Helsingin Yliopisto

Tietojenkäsittelytieteen laitos

Tietorakenteiden harjoitustyö

Adaptive Huffman encoding

Timo Sand
timo.sand@cs.helsinki.fi

24. huhtikuuta 2009

Sisältö

1 Aiheen määrittely

Harjoitustyön aiheena on tehdä ohjelma, jokaa pystyy pakkaamaan ja purkamaan esim. tekstitiedoston. Tähän tarkoitukseen käytetään "Adaptiivista Huffman" -koodausta. Huffman -koodauksessa pakattavaa dataa käsitellään yksi merkki kerrallaan, eli luetaan merkit bittijonona. Kukin merkki koodataan bittijonoksi, joka on sitä lyhyempi, mitä useammin merkki esiintyy datassa. Adaptiivisessa Huffman-koodauksessa luodaan binääripuu ns. "lennosta", eli jokainen luettu merkki päivittää puuta ja täten merkkejen bittijonoja. Purku tapahtuu samassa järjestyksessä, eli luetaan bittijono ja käännetään se merkiksi, joka lisätään puuhun. Kun uusi merkki luetaan ensimmäistä kertaa puuhun, niin sen desimaaliesitys kirjoitetaan kokonaisena bittijonona tiedostoon, jotta siitä pystytään rakentamaan puuta.

Käytettyjen työtuntien määrä: n. 80-85

2 Käyttöohje

Ohjelma käynnistetään komennolla huffman.sh. Ilman parametrejä se tulostaa ohjeet käyttäjälle. Parametrillä *-d* ohjelma ajetaan gdb:ssä ja tiedostopolku paramaterinä käynnistää itse ohjelman.

3 Algoritmit ja tietorakenteet

3.1 Huffman-puu

Huffman-puu on binääripuu. Huffman-puussa on solmuja $2n - 1$, joten, kun oletetaan että käytetään ainoastaan ASCII merkkejä, niin voidaan sanoa, kun lehtiä on 257, että solmuja on maksimissaan 513. Huffmann-puu on aina täysi, elikkä jokaisella solmulla on 0 tai 2 lasta. Lehtien määrä siis merkkien desimaaliesityksen mukaan 0-255, sekä NYA-solmu, joka siis kuvastaa ei vielä koodattuja merkkejä.

Binääripuun lisäksi käytetään kahta taulukkoa, toisessa on suorat viitteet puun alkioihin, täten kaikki puuhun liittyvät operaatiot tulevat vakioaikaiseksi ja toisessa taulukossa on kaikki puun solmujen painot. Ensimmäinen indeksoidaan alkioitten merkin bittiesityksen mukaan ja toinen solmuje indeksin mukaan.

3.2 FGK-algoritmi

Algoritmi FGK on kolmen miehen kehittämä algoritmi: Faller, Gallager ja Knuth. Algoritmin pohjana on solmujen sisar-ehto. Binääripuun solmuilla, jossa ei ole negatiivisiä painoarvoja, on sisar-ehto voimassa, jos jokaisella solmulla on sisar ja jos solmut voidaan numeroida nousevassa järjestyksessä, jossa jokainen solmu on sen sisar-solmun vieressä. Myöskin solmun vanhempi on koreammalla numerojärjestyksessä.

Algoritmin toimintaperiaate on seuraavanlainen. Niin pakkaus kun purku rakentaa syötteenä annetusta tiedostosta huffman-puun samalla tavalla. Jokaista merkkiä kohden lähetetään koodijono puuhun, joka sitten päivitetään.

4 Testaus

Ohjelman pakkaamista on testattu syötteillä: lorem.txt ja ly-ebook.txt Ohjelma purkua on testattu syötteilä: lorem.bin ja ly-ebook.bin Kummatkin testisyötteet ovat mukana. Kaikilla testeillä tähän asti ohjelma suoriutuu. Ohjelmaa on myös testattu intensiivisesti toteutuksen aikana.

5 Toteutuksen puutteet

Toteutus ei tällä hetkellä osaa kirjoittaa bittijonoja, joten pakkaamisen sijaan, se tekee alkuperäisistä tiedostoista isompia. Purun yhteydessä ei osata käsitellä tiedoston lopussa olevia roskabittejä, joten tiedoston loppuun saatetaan kirjoittaa "roskaa".

Viitteet

[1] *http://www.cs.sfu.ca/CC/365/li/squeeze/AdaptiveHuff.html* - Java appletti huffmannin toiminnan visualisoimiseksi

[2] *http://www.cs.duke.edu/csed/curious/compression/adaptivehuff.html* - Sivusto, jossa hyvin selitetty ja havainnollistavat kaaviot.