Mika Huttunen

Määrittelydokumentti

Yleistä

Tarkoituksenani on kehittää ohjelma, jolla voi pakata ja purkaa tavallisia tekstitiedostoja ja se tukee myöhemmin ehkä myös muita tiedostotyyppejä.
Ohjelmalla voi pakata tiedostoja ainakin kahta eri pakkausalgoritmia käyttäen; käyttäjä saa valita käyttääkseen joko Huffman -algoritmin tai Lempel-Ziv-Welch -toteutusta.

Valitsin että käytän ym. algoritmeja ongelmanratkaisuun, sillä ne ovat muutenkin yleisesti käytettyjä ja lisäksi minua ohjeistettiin toteuttamaan ohjelmani niitä käyttäen.

Toteutan ohjelman käyttäen tietorakenteina ainakin itsetoteutettua maksimikekoa sekä binääripuuta. Lisäksi toteutan hajautustietorakenteen jollain lailla, jota voidaan hyödyntää eri merkkien laskemisessa, kun pakataan tekstitiedostoja Huffman -algoritmia käyttäen.
Muita mahdollisia aputietorakenteita tulee tod. näk. käyttöön myöhemmin, mutta nämä vaikuttavat näin alkuun ohjelman toimimisen kannalta tärkeimmiltä.
Lempel-Ziv-Welchin toteutusta en ole vielä juuri edes miettinyt, mutta se saattaa hyvinkin tarvita muita tietorakenteita, mutta katselen sitä myöhemmin kun saan ensin Huffman -algoritmin toimimaan kuten haluankin.

Syötteenä ohjelma saa tiedoston polun, joka annetaan sille kirjoittamalla tekstikäyttöliittymän kautta.

Algoritmien toimintaperiaatteet

Huffman

<u>Pakkaamis- ja purkuvaiheet:</u>
Aikavaativuus: O(m * n), missä m on syötteenä olevan tekstitiedoston koko ja n siinä olevien erilaisten tavujen lukumäärä
Tilavaativuus: O(m)

<u>Alustus</u>

Huffmanin algoritmi toimii siten että ennen sitä luetaan pakattava tekstitiedosto ja luoda solmuja jokaista siinä esiintyvää merkkiä varten. Solmuja syntyy siis n kpl, missä n = “merkkien lkm.”. Näille solmuille kirjataan lisäksi tietoa siitä, kuinka monta kertaa ne ovat tiedostoa lukiessa esiintyneet, ja ne asetetaan tämän tiedon nojalla suuruusjärjestykseen maksimikekoon.

Huffmanin algoritmi

Huffmanin koodauksessa kaikki solmut poistetaan keosta (ajassa O(1)) luoden samalla myös ylimääräisiä solmuja (n - 1) kpl, joten käsiteltäviä solmuja on siis 2n - 1 = O(n) kpl. Jokainen näistä luoduista solmuista asetetaan myös maksimikekoon käsittelyä varten. Tämä lisäysoperaatio vie aikaa O(logn), joten algoritmin aikavaativuus on suuruusluokkaa O(nlogn).

Pakkausvaihe

Kun algoritmi on suoritettu, on saatu aikaiseksi binääripuu, joka kertoo, mitä binääriesitystä kukin tiedoston merkki vastaa. Nyt luodaan uusi tiedosto, kelataan alkuperäinen tiedosto läpi ja kirjoitetaan tähän uuteen tiedostoon alkuperäisen tiedoston data käyttäen siinä olevien merkkien binääriesityksiä, jotka Huffmanin algoritmilla määritettiin. Tähän tiedostoon sisällytetään myös ko. binääripuu, jotta muodostettu pakkaus voidaan myöhemmin purkaa.

Purkuvaihe

Muodostetun pakkauksen purku tapahtuu käytännössä päinvastoin kuin pakkaaminen; tarkastellaan alkuperäisen tiedoston dataa vastaavaa binääriesitystä (pakkauksen datassa) ja käytetään apuna pakkauksessa mukana olevaa Huffman puuta. Näiden avulla löydetään merkit, jotka uuteen muodostettavaan tiedostoon tullaan kirjoittamaan.

Lempel-Ziv-Welch

Tarkoituksenani on toteuttaa Huffmanin algoritmilla toimiva pakkaaminen ensin valmiiksi ja kun se toimii kuten haluan, laajennan joko sitä jotenkin tai toteutan todennäköisesti tämän lisätyönä. Itse Lempel-Ziv-Welchin menetelmään en ole vielä tutustunut.

Lähteet

Huffman Coding, Wikipedia, 18.5.2014, http://en.wikipedia.org/wiki/Huffman_coding

Lempel-Ziv-Welch, Wikipedia, http://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch