Tampereen yliopisto

Teo Turkkila

ALGORITMIEN TEHOKKUUDEN ANALYSOINTI AMORTISOIDUSTI JA ASYMPTOOTTISESTI

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Toukokuu 2022

TIIVISTELMÄ

Teo Turkkila: Algoritmien tehokkuuden analysointi amortisoidusti ja asymptoottisesti
Amortized and asymptotic analysis of algorithm efficiency
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2022

Algoritmin tehokkuuden analysointi on tärkeää, koska liian tehoton algoritmi voi tehdä ohjelmasta käyttökelvottoman. Tehottomat algoritmit johtavat hitaaseen ohjelmaan ja siten ohjelman kaupallistaminen voi osoittautua mahdottomaksi, vastaavasti tutkimuskäytössä liian hitaalla algoritmilla tuloksia ei saada riittävän nopeasti käytettäväksi. Tässä työssä perehdytään algoritmien amortisoituun ja asymptoottiseen analysointiin, analysointitapojen eroihin, sekä analysointitapojen yhtäläisyyksiin. Työn tavoite on esitellä analysoinnilla saavutettavissa olevia hyötyjä teoreettisella, sekä konkreettisella tasolla.

Työ tehtiin kirjallisuuskatsauksena ja lähteinä käytettiin olemassa olevia tutkimuksia. Lähteet työhön valittiin siten, että saavutettiin mahdollisimman kattava kuvaus algoritmien analysointitavoista, niihin liittyvistä eroista ja yhtäläisyyksistä. Lähteistä valikoitiin mukaan myös sellaiset, jotka esittelivät analysoinnin tärkeyttä ja saavutettavissa olevia hyötyjä.

Työ jakautuu kolmeen osaan, analysointitapojen esittelyyn, analysointitapojen vertailuun ja saavutettavissa olevien hyötyjen esittelyyn. Analysointitapojen vertailun perusteella voidaan päätellä oikean analysointitavan valinnalla olevan suuri vaikutus saatuihin tuloksiin. Työssä havaittiin analysointitavan vaikuttavan oleellisesti mahdollisten tulosten hyödyntämiseen. Asymptoottinen analyysi asettaa raja-arvon, jota algoritmi ei koskaan ylitä tai alita. Amortisoitu analyysi kertoo algoritmin keskimääräisen tehokkuuden, mutta ei kerro raja-arvoista.

Saatujen tulosten perusteella voidaan päätellä konkreettisten hyötyjen olevan aina saavutettavissa, oikealla analyysin valinnalla, sekä helposti todennettavissa algoritmin optimoinnin seurauksena. Työn perusteella voidaan osoittaa algoritmin analysointitavan valinnalla olevan suora yhteys saavutettavissa oleviin hyötyihin. Asymptoottisella analysoinnilla saaduista raja-arvoista voidaan päätellä ohjelmaa suorittavan laitteiston vaatimukset, sekä algoritmin suorittamiseen kuluva aika huonoimmassa mahdollisessa tapauksessa. Amortisoitu analyysi osoittaa tarkasti algoritmin keskimääräisen tehokkuuden, kun algoritmi suoritetaan useamman kerran. Työssä havaittiin asymptoottisen analysoinnin tuloksen olevan subjektiivinen arvo siinä, missä amortisoidun analyysin tulos on objektiivinen.

Avainsanat: Algoritmit, asymptoottinen tehokkuus, amortisoitu tehokkuus, algoritmien analysointi

Sisällysluettelo

1 Johdanto

Nykyaikaisessa tietoyhteiskunnassa algoritmit ovat alati kasvavassa roolissa, algoritmit vaikuttavat päivittäiseen elämään aina sosiaalisen median alustoilta kriittisen infrastruktuurin toimintaan. Algoritmien tulee olla tehokkaita, jotta ne voivat täyttää roolinsa ja siten on ensiarvoisen tärkeää, että niiden tehokkuutta voidaan analysoida luotettavasti. Analysoinnin tulee paljastaa algoritmin teoreettisen tehokkuuden rajat, johon käytetään asymptoottista analysointia, ja algoritmin tehokkuus oletetuissa todellisissa olosuhteissa, jonka arviointiin käytetään amortisoitua analysointitapaa.

Tässä tutkielmassa tehdään kirjallisuuskatsaus algoritmien analysoinnista amortisoidusti ja asymptoottisesti. Työssä esitellään analysointitapojen samankaltaisuuksia, eroavaisuuksia ja analysoinnilla saavutettavissa olevia hyötyjä. Työn ollessa kirjallisuuskatsaus, keskitytään alan kirjallisuuteen ja perustutkimukseen aihepiirin ympäriltä. Lähteiksi on valittu ensisijaisesti yleisen tason tutkimuksia, eikä yksittäisen algoritmin analysointiin keskittyviä tutkimuksia.

