Testausdokumentti

1 Oikeellisuus

Pakkaus- ja purkualgoritmeja on testattu antamalla niille syötteeksi (a) käsin muodostettuja pieniä testisyötteitä, (b) ohjelmallisesti generoituja suurempia syötteitä sekä (c) real-world tiedostoja. Syöte annetaan pakkausalgoritmille, jonka antama pakattu versio annetaan heti purkualgoritmille. Purkualgoritmin antaman tuloksen on oltava täsmälleen sama kuin alkuperäinen syöte. Real-world-tiedostoja lukuun ottamatta testit on automatisoitu jUnitilla.

Automaattitestit kattavat käytännössä kaiken; Jacocoverage/Pitest eivät anna 100% kattavuutta, mutta melkein kaikissa tapauksissa se johtuu jommasta kummasta seuraavista:

1. Koodissa on virheentarkistus (index out of bounds tms.), mutta algoritmit eivät aiheuta kyseistä virhettä.
2. Kyseessä on työn kannalta vähemmän tärkeä osa koodia, kuten `main()` tai `Options`-luokan metodit tai lokiviestit.

2 Suorituskyky

Suorituskykyä on testattu eri kokoisilla syötteillä ja verrattu bzip2-ohjelmaan. LZW-x tarkoittaa LZW-koodausta, missä koodisanan maksimikoko on x bittiä (komentoriviparametrilla `-ls x`. Ajastus on tehty `time`-komennon avulla. Lopputuloksen oikeellisuus on varmistettu `diff`-komennolla. Testaus on puoliautomatisoitu `runtest`-skriptillä.

- 537K tekstitiedosto (suomenkielinen lyx-tiedosto)

Algoritmi	Pakattu koko	Pakkausaika	Purkuaika
bzip2	40K (7%)	0s	0s
Huffman	341K (64%)	1s	0s
LZW-9	339K (63%)	1s	2s
LZW-12	163K (30%)	1s	1s
LZW-16	104K (19%)	1s	1s

- 1038K tekstitiedosto (englanninkielinen kirja)

Algoritmi	Pakattu koko	Pakkausaika	Purkuaika
bzip2	280K (27%)	0s	0s
Huffman	604K (58%)	1s	0s
LZW-9	776K (75%)	2s	3s
LZW-12	530K (51%)	1s	1s
LZW-16	420K (40%)	2s	1s

- 13420K tar-tiedosto (`/etc` -hakemiston sisältö)

Algoritmi	Pakattu koko	Pakkausaika	Purkuaika
bzip2	3085K (23%)	5s	1s
Huffman	8915K (66%)	6s	2s
LZW-9	7530K (56%)	16s	16s
LZW-12	6008K (45%)	7s	6s
LZW-16	5315 (40%)	8s	7s

- 25847K mp4-tiedosto

Algoritmi	Pakattu koko	Pakkausaika	Purkuaika
bzip2	20151K (78%)	13s	5s
Huffman	25803K (100%)	17s	7s
LZW-9	28667K (111%)	57s	50s
LZW-12	33881K (131%)	35s	19s
LZW-16	34319K (133%)	36s	38s

3 Hyvät ja huonot syötteet

Taulukoista näkyy, että sekä Huffman että LZW pakkaavat tekstitiedostoja kohtalaisen hyvin. (Pakattu) binääritiedosto taas ei pakkaudu hyvin. Tässä Huffmanin pakkaussuhde on olematon ja LZW:n jopa negatiivinen.

Tekstitiedostoissa toiset merkit ovat toisia yleisempiä, minkä ansiosta Huffman toimii hyvin; sehän nimenomaan antaa yleisimmille merkeille lyhimmät koodisanat. Jos taas syötteessä on kaikkia merkkejä suunnilleen yhtä monta, niin Huffman-puusta tulee tasapainoinen, jolloin jokaisen koodisanan pituus on suunnilleen $\log_2 256 = 8$ bittiä. Koska myös ei-koodatun merkin pituus on 8 bittiä, niin pakkausta ei tapahdu (eikä toisaalta laajenemistakaan).

Tekstitiedostoissa on myös paljon toistoa, joita LZW pystyy hyödyntämään. Nimittäin jos algoritmi törmää merkkijonoon, joka jo on sanastossa, niin tämä merkkijono (joka voi olla 12 bitin koodisanoilla parhaimmillaan n. 4000 merkkiä pitkä) voidaan korvata

yhdellä (tässä esimerkissä 12 bitin) koodisanalla. Kyseinen merkkijono pakkautuu silloin parhaimmillaan n. $12/(4000 \cdot 8) \approx 2\%$ alkuperäisestä. Jos taas tiedostossa ei ole yhtään toistoa, niin jokainen merkki koodataan itsellään. Mutta pienin koodisanan pituus on 9 bittiä, jolloin syöte kasvaa n. $9/8 \approx 113\%$ alkuperäisestä. Tämä vastaa yllä viimeisen taulukon LZW-9 -riviä. Se, että suuremmilla koodisanoilla teho edelleen huononee johtuu siitä, että koodisanan koko kasvaa sitä mukaa kun sanasto täyttyy, jolloin pahimmillaan kasvu on esimerkiksi 12 bitin tapauksessa n. $12/8 = 150\%$ merkkiä kohden.