Testausdokumentti, Eclipse calculator

Yksikkötestit

Yksikkötestein on testattu jokaisen funktion toiminta, keskittyen raja-tapauksiin. Testauksessa on keskitytty pieniä yksittäisiä asioita tekevien funktioiden testaukseen ja näitä yhteen nivoavia funktioita on testattu lähinnä niin että nähdään että ne tuottavat oikean yhteistuloksen, koska yksittäisten funktioiden testien tulisi taata myös niiden oikea toiminta.

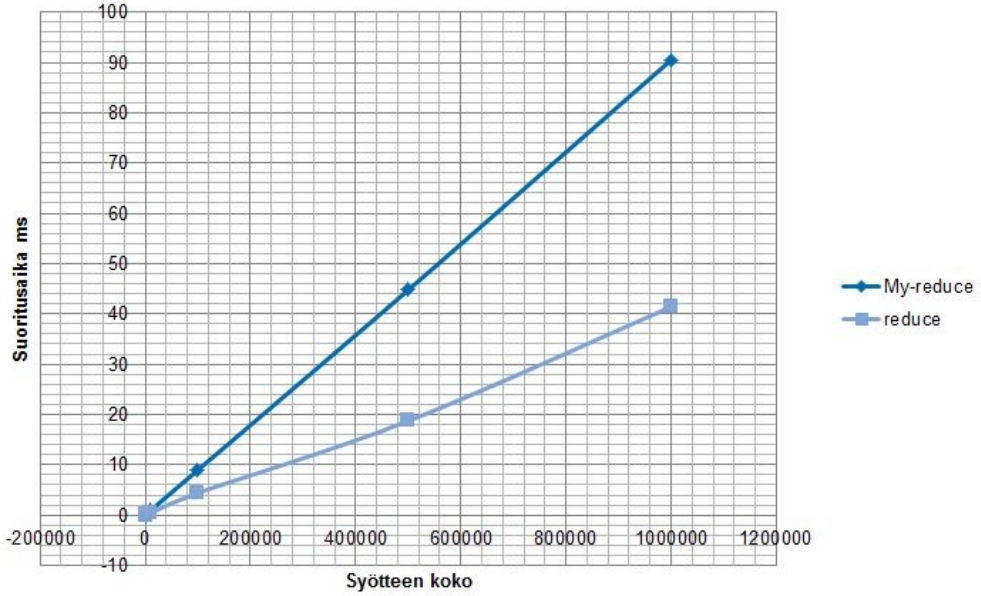
Testaus on onnitunut hyvin, omien funktioiden implementointi ja koodin refaktorointi on tapahtunut melko huoletta, sillä testit on hyvin onnistuneet varmistamaan ohjelman toiminnan säilymisen ennallaan.

Suorituskykytestit

my-reduce

my-reducea verrattiin Clojuren reduce funktioon. Syötekoot suorituskykytestissä on miljoonaan alkioon asti ja kaikille alkioille tehdään summausoperaatio.

Clojuren reduce toimii hyvin eri tavoin kun oma toteutukseni ja Clojuressa onkin syötteen tyypin mukaan useita eri funktioita joilla toiminto toteutetaan, niin että lopputulos olisi mahdollisimman nopea juuri syötteen tyyppisille muuttujille. Oma reduceni on ehkäpä tästä syystä alkuperäistä hitaampi. Se toimii käymällä kaikki luvut häntäoptimoidulla rekursiolla läpi ja sijoittamalla ne yksitellen parametrina annetulle funktiolle.

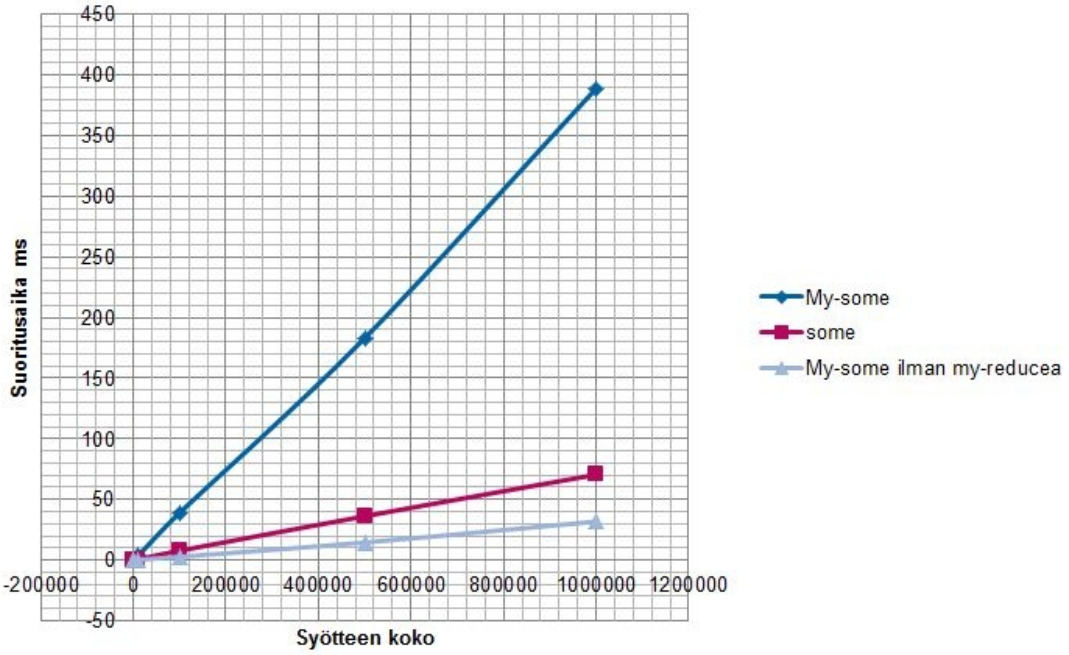


Lopputuloksena kuvaajastakin havaittu; oma toteutukseni on noin kaksi kertaa alkuperäistä hitaampi. Ohjelman syötekoolla ero on onneksi merkityksettömän pieni.

my-some

my-somea verrattiin Clojuren some funktion suoritusaikoihin. Suurin syöte on miljoona alkiota sisältävä vektori, josta etsitään parillista lukua. Parillinen arvo on vektorissa aina viimeisenä, eli kyseessä on pahin mahdollinen tapaus.