Työssä keskitytään esittelemään algoritmien analysointi asymptoottisesti ja amortisoidusti, sekä analysointitapojen samankaltaisuudet ja eroavaisuudet. Algoritmien analysointitapojen jälkeen kerrotaan millaisissa tilanteissa tulisi valita asymptoottinen analysointitapa ja milloin amortisoitu analysointitapa on oikea valinta. Valinnan perusteluna käytetään saavutettavissa olevaa hyötynäkökulmaa. Tällä lähestymistavalla työ luo lukijalleen kokonaiskuvan aiheesta ja konkreettiset hyödyt on helppo tuoda esille.

Miten algoritmien tehokkuutta analysoidaan amortisoidusti ja asymptoottisesti, miten nämä eroavat toisistaan ja millaisia hyötyjä on mahdollista saavuttaa analysointitavan valinnalla? Tähän tutkimuskysymykseen vastataan työssä.

Asymptoottinen algoritmin analyysi kertoo aina huonoimman mahdollisen tilanteen algoritmin tehokkuuden suhteen. Amortisoitu algoritmin analyysi kuvastaa algoritmin oletettua keskimääräistä tehokkuutta useilla perättäisillä suorituksilla. Analysointitavan valinnalla voidaan varmistua algoritmin toiminnasta kaikissa olosuhteissa, huonoin mahdollinen tilanne, tai algoritmin toiminnasta todellisessa käyttöympäristössä, jossa huonointa tilanne esiintyy harvoin tai ei koskaan.

Tutkielmassa esitellään ensin molemmat analysointitavat. Seuraavaksi niitä vertaillaan keskenään. Tämän jälkeisessä luvussa esitellään hyötyjä, jotka on mahdollista saavuttaa valitsemalla oikea analysointitapa. Toiseksi viimeisessä luvussa on keskustelua tutkielman aiheesta ja saaduista tuloksista. Lopuksi esitellään vielä yhteenveto saaduista tuloksista.

2 Tutkimusmenetelmä

Tehty tutkimus on tyypiltään kirjallisuuskatsaus. Työn ollessa kirjallisuuskatsaus uutta tutkimusta ei ole suoritettu, vaan on tutkittu aiheesta olemassa olevia aiempia tutkimuksia. Lähteiksi valittiin tutkimuksia ja aiheeseen liittyvää peruskirjallisuutta.

Olemassa olevia tutkimuksia on etsitty erilaisilla hakusanayhdistelmillä hakuportaaleista ja artikkelitietokannoista. Käytettyjä hakuportaaleja ja artikkelitietokantoja olivat Tampereen yliopiston Andor-palvelu ja IEEE-tietokanta.

Hakuja tehtäessä käytettiin samoja hakusanoja suomen ja englannin kielellä. Hakutuloksien valinnassa painotettiin tutkimuksia, jotka olivat vertaisarvioituja ja keskittyivät asian käsittelyyn yleisellä tasolla yksittäisten algoritmien tarkastelun sijaan. Käytettyjä hakusanoja olivat esimerkiksi: algorithm, algorithm analysis, amortized analysis ja big oh. Hakusanoja käytettiin myös yhdistelminä, mutta kieltäviä termejä ei käytetty.

Yhdistämällä hakusanoja tai rajaamalla hakutietokantaa saatiin riittävän pienet, mutta tarpeeksi kattavat hakutulokset. Jokainen hakutulos käytiin tämän jälkeen lävitse ja valittiin tuloksen relevanssin perusteella se joko osaksi työtä tai jätettiin huomioimatta. Lähteeksi valikoitui myös aiheeseen liittyvää peruskirjallisuutta, jossa on esitelty kattavasti perusteoriaa aiheeseen liittyen.

3 Algoritmien analysointi

Tässä luvussa esitellään algoritmien analysointi yleisellä tasolla ja sen jälkeen esitellään tarkemmin, mitä algoritmin analysointi tarkoittaa asymptoottisesti tarkasteltuna ja amortisoidusti tarkasteltuna. Tarkastelun näkökulmaksi on valittu erityisesti tehokkuus, koska sen arviointi on yleisin algoritmin analysoinnin tavoite.

Tutkimuksessa Levitin (2012) esitetyn määritelmän mukaan algoritmia analysoidaan kahdella mittarilla, aikatehokkuuden suhteen ja tilatehokkuuden suhteen. Aikatehokkuutta tarkasteltaessa arvioidaan yleensä kauanko algoritmin suorittaminen vie aikaa tai montako matemaattista operaatiota tulee suorittaa. Tilatehokkuutta taas arvioidaan tarkastelemalla, paljonko algoritmi vie muistia.

Rosendahl (1989) arvioi tutkimuksessaan analysoinnin olevan hyödyllistä tehokuuden arvioinnin ja parempien algoritmien kehittämisen suhteen. Rosendahlin molempia arvioita voidaan pitää oikeina, sillä nykyään algoritmit esitellään tavallisimmin asymptoottisen tai amortisoidun tehokkuusarvion kanssa, joka on saatu selville suorittamalla algoritmin analysointia. Vakiintunut tapa esitellä algoritmin tehokkuus, analysoinnin perusteella saadulla tuloksella, osoittaa itsessään analysoinnin tärkeyden tehokkuutta tarkasteltaessa. Algoritmin analysoinnista saatua tehokkuuden arvoa käytetään myös algoritmien keskinäiseen vertailuun, joka todistaa Rosendahlin arvion algoritmien kehittämisestä oikeaksi.

