Mihin algoritmeja tarvitaan?

Esko Ukkonen

Alkulukutestaus ja bioinformatiikka ovat ajankohtaisia algoritmitutkimuksen alueita. Kumpikin on kiinnostava sekä algoritmiteorian että sovellusten kannalta. Äskettäin esitetty nopea alkulukutesti ratkaisi klassisen lukuteoreettisen ongelman ja toi samalla uutta puhtia tiedonsuojausmenetelmien tutkimukseen. Bioinformatiikasta on puolestaan tullut uuden molekyylibiologian kehityksen seurauksena voimakkaasti laajeneva monitieteinen tutkimusala, joka tarjoaa uudentyyppisiä haasteita algoritmitutkimukselle. Suomalaiset ovat olleet mukana bioinformatiikan algoritmien kehittämisessä alusta alkaen.

Algoritmi on tietojenkäsittelytieteen keskeinen käsite. Algoritmeja tarvitaan, jotta tietokoneita osattaisiin ohjelmoida. Kukin algoritmi määrittelee äärellisistä ja täsmällisistä laskenta-askelista muodostuvan kokonaisuuden, joka tuottaa annetuista syöttötiedoista halutun tuloksen. Tietokoneohjelmat ovat abstraktien algoritmien konkreettisia toteutuksia, joita tietokone pystyy suorittamaan erilaisten tehtävien ratkaisemiseksi.

Algoritmien tutkimuksella on kaksi päätavoitetta. Toisaalta kehitetään yhä parempia algoritmeja yhä uusille laskentaongelmille. Tämä työ on usein hyvin sovellushakuista ja käytännöllistä. Toisaalta pyritään ymmärtämään algoritmien yleisiä rajoituksia ja laskentaongelmien sisäistä vaikeutta eli ns. laskennallista vaativuutta.

Tämä vastakkaisista suunnista etenevä tutkimusohjelma luo täsmällisen perustan arvioida algoritmitutkimuksen edistystä. Ongelman sisäisen vaikeuden analysointi antaa alaraja-arvion siitä kuinka paljon laskentaa ongelman ratkaiseminen vähintään vaatii, käytettiinpä mitä algoritmia tahansa. Ongelmalle esitetty uusi ratkaisualgoritmi antaa puolestaan erään ylärajan sille, kuinka suuri laskenta ainakin riittää ongelmamme ratkaisemiseksi. Kun nämä kaksi arviota kohtaavat, tiedetään että käsillä oleva algoritmi on tietyssä mielessä paras mahdollinen ongelman ratkaisu.

Algoritmitutkimuksen keskeinen asema tietojenkäsittelytieteessä näkyy esimerkiksi siitä, että Kansainvälisen matemaattisen unionin myöntämä Nevanlinna-palkinto on tähän mennessä aina jaettu tutkijalle, jonka työ liittyy syvällisellä tavalla algoritmiteoriaan. Tämä arvostettu palkinto annetaan merkittävästä tietojenkäsittelytieteen matemaattisiin aspekteihin liittyvästä tutkimustyöstä.

Alkulukutestaus helppoa

Eräs algoritmitutkimuksen tuore huomattava tulos on lausuttavissa lyhyesti: *alkulukutestaus kuuluu joukkoon P*. Tieto tästä saavutuksesta levisi vuoden 2002 kesällä ja jopa *New York Times* kertoi siitä. Mistä on kyse, kun tällainen teoreettisluonteinen tulos pystyi poikkeuksellisesti ylittämään uutiskynnykset?

Intian teknologisen instituutin tutkijat Manindra Agarwal, Neeraj Kayal ja Nitin Saxena onnistuivat löytämään algoritmin, joka testaa onko mielivaltainen annettu kokonaisluku ns. alkuluku, siis sellainen luku, että jos se yritetään jakaa millä tahansa sitä pienemmällä kokonaisluvulla, jako ei mene tasan (paitsi jos jakajana on 1). Tämä keksijöidensä nimien perusteella AKS-algoritmiksi nimetty algoritmi on lisäksi sellainen – ja tämä on tuloksen uusi tärkeä ominaisuus – että sen toiminta-aika kasvaa tutkittavan luvun pituuden kasvaessa varsin kohtuullisesti.

