D}$ päättyy loppumerkkiin $\$_i$, ja että kaikilla loppumerkeillä pätee aakkosjärjestys $\$_i < \$_{i+1}$, kun $1 \leq i < r$. Katenoidaan joukon $\mathcal{D}$ merkkijonot yhdeksi merkkijonoksi T, jonka pituutta merkitään $n = |T|$, ja muodostetaan merkkijonolle T loppuosataulukko SA sekä LCP-taulukko. Esimerkki näistä rakenteista joukolle $\mathcal{D} = \{\texttt{aaba}\$_1, \texttt{abaaab}\$_2, \texttt{bbabb}\$_3, \texttt{abba}\$_4\}$ on kuvassa 2.

Määritelmä 1 *Osajono $\phi \preceq T$ on* haarautuva*, jos on olemassa symbolit $a, b \in \Sigma$ siten, että $a \neq b$ ja katenoiduille merkkijonoille ϕa ja ϕb pätee $\phi a \preceq T$ ja $\phi b \preceq T$.*

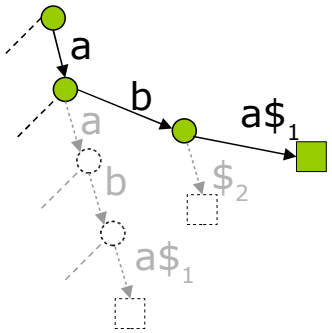
Merkkijonon haarautuvat osajonot vastaavat merkkijonon *loppuosapuun* sisäsolmuja [KLA+01]. Loppuosapuu on puurakenne, jonka polut juuresta lehtisolmuihin vastaavat merkkijonon loppuosia [Wei73]. Loppuosapuun sisäsolmujen läpikäynti voidaan simuloida Kasain et al. [KLA+01] algoritmilla käyttäen apuna vain taulukoita SA ja LCP. Loppuosapuurakennetta ei muodosteta, mutta sen yhteys haarautuvien osa-



Kuva 3: Lähtötilanne.



Kuva 4: Päivitysaskel.



Kuva 5: Päivitysaskeleen erikoistapaus.

jonojen läpikäyntiin on hyvä ymmärtää.

Läpikäydään merkkijonon T haarautuvat osajonot taulukoiden SA ja LCP avulla seuraavasti. Perusideana on pitää jokaisella askeleella i muistissa vain *oikean puoleisin polku*, joka vastaa loppuosaa $SA[i]$. Oikean puoleisin polku pidetään pinossa R, jossa päällimmäisenä on lehtisolmu ja tämän alla yksi tai useampi sisäsolmu. Lehtisolmu vastaa koko loppuosaa $T[SA[i] \ldots n]$ ja sisäsolmut tämän loppuosan alkuosia. Jokaiselle solmulle v pinossa tallennetaan *merkkijonosyvyys* ℓ_v siten, että solmu v edustaa loppuosan $SA[i]$ alkuosaa $T[SA[i] \ldots SA[i] + \ell_v - 1]$.

Esimerkki 4 *Kuvassa 3 on pinon R tilanne, kun loppuosat 1-9 on jo käsitelty. Kuvassa oleva polku vastaa siis loppuosaa $T[SA[9] \ldots n] = \texttt{aaba\$}_1$. Solmujen merkkijonosyvyydet ovat juurisolmusta alkaen $0, 1, 2, 3$ ja 5, missä esimerkiksi $|\texttt{a}| = 1$ ja $|\texttt{aaba\$}_1| = 5$.*

Tarkastellaan seuraavaksi, kuinka pinon R solmuja päivitetään askeleella i. Oletetaan, että ensimmäiset $i-1$ loppuosaa on jo käsitelty, ja pinossa R on loppuosaa $SA[i-1]$ vastaava polku. Kun halutaan siirtyä seuraavaan loppuosaan $SA[i]$, poistetaan ensin pinosta R kaikki solmut, joiden merkkijonosyvyys on suurempi kuin arvo $LCP[i]$. Jos pinossa seuraavana olevan solmun merkkijonosyvyys on pienempi kuin arvo $LCP[i]$, lisätään pinoon uusi sisäsolmu, jonka merkkijonosyvyys on $LCP[i]$. Lopuksi pinoon lisätään aina uusi lehtisolmu merkkijonosyvyydeltään $n - SA[i] + 1$. Nyt pino R on päivitetty vastaamaan loppuosaa $SA[i]$.