Teknologian kehityksen myötä muistin määrä on kasvanut suhteessa laskentatehoon suureksi. Tämän kehityksen seurauksena ei useinkaan ole mielekästä tai tarpeellista suorittaa tilatehokkuuden analysointia. Tämän seurauksena puhuttaessa algoritmin tehokkuuden analysoinnista viitataan usein vain aikatehokkuuteen, toisin sanoen suoritusaikaan.

Laskennallisen tehokkuuden näkökulmasta ollaan yleensä kiinnostuneita kahdesta asiasta, laskennallisten ongelmien monimutkaisuudesta, sekä algoritmien kyvystä ratkaista niitä (Oliveira & Barbosa, 2016). Analysointia varten algoritmin tulee kyetä ratkaisemaan annettu ongelma tai saavuttaa loppu ilman ratkaisua.

Ongelmien monimutkaisuuden pienentäminen on usein käytännössä vain ongelman pilkkomista pienempiin osiin, hajota ja hallitse -periaatetta käyttäen. Näiden pienempien ongelmien ratkaisuun on näin ollen helpompi löytää parempi algoritmi, mutta tehokkuuden näkökulmasta samat muutokset olisi hyvin usein mahdollista tehdä jo alkuperäiseen algoritmiin ja tehokkuutta todellisuudessa parannetaan muokkaamalla algoritmeja, eikä muokkaamalla ongelmaa.

3.1 Asymptoottinen tehokkuuden analysointi

Asymptoottinen tehokkuus tunnetaan myös termillä O-notaatio tai englanniksi ”big-Oh notation”. Asymptoottisen tehokkuuden voi kuvitella tarkoittavan ”vähemmän kuin tai yhtä kuin” (Mailund, 2021). Toisin ilmaistuna asymptoottinen tehokkuus kertoo siis, miten tehokas algoritmi on huonoimmassakin mahdollisessa tilanteessa.

Algoritmien asymptoottisen tehokkuuden analysointi on oleellinen osa tietojenkäsittelytieteitä ja optimointia, mukaan lukien teoreettiset tutkimukset ja käytännön sovellukset (Oliveira & Barbosa, 2016). Sovelluksien optimoinnin kannalta analysointi on lähes välttämätöntä, koska pelkällä lopputuotteen testaamisella ei saada selville, miten suorituskykyä voidaan oikeasti parantaa.

Asymptoottisen tehokkuuden analysointi kertoo selkeästi huonoimman mahdollisen tilanteen algoritmin suoritukselle. Huonoimman tilanteen ollessa selvillä voidaan määrittää tarkasti, missä suhteellisessa suoritusajassa algoritmi on aina suoritettu. Tämä ominaisuus tekee asymptoottisesta analysoinnista erittäin hyvin soveltuvan analysointitavan aikakriittisille toiminnoille. Hyvänä esimerkkinä aikakriittisestä toiminnosta ovat käyttäjäkokemukseen vaikuttavat toiminnot.

Tarkastelemalla algoritmia asymptoottisesti ei saada selville mitään algoritmin käyttämästä muistin määrästä, sillä käytetyn muistin määrällä ei ole vaikutusta algoritmin hitaimpaan mahdolliseen ajoaikaan. Jos muistia ei ole riittävästi saatavilla algoritmin tarpeisiin, hidastuu algoritmin suhteellinen suoritusaika, mutta tässäkin tilanteessa hidastuminen tapahtuu huonoimmillaan asymptoottisen analyysin asettaman raja-arvon alapuo-

lella. Asymptoottinen analysointi ei myöskään kuvasta algoritmin todellista suoritusaikaa, sillä se asettaa raja-arvon huonoimmalle mahdolliselle tilanteelle ja tällainen tilanne voi olla äärettömän harvinainen.

Cormen (2009) todistaa kirjassaan, kuinka asymptoottisen tehokkuuden arvioinnissa on helppoa asettaa liian löysä yläraja, jolloin asymptoottisen tehokkuuden arvion hyödyt menetetään. Ylärajan ollessa huonosti määritelty saatetaan algoritmi todeta käyttökelvottomaksi liiallisen hitauden takia, vaikka todellisuudessa algoritmi olisikin voinut olla riittävän nopea kaikissa tilanteissa.

3.2 Amortisoitu tehokkuuden analysointi

Amortisoitu tehokkuuden analyysi kertoo algoritmin yksittäisen ajon tehokkuuden sijaan keskimääräisen tehokkuuden, kun algoritmia ajetaan n kertaa (Levitin, 2012). Amortisoitu tehokkuus on tärkeä keino analysoida algoritmin tehokkuutta, sillä algoritmi voi helposti olla hidas yksittäisellä suorituskerralla, mutta useammalla suorituksella saadaan keskimääräiseksi suoritusajaksi todella hyvä arvo. Amortisoitu tehokkuuden analyysi suodattaa myös harvinaiset poikkeamat pois arviostaan tehokkaasti, koska niillä on usein vain pieni vaikutus keskimääräiseen tehokkuuteen.

