Tiralabra 2013 periodi III
Toteutusdokumentti

Mika Viinamäki

21. helmikuuta 2013

1 Ohjelman yleisrakenne

Ohjelma koostuu muutamasta, jossain määrin itsenäisestä osiosta:

- LZW-pakkaaja ja -purkaja, `kauhsa.compression.lzw`-paketti
- `BitGroup` ja siihen liittyvät apuluokat, jotka mahdollistavat tiedon kirjoittamisen ja lukemisen bitti kerrallaan, `kauhsa.utils.bitgroup`-paketti
- Oma hajautustaulu-implementaatio, `kauhsa.utils.hashmap`-paketti
- Satunnaiset apuluokat, `kauhsa.utils`-paketin juuressa
- Komentoriviltä käytettävä käyttöliittymä, `kauhsa.LZWApp`-luokka

Näiden ohjelman mukana on erinäisiä suorituskykytestejä sekä yksikkötestejä.

2 LZW

Algoritmi on toteutettu enimmäkseen Wikipedian LZW-artikkelin[1] perusteella. LZW ei algoritmina ota kantaa siihen, miten sen ytimenä oleva sanakirja on toteutettu — tässä implementaatiossa on käytetty sekä pakkaamisessa että purkamisessa sanakirjana hajautustaulua.

Koska LZW-toteutus pystyy pakkaamaan mitä tahansa binäärimuotoista dataa, sanakirja alustetaan kaikilla mahdollisilla yksittäisillä tavuilla (0-255). Lisäksi sanakirjaan lisätään erityinen EOF-koodi[2], jolla merkitään pakatun datan loppu. Toteutus ei kuljeta näitä tietoja pakatun datan mukana, vaan luottaa siihen että purkaja on yhteensopiva pakatun datan kanssa.

[1] `http://en.wikipedia.org/wiki/LZW`
[2] "End of File"

LZW-implementaatiossa on kaksi eri pakkaustapaa:

- "Tavallinen" pakkaus, joka pakkaa kaiken datan käyttäen samaa sanakirjaa alusta loppuun. Hyvä pakkausteho hyvin pakkautuvalla datalla, mutta saattaa olla todella hidas suurella määrällä huonosti pakkautuvaa dataa.
- Pakkaus lohkoissa, joka aloittaa pakkaamisen alusta aina kun sanakirja kasvaa liian suureksi. Nopeampi kuin tavallinen pakkaus, mutta etenkin hyvin pakkautuvalla datalla pakkausteho on yleensä jonkin verran huonompi. Sanakirjan koon yläraja on säädettävissä.

Tietyllä pakkaustavalla pakattu data on purettava sitä vastaavalla purkutavalla.

3 Hajautustaulu

Hajautustaulu on toteutettu Tietorakenteet ja algoritmit -kurssin luentomateriaalissa[3] esitettyä avointa hajautusta sekä kaksoishajautusta käyttäen. Käytännössä toteutus on identtinen materiaalissa esitetyn pseudokoodin kanssa, joskaan esimerkiksi poistamista ei ole toteutettu koska se ei ole LZW-algoritmin kannalta tarpeellinen ominaisuus.

4 Puutteet

Koodin rakenteessa on joitain puutteita — olisi muun muassa kätevää, mikäli LZW-sanakirja olisi rajapinta ja `LZWEncoder` ja `LZWDecoder` -luokille pystyisi antamaan konstruktorin parametrina instanssin haluttuun sanakirja-implementaatioon. Kyseiset luokat voisivat muutenkin olla selkeämmin toteutettuja. Lisäksi esimerkiksi koodin esittämiseen tarvitun bittimäärän laskemisen voisi siirtää pois `LZWDictionary`-luokasta.

Lisäksi LZW-pakkaus ei ole niin tehokas kuin voisi olla - hajautustaulun käyttäminen sanakirjana on tarpeettoman tehotonta. Parempia ratkaisuja löytyy muun muassa täältä — sääli, etten löytänyt sivua ennen kuin aloitin algoritmini toteuttamisen. Parempi sanakirja ei kuitenkaan ehkä olisi aivan valtaisa muutos ohjelmaan.

[3] `http://www.cs.helsinki.fi/u/floreen/tira2012/tira.pdf`, s. 310–325