Annetaan seuraavaksi kaksi esimerkkiä pinon R päivitysaskeleesta joukolle $\mathcal{D} = \{\texttt{aaba\$}_1, \texttt{abaaab\$}_2, \texttt{bbabb\$}_3, \texttt{abba\$}_4\}$, jonka SA- ja LCP-taulukot ovat nähtävissä kuvasta 2. Ensimmäisessä esimerkissä ensin poistetaan solmuja niiden merkkijono-

syvyyden mukaan ja lopuksi lisätään uusi lehtisolmu. Jälkimmäisessä esimerkissä joudutaan tämän lisäksi lisäämään uusi sisäsolmu.

Esimerkki 5 *Oletetaan, että loppuosat 1-9 on jo käsitelty ja pinossa R on kuvan 3 mukaiset solmut. Kun halutaan edetä seuraavaan loppuosaan $SA[10]$, poistetaan ensin pinosta kaikki solmut, joiden merkkijonosyvyys on suurempi kuin $LCP[10] = 1$. Poistetut solmut on piirretty kuvassa 4 katkoviivalla. Tämän jälkeen pinoon lisätään uusi lehtisolmu, joka vastaa loppuosaa $T[SA[10] \ldots n] =$ $\texttt{ab\$}_2$. Lopulta tilanne pinossa R näyttää samalta kuin kuvassa 4.*

Esimerkki 6 *Oletetaan, että loppuosat 1-10 on jo käsitelty ja pinossa R on kuvan 4 mukaiset solmut. Kun halutaan edetä seuraavaan loppuosaan $SA[11]$, poistetaan ensin pinosta kaikki solmut, joiden merkkijonosyvyys on suurempi kuin $LCP[11] = 2$. Koska pinosta ei löydy sisäsolmua merkkijonosyvyydellä $LCP[11]$, lisätään uusi sisäsolmu, jonka merkkijonosyvyys on $LCP[11]$. Tämän jälkeen pinoon lisätään uusi lehtisolmu, joka vastaa loppuosaa $T[SA[11] \ldots n] =$ $\texttt{aba\$}_1$. Lopulta tilanne pinossa R näyttää samalta kuin kuvassa 5.*

Pinoon R lisätty sisäsolmu on aina kahden peräkkäisen lehtisolmun *syvin yhteinen esi-isä* (lowest common ancestor) [KLA+01]. Toisin sanoen, merkkijonon T haarautuvia osajonoja ovat siis kaikki ne osajonot, joita pinoon R lisätyt sisäsolmut edustavat. Kun edellä kuvattu pinon päivitysaskel toistetaan kaikille $1 \leq i \leq n$, tulevat kaikki merkkijonon T haarautuvat osajonot läpikäydyiksi.

Esimerkki 7 *Kuvasta 5 nähdään, että sisäsolmua edustava osajono* `ab` *on haarautuva: selvästi sekä osajono* `aba` *että* $\texttt{ab\$}_2$ *kuuluvat merkkijonoon T.*

3.2 Osajonojen kattavuuden laskeminen

Haarautuvien osajonojen kattavuudet voidaan laskea Huin [Hui92] menetelmällä, jossa jokaiselle haarautuvalle osajonolle ϕ ylläpidetään kahta laskuria $S(\phi, \mathcal{D})$ ja $C(\phi, \mathcal{D})$. Laskuri S kertoo osajonon ϕ esiintymien lukumäärän joukossa $\mathcal{D}$. Tarkemmin sanottuna,

$$S(\phi, \mathcal{D}) = \sum_{T' \in \mathcal{D}} |\{\ 1 \leq i \leq |T'| \quad : \quad \phi = T'[i \ldots i + |\phi| - 1]\ \}|.$$

Laskuri C kertoo osajonon ϕ *toistuvien esiintymien* lukumäärän saman merkkijonon sisällä siten, että

$$C(\phi, \mathcal{D}) = \sum_{T' \in \mathcal{D} \text{ ja } \phi \preceq T'} \Big(\; |\{ \; 1 \leq i \leq |T'| \quad : \quad \phi = T'[i \ldots i + |\phi| - 1] \; \}| \; - 1 \; \Big).$$

Laskureista S ja C saadaan yksinkertaisella vähennyslaskulla osajonon ϕ kattavuudeksi $freq(\phi, \mathcal{D}) = S(\phi, \mathcal{D}) - C(\phi, \mathcal{D})$.