Algoritmin rakenne paljastaa usein suoraan algoritmin aikatehokkuuden raja-arvon mutta ei todellista kuvaa algoritmin tehokkuudesta (Laaksonen, 2018). Tämä on usein seurausta käytetyistä tietorakenteista ja niiden taustalla vaikuttavista operaatioista.

Amortisoitu tehokkuuden analyysi esittää aina todellisen keskimääräisen tehokkuuden ja sen määrittelyssä tulee siten olla tarkka, koska amortisoidun tehokkuuden arvon tulisi olla aina sama riippumatta siitä kuka sen määrittää. Amortisoitu tehokkuuden analyysi ei ota kantaan yksittäiseen suoritukseen vaan aina useamman suorituksen muodostamaan kokonaisuuteen (Ackerman et al., 2010).

4 Analysointitapojen vertailu

Tässä luvussa esitellään asymptoottisen tehokkuuden analysointitavan ja amortisoidun tehokkuuden analysointitavan eroavaisuuksia ja samankaltaisuuksia. Työssä painotetaan käytännön vaikutuksia teoreettisten jäädessä vähemmälle huomiolle.

Kerrotaan analysointitapojen eroavaisuuksista ja niiden vaikutuksista käytännössä. Käytännön vaikutuksissa keskitytään analysoinnin suorittamiseen, toteutusmahdollisuuksiin ja analysointitavan valintaan.

4.1 Analysoinnin yhtäläisyydet

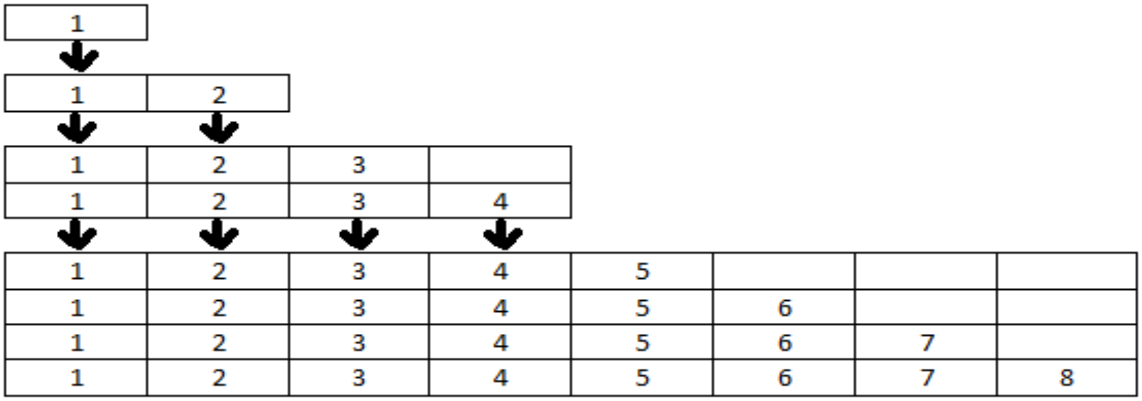
Analysointitapoja vertaillessa voidaan huomata, että molemmat nojautuvat vahvasti matematiikkaan ja niiden määrittelyyn käytetään yleisesti monikertaa *n*. Tulokset ovat molemmissa suhteellisia tehokkuuksia ajan suhteen, eivätkä ne kerro itsessään tarkkaa suoritusaikaa vaan se on laitteistoriippuvainen. Tarkka suoritusaika on mahdollista laskea

sijoittamalla laitteiston mahdollistama suoritusaika *n*:n tilalle analysoinnin tuloksena saatuun laskentakaavaan.

Yhteistä molemmille analysointitavoille on mahdollisuus vääriin tuloksiin. Asymptoottisen analyysin tulokseksi saatu raja-arvo voi olla määritelty liian suureksi. Amortisoidun analyysin tulos taas voi olla väärin määritelty erilaisten laskuvirheiden takia, sillä analyysin matemaattinen määrittely voi olla hyvin haasteellista.

4.2 Analysoinnin eroavaisuudet

Asymptoottisessa tehokkuutta analysoitaessa muistin määrää ei tarvitse huomioida, koska raja-arvo ei muutu, vaikka suhteellinen suoritusaika muuttuisi. Amortisoidun tehokkuuden analysoinnissa muistin määrä on vaikuttava tekijä ja usein muistin käyttöä lisätään, jotta saadaan parempi keskimääräinen tehokkuus useammalla suorituskerralla.



Kuva 1. C++ Vektorin varaama muisti ja alkioiden kopiointi.

Kuvassa 1 nähdään, miten C++-ohjelmointikielen vektori varaa muistia ja milloin alkioita joudutaan kopioimaan. Huomataan aina vektorin täyttyessä sen varaavan muistia kaksi kertaa niin paljon kuin sillä oli varattuna aiemmin. Alkiot tulee myös aina kopioida uuteen paikkaan, kun varattu muisti ei riitä. Nähdään ettei tehdyn työn määrä ole riippuvainen käytettävissä olevasta muistista, kun peräkkäisiä suorituksia on useita.