My-some on toteutettu reducen avulla, funktio palauttaa aina ensisijaisesti falsen, mutta se vertaa alkuoletustaan jokaiseen syötteen alkioon ja palauttaa näistä toden jos sellaista löytyy. Clojuren core reducea käyttäessä funktio on noin puolet core somea nopeampi, jossa arvot käydään läpi häntäoptimoidulla rekursiolla. Sen sijaan omaa reduce funktiotani käyttäessä se on selvästi core toteutusta hitaampi, joten päädyin käyttämään core version reducea sen kanssa.



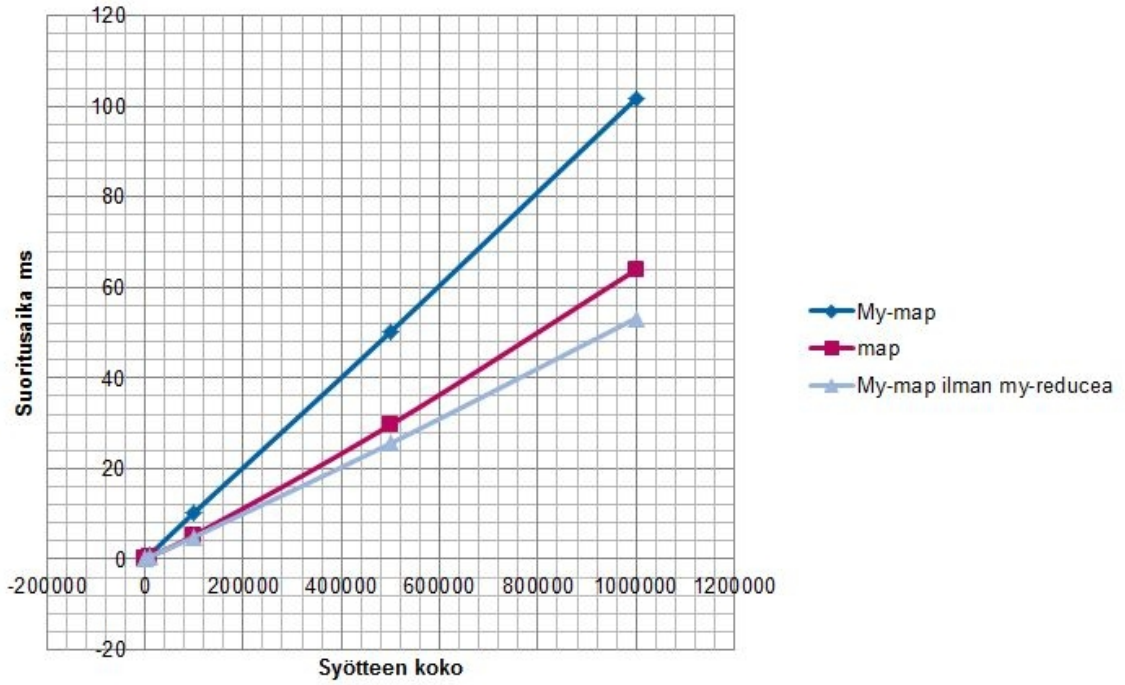
my-map

my-map on verrattavissa Clojuren mapv funktioon ja olen testannut sen suorituskykyä verraten suoritusaikoja mapv funktion suoritusaikoihin. Testattu syöte oli vastaava kuin mitä ohjelmassa käytetään, eli vektori joka palautetaan vektorina. Testissä jokaiseen lukuun lisätään 1 ja suurimman syötteen koko on miljoona alkiota sisältävä vektori.

Alunperin my-map oli toteutettu rekursiivisena, jolloin miljoona alkiota sisältävällä syötteellä ja jokaisen arvon kasvattamisella yhdellä kesti

suorituksessa noin 8 sekunttia, mutta algoritmin muuttaminen mutableksi ja suoritus loopissa laski suoritusajan noin 50 millisekunttiin. Tästä nähdään kuinka joissain tilanteissa clojuren immutable muuttujat (varsinkin vektorien ja listojen läpikäyntiin liittyvissä tehtävissä) kannattaa väliaikaisesti muuttaa mutableiksi, jotta suoritusaikaa saadaan lyhennettyä.

Myös map käyttää reducea ja siinäkin ero on selvä (joskin ei yhtä suuri kuin somessa) käytettäessä omaa toteutustani reducesta tai clojuren core toteutusta.



Testien toistaminen

Testit voidaan toistaa komentoriviltä ajamalla komento:

```
% lein midje
```

Tai mikäli lein midje komentoa ei löydy, aja ensin lein deps ja kokeile REPL:istä:

(Jos lein komentoa ei löydy, asenna Leiningen http://leiningen.org/)

```
user=> (use 'midje.repl)
user=> (autotest)
```

Tämän jälkeen testit ajetaan ja myös mitatut suoritusajat tulostuvat. Jos halutaan pelkät suoristusajat, voidaan ajaa midjen lataamisen jälkeen REPL:istä myös komento:

```
user=> (eclipse.core.internal.performance-tests)
```