Esimerkki 8 *Tarkastellaan merkkijonojoukkoa* $\mathcal{D} = \{\texttt{ababa}, \texttt{abacac}\}$ *ja sen osajonoja* `ab` *ja* `ac`*. Osajonolla* `ab` *on yhteensä* $S(\texttt{ab}, \mathcal{D}) = 3$ *esiintymää, joista* $C(\texttt{ab}, \mathcal{D}) = 1$ *esiintymä on toistuva esiintymä joukon ensimmäisessä merkkijonossa. Osajonon* `ab` *kattavuus on* $freq(\texttt{ab}, \mathcal{D}) = 3 - 1 = 2$*. Osajonolla* `ac` *on vastaavasti yhteensä* $S(\texttt{ac}, \mathcal{D}) = 2$ *esiintymää, joista* $C(\texttt{ac}, \mathcal{D}) = 1$ *esiintymä on toistuva esiintymä. Osajonon* `ac` *kattavuus on siis* $freq(\texttt{ac}, \mathcal{D}) = 2 - 1 = 1$.

Laskurien S ja C arvot lasketaan samalla, kun haarautuvia osajonoja läpikäydään luvussa 3.1 kuvatulla menetelmällä. Jokaiseen solmuun pinossa R liitetään muuttujat S ja C. Laskurin S arvot saadaan yksinkertaisesti alustamalla lehtisolmun muuttuja S aina arvoon 1 ja sisäsolmun arvoon 0. Kun pinosta R poistetaan päällimmäinen solmu, sen S-arvo lisätään aina pinossa seuraavana olevan solmun S-muuttujaan. Sisäsolmun v muuttujan S arvo summautuu lopulta yhtäsuureksi kuin sen alipuun lehtisolmujen lukumäärä: lehtisolmujen lukumäärä on samalla solmua v edustavan osajonon esiintymien lukumäärä joukossa $\mathcal{D}$ [Hui92].

Muuttujien C päivittäminen on hankalampaa. Otetaan käyttöön aputaulukko D, johon tallennetaan tieto, mitä merkkijonoista $T_j \in \mathcal{D}$ kukin loppuosista $SA[i]$ vastaa. Asetetaan $D[i] = j$, jos loppuosa $T[SA[i] \ldots n]$ alkaa merkkijonosta T_j. Taulukon D päälle määritellään toinen aputaulukko P, joka kytkee toisiinsa taulukon D arvot. Taulukko P sisältää viittauksen aina edelliseen yhtä suureen arvoon taulukossa D: arvo $P[i]$ on suurin mahdollinen i', jolla $D[i'] = D[i]$ ja $i' < i$. Jos ehto ei täyty millään $i' < i$, niin $P[i] = -1$.

Esimerkki 9 *Kuvassa 6 on esimerkki taulukoista* D *ja* P *joukolle* $\mathcal{D}$= $\{\texttt{aaba\$}_1, \texttt{abaaab\$}_2, \texttt{bbabb\$}_3, \texttt{abba\$}_4\}$*. Esimerkiksi* $D[14] = 4$*, koska loppuosa* $T[SA[14] \ldots n]$ *alkaa neljännestä merkkijonosta. Vastaavasti* $P[14] = 6$*, koska arvon 4 edellinen esiintymä taulukossa* D*, ennen positiota 14, on positiossa 4.*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
T:	a	a	b	a	$_1	a	b	a	a	a	b	$_2	b	b	a	b	b	$_3	a	b	b	a	$_4
SA:	5	12	18	23	4	22	8	9	1	10	2	6	15	19	11	17	3	21	7	14	16	20	13
D:	1	2	3	4	1	4	2	2	1	2	1	2	3	4	2	3	1	4	2	3	3	4	3
P:	-1	-1	-1	-1	1	4	2	7	5	8	9	10	3	6	12	13	11	14	15	16	20	18	21

Kuva 6: Taulukot D ja P joukon $\mathcal{D} = \{\texttt{aaba\$}_1, \texttt{abaaab\$}_2, \texttt{bbabb\$}_3, \texttt{abba\$}_4\}$ katenoidulle merkkijonolle T.