Asymptoottista tehokkuutta analysoidessa tarkastellaan huonointa mahdollista tilannetta, mikä on mahdollista jo kahden alkion kohdalla, kun toinen alkio lisätään. Huonoimmassa tilanteessa joudutaan kaikki aiemmat alkiot kopioimaan myös, jolloin työtä tehdään *n* verran, jossa *n* on alkioiden määrä.

Amortisoitua tehokkuutta analysoidessa otetaan useampi peräkkäinen suoritus huomioon ja kuvassa 1 havaitaan, että 8 alkion lisäyksen kohdalla kopiointia on tehty 7 kertaa eli *n-1* kertaa. Amortisoidun tehokkuuden analyysissä tehokkuus lasketaan kaavalla $\frac{T_n}{n}$, jossa T_n, on suoritettujen operaatioiden kokonaiskustannus ja *n* operaatioiden määrä. Sijoittamalla *n-1* kyseiseen kaavaan saamme $\frac{n-1}{n} \approx 1$.

Kuvan 1 perusteella voidaan siis osoittaa asymptoottisen tehokkuuden analyysin ja amortisoidun tehokkuuden analyysin antavan eri tuloksen. Huomataan myös, ettei muistinkäytöllä ole vaikutusta asymptoottisen tehokkuuden analyysiin, mutta varaamalla ylimääräistä muistia saadaan amortisoitu tehokkuus huomattavasti paremmaksi, verrattuna tilanteeseen, jossa muistia käytetään aina mahdollisimman vähän.

Asymptoottista tehokkuutta määrittäessä voidaan aina valita löysempi raja-arvo, kuin mikä todellisuudessa olisi absoluuttinen hitain mahdollinen tapaus. Asymptoottinen tehokkuus on siten aina määrittelijästä riippuvainen, sillä O(n) tehokkuuden omaavalle funktiolle myös O(n^2) on oikein määritetty asymptoottinen tehokkuus, vaikka se ei ole lähelläkään tiukinta mahdollista raja-arvoa. Asymptoottisen tehokkuuden määrittelyssä sorrutaan usein liian löysän raja-arvon valintaan (Cormen, 2009).

Amortisoitu tehokkuus on oikein määriteltynä aina sama, koska se kertoo keskimääräisen suoritusajan useammalla peräkkäisellä suorituksella, joka on aina sama riippumatta siitä, kuka toimii määrittelijänä. Amortisoidun tehokkuuden analyysissä voi olla haastavaa saada oikein määritelty arvo jo verrattain yksinkertaisille algoritmeille. Lisäämällä muuttujia osaksi algoritmia analysoinnin aikana on mahdollista saavuttaa tarkemmat arvot (Gulwani et al., 2009).

Asymptoottisen tehokkuuden analyysin antamaa arvoa ei useinkaan voida sellaisenaan hyödyntää käytännössä, kun taas oikein suoritetun amortisoidun tehokkuuden analyysin tulosta on mahdollista hyödyntää sellaisenaan. Käyttäjäkokemuksen arvioinnissa keskimääräinen suoritusaika on oleellisessa roolissa ja siten amortisoidun tehokuuden analyysin tuloksesta voidaan suoraan päätellä, onko valittu algoritmi sopiva.

Algoritmin tehokkuutta analysoitaessa testaamalla täytyy virheisiin varautua. Yucesan et al. (2020) havaitsivat tutkimuksessaan, että lopettamalla ensimmäiseen virheeseen saavutetaan hieman nopeampi tulos, mutta se ei ole peruste olla ajamatta testejä myös loppuun asti, sillä saadut tulokset ovat tilastollisesti hyödyllisiä.

4.3 Eroavaisuuksien vaikutus

Hoffmann J., Aehling K., Hoffman M. (2012), esittävät tietokoneohjelman tärkeimmäksi ominaisuudeksi tehokkuuden, joka voidaan määrittää käytetyn ajan, muistin ja tehokkuuden avulla. Heidän näkemyksensä mukaan ideaalitilanteessa kokenut ohjelmoija kykenee kertomaan suoraan hyvästä ohjelmakoodista asymptoottisen tehokkuuden. Amortisoitua tehokkuuta ei ole mahdollista kuitenkaan useinkaan päätellä pelkän koodin perusteella.

Koska amortisoitua tehokkuutta ei ole useinkaan samalla tavalla mahdollista päätellä koodista kuin asymptoottista tehokkuutta, tulisi luoda tehokkaampia tapoja analysoida algoritmien amortisoitu tehokkuus, sekä miettiä, mitkä asiat mahdollisesti rajoittavat amortisoidun tehokkuuden määrittämistä. Jotta amortisoitu tehokkuus olisi helpompi määritellä, voidaan harkita algoritmin jakamista pienempiin osiin.

