Tietorakenteiden ja algoritmien harjoitustyön testausdokumentti

Risto Tuomainen

31. elokuuta 2014

Suorituskyky testauksessa testattiin Strassen algoritmia, determinantin laskemista, matriisin kääntöä ja harvan matriisin kertomista tiiviillä. Determinantin laskeminen testaa oikeastaan LUP-hajotelmassa käytettävää algoritmia, koska determinantin saa hajotelmasta triviaalisti. Naiivia matriisikertolaskua käytettiin kertolaskujen vertailukohtana.

Testauksen toteuttamisessa ei varsinaisesti kannettu huolta clean code -puolesta. Testit voi toistaa poistamalla pääohjelmametodin koodista kommentteja halutuilta kohdilta. Ohjelma tuottaa .csv-tiedostoja, joista voi piirtää kuvioita esim. R:llä kuten tässä on tehty.

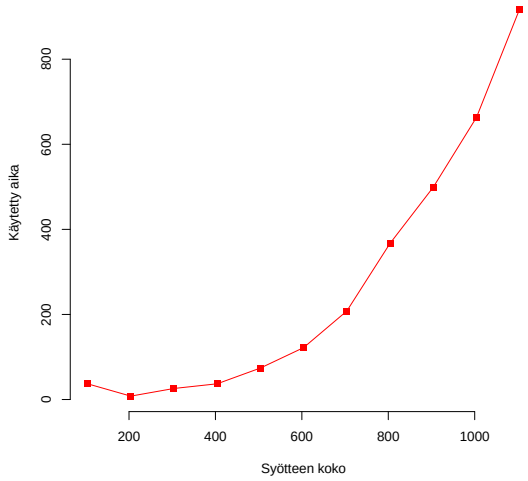
Determinantin laskeminen

Determinanttia testattiin eri kokoisilla neliömatriiseilla, jotka oli täytetty satunnaisilla kokonaisluvuilla väliltä $(0, 100)$. Determinantin laskeminen toteutettiin $\mathbf{PA} = \mathbf{LU}$-dekomponoinnin avulla. Ratkaisussa seurattiin melko tarkasti oppikirjan pseudokoodia [1, s. 824], jossa algoritmi oli toteutettu melko tehokkaasti. Toteutus näyttääkin saavuttavan suunnilleen $O(n^3)$-aikavaativuuden, noin ainakin kuvan perusteella arvioituna.

Tiiviiden matriisien kertolasku

Harjoitustyössä toteutin Strassen algoritmin neliömatriisien kertomista varten. Suorituskykytestissä kokeilin algoritmin käyttämää aikaa eri kokoisilla neliömatriiseilla. Lisäksi vertailun vuoksi sama kertolasku tehtiin myös naiivilla algoritmilla. Näistä kahdesta Strassen algoritmin pitäisi teoriassa olla nopeampi, mutta kuten kuvasti voi havaita, testissä naiivialgoritmi voitti Strassen huomattavalla erolla. Eräs syy tähän voi olla, että toteutin Strassen algoritmin siten, että sen aikana kopioidaan kerrottavien matriisien osamatriiseja (sen sijaan, että olisin laskenut indeksejä ja tehnyt laskutoimituksia syötematriisiin). Tämä lisää tietysti suoritusaikaa hieman. Asymptoottisen aikavaativuuden kannalta ratkaisu on kuitenkin neutraali [1, s. 77] Lisäksi toteustapa lisää huomattavasti algoritmin muistivaatimusta, ja testin perusteella vaikuttaakin siltä, että juuri muistivaatimus dominoi aikavaatimusta. Hyppäykset suoritusajassa nimittäin voisivat selittyä luontevasti sillä, että kaikki suorituksessa vaadittu tieto ei ole mahtunut muistiin, ja suorituksen aikana on jouduttu hakemaan tietoa kiintolevyltä.

Kuva 1: Determinantin laskemisen aikavaativuus



Yhteenvetona Strassen algoritmia siten kuin se tässä on toteutettu voi pitää katastrofaalisen huonona.

Harvan ja tiiviin matriisin kertolasku

Ainut epätriviaalialgoritmi, jonka toteutin harjoitustyössä ja joka käyttää Yale Matrix -esitystä hyödykseen, on harvan ja tiiviin algoritmin kertolasku. Algoritmia testattiin kertomalla kahta neliömatriisia keskenään. Algoritmi toimisi samoin myös yleisessä tapauksessa. Testissä harvassa matriisissa oli 5 % alkioista muita kuin nollia ja tiiviissä matriisissa kaikki arvot olivat muita kuin nollia. Algoritmi osoittautui varsin menestyksekkääksi, ja kuten kuvista voi nähdä aikavaatimus on selvästi eri suuruusluokkaa kuin naiivilla kertolaskulla. Testissä ei edes kokeiltu erityisen suuria matriiseja, mutta jo näissä testeissä ero oli jo käytännönkin kannalta ratkaiseva.

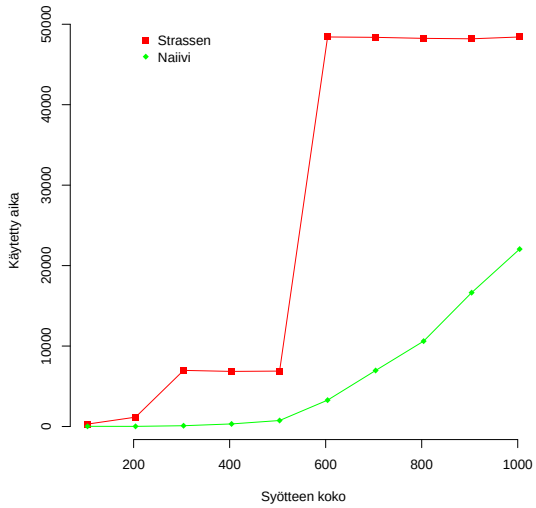
Matriisin kääntö

Matriisin kääntö tehtiin naiivilla Gauss-Jordan-eliminoinnilla, jonka aikavaativuus on luokkaa $O(n^3)$. Kääntämistä testattiin eri kokoisilla, verrattain pienillä neliömatriiseilla. Aika vaativuuden voikin kuviosta nähdä olevan sen tyyppistä kuin teoreettinen vaativuus antaa ymmärtää.

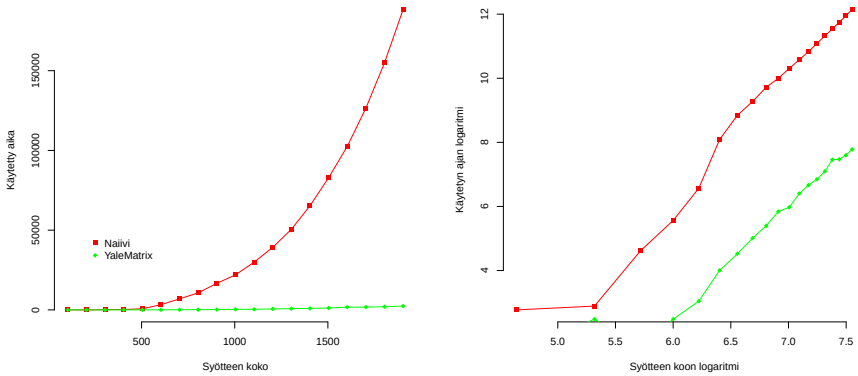
Viitteet

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, 2009.

Kuva 2: Tiiviiden matriisien kertolasku



Kuva 3: Harvan matriisin kertominen tiiviillä



Kuva 4: Matriisin kääntö