Pinoon R lisättävien solmujen C-arvo on aluksi 0. Kun pino R on päivitetty vastaamaan loppuosaa $SA[i]$, etsitään pinosta R merkkijonosyvyydellä

$$\ell = LCP[\ RMQ_{LCP}(P[i]+1, i)\ ]$$

oleva solmu ja kasvatetaan sen C-muuttujaa yhdellä. Merkkijonosyvyydellä ℓ oleva solmu löydetään vakioajassa, kun pinon R ohella ylläpidetään n alkion taulukkoa, jonka ℓ_v:s alkio osoittaa aina solmuun, jonka merkkijonosyvyys on ℓ_v [FMV08].

Esimerkki 10 *Olkoon joukko $\mathcal{D}$ kuten kuvassa 2. Kun loppuosa $SA[11]$ on käsitelty, lasketaan osavälin $[P[11]+1, 11] = [10, 11]$ minimi taulukosta LCP, eli $RMQ_{LCP}(10, 11) = 10$ ja $LCP[10] = 1$. Tämän jälkeen solmun, jonka merkkijonosyvyys pinossa R on $LCP[10] = 1$, muuttujaa C kasvatetaan yhdellä.*

Tähän mennessä on näytetty, kuinka pinossa R olevien solmujen S- ja C-muuttujia päivitetään samalla, kun merkkijonon haarautuvat osajonot läpikäydään. Muuttujien S ja C avulla voidaan lopulta laskea osajonon kattavuus. Kun osajonoa ϕ vastaava solmu v poistetaan pinosta, lasketaan osajonon kattavuus $freq(\phi, \mathcal{D}) = S - C$ ja tarkistetaan täyttyvätkö annetut kattavuusrajoitteet. Jos osajonon kattavuudelle pätee $min \leq freq(\phi, \mathcal{D}) \leq max$, niin tulostetaan kaikki osajonot $T[SA[i] \ldots SA[i] + h - 1]$, missä i on askel, jolla solmu v poistettiin pinosta. Muuttuja h saa arvonsa merkkijonosyvyyksien suljetulta väliltä $[\ell_{v'}, \ell_v]$, missä v' on v:n isäsolmu.

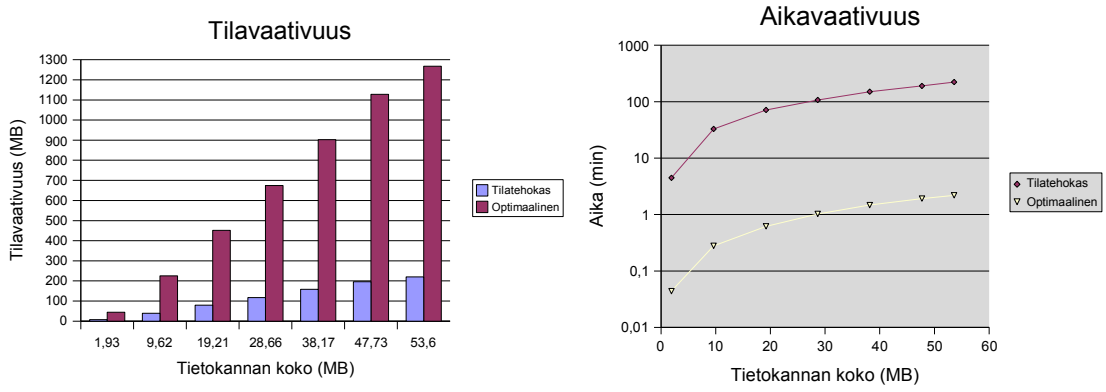
Kuvassa 7 on louhinta-algoritmin esitys kokonaisuudessaan pseudokoodina. Algoritmin suoritusaika on optimaalinen $O(n+s)$, missä n on syötteen pituus ja s tuloksen koko. Rivejä 14–15 lukuunottamatta algoritmi toimii selvästi ajassa $O(n)$: algoritmin suorituksen aikana pinoon R lisätään $O(n)$ solmua, joten silmukkaa riveillä 7–19 toistetaan yhteensä korkeitaan $O(n)$ kertaa algoritmin koko suorituksen aikana. Riveillä 14–15 kuluu yhteensä $O(s)$ aika.

