Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit (alkukesä 2014)

Aleksi Paavola

Tiedon tiivistys

”Tiedosto tulisi saada mahtumaan pienempään tilaan, miten tämä onnistuu? Toivottava lopullinen koko on 40-60% alkuperäisestä koosta. Tiedosto pitää myös pysytä avaamaan alkuperäiseen muotoon myöhemmin.” - Aihemäärittely, kurssiwiki.

Luodaan ohjelma tiedoston pakkausta ja purkamista varten toteuttamalla Huffman-koodaus ja LZW (Lempel-Ziv-Welch) algoritmit.

Näiden lisäksi toteutetaan algoritmien käyttämät tietorakenteet, niiden nopeus -ja pakkaustehokkuusvertailu, graafinen käyttöliittymä, tiedostoon kirjoittaminen ja tiedostosta luku pakattujen tiedostojen käsittelyä varten sekä koodin kattava testaus ja Javadoc.

Ohjelma saa syötteekseen pakkaamattoman tai pakatun tiedoston ja joko pakkaa tai purkaa sen valitulla algoritmilla, ilmoittaen samalla operaatioon kuluneen ajan ja ulostulotiedoston koon.

Toteutettavat algoritmit ja niiden tietorakenteet

Huffman

Valitsee tiedoston merkkien esitysmuodon niin, että korkeimman tiheyden omaava (yleisin) merkki ilmoitetaan lyhyimpänä bittijonona.

Aikavaativuustavoite pakkaamiselle ja purkamiselle O(n logn).

Pakkaus: Ensin lasketaan merkkien tiheydet, sitten rakennetaan niiden perusteella rekursiivisesti Huffman-puu. Lopuksi muunnetaan puun avulla alkuperäinen tiedosto tiivistettyyn bittiesitykseen ja kirjoitetaan se levylle. Huffman-puun uudelleenrakentamista varten tiedostoon pitää sisällyttää headeri, josta voidaan lukea alkuperäinen merkki ja sen tiheysarvo.

Purku: Rakennetaan Huffman-puu pakatun tiedoston headerin avulla, jonka jälkeen pakattu tiedosto puretaan kulkemalla bittijonoa lukemalla puuta alas, muuntaen sitä varsinaiseksi merkkijonoksi.

- **Huffman-puu**

  Binääripuu, joka pitää lehdissään tietoa merkeistä. Puu rakennetaan rekursiivisesti merge-funktion avulla yhdistämällä alipuita niiden yhteenlasketun tiheysarvon mukaisesti. Puuta alas kuljettaessa jokainen siirtymä vasempaan lapseen tarkoittaa bittiä 0, oikeaan 1. Kun saavutaan lehteen, kirjoitetaan pakatessa reitti bittijonoon. Purettaessa taas kirjoitetaan lehdestä löytyvä merkki.

  Koska tätä binääripuuta ei sen rakentamisen jälkeen muokata, ei sille toteuteta rotate- tai delete-funktioita.

- **Prioriteettijono**

  Käytetään Huffman-puuta rakennettaessa. Jonossa korkeimman prioriteetin saa sellainen puun lehti, jolla on kaikista pienin todennäköisyys (paino).

Lempel-Ziv-Welch

Etsii tiedostosta merkkijonojen toistuvuuksia ja rakentaa niiden perusteella sanakirjan siten, että yhdellä bittijonolla voidaan kuvata useampaa merkkiä. Merkit esitetään yli 8 bitin kokoisina (yleensä 9 – 12 bittiä).

Aikavaativuustavoite pakkaamiselle ja purkamiselle O(n).

Pakkaus: Luetaan merkki ja siitä seuraava merkki. Jos näiden kahden merkin yhdistelmää ei ole sanakirjassa, lisätään se sinne ja kirjoitetaan yksittäinen merkki jonoon. Jos yhdistelmä löytyy sanakirjasta, katsotaan, löytyykö yhdistelmän ja siitä seuraavan merkin yhdistelmä, jne.

Purku: Samalla tavalla kuin pakkaus, mutta nyt luetaan ennaltamäärätyn kokoisia bittijonoja.

- **Sanakirja**

  Sisältää aluksi kaikki yksittäiset merkit. Uusia merkkien yhdistelmiä lisätään löydettäessä. Sanakirja voidaan rakentaa suoraan pakatusta tiedostosta, eikä sitä tarvitse sisällyttää esim. pakatun tiedoston headerissa.

Muut toteutettavat ohjelmiston osat

Ohjelmiston osat jotka eivät sisälly varsinaisiin algoritmeihin.

Nopeus -ja pakkaustehokkuusvertailu

Toiminto, joka mahdollistaa Huffmanin ja LZW:n vertailun valitulla tiedostolla. Suoritetaan molemmat algoritmit ja verrataan tuloksia.

Ajastin: Mittaa algoritmien suoritukseen kulunutta aikaa ja ilmoittaa sen.

Tiedostokoko: Lukee ja ilmoittaa tiedostokoon.

Graafinen käyttöliittymä

Tietokonehiirellä käytettävä. Sisältää seuraavat ominaisuudet:

1. Tiedoston valinta.
2. Algoritmin valinta.
3. Pakkaus ja purku.
4. Tehokkuusvertailu.
5. Konsoli tai jokin muu käyttäjälle luettavaa palautetta antava rajapinta.

Tiedoston luku ja tiedostoon kirjoittaminen

Ohjelmiston pitää pystyä lukemaan ja kirjoittamaan sekä pakkaamattomia että pakattuja tiedostoja. Tätä varten toteutetaan bittien luku-, kirjoitus- ja kompressointioperaatiot käyttäen apuna Javan valmiita kirjastoja.

Testaus

Testaamiseen käytetään JUnit-viitekehystä. Testikattavuuden tavoite on 100% kaikkien muiden ohjelmiston toimintojen paitsi graafisen käyttöliittymän osalta. Luokat ja metodit jätetään julkisiksi yksittäisten olioiden testaamisen helpottamiseksi.

Käytetyt lähteet

https://github.com/TiraLabra/Alkukes-2014/wiki

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

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