Ollaksemme täsmällisiä, AKS-algoritmi vaatii laskenta-ajan joka on verrannollinen testattavan luvun pituuden potenssiin 12. Asteluku 12 on tosin algoritmin käytännön soveltamisen kannalta kiusallisen korkea, mutta oleellista onkin vain se, ettei asteluku riipu testattavasta luvusta. Teknisemmin ilmaistuna sanotaan, että tällaisen algoritmin aikavaatimus on polynominen syötteen pituuden suhteen. Edellä mainittu joukko *P* koostuu laskentaongelmista, joilla on tällainen enintään polynomisessa ajassa toimiva ratkaisualgoritmi. Intialaisten tulos lopetti pit-

kään jatkuneen epätietoisuuden alkulukutestauksen vaikeudesta: alkulukutestaus on kuin onkin joukon *P* jäsen.

Miksi sitten joukon *P* jäsenyys on arvokasta? Eikö riitä että ongelmalla on ylipäänsä jokin ratkaisualgoritmi, jolloin algoritmi voidaan ohjelmoida tietokoneelle ja sitten pannaan kone vain töihin? Asia ei ole näin yksinkertainen. Jos halutaan, että tietokone saa tulokset valmiiksi säällisessä ajassa, algoritmin pitää olla riittävän nopea.

Alkulukutestaus tarjoaa tästä asiasta valaisevan esimerkin. Välittömästi nimittäin huomataan, että alkulukutestaus onkin aivan helppo ratkaista: annetun luvun *i* alkulukuominaisuuden testaamiseksi kokeillaan vuoronperään kaikilla lukua *i* pienemmillä kokonaisluvuilla luvusta 2 alkaen, jakavatko ne tasan luvun *i*. Tasan jakaminen selviää tunnetulla koulussa opitulla menetelmällä, ja jos yksikin tasan menevä jako löytyy, *i* ei ole alkuluku, mutta muuten on.

Tämä algoritmi on kuitenkin hitautensa takia täysin käyttökelvoton. Jos luvussa *i* on vaikkapa 300 numeroa (näin pitkiä kokonaislukuja käytetään nykyaikaisissa tiedonsuojausmenetelmissä), on helppo arvioida, että algoritmimme tarvitsema jakolaskujen määrä on suuruudeltaan niin supertähtitieteellinen, että mikään nykyisten tietokoneiden kyvyillä varustettu laskentalaite (tai rakennettavissa oleva rinnakkaisten laskentalaitteiden kokoelma) ei pysty saamaan tulosta valmiiksi siedettävässä ajassa, ei vaikka koneiden nopeus tunnetun Mooren lain mukaisesti kasvaisi rajustikin. Tarvitaan parempi algoritmi.

Luokkaa *P* voidaan myös luonnehtia niin, että se on ”käytännössä” algoritmeilla ratkeavien ongelmien luokka. Kokemusperäisesti on nimittäin havaittu että polynomisessa ajassa toimiva algoritmi on yleensä käyttökelpoisen nopea. Jos toisaalta aikavaatimus kasvaa nopeammin kuin mikään polynomi, ei algoritmi voi olla yleisesti käyttökelpoinen. Polynomisissa algoritmeissa on myös teoreettista viehätystä: niiden edellytyksenä yleensä on, että ongelman sisäisestä kombinatorisesta rakenteesta on paljastunut vahvaa säännönmukaisuutta, joka avulla ratkaisu löytyy ”suoraan” ilman että on tarpeen turvautua edellä mainittuun hitaaseen yrityksen ja erehdyksen menetelmään.

Alkulukutestauksen saaminen luokkaan *P* onkin huomattava saavutus, onhan alkulukuominaisuus klassinen matemaattinen käsite jonka tutkimuksella on vuosituhantinen perinne. Kiinalaisilla oli jo noin 10. vuosisadalla ennen ajanlaskumme alkua (virheelliseksi osoittautunut) ehdotus alkulukutestausalgoritmiksi, ja (oikein toimiva mutta hidas) ns. Eratosteneen seula on noin vuodelta 240 ennen ajanlaskun alkua.

Lisäksi tunnetaan useita uudempia alkulukutestejä, joiden tehokkuus on jonkun lisäoletuksen (kuten laajennetun Riemannin hypoteesin) varassa tai jotka eivät ole deterministisiä vaan käyttävät apuna lantinheittoa. Tällaiset testit ovat jo laajassa käytössä eikä ole selvää, että uusi deterministinen AKS-algoritmi on käytännössä näitä parempi. Deterministinen algoritmi on kuitenkin ratkaissut ongelman sisäistä vaikeutta koskevan pitkäaikaisen avoimen kysymyksen. Mahdollisesti algoritmia pystytään vielä parantamaan paljonkin, nyt kun pää on saatu auki.