Tuloksen koko s voi pahimmassa tapauksessa olla $\Theta(n^3)$. On kuitenkin osoitettu, että tulosjoukon esitystapaa muuttamalla sekä algoritmin aikavaativuus että tuloksen

Syöte: Tietorakenteet SA, LCP, RMQ_{LCP}, T ja P joukolle $\mathcal{D}$. Kattavuusrajoite (min, max).
Tulos: Kaikki joukon $\mathcal{D}$ osajonot, joilla kattavuusrajoite on voimassa.

```
1  Olkoon R pino, johon tallennetaan kolmikkoja (v.h, v.S, v.C).
2  Lisätään pinoon R pysäytyssolmu merkkijonosyvyydeltään v.h = −∞.
3  Olkoon B aputaulukko, missä B[ℓ_v] osoittaa solmuun merkkijonosyvyydellä ℓ_v.
4  for i = 1, ..., n + 1 do
5      v ← top(R)        { v edustaa seuraavaksi tutkittavaa osajonoa }
6      S ← 0,   C ← 0
7      while v.h > LCP[i] do        { Oletetaan, että LCP[n + 1] = −1 }
8          v ← pop(R),   w ← top(R)        { w osoittaa pinon päälle }
9          if w.h ≥ LCP[i] then
10             w.S += v.S,   w.C += v.C
11         end if
12         freq ← v.S − v.C        { Solmua v edustavan osajonon kattavuus }
13         if min ≤ freq ≤ max then
14             for h = max{w.h, LCP[i]} + 1, ..., v.h do
15                 Tulosta T[SA[i] ... SA[i] + h − 1].
16         end if
17         S ← v.S,   C ← v.C
18         v ← w
19     end while
20     if v.h < LCP[i] then
21         Lisää (LCP[i], S, C) pinoon R ja päivitä osoitin B[LCP[i]].
22     end if
23     if i ≤ n then
24         ℓ ← LCP[ RMQ_LCP(P[i] + 1, i) ]
25         Kasvata solmun B[ℓ] muuttujaa C yhdellä.
26         Lisää (n − SA[i] + 1, 1, 0) pinoon R ja päivitä osoitin B[n − SA[i] + 1].
27     end if
28 end for
```

Kuva 7: Merkkijonojoukkojen louhinta-algoritmi [FHK06].



Kuva 8: Vasemmalla louhinta-algoritmien tilavaativuus, ja oikealla louhintaan kulunut aika logaritmisessa asteikossa.

koko saadaan pidettyä luokassa $O(n)$ [Fis07]. Louhinta-algoritmin tilavaativuus on $O(n)$ sanaa, eli $O(n \log n)$ bittiä RAM-laskennan mallissa.

4 Algoritmin suorituskyky käytännössä

Louhinta-algoritmin toteutusta[1] testattiin aineistolla, joka kerättiin ihmisen ja hiiren proteiineista. Positiivinen merkkijonojoukko sisälsi 71 622 ihmisen proteiinia yhteispituudeltaan noin 28,6 MB ja negatiivinen 62 562 hiiren proteiinia yhteispituudeltaan noin 27,6 MB. Testit ajettiin tietokoneella, jossa oli Intel Pentium 4 (3.00GHz) -suoritin ja 3 GB keskusmuistia.

Luvussa 3 kuvatusta louhinta-algoritmista on olemassa myös tilatehokas toteutus[2]. Tilatehokas variaatio [FMV08] toimii $O(n \log n)$ ajassa ja käyttää $O(n \log \sigma + r \log n)$ bittiä muistia, missä n on syötteen pituus, σ on aakkoston koko ja r merkkijonojen lukumäärä syötteessä. Tilatehokas variaatio louhinta-algoritmista ei saavuta optimaalista aikavaativuutta, mutta sen tilavaativuus on asymptootisesti pienempi kuin $O(n \log n)$ bittiä, jos oletetaan $r = o(n)$. Tilatehokasta variaatiota käytettiin testeissä vertailukohteena.

Tulokset olivat odotetun kaltaisia. Kuvasta 8 nähdään, että tilatehokas algoritmi vaati suurimmalla aineistolla vain noin kuudesosan optimaalisessa ajassa toimivan algoritmin vaatimasta muistimäärästä. Vastaavasti optimaalisessa ajassa toimiva al-

[1] http://www.bio.ifi.lmu.de/~fischer/frequentLinear.tgz
[2] http://www.cs.helsinki.fi/group/suds/fsm/

goritmi selvisi louhinnasta muutamassa minuutissa, kun tilatehokkaalla algoritmilla kului samaan tehtävään muutamia tunteja. Optimaalisessa ajassa toimivan algoritmin tila- ja aikavaativuus käyttäytyivät lineaarisesti syötteen pituuden suhteen.