Algoritmien analysointitapojen eroavaisuuden takia on myös tärkeää tunnistaa käytettävä ympäristö ja laitteiston mahdollistamat vaihtoehdot. Kuten kuvassa 1 havaittiin, voidaan runsaammalla muistinkäytöllä saavuttaa merkittäviä hyötyjä amortisoituun tehokkuuteen. Tilanteessa, jossa muistia vartaan aina yksi paikka lisää, kun alkio lisätään, tehdään työtä keskimäärin n verran. Varaamalla muistia enemmän saatiin tehokkuudeksi keskimäärin 1 eli vakioaikainen operaatio.

Asymptoottisen tehokkuuden analysoinnin kertoessa vain huonoimman mahdollisen tilanteen raja-arvon on tärkeää tunnistaa, miten yleinen tapahtuma tuo huonoin tilanne on. Huonoimman tilanteen ollessa harvinainen, eivät sen perusteella tehdyt muutokset algoritmiin välttämättä tuo haluttuja muutoksia. Vastaavasti tilanteessa, jossa huonoin mahdollinen tapahtuma ilmenee edes kohtalaisella todennäköisyydellä ei amortisoitu tehokkuuden analyysi tuo useinkaan tarpeeksi kattavaa tulosta.

5 Saavutettavissa olevat hyödyt

Tässä luvussa esitellään, millaisia teoreettisia ja konkreettisia hyötyjä on mahdollista saavuttaa analysoimalla algoritmeja. Luvussa esitellään myös hyötynäkökulman vaikutusta analysointitavan valintaan.

5.1 Teoreettiset hyödyt

Amortisoinnin idea suosittelee suunnittelemaan tietorakenteita siten, että käytettäessä tietorakennetta sitä muokataan yksinkertaisella tavalla, jotta tulevat käyttötapaukset olisivat suoritusajaltaan nopeampia (Tarjan, 1985). Näin ei usein kuitenkaan todellisuudessa tehdä, sillä tietorakenteen muuttaminen jokaisella algoritmin ajokerralla hidastaa muuta ohjelmaa usein enemmän kuin, mitä se nopeutuu yksittäisen algoritmin amortisoidun tehokkuuden parantuessa.

Teoreettisella tasolla, tietorakenteen muuttamisesta jokaisella suorituskerralla tehokkaampaan muotoon seuraavaa suoritusta varten on saavutettavissa oleva hyöty suuri. Esimerkiksi tilanteessa, jossa algoritmin ajosta ja tietorakenteen muuttamisesta vastaavat eri tahot, voi näiden suorittaminen rinnakkain olla mahdollista. Rinnakkainen suoritettu tietorakenteen muuttaminen ei silloin välttämättä hidasta ohjelmaa yhtään, mutta algoritmin amortisoidun tehokkuuden paraneminen nopeuttaa silti ohjelman toimintaa. Konkreettisesti tämä on vaikea toteuttaa, koska normaalissa käyttöympäristössä pyörii usein samaan aikaan useita ohjelmia ja resursseja ei ole käytettävissä rajattomasti rinnakkaiseen suorittamiseen.

Valitsemalla asymptoottisen tehokkuuden analysoinnin tuloksen perusteella käytettävä algoritmi on mahdollista valita suorituskyvyltään riittävän tehokas algoritmi kaikkiin mahdollisiin tilanteisiin. Todellisuudessa asymptoottisen tehokkuuden perusteella valitun

algoritmin hyödyt jäävät usein saavuttamatta, sillä teoreettinen hitain tapaus voi olla äärimmäisen harvinainen.

Asymptoottisen tehokkuuden analysoinnin perusteella valitun algoritmin käyttäminen mahdollistaa käytetyn muistin minimoimisen monessa tilanteessa. Todellisuudessa näin ei kannata menetellä, koska suhteellinen suoritusaika voi kärsiä liian vähäisestä muistista ja muistia on nykyisellä teknologialla helppo lisätä.

5.2 Konkreettiset hyödyt

Valitsemalla käytettävät algoritmit amortisoidun tehokkuuden analyysin perusteella on mahdollista valita nopeita algoritmeja, jotka ovat suoritusajaltaan riittävän tehokkaita lähes kaikissa tilanteissa. Valitsemalla asymptoottisen tehokkuuden analyysin perusteella sopiva algoritmi voidaan olla varmoja, että se on aina kaikissa tilanteissa suoritettu tietyssä ajassa ja siten saavutetaan haluttu lopputulos riittävän nopeasti.

Analysoimalla algoritmeja molemmilla tavoilla on mahdollista saavuttaa eri tilanteisiin optimoitu lopputulos, joka vastaa haluttua. Analysoimalla omia algoritmeja ohjelmoija kehittyy myös paremmaksi ohjelmoijaksi, sillä hän oppii ottamaan huomioon algoritmin tehokkuuden jo suunnittelu- ja ohjelmointivaiheessa.

Analysoimalla algoritmeja saadaan selville tehokkuus suhteellisen suoritusajan monikertana. Tätä tietoa voidaan hyödyntää laitteiston optimoinnissa ja valinnassa. Osataan valita riittävän tehokas laitteisto, jos suoritusaika on kriittinen osa prosessia. Vastaavasti on mahdollista laskea kustannus-suoritusaikasuhteita, joka voi olla esimerkiksi asiakkaille suunnitelluissa laite-ohjelmisto-paketeissa tärkeä tieto, jotta saavutetaan suurimmat mahdolliset voitot.