Tekijöihin jako vaikeaa

Alkulukutestaus on klassista lukuteoriaa ja sellaisenaan kiehtova ongelma, jonka ratkaisulla voi olla odottamattomiakin seurauksia. Vielä kiinnostavampi on sen lähisukuinen ongelma, nimittäin annetun luvun jakaminen tekijöihinsä. Nyt tehtävänä on selvittää, mitkä ovat annetun kokonaisluvun tekijät ts. luvut, joiden tulo annettu luku on.

Toistaiseksi ei tunneta nopeaa tekijöihinjakoalgoritmia. Toisaalta käänteinen tehtävä, luvun määrittäminen kun sen tekijät on annettu, on nopeasti laskettavissa koulussa opitulla kertolaskualgoritmilla. Tekijöihin jako onkin lupaava ehdokas niin sanotuksi yksisuuntaiseksi funktioksi. Tällaisen funktion laskeminen on nopeaa toiseen suuntaan mutta hidasta toiseen.

Yksisuuntaisia funktioita tarvitaan nykyaikaisissa julkisen avaimen salakirjoitukseen perustuvissa tiedonsuojausmenetelmissä. Näiden menetelmien turvallisuus on yksisuuntaisuusoletuksen varassa. Paljon käytetty Rivest-Shamir-Adleman salakirjoitusmenetelmä käyttää tekijöihin jakoa yksisuuntaisena funktiona. Sen turvallisuus perustuu siis oletukseen, että tekijöihin jako on hidasta. Nopean tekijöihinjakoalgoritmin löytyminen murtaisi samalla tämän salakirjoitusmenetelmän.

Tätä kirjoitettaessa ei ole tiedossa mitään siihen viittaavaa, että uuden AKS-alkulukutestin seurauksena myös tekijöihin jakaminen ratkeaisi nopeammin. Uusi testi ei näytä antavan viitteitä testattavan luvun tekijöiden arvoista; se ainoastaan sanoo onko tekijöitä olemassa.

Tilanne saattaa muuttua, jos ns. kvanttitietokone onnistutaan joskus rakentamaan. Tällainen kone osaisi toteuttaa uudentyyppisiä laskenta-

operaatioita, joita käyttäen on mahdollista suorittaa tiettyjä tehtäviä oleellisesti nopeammin kuin nykyisillä tietokoneilla. Nevanlinna-palkittu Peter Schorr osoitti jo lähes 10 vuotta sitten, että kvanttitietokoneen laskentaoperaatioiden avulla voidaan tekijöihin jakaminen suorittaa nopeasti. Kvanttitietokoneen tultua jouduttaisiinkin salakirjoitusmenetelmät miettimään uudestaan.

Merkkijonoalgoritmit ja bioinformatiikka

Toinen algoritmitutkimuksen ajankohtainen ongelmien lähde ja sovellusalue on molekyylibiologia. Lähtökohtana on DNA-molekyylin sisältämän perinnöllisen informaation koodaustapa. DNA-molekyylin oleellinen sisältö voidaan kuvata neljää erilaista merkkiä (*A*, *C*, *G*, *T*) sisältävänä sekvenssinä eli yksinkertaisena merkkijonona. Kun tietokoneet käyttävät tiedon esittämiseen kaksiarvoisia bittejä ja ihmiset joitakin kymmeniä erilaisia kirjainmerkkejä tai tuhansia sanamerkkejä, on luonto siis omaksunut neliarvoiset ”kvatit” DNA:n sisältämän perinnöllisen informaation koodaamiseen.

Useiden organismien DNA:n merkkijonot on jo selvitetty (eli ”sekvensoitu”) ja useita on valmistumisvaiheessa. Näin kertyy valtavasti mielenkiintoista merkkijonomuotoista dataa. Tämän datan käsittely ja analyysi on tehtävä, jossa tietojenkäsittelytieteen piirissä pitkään jatkuneella merkkijonomenetelmien ja muiden kombinatoristen algoritmien tutkimuksella on paljon annettavaa, koska kyseessä on luonteeltaan diskreeteistä symboleista (eikä enemmän tai vähemmän epätarkoista luvuista) koostuva data.

