

Ohjelman yleisrakenne

Tällä hetkellä ohjelmassa on kaksi eri tiedostonpakkausalgoritmia, Huffmanin koodaus ja LZW. Ohjelma ja tämä toteutusdokumentti ovat molemmat tällä hetkellä pahasti kesken.

Huffmanin koodauksen olen toteuttanut käyttäen hyväksi hajautustaulua ja prioriteettijonoa. Hajautustaulun teossa on myöskin käytetty ArrayList tietorakennetta. Nämä kaikki olen itse tehnyt ja vain niin laajana että Huffmanin koodaus toimisi.

Huffmanin koodauksessa pakatessa käytetään pääasiassa luokkaa "HuffmanCompression". Ensin käydään teksti läpi ja lisätään kaikkien merkkien esiintyvyydet hajautustauluun. Tästä hajautustaulusta muodostetaan prioriteettijonon avulla Huffmanin puu. Puu käydään läpi ja merkitään jokaista mahdollista merkkiä vastaava Huffmanin koodi hajautustauluun. Tämän jälkeen kirjoitetaan merkkien esiintyvyydet pakatun tiedoston alkuun. Sitten käydään alkuperäinen tiedosto läpi merkki merkiltä ja kirjoitetaan merkkejä vastaavat koodit hajautustaulusta pakattuun tiedostoon. Pakkauksen suoritus on nyt valmis.

Huffmanin koodauksessa purettaessa käytetään pääasiassa luokkaa "HuffmanDecompression". Ensin luetaan pakatun tiedoston alkuosasta merkkien esiintyvyydet hajautustauluun. Tästä hajautustaulusta muodostetaan prioriteettijonon avulla Huffmanin puu, samalla tavalla kuten tiedostoa pakatessa. Tämän jälkeen luetaan pakatun tiedoston loppuosa läpi ja puretaan Huffmanin koodi edellä mainitun puun avulla.

LZW algoritmissa olen toteuttanut käyttäen hyväksi hajautustaulua ja ArrayListiä, jotka olen itse toteuttanut niin laajana että LZW toimisi.

LZW algoritmin pakkaus käyttää luokkaa "LZWCompression". LZW tiedoston pakkauksessa aluksi alustetaan sanakirja hajautustaulu kaikilla 256 ascii merkillä ja niitä vastaavalla koodilla. Sitten tämän jälkeen käydään tiedosto läpi, jos sanakirja sisältää wc (missä c on luettumerkki) niin $w=wc$, muuten koodi ArrayListaan lisätään w vastaava koodi sanakirjasta ja sanakirjaan lisätään wc ja uusi $w = c$. Tämän jälkeen kirjoitetaan koodisto tiedostoon 24 bittiä kerrallaan.

LZW algoritmin purkaus käyttää luokkaa "LZWDecompression". Purkaminen sujuu melkein samalla tavalla kuin pakkaaminen. Ensin luetaan tiedostoa 24 bittiä kerrallaan ja tästä saadaan tätä vastaava koodi joka lisätään ArrayListaan. Seuraavaksi alustetaan sanakirja kuten pakatessakin. Nyt voidaan koodit purkaa sanakirjan avulla.

Saavutetut aika- ja tilavaativuudet (m.m. O-analyysi pseudokoodista)

Teen myöhemmin.

Suorituskyky- ja O-analyysivertailu (mikäli työ vertailupainotteinen)

Suoritus kykyyn liittyen suorituskykytestaus.pdf tiedostossa.

Työn mahdolliset puutteet ja parannusehdotukset

LZW toimii tosi hitaasti isoilla tiedostoilla, eli se pitäisi korjata.

Lähteet

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

<http://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch>

Tosi monta eri stackoverflow sivua liittyen aiheeseen.