Samu Tamminen 013976176

Testausdokumentti

Yksikkötestit

Testasin ohjelmaa manuaalisesti ja yksikkötesteillä koodia kirjoitettaessa. Clujure:lle suunnattu riippuvuuksien hallinta-työkalu Leiningen ajaa testit komennolla ‘lein test’. Yksikkötestit kattavat lopullisessa ohjelmakoodissa riveistä 99.61 prosenttia ja muodoista(forms) 92.30 prosenttia.

Lines	Non-Blank	Instrumented	Covered	Partial	
90	79	43	39	4	matrix_calculator/helpers.clj
48	42	23	22	1	matrix_calculator/basic_features.clj
160	143	93	92	1	matrix_calculator/multiply.clj
146	127	69	55	13	matrix_calculator/determinant.clj
57	50	29	27	2	matrix_calculator/paaluokka.clj

(lähde: Cloverage-koodikattavuus työkalun generoima https://github.com/lshift/cloverage)

Yksikkötestit kattavat huonoiten determinantti-tiedoston determinant-funktiota, sillä sen toimintaa ei hitauden takia pystynyt testaamaan suuremmilla syötteillä.
Yksikkötestit auttoivat hahmottamaan oman koodin toimintaa selkeämmin ja auttoivat varmistamaan koodin toimivan oletetusti. Sovelluslogiikan tasolla olevissa erehdyksissä yksikkötestaus ei auta.

Suorituskykytestaus

Vertailen ohjelmani funktioiden matrix-multipcation ja strassen sekä determinant ja LU-determinant suoritusaikoja. Keksin testimatriisit päästäni(löytyvät dokumentin lopusta) ja ajan suoritukset komentorivillä. Suoritusajan mittaamisen käytän Linux-komentorivikäskyä time ja merkitsen ylös viiden ajokerran keskiarvon.

Matriisikertolasku

Funktio	matrix_multipcation	strassen
Aikavaativuus	O(n^3)	O(n^2.71)
Aika matriiseilla 1 ja 2 (ms)	790.2	785.8
Aika matriiseilla 3 ja 4	775.6	784.6

Aika matriiseilla 6 ja 6	812.4	840.6

Funktioiden aikavaativuudet ovat melko samat, joten niiden ero tehokkuudessa ei tule ilmi näin pienillä syötteillä. Aikavaativuuden puolesta strassenin pitäisi olla nopeampi. Algoritmin parempi teho saadaan vähentämällä rekursiokutsujen tarvetta vakioaikaisella laskennalla. Syötteen ollessa pieni ei rekursiokutsujakaan vähennetä paljoa. Parempia mittauksia saataisiin aikaiseksi generoimalla suuria, satojen levyisiä, neliömatriiseja testejä varten. Rajoitun aikani puitteissa näihin testituloksiin.

Determinantti

Funktio	determinant	LU-determinant
Aikavaativuus	O(n!)	O(n^3)
Aika matriisilla 1	763.4	757.0
Aika matriisilla 5	893.6	766.0
Aika matriisilla 7	13 422	790

Determinanttia laskettaessa kahden eri aikavaativuuden algoritmien välinen ero näkyy selvemmin. Kolmen levyisellä neliömatriisilla ajat ovat melko samat, kuuden levyisillä eroa syntyy sekunnin kymmenesosa, mutta kymmenen levyisellä neliömatriisilla raakaan voimaan perustuva funktio toimii jo kohtuuttoman hitaasti. Determinantti-funktio kutsuu jokaisella iteraatiollaan itseään n-1 kertaa ja on aikavaativuudeltaan hitain mahdollinen. LU-determinant-funktio taas esilaskee n^3 ajassa hajotelmamatriisin, josta determinantti voidaan lukea O(n) ajassa.

Suorituskykyanalyysini ei tuonut suuria yllätyksiä. Aikavaativuuksien eron ollessa pienempi, ei pienillä syötteillä saada suuria eroja. Jos taas aikavaativuus eroaa huomattavasti, ovat ajalliset testitulokset sen mukaisia.

Matriisi 1: [[1,4,7][9,6,4][4,5,6]]
Matriisi 2: [[1,2,8][5,4,3][8,7,1]]
Matriisi 3: [[3,2,1,5,6,7][32,1,-5,6,-5,7][-1,-2,3,4,9,0][20,10,-43,2,0,0]]
Matriisi 4: [[-11,32,6,0][0,0,-1,-4][2,3,9,8][-1,4,98,3][4,2,1,-1][34,-6,0,-3]]
Matriisi 5:
[[-11,32,6,0,4,10][0,12,-4,0,-1,-4][2,3,20,32,9,8][-1,66,10,4,98,3][4,2,10,2,1,-1][34,-6,0,-3,8,-1]]
Matriisi 6:
[[4,3,2,5,6,3,2,-2,0,2001][-4,5,6,344,78,5,8,4,2,-7][98,-121,6,54,2,1,3,45,6,-10][1,1,1,503,8,7,6

,4,-3,50][0,54,32,22,1,-56,43,90,6,90][6,54,2,-4,4,32,22,19,72,4][3,2,-27,21,5,31,98,69,-4,0][1
1,33,1,2,3,45,87,94,28,0][756,-4,-777,0,0,4,0,-3,6,77][6,3,1,55,0,0,0,11,-889,81]]
Matriisi 7:
[[4,3,2,5,6,3,2,-2,0,1][-4,5,6,2,78,5,8,4,2,-7][1,2,6,1,2,1,3,2,6,-10][1,1,1,2,8,7,6,4,-3,3][0,1,3,1,
1,3,1,2,6,4][6,2,2,-4,4,2,22,19,1,4][3,2,1,21,5,31,1,1,-4,0][2,1,1,2,3,3,2,4,3,0][2,-4,1,0,0,4,0,-3,
6,3][6,3,1,4,0,0,0,11,2,2]]