Uuden molekyylibiologisen datan analyysitarpeet ovat synnyttäneet uuden tutkimusalan, bioinformatiikan. Sillä tarkoitetaan biologisen informaation ja biologisten systeemien rakenteen analyysiä ja mallittamista tietojenkäsittelytieteen, tilastotieteen ja matematiikan tarjoamien välineiden avulla. Bioinformatiikasta on tullut bioteknisen tutkimuksen ja teollisuuden eräs keskeinen tekijä, jonka osaajista on maailmanlaajuinen pula.

DNA-palapeli

DNA-jonojen kokoaminen lyhyemmistä palasista on eräs välttämätön laskentatehtävä, joka pitää ratkaista kaikissa genomien sekvensointihankkeissa. Ongelma syntyy siitä, että DNA:n rakennetta osataan lukea laboratoriossa sekvensointilaitteita käyttäen vain suhteellisen lyhyissä palasissa. Näiden palasten pituus on tyypillisesti 150–800 merkkiä. Paloista pitäisi koota koko genomia esittävä oikeassa järjestyksessä oleva yhtenäinen merkkijono, jonka pituus on kenties miljoonia tai miljardeja merkkejä.

Palojen oikea sijainti ja kulkusuunta lopullisessa jonossa ei ole etukäteen tarkalleen tiedossa vaan ne voivat olla enemmän tai vähemmän ‘haulikolla ammuttuja’. Lisäksi paloissa voi olla pieni määrä eri syistä johtuvia virheitä (puuttuvia, muuttuneita tai lisättyjä merkkejä).

DNA-jonon palasista muodostuu tällä tavoin varsin ainutlaatuinen, valtaisa palapeli. Esimerkiksi ihmisen genomin kokoamista varten näitä paloja on tuotettu lähes 30 miljoonaa ja palapelin lopputuloksena pitäisi muodostua noin 3 miljardia merkkiä pitkä jono. Palapelin ratkaiseminen on hieno algoritmitutkimuksen ongelma, joka on algoritmiteoreettisesti haastava ja jolla ilmeisestikin on relevanttia käyttöä.

Ratkaisu lähtee liikkeelle siitä, että joidenkin palasten tiedetään sijaitsevan osittain päällekkäin alkuperäisessä jonossa. Sekvensointi on tätä varten tarkoituksellisesti tehty niin, että palaset peittävät saman alueen moneen kertaan. Silloin palasten oikeasta keskinäisestä järjestyksestä saadaan vihiä etsimällä pareja, joilla on samanlainen alku- ja loppuosa ja jotka näin ollen voisivat olla alkuperäisessä jonossa osittain päällekkäin.

Näin muodostuu palaparien keskinäistä järjestystä koskevia paikallisia ehdotuksia. Näiden perusteella voidaan seuraavaksi muodostaa ehdotuksia yhä laajempien palajoukkojen keskinäiseksi asetelmaksi. Tehtävää vaikeuttaa se, että usein on tarjolla useita erilaisia lupaavia asetelmavaihtoehtoja. Tämä voi johtua siitä, että paloja on sekvensoitu liian vähän tai paloissa on virheitä. Suurin vaikeus on kuitenkin se, että DNA-jonot voivat sisältää useaan kertaan toistuvia pitkiä jaksoja, mikä voi sotkea palapelin sallimalla useita ratkaisuja.

Kuvattu palapeli johtaa useisiin algoritmisiin osatehtäviin. Tarvitaan esimerkiksi merkkijonomenetelmiä palaparien likimääräisten päällekkäisyyksien hakemiseen ja erilaisia verkkoteorian algoritmeja palapelin kokonaisratkaisun muodostamiseen. Jos ratkaisu muotoillaan kombinatoriseksi optimointiprobleemaksi, se osoittautuu tunnetun ”kauppamatkustajan ongelman” sukulaiseksi ja on luonteeltaan ns. NP–kova-ongelma. Tämän ongelmaluokan tarkka ratkaiseminen on kaikilla tunnetuilla menetelmillä toivottoman hidasta, joten erilaiset likimääräisratkaisut ovat tarpeen. Jonojen toisteisuus vaatii vielä erikseen rää-

tälöityjä menetelmiä, jotta lopputulos olisi biologisesti mahdollisimman tarkka.

Kaikki palapelin vaiheet osataan toteuttaa tehokkailla algoritmeilla, mikä datan suuresta koosta johtuen onkin välttämätöntä. Silti isot genomihankkeet tarvitsevat varsin järeän tietokonepatterin merkkijonojensa hallintaa ja analyysiä varten. Käsityökin on edelleen välttämätöntä ratkaisun viimeistelyvaiheessa.