Paremmin ja vakaammin toimivat algoritmit johtavat vakaampiin ja paremmin suoriutuviin ohjelmistoihin. Kun ohjelmistot ovat luotettavampia, johtaa tämä suoraan kaupallisessa ympäristössä suurempiin tuottoihin sillä ohjelmistot ovat suuremman osan ajasta käytettävissä.

Asymptoottinen algoritmin analysointi voi myös paljastaa tilanteen, jossa algoritmin suorittaminen kaataa ohjelman. Jos algoritmi hidastuu liikaa, on huonoimmassa tilanteessa mahdollinen suoritusaika niin suuri, että käyttöjärjestelmä lopettaa ohjelmakoodin ajamisen, vaikka koodissa ei mitään vikaa muuten olisikaan.

5.3 Hyötyjen vaikutus analysointitavan valintaan

Hyötynäkökulmaa tarkasteltaessa on tärkeää määrittää heti alussa, että mitä halutaan. Valittavia hyötynäkökulmia ovat esimerkiksi suoritusaika, suoritusvarmuus tai kustannukset.

Mikäli halutaan kaikissa tilanteissa vähintään tietyn tehokkuuden omaava algoritmi, tulee analysointi tehdä asymptoottisesti. Tämä hyötynäkökulma on usein tieteellisissä prosesseissa ja muissa suurta laskentatehoa vaativissa prosesseissa tärkeä.

Tilanteessa, jossa on tärkeää, miten algoritmin käyttäytyy keskimäärin ja mahdollisella huonoimmalla mahdollisella tapauksella ei ole juurikaan merkitystä, esimerkiksi sen harvinaisuuden takia, tulisi aina analysointi suorittaa amortisoidusti. Tämä hyötynäkökulma on tärkeä esimerkiksi miettiessä käyttäjäkokemusta tai usein suoritettavan aikakriittisen toiminnon tehokkuutta.

Kustannus näkökulma valitaan usein, kun suoritusaika on tärkeä osatekijä, mutta ei tärkein tekijä. Esimerkiksi tutkimusta tehtäessä voidaan budjettisyistä hyväksyä hieman pitempi aika, jolloin laitteisto kustannukset saadaan pienemmäksi. Kustannusnäkökulman valinta kannattaa usein vain tilanteissa, joissa algoritmi käyttäytyy ennakkoon määritellyllä tavalla ja sen seurauksena amortisoitu tehokkuuden analysointi on oikea valinta.

6 Keskustelu

Analysoimalla algoritmien tehokkuuksia saavutetaan lähes aina konkreettisia hyötyjä. Koska hyötyjä pystytään saamaan helposti ja niiden mittaaminenkin on toteutettavissa, tulisi tutkia algoritmianalysoinnin ottamista kiinteäksi osaksi ohjelmointiprosessia.

Tulevaisuudessa olisi hyvä myös tutkia tapoja kehittää analysointitavan valintaa, jotta se saataisiin selkeämmäksi ja onnistuttaisiin paremmin käyttämään aina oikeaa analysointitapaa, jolloin resursseja ei käytetä epäolennaisiin asioihin. Amortisoidun tehokkuuden analysointiin olisi oleellista kehittää tarkempia, tehokkaampia ja helpompia malleja, sillä nykyiset mallit ovat usein epätarkkoja ja vaativat todella edistynyttä matematiikkaa. Amortisoidun tehokkuuden analysoinnin suorittaminen oikein on vaativa prosessi ja on aiheellista pohtia, onko se jopa liian vaativaa jokaisen ohjelmoijan suoritettavaksi.

Algoritmien analysoinnilla on mahdollista saavuttaa myös pitkäkestoisia hyötyjä, kun ohjelmoija kehittyy analysoinnissa ja alkaa suorittamaan sitä alitajuntaisesti jo ohjelmoinnin aikana. Tulevaisuudessa olisi hyvä tutkia saavutettavissa olevia kustannushyötyjä, jos ohjelmoija suorittaa algoritmien analysointia. Saavutetaanko hyötyjä suoraan siitä, että algoritmien muokkaamiseen ja optimointiin menee ensimmäisen version jälkeen vähemmän aikaa?

Eräs asia, jota tulevaisuudessa olisi syytä tutkia on se, miten nopeasti ohjelmoijan algoritmien laatu alkaa paranemaan ja millä tavalla ohjelmoijan ymmärrys algoritmeista on kehittynyt analysoinnin seurauksena. Aronshtam et al. (2021) huomasivat tutkimuksessaan, että algoritmien analysointi vaatii huomattavasti enemmän kognitiivisia kykyjä kuin normaali ohjelmointitehtävän suorittaminen.

Analysoinnista aiheutuvia haittoja on myös tärkeää tutkia tulevaisuudessa. Liiallinen algoritmien analysointi voi hidastaa kehitysprosessia ja on myös kartoitettava, onko tilanteita, jolloin analysoinnilla ei ole mahdollisuutta saavuttaa todellisia hyötyjä. Mikäli hyötyjä ei voida saavuttaa, ovatko analysointiin käytetyt resurssit menetettyjä resursseja?

