Määrittelydokumentti

Työssä toteutan tiedoston tiivistämiseen vaaditun algoritmin. Alustavana ajatuksena on toteuttaa Huffmannin koodeihin perustuva algoritmi joka korvaa tiedostossa olevat symbolit pienemmillä, vaihtelevamittaisella koodeilla. Algoritmin toteutukseen vaaditaan vähintään prioriteettijono ja puu Huffmannin puun toteuttamiseksi. Lisäksi tarvitsen hajautustaulun. ArrayListiä tarvitaan tiedoston lukuvaiheessa sekä prioriteettijonon sisällä heapin toteutukseen. Tarvitaan ainakin hajautustaulu, prioriteettijono, puurakenne, arraylist.

Perusideana minulla olisi lukea tiedostosta haluttu määrä bittejä hajautustauluun, jossa avaimena on bittikuvio (n bittiä – koska java näemmä osaa lukea vain tavuja, on n jokin tavun moninkerta ainakin aluksi - , esimerkiksi tekstitiedoston tapauksessa voisi olla yksi kirjain) ja arvona kuinka monta kertaa tiedostossa kyseinen bittikuvio esiintyy. Lisäksi pidettäisiin yhtä muuttujaa kirjaamassa bittikuvioiden kokonaismäärää jotta saadaan esiintymistiheys helposti laskettua. Bittikuviot asetetaan minimiprioriteettijonoon esiintymistiheyden mukaan, josta niitä aletaan yhdistää Huffmannin puuksi. Puu luettaisiin uuteen hajautustauluun jossa tälläkin kertaa avaimena olisi bittikuvio, mutta tietona olisi korvaava koodi. Nyt lähdetiedosto voidaan kirjoittaan tulostiedostoon, jossa alussa on tallessa merkki – koodi-parit tiedoston purkamista varten, mitä seuraa tiivistetyt koodit. Purkuvaiheessa luetaan tiedostosta merkki-koodiparit hajautustauluun siten että koodi on avaimena, ja kirjoitetaan uusi hajautustaulun avulla jossa alkuperäinen teksti on purettuna.

Syötteenä ohjelma siis saisi pakattavan tiedoston ja luo uuden pakatun tiedoston. Ohjelma todennäköisesti tulee olemaan komentorivipohjainen koska inhoan graafisten käyttöliittymien rakentamista.

Huffmannin puu on mahdollista koota O(n log n) ajassa, minkä lisäksi luku\kirjoitusoperaatiot ovat verrannollisia merkkien määrään O(n), ennen koodin kirjoittamista kuvittelisin että hajautuskarttojen kokoaminen\lukeminen olisi vakioaikaista jokaiselle merkille -> koko operaatio O(n). Näin ollen kuvittelisin että ohjelma toimisi ajassa O(n log n) jossa n on merkkien määrä. Tilavaatimuksen kannalta ainakin kaikki merkit on pidettävä hajautuskartassa -> O(n) avain\arvo-paria -> vähintään O(n) tilavaatimuus. Tarkennan tätä osiota kun alan kirjoittamaan koodia jos arviot tässä vaiheessa osoittautuvatkin vääräksi.

Lähteet:

http://en.wikipedia.org/wiki/Huffman_coding – sivu tarkistettu 1.8.2013
Introduction To Algorithms, Third edition sivut 428 – 435 – Thomas H. Cormen et al.