DNA-jonojen tulkinta

DNA-jonojen kokoaminen kokonaisia genomeja esittäviksi sekvensseiksi on vasta lähtölaukaus jonojen sisällön analysoinnille. Koottukin jono on vain raakadataa joka odottaa tulkitsijaansa. Esimerkiksi varsinaisten geenien paikallistaminen DNA-jonosta laboratoriossa on työläs tehtävä, jota voidaan auttaa laatimalla tietokone-ennusteita geenien paikoista. Tämä on luonteeltaan merkkijonojen luokitteluongelma, joka on onnistuttu ratkaisemaan kohtuullisen hyvin. Uudet arviot eri organismien geenien määristä perustuvat tällaisiin ennusteisiin. Esimerkiksi ihmisen geenien määrä on näin ennustettu selvästi vanhoja arvioita alhaisemmaksi.

On myös havaittu että samankaltaisten jonojen biologinen merkitys on usein sama. Näin syntyy tarve verrata merkkijonoja ja löytää samankaltaisia jonoja tai jonojen alueita (ns. homologioita) tai jonoissa esiintyviä yhteisiä merkkihahmoja. Tähän tarkoitukseen on kehitetty laaja joukko merkkijonojen etäisyysmittoja ja niihin perustuvia algoritmeja ja tietokoneohjelmia, joilla verrataan jonoja toisiinsa ja etsitään annetun jonon sukulaisjonot erilaisista DNA-jonojen tietokannoista. Jälleen algoritmien tehokkuus on tärkeää, koska laajamittaisten sekvensointihankkeiden ansiosta tietokantojen koko kasvaa räjähdysmäisesti.

Merkkijonojen vertailu- ja hakuohjelmat ovat nykyään laajassa päivittäisessä käytössä lähes kaikessa molekyylibiologisessa tutkimuksessa. Biologeista on tullut esimerkiksi Suomen supertietokoneiden suurin käyttäjäryhmä.

Kunkin organismin DNA voidaan ymmärtää eräänlaiseksi ohjelmaksi, jonka ohjaamana organismi elää, kasvaa ja kehittyy. Nykyisen molekyylibiologian suuria kysymyksiä ja samalla bioinformatiikan tutkimuksen keskeinen ajankohtainen haaste on selvittää, miten tämä ohjelma toimii. Työ etenee vähittäin. Koko genomin toiminnasta järjestelmällistä tietoa antavien mittausmenetelmien kehittäminen ja systemaattisten mittausten suorittaminen ”systeemibiologian” hengessä on tässä avainasemassa. Eräs vallankumouksellinen uusi mittauslaite on ns. DNA-siru. Sillä voidaan samanaikaisesti mitata organismin kaikkien – mahdollisesti kymmenien tuhansien – geenien aktiivisuutta eri olosuhteissa.

Mittaukset antavat tietoa DNA-ohjelman toiminnan sisäisistä tiloista, minkä perusteella pyritään muodostamaan esimerkiksi geenien välisiä säätelysuhteita kuvaavia verkostorakenteita tai jopa kokonaisen solun toimintaa kuvaavia tietokonemalleja. Tällaisten säätelyverkkojen ja mallien laskeminen mittausdatan perusteella on jälleen erittäin haasteellinen algoritmitutkimuksen ongelma. Sen ratkaisemiseksi on intensiivinen tutkimus käynnissä eri puolilla maailmaa. Tulokset auttavat ymmärtämään biologisia ilmiöitä yhä täsmällisemmin ja voivat johtaa bioteknisiin sovelluksiin ja jopa molekyylibiologisia toimintaperiaatteita käyttävien uudentyyppisten tietokoneiden kehittämiseen.

KIRJALLISUUTTA:

Agrawal, M., Kayal N. & Saxena N. (2002): *PRIMES is in P*. Indian Institute of Technology, Kanpur, India, August 6 2002, (www.cse.iitk.ac.in/users/manindra/index.html).

Brazma, A., Jonassen, I., Vilo J. ja Ukkonen E. (1998): ”Predicting gene regulatory elements in silico on a genomic scale”. *Genome Research* 8, 1202-1215.

Gusfield, D. (1997): *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.

Myers, E. W. et al. (2000): ”A whole-genome assembly of Drosophila”. *Science* 287, 2196-2204.

Kirjoittaja on akatemiaprofessori Tietojenkäsittelytieteen laitoksella Helsingin yliopistolla. Kirjoitus perustuu esitelmään Tieteen päivillä 8.–12.1.2003.
esko.ukkonen@helsinki.fi