Viikkoraportti 1

Lähdin toteuttamaan harjoitustyön ensimmäisellä viikolla pakkausalgoritmia, joka käyttäisi pelkkää huffman-koodia tiedon tiivistämiseen. Sain ohjelman perustan koodattua melko nopeasti. Ohjelmassa on vielä luultavasti bugeja, mutta pakkaaminen ja purkaminen toimii.

Koodatessani tulin miettineeksi, että mitä jos jokaisella aakkoston kirjaimella olisikin oma puunsa? Eli kirjaimen koodi riippuisi siitä, mikä on sen edellinen merkki. Tämä mahdollistaa vielä lyhyempien koodien käyttämisen useammalla merkillä, sillä jotain kieltä sisältävässä datassa tietyt merkit ovat todennäköisempiä tiettyjen merkkien jälkeen. (esimerkiksi html:ssä “<” merkkiä seuraa usein “/”)

Koodasin aluksi ihan huvikseni tätä mekaniikkaa käyttävän pakkausalgoritmin, ja huomattuani, että tiivistyssuhteessa on oikeasti eroa, päätin tehdäkin koko algoritmin tällä tavalla. Toki on ongelmana, että puiden määrä kasvaa potenssiin kaksi, mutta ottaen huomioon koodipuiden vaatiman tilan, on pakkaussuhde silti parempi. Tässä muutaman testitiedoston pakkaustulokset:

Tiedosto/pakkausmenetelmä	Huffman	Viritelty huffman	Zip (linuxin oletuspakkaaja)
Kalevala, 571.3 KB	304.1 KB / 53.2%	242.9 KB / 42.5%	201.3 KB /35.2 %
Html-tiedosto, 937.9 KB	625.1 KB / 66.6%	406.6 KB / 43.3 %	70.1 KB / 7.4%
Java-koodia, 25KB	14.4 KB / 57.6%	12.4 KB / 49.6 %	6.2 KB / 24.8 %

Kuten taulukosta näkee, pakkaa viritelty Huffman selkeästi pienenpään tilaan, kuin pelkkä tavallinen huffman. Selvä voittaja toki on zip, joka on paljon toistoa sisältävässä testitiedostossa (kooditiedostot) aika lyömätön. Tämä toki johtuu zipin pakkaustavasta, jossa pystytään esittämään toistetut datapätkät pelkkinä viittauksina edelliseen samaan dataan.

Ongelmana projektissa on ollut testien kirjoittaminen. En oikeen tiedä, kuinka testejä kannattaisi kirjoittaa ohjelmalle, jossa suurin osa metodeista liittyy tiedoston kirjoittamiseen ja lukemiseen. Ja metodit toimivat oikeastaan aina yhtenä kokonaisuutena. Testit ovat siksi tässä vaiheessa vajavaiset, vaikka vaatimuksissa olikin 100% testikattavuus alusta asti. Olisikohan tämän ohjelman tapauksessa testaus järkevämpi suorittaa testausdokumentilla?

Ohjelma on nyt siinä vaiheessa, että kaikilla testitiedostoilla, mitä olen pakannut ja purkanut, algoritmi näyttäisi toimivan oikein. Joissain tiedostoissa tiedoston loppuun tulee ylimääräinen merkki. Tämä luultavasti johtuu siitä, että koska bitit eivät jakaudu tasaisesti kahdeksan bitin tavuihin, jää osassa tiedostoista viimeiseen tavuun merkityksettömiä bittejä, jotka ohjelma toki tulkitsee merkeiksi.

Seuraavaksi alan toteuttamaan omia tietorakenteita, eli ohjelmassa tarvittavan hajautustaulun ja keon. Uskoisin, että näissä riittää koodattavaa vähäksi aikaa. Jossain vaiheessa pitää muuttaa myös ohjelma kirjoittamaan sanakirja ja tiedosto samaan tiedostoon, koska ei ole kovin tyylikästä, että pakattu tiedosto on kahdessa osassa.