Testausdokumentti

Tietorakenteiden harjoitustyö

Joosua Laakso

8. tammikuuta 2014

1 Testauksen toteutus

Ohjelmakoodin testaus on toteutettu JUnit-testausjärjestelmällä, jota käytetään Java-koodin testaukseen. Testien lähdekoodi on jaettu useaan eri tiedostoon, joista jokainen testaa omaa osa-aluetta ohjelman toiminnasta. Tärkeimmissä roolissa ovat EvaluatorTest.java-tiedoston ja PerformanceTest.java-tiedoston sisältämät testikoodit. EvaluatorTest-tiedoston koodi testaa ohjelman toimivuutta kokonaisuutena erilaisilla syötteillä, kun taas PerformanceTest-tiedoston sisältämä testilähdekoodi testaa ohjelman suorituskykyä syötteen koon kasvaessa.

2 Testisyötteet

2.1 EvaluatorTest

Testiluokka pyrkii testaamaan säännöllisiä lausekkeita siten, että kaikkia niitä eri säännöllisten lausekkeiden ominaisuuksia, joita tämä toteutus tukee, testataan jollain tavalla. Näitä ominaisuuksia ovat tähti-, unioni-, plus-, kysymysmerkki- ja jokerikorttioperaattorit, joiden lisäksi testataan myös hakasulkusyntaksia vaihtoehtoisille merkeille, sekä aaltosulkusyntaksia määrälliselle toistolle. Näitä erilaisia säännöllisiä lausekkeita testataan useilla eri syötteillä, joista osa on sellaisia, että lausekkeen kuuluu hyväksyä ne, ja osa sellaisia, että lausekkeen kuuluu hylätä ne.

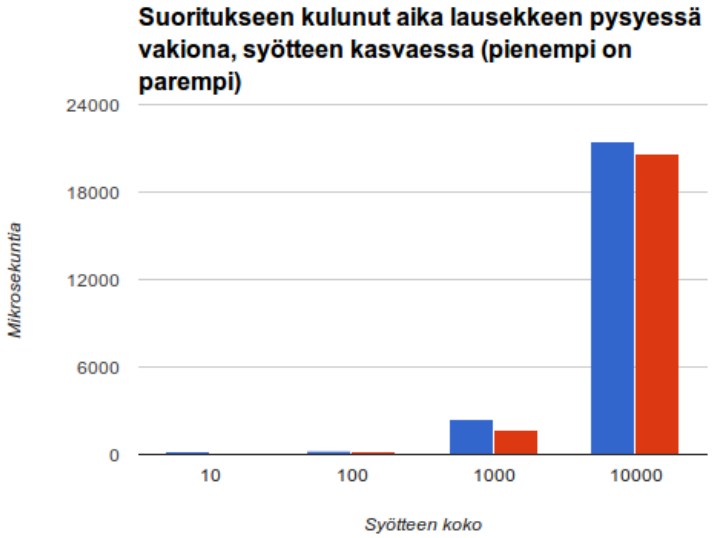
2.2 PerformanceTest

Testiluokka testaa toteutuksen suorituskykyä kahdella erityyppisellä testillä. Toinen testi testaa säännöllistä lauseketta, joka on muotoa "[A-D]+", eli joka hyväksyy kaikki epätyhjät merkkijonot, jotka sisältävät vain merkkejä A, B, C ja D. Tätä lauseketta testataa hyväksyvillä syötteillä, joiden pituudet ovat 10, 100, 1000 ja 10000. Toisen testin tarkoitus on puolestaan testata suoritusajan kasvua, kun sekä lausekkeen, että syötteen pituuden kasvavat. Testi testaa lauseketta, joka on muotoa "A?^{n}A^n", n on luonnollinen luku, eli joka hyväksyy merkkijonot joissa on n:stä 2n:ään merkkiä "A". Testi testaa suoritusaikaa säännöllisillä lausekkeilla siten, että jokaisella testausiteraatiolla n kasvaa, ja syöte on vastaavasti merkkijono, joka on merkki "A" n kertaa peräkkäin, eli jonka lauseke hyväksyy.

3 Suorituskykytestauksen tulokset

3.1 Empiiriset tulokset

Suorituskykytestit suoritettiin PerformanceTest-luokan testeillä. Vertailun vuoksi suoritin myös samat testit Javan standardikirjaston säännöllisten lausekkeiden toteutuksella vartaillakseni toteutustani siihen. Testissä, jossa lauseke pysyi



Kuva 1: Punainen: Javan standardikirjaston toteutus Sininen: oma toteutus

vakiona, oman toteutuksen suoriutuminen oli samaa tasoa Javan standardikirjaston toteutuksen kanssa. Javan standardikirjaston toteutus oli nopeampi käsittelemään tässä testissä kaiken kokoisia syötteitä, mutta nopeusero oli hyvin pieni. Toisessa testissä, jossa sekä lausekkeen pituus että syötteen koko kasvoivat, oman toteutuksen suoritusajan kasvu oli eksponentiaalista syötteen koon (ja siten myös lausekkeen pituuden) suhteen. 23 pituinen syöte oli suurin, jonka oma toteutus pystyi käsittelemään, suuremmilla syötteillä muisti loppui. Javan standardikirjaston toteutus taas selvisi samoista syötteistä siten, että aikaa kului vain alle 100 mikrosekuntia jokaisella syötteellä samoilla säännöllisillä lausekkeilla.

3.2 Johtopäätökset

Vakiolausekkeisessa testissä Javan toteutus ja oma toteutukseni eivät eronneet paljoa suorituskyvyssä, joten pidän sitä merkkinä siitä, että olen ainakin jollain tasolla onnistunut työssäni. Sen sijaan toisen testin tulokset, jossa sekä lauseke että syöte kasvoivat, ovat kummallisempia. Kun 23 pituisella syötteellä omalta toteutukseltani kului lähes minuutti suoriutua, Javan toteutus suoriutui samasta tehtävästä silmänräpäyksessä. Luulen että tämä johtuu siitä, että Javan toteutus käyttää jotain optimointia, jonka avulla tällaisen tapauksen, kun testissä oli, pystyy suorittamaan nopeasti. Tämä optimointi saattaa liittyä siihen, että lauseke, jota testattiin, oli sellainen että se hyväksyi merkkijonoja,

Kuva 2: Oma toteutus

jotka sisälsivät vain yhtä merkkiä.