5 Yhteenveto

Tämä tutkielma tarkasteli merkkijonojoukkojen louhintaa, kun halutaan löytää kaikki osajonot, jotka täyttävät annetut kattavuusrajoitteet. Ongelma ratkeaa optimaalisessa ajassa syötteen pituuden suhteen, mutta algoritmi vaatii käytännössä huomattavan määrän muistia toimiakseen. Tilatehokkaita tietorakenteita käyttämällä tilavaativuus saadaan pienemmäksi, mutta samalla aikavaativuus kasvaa optimaalisesta $O(\log n)$ kertoimen verran. Koska käytössä olevan muistin määrä muodostaa yleensä konkreettisemman rajoitteen kuin käytössä oleva laskenta-aika, on tilatehokas menetelmä kuitenkin mielenkiintoinen vaihtoehto merkkijonojoukkojen louhintaan.

Merkkijonojoukkojen louhinta liittyy läheisesti *tietoalkiojoukkojen louhintaan* [AIS93, AS94], joka on yksi keskeisimmistä tiedonlouhinnan ongelmista. Tietoalkiojoukkojen louhintaa käytetään muun muassa assosiaatiosääntöjen louhinnassa. Ikävä kyllä tietoalkiojoukkojen ja merkkijonojoukkojen louhinnan välille ei näyttäisi löytyvän suoraa palautusta. Tietoalkiojoukkojen louhinta optimaalisessa ajassa, algoritmilla joka olisi mahdollisesti myös tilatehokas, jätetään tulevaisuuden tutkimushaasteeksi.

Lähteet

AIS93 Agrawal, R., Imielinski, T. ja Swami, A. N., Mining association rules between sets of items in large databases. *Proc. Int. Conf. on Management of Data*. ACM Press, 1993, sivut 207–216.

AS94 Agrawal, R. ja Srikant, R., Fast algorithms for mining association rules. *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. Morgan Kaufmann, 1994, sivut 487–499.

BFC00 Bender, M. A. ja Farach-Colton, M., The LCA problem revisited. *Proc. LATIN*, osa 1776 sarjasta *LNCS*. Springer, 2000, sivut 88–94.

CKYT03 Chan, S., Kao, B., Yip, C. L. ja Tang, M., Mining emerging substrings. *Proc. of the Itl. Conf. on Database Systems for Advanced Applications (DASFAA)*. IEEE Computer Society, 2003, sivut 119–126.

DL99 Dong, G. ja Li, J., Efficient mining of emerging patterns: Discovering trends and differences. *Proc. KDD*. ACM Press, 1999, sivut 43–52.

RJL+02 De Raedt, L., Jäger, M., Lee, S. D. ja Mannila, H., A theory of inductive query answering. *Proc. ICDM*. IEEE Computer Society, 2002, sivut 123–130.

FHK06 Fischer, J., Heun, V. ja Kramer, S., Optimal string mining under frequency constraints. *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, osa 4213 sarjasta *LNCS*. Springer, 2006, sivut 139–150.

Fis07 Fischer, J., *Efficient Data Structures for String Algorithms*. Väitöskirja, LMU München, 2007.

FMV08 Fischer, J., Mäkinen, V. ja Välimäki, N., Space-efficient string mining under frequency constraints. Submitted to appear 2008.

Hui92 Hui, L. C. K., Color set size problem with application to string matching. *Proc. CPM*, osa 644 sarjasta *LNCS*. Springer, 1992, sivut 230–243.

KLA+01 Kasai, T., Lee, G., Arimura, H., Arikawa, S. ja Park, K., Linear-time longest-common-prefix computation in suffix arrays and its applications. *Proc. CPM*, osa 2089 sarjasta *LNCS*. Springer, 2001, sivut 181–192.

KSB06 Kärkkäinen, J., Sanders, P. ja Burkhardt, S., Linear work suffix array construction. *Journal of the ACM*, 53,6(2006), sivut 918–936.

MM93 Manber, U. ja Myers, E. W., Suffix arrays: A new method for on-line string searches. *SIAM J. Comput.*, 22,5(1993), sivut 935–948.

Wei73 Weiner, P., Linear pattern matching algorithms. *Proc. Annual Symp. on Switching and Automata Theory*. IEEE Computer Society, 1973, sivut 1–11.