Tietorakenteiden harjoitustyö: Pakkausalgoritmi

Tietorakenteiden harjoitustyön aiheenani on pakkausalgoritmi. Ensimmäinen tavoite on toteuttaa toimiva, suhteellisen tehokas ja tietoa riittävästi tiivistävä algoritmi. Teen ohjelmani java-kielellä, koska se on minulle tutuin kieli. Tutulla kielellä voi keskittyä itse asiaan, eli ohjelmointiin. Kun algoritmi on valmis, toteutan harjoitustyön ohjeiden mukaisesti ohjelmointikielen valmiit tietorakenteet itse.

Toteutan pakkausalgoritmina Huffmanin algoritmin. Huffmanin algoritmissa pakattavan tiedoston merkit järjestetään yleisyysjärjestykseen, antaen yleisimälle merkille lyhyin koodi. Tästä toki seuraa, että harvinaiset merkit saavat pidemmät koodit, mutta kunhan yleisten ja harvinaisten merkkien määrässä on riittävän suuri ero, pienenee tiedoston koko huomattavasti. Huffman-koodi antaa jokaiselle merkille niin sanotun etuliitekoodin (prefix code tai prefix-free code). Etuliitekoodissa minkään merkin koodi ei voi olla toisen merkin koodin alkuosa, joten koodi on aina yksiselitteinen, vaikka merkkien koodien pituudet vaihtelevatkin.

Harjoitustyössäni tulen toteuttamaan tietorakenteista ainakin keon, puun ja hajautustaulun. Keon avulla tiedostosta löytyneet merkit on helppo järjestää puun rakentamista varten. Koodin tärkein osa onkin puu, mutta siinä juuri ei ole ohjelmointia, sillä puu on vain yksinkertainen binääripuu. Hajautustaulu on kätevä algoritmin eri vaiheissa, esimerkiksi kun pitää saada nopeasti merkkiä vastaava binääriesitys tiedostoa pakatessa.

Huffman-koodin muodostaminen on melko nopea operaatio, kirjallisuuden mukaan O(n log n). Tiedoston kirjoittaminen ja läpikäyminen vie kuitenkin aikaa pitkällä tiedostolla, sillä jokainen merkki pitää kirjoittaa bittitasolla erikseen tiedostoon. Merkkien koodit eivät siis noudata tasaisia 2:n potensseissa meneviä lohkoja, vaan ovat tiiviisti peräkkäin, jotta tiedostosta saadaan mahdollisimman pieni.

Käytän testisyötteinä pitkiä tekstitiedostoja. Tekstitiedosto on helppo syöte, sillä aakkosto on hyvin rajattu. Jos tiedosto sisältää järkevää luonnollista kieltä, pääsee huffman-koodilla hyvään tiivistyssuhteeseen, koska luonnollisessa kielessä esiintyvät tietyt merkit huomattavasti toisia merkkejä useammin.

Algoritmin valmistuttua aion keskittyä ohjelman optimointiin. Algoritmia voisi ehkä suunnata tietyntyyppisille tiedostoille pienentäen pakatun tiedoston kokoa. Minua kiinnostaisi myös pakkaus- ja purkualgoritmin nopeuttaminen. Algoritmia voisi nopeuttaa toki tehostamalla algoritmia, mutta myös käyttämällä useampaa säiettä, joita suoritettaisiin samanaikaisesti usealla prosessorilla. Useammalla kuin yhdellä ytimellä suoritettavan ohjelman tekeminen olisi hyvin mielenkiintoista. Kävin tänä keväänä käyttöjärjestelmät -kurssin, ja haluaisin päästä hyödyntämään siellä opittuja rinnakkaisohjelmoinnin taitoja. Olisi mielenkintoista vertailla moniajetun ja tavallisen ohjelman nopeuseroja isoilla testitiedostoilla.

Lähteinä olen käyttänyt Thomas H. Cormenin kirjaa “Introduction to Algorithms” sekä Wikipedian Huffman-koodista kertovaa sivua (http://en.wikipedia.org/wiki/Huffman_coding)