Onko ohjelmoijalle analysoinnin seurauksena karttuva kyvyt aina niin hyödyllisiä, että resursseja kannattaa käyttää analysoinnin suorittamiseen?

7 Yhteenveto

Algoritmien tehokkuuden analysointia voidaan tehdä asymptoottisesti tai amortisoidusti. Analysointitavat eroavat toisistaan oleellisesti ja oikean tavan valinta on äärimmäisen tärkeää.

Analysoidessa asymptoottista tehokkuutta tarkastellaan aina vain huonointa mahdollista tilannetta. Asymptoottisesta tehokkuuden analysoinnista saatu tulos on raja-arvo, jota algoritmi ei suorittamisen aikana koskaan ylitä, se voidaan helposti määritellä liian epätarkasti. Epätarkkakin raja-arvo on asymptoottisessa mielessä oikea, mutta hyödyt menetetään. Asymptoottisen tehokkuuden analysoinnin suurin ongelma on sen epäoleellisuus, jos huonoin tilanne tapahtuu todella harvoin.

Amortisoidun tehokkuuden analysoinnissa tarkastellaan keskimääräistä tehokkuutta, kun sama toiminto tehdään useita kertoja. Analysoinnin tuloksena saatu arvo on aina sama riippumatta siitä, kuka sen määrittelee. Amortisoitu tehokkuuden analysointi soveltuu huonosti tilanteisiin, joissa huonoin mahdollinen tilanne tapahtuu usein.

Suorittamalla analysointia on mahdollista saavuttaa selkeitä hyötyjä. Teoreettisetkin hyödyt voivat tulevaisuudessa olla konkreettisia hyötyjä. Konkreettisia hyötyjä on jo nyt mahdollista mitata ja ne voivat olla hyvinkin monimuotoisia.

Tekemällä algoritmien analysointia on ohjelmoijalla mahdollista kehittää omaa osaamistaan ja tulla paremmaksi ohjelmoijaksi. Syvempi ymmärrys algoritmien tehokkuudesta auttaa luomaan paremmin toimiva ohjelmistoja ja tuottamaan paremman käyttäjäkokemuksen.

Algoritmien analysoinnin perusteella on mahdollista saavuttaa kustannussäästöjä, kun algoritmin optimoinnin jälkeen voidaan myös laitteisto optimoida sopivaksi. Analysoinnin seurauksena saadaan myös vakaampia ja paremmin suoriutuvia ohjelmistoja, josta seuraa kaupallisessa ympäristössä usein suoraa rahallista hyötyä, kun toiminnot ovat koko ajan käytössä. Analyysillä saavutettavien hyötyjen takia olisi tärkeää tutkia aihetta lisää, jotta tulevaisuudessa voidaan saavuttaa vielä suurempia hyötyjä.

Lähdeluettelo

Ackermann, H., Röglin, H., Schellbach, U., Schweer, N. (2010). *Chapter 4. Analysis of Algorithms. In: Müller-Hannemann, M., Schirra, S. (eds) Algorithm Engineering. Lecture Notes in Computer Science, vol 5971.* Springer, Berlin, Heidelberg.

Aronshtam, L. et al. (2021) Can we do better? a classification of algorithm run-time-complexity improvement using the SOLO taxonomy. *Education and information technologies.* [Online] 26 (5), 5851–5872.

Cormen, T. H. (2009) *Introduction to algorithms. 3rd* ed. Cambridge, MA: MIT Press.

Gulwani, S. et al. (2009) 'SPEED: precise and efficient static estimation of program computational complexity', *in Conference Record of the Annual ACM Symposium on Principles of Programming Languages.* 2009 ACM. pp. 127–139.

Hoffmann, J. et al. (2012) Multivariate amortized resource analysis. *ACM transactions on programming languages and systems.* 34 (3), 1–62.

Laaksonen, A. (2018) 'Algorithm Design Topics', in Guide to Competitive Programming. *Cham: Springer International Publishing*. pp. 107–117.

Levitin, A. (2012) *Introduction to the design & analysis of algorithms. 3rd ed.* Boston: Pearson/Addison-Wesley.

Mailund, T. (2021) 'Algorithmic Efficiency', *in Introduction to Computational Thinking.* Berkeley, CA: Apress. pp. 65-96.

Oliveira, F. de S. & Barbosa, V.C. (2016) A note on computing independent terms in asymptotic expressions of computational complexity. *Optimization letters.* 11 (8), 1757-1765.

Rosendahl, M. (1989) Automatic Complexity Analysis. *In FPCA.* ACM Press. pp. 144-156.

Tarjan, R.E. (1985) Amortized computational complexity. *SLAM journal on algebraic and discrete methods.* 6 (2), 306-318.

Yucesan, O. & Ozkil, A. (2020) Time Complexity Comparison of Stopping at First Failure and Completely Running the Test. *Journal of electronic testing.* 36 (3), 409-417.