Tiralabra 2013 periodi III
Viikkoraportti III

Mika Viinamäki

31. tammikuuta 2013

1 Viikon saavutukset

Tällä viikolla sain hajautustaulun korvattua omalla toteutuksella. LZW:n testeissä vaihto siihen ei merkittävästi huonontanut suorituskykyä verrattuna Javan omaan `HashMap`iin - ehkä 10% luokkaa. Sen syvällisemmin hajautustaulun suorituskykyä en ole vielä tutkinut. Valmiita tietorakenteita ei siis enää periaatteessa ole jäljellä koodissa - joskin taulukoihin liittyviä apumetodeja en vielä korvannut itse, mutta niiden korvaaminen pitäisi olla triviaalia.

Implementoin lohkopohjaisen enkoodauksen ja dekoodauksen ja se tuntui toimivan ihan hyvin, mutta hirveän kattavasti en ole sitä testannut. Nopean kokeilun perusteella suorituskyky isoilla, satunnaisilla syötteillä putosi aika reippaasti (luokkaa 10 minuutista 20 sekuntiin), mutta vielä en ole tehnyt mitään kattavampaa analyysia sen vaikutuksista muun muassa pakkaustehoon.

Lisäksi tuli refaktoroitua koodia noin muuten aika paljon. `LZWEncoder` ei ole enää täysin staattinen luokka, ja toiminnallisuus on pilkottu pienempiin metodeihin — joskin väittäisin, että algoritmin toimintaa on nyt vaikeampi seurata, kun state on siirretty "globaaleihin"muuttujiin.

Suorituskykytestejä tein enkoodauksesta ja dekoodauksesta, ja vaihtoehtoja tutkittuani päädyin käyttämään frameworkia nimeltä *Caliper* apunani. Käytännössä se hoitaa tarpeellisten toistojen tekemisen ja ajanoton automaattisesti, ja se osaa testauksen lopuksi piirtää hienoja palkkeja konsoliin tai vaikka webiin. Lisäksi se tekee suorituskykytestien kirjoittamisesta noin muutenkin karvan verran miellyttävämpää — esimerkkinä käy vaikka yllä olevan benchmarkin lähdekoodi, joka löytyy täältä.

En oikein tiedä mitä seuraavaksi kannattaisi tehdä enää itse pakkauksen suhteen — varmaan keskityn suorituskykytestien ja optimointien tekemiseen, mutta mitään erikoisia ideoita itse algoritmin toiminnan parantamiseen ei ole. Mielessä kävi LZW:n dictionaryn muodostaminen datassa esiintyvien eri kirjaimien pe-

rusteella, mutta se vaatisi että kaikki data käydään läpi ennen enkoodauksen aloittamista enkä usko siitä edes seuraavan merkittävää pakkaustehon parantumista.

Osa viime viikolla antamistasi kommenteista herätti hieman kummastusta — ilmeisesti olit listannut kaikki metodit, jotka ovat yli 10 riviä pitkiä? En itse usko että "metodi on liian pitkä"on mikään oikea ongelma. Metodi voi tietty olla liian sekava johtuen liiallisesti pituudestan — ja joitain metodeja mistä kommentoit oli ihan aiheellista lyhentää ja pilkkoa osiin. Mutta onko se, että (automaattisesti generoitu) `equals`-metodi on liian pitkä tai että siitä puuttuu Javadoc mikään oikea puute tai virhe?

2 Seuraavaksi

- Suorituskykytestejä
- Pakkaustehotestejä
- Dokumentaatioiden kirjoittelua
- Yleistä siistimistä ja refaktorointia