Mitä algoritmeja ja tietorakenteita toteutan työssäni

Tietorakenteiden ja algoritmit harjoitustyöni aiheena on tiedontiivistys. Suunnitelmissa olisi toteuttaa 4 erilaista häviötöntä tiedostontiivistys algoritmia ja sitten vertailla niiden suorituskykyä, aika- ja tilavaativuuksia.

Varmasti aion toteuttaa Huffmanin koodauksen ja Lempel-Ziv-Welchin algoritmin ja niistä tarkemmin lisää alempana. Lisäksi aion tehdä molempien algoritmien variaatioista yhden tapauksen. Esimerkiksi Huffmanin koodauksesta joko adaptiivisen-, pituusrajoitteisen-, template tai n-ary Huffman koodauksen. Ja neljäs algoritmi tulee olemaan jokin muu Lempel-Ziv parivaljakon kehittämistä algoritmeista, luultavasti joko LZ77 tai LZ78, mahdollisesti joku muu. Päätän tarkemmin mitkä kaksi variaatiota valitsen toteuttavaksi sitten kun Huffmanin koodaus ja Lempel-Ziv-Welchin algoritmit toimivat.

Huffmanin koodauksessa aion käyttää seuraavia tietorakenteita: binäärihakupuu, minimikeolla toteutettu prioriteetti jono ja muutama hajautustaulu.

Huffanin koodauksen toiminta perustuu siihen että tiedostossa yleisimmin käytetyt merkit koodataan lyhyimmiksi bittijonoiksi kuin harvinaisemmat merkit ja täten tiedosto tulee viemään vähemmän tilaa kun se on pakattu. Huffmanin koodauksessa pakkaamisessa käydään ensimmäiseksi tiedosto läpi merkki merkiltä ja lasketaan niiden esiintyvyydet tämä talletettaan hajautustauluun. Sitten tehdään solmu jokaiselle käytössä olevalla merkille ja laitettaan tämä prioriteettijonoon. Prioriteettijonon avulla muodostetaan binäärihakupuu, jonka lehtisolmuina ovat kaikki tiedostossa olevat merkit. ”Huffmanin koodi” muodostuu kun juuresta kuljetaan niihin merkkeihin. Tämä koodi ja sitä vastaava merkki talletettaan hajautustauluun tiedoston pakkaamista varten. Pakatun tiedoston alkuun tulee purkua varten tiedoston pakkaamiseen käytetty puu. Tämän jälkeen alkuperäinen tiedosto käydään läpi ja jokainen siinä oleva merkki koodataan pakattuun tiedostoon Huffmanin koodilla. Tiedoston purkamisessa käytettään apuna tiedoston alussa olevaa Huffmanin puuta, siitä saadaan mitä jokaista kooditavua vastaava merkki on selkokielellä.

Lempel-Ziv-Welchin algoritmin toiminta perustuu tekstissä esiintyviin toistuvuuksiin. LZW pakkauksessa aluksi lisätään sanakirjaan kaikki merkit ja niitä vastaavat koodit mitä tekstissä on. Tämän jälkeen otettaan merkkijono w = ”” ja käydään kaikki alkuperäisen tiedoston merkit läpi. Jos w + c(missä c o joku sillä hetkellä läpi käytävä merkki) on sanakirjassa niin w = w + c. Muuten lisätään w + c sanakirjaan ja lisätään w:n sanakirjan koodi pakattuun tiedoston ja w = c. LZW purkaminen toimii samalla tavalla. Siinä käydään läpi syötettä ja kun uusi merkkijonokombinaatio tulee vastaan se lisätään sanakirjaan ja sitten sitä käytettään syötteen koodin purkuun.

LZW algoritmin sanakirjan ajattelin toteuttaa hajautustaulun avulla, en vielä tiedä tarvitsenko jotain muita tietorakenteita tätä algoritmia varten.

Mitä ongelmaa ratkaisen ja miksi valitsin kyseiset algoritmit/tietorakenteet

Ongelmana on että tiedosto pitäisi saada mahtumaan pienempään tilaan kun sitä säilytettään ja kun sitä käytettään se pitää pystyä avaamaan alkuperäiseen muotoonsa. Kurssisivun mukaan olisi pyrittävää 40-60% kokoon alkuperäisestä koosta.

Otin tiedoston tiivistämisen aiheeksi koska en vuoden passiivisen miettimisen jälkeen löytänyt mitään mielenkiintoisempaa aihetta. Olen myös sitä mieltä että olisi hyödyllistä oppia tämä asia kun tätä ei käydä oikein missään kandivaiheen kursseilla läpi.

Valitsin Huffmanin koodauksen ja Lempel-Ziv-Welchin algoritmit, koska ne ovat tunnettuja algoritmeja, jotka ovat riittävän helppo toteuttaa. Lisäksi niiden toiminta eroaa tosistaan riittävissä määrin jotta niiden toteutus ja vertailu olisi tarpeeksi vaihtelevaa.

Lisäksi valitsen toteuttavaksi kaksi variaatiota edellä mainituista algoritmeista, jotta työn laajuus olisi riittävän suuri viiden opintopisteen suoritukseen. Näiden algoritmien valinta tule perustumaan siihen, että haluan niiden olevan alkuperäisestä riittävän erilaisia, mutta silti tarpeeksi samankaltaisia. Näin koska haluan algoritmien vertailusta mielenkiintoisempaa ilman että teen liikaa työtä kurssin eteen (näin kävisi jos tekisin kaksi täysin uutta pakkaus/purku algoritmia).

Mitä syötteitä ohjelma saa ja miten näitä käytetään

Ohjelma saa syötteenä tekstitiedoston joka joko pakataan tai puretaan käyttäjän valinnan mukaan. Käyttäjä pystyy myös valitsemaan käytettävän algoritmin.

Aion käyttää erikokoisia tekstitiedostoja ohjelman testauksessa ja vertailla niiden avulla algoritmien suorituskykyä.

Jos aikaa jää saatan yrittää saada algoritmeja toimimaan myös muilla tiedostotyypeillä kuin tekstitiedostoilla.

Tavoitteena olevat aika- ja tilavaativuudet

Huffman koodauksen operaatioiden aika- ja tilavaativuudet ovat seuraavat:

Tekstin frekvenssien laskenta: Aikavaativuus O(n), tilavaativuus O(n).

Huffmanpuun muodostus prioriteettijonon avulla: Aikavaativuus O(n log n), tilavaativuus O(n)

Lähdetiedoston lukeminen vie aikaa O(n) ja kohdetiedoston kirjoitus O(n) ne vievät myös tämän määrän tilaa.

Purkaessa lähdetiedoston lukeminen vie O(n) ja merkkien haku lähdetiedoston puusta vie aikaa O(log n). Kohde tiedoston kirjoittaminen vie aikaa O(n).

Joten tämän takia algoritmin aikavaativuus tulee olemaan O(n log n) ja tilavaativuus O(n), missä n on tiedostossa olevien merkkien määrä.

Lempel-Ziw-Welchin algoritmin aika- ja tilavaativuudet ovat sekä pakatessa että purettaessa O(n) missä n on tiedostossa olevien merkkien määrä.

Lähteet

http://en.wikipedia.org/wiki/Huffman_coding

http://en.wikipedia.org/wiki/Lempel–Ziv–Welch

http://en.wikipedia.org/wiki/Lossless_compression