TRAK II -harjoitustyö: Huffman-koodaus

Simo Heikinheimo (67923 TKT) simo.heikinheimo@utu.fi
Samuel Laurén (500074 TKT) samuel.lauren@utu.fi
Jarko Papalitsas (501250 TKT) jastpa@utu.fi

1. huhtikuuta 2012

Sisältö

Luku 1

Yleistä

1.1 Tehtävän kuvaus ja analyysi

Tehtävänä oli toteuttaa Huffman-koodauksen toteuttava algoritmi ja siihen olellisesti liittyvät tietorakenteet. Ohjelmalle annetaan syötteeksi koodattava tai koodauksen purkua vaativa tiedosto. Ohjelma purkaa tai koodaa tiedoston riippuen vivuista. Myös koodauksen havainnollistaminen on mahdollista tulostusoptiolla. Vivuista enemmään liitteenä olevassa käyttöohjeessa.

1.2 Ratkaisuperiaate

Huffman-koodausta käytetään häviöttömään datan pakkaukseen. Periaatteena on järjestää merkit toistuvuusjärjestykseen prioriteettijonoon. Prioriteettijonossa vähiten toistuvilla merkeillä on suurin prioriteetti. Puun kasaaminen aloitetaan ottamalla prioriteettijonosta aina kaksi pienintä alkiota (suurin prioriteetti) ja yhdistämällä näiden esiintymismäärä luotavaan isäsolmuun. Solmu siirretään takaisin prioriteettijonoon tämän jälkeen. Kun prioriteettijonossa on enää yksi solmu, on Huffman-puu valmis.

Valmiissa Huffman-puussa vain lehtisolmuissa on symboleita. Puussa alkuperäistä symbolia kuvaa polku tähän kyseenomaiseen lehtisolmuun. Reitin yksittäistä suuntaa vasemmalle tai oikealle kuvataan yksittäisellä bitillä. Esimerkiksi vasen = 0 ja oikea = 1. Täten yleisimmin toistuvat merkit ovat lähempänä juurta ja niitä voidaan kuvata vähemmällä määrällä dataa.

Ohjelmassa parannettavaa on tietenkin enemmät testit, muistivuotojen tutkimiset sekä tiedostoformaatin parantelu.

Luku 2

Ohjelman ja sen osien kuvaus

2.1 Prioriteettijono

Tietueet

- struct **Node**

  Prioriteettijonon alkio.
- struct **PriorityQueue**

  Prioriteettijono.

Määrittelyt

- #define **parent**(index) index / 2

  Palauttaa isäsolmun indeksin.
- #define **left**(index) 2 ∗ index + 1

  Palauttaa vasemman lapsisolmun indeksin.
- #define **right**(index) 2 ∗ index + 2

  Palauttaa oikean lapsisolmun indeksin.

Funktiot

- **PriorityQueue** ∗ **PriorityQueue_new** ()

  Alusta uusi prioriteettijono.
- void **PriorityQueue_free** (**PriorityQueue** ∗queue)

  Vapauta prioriteettijonon varaama muisti.
- **Node** ∗ **PriorityQueue_peekHighestPriority** (**PriorityQueue** ∗queue)

Palauttaa jonon korkeimman prioriteetin alkion, jättäen sen kuitenkin edelleen osaksi jonoa.

- **Node** ∗ **PriorityQueue_removeHighestPriority** (**PriorityQueue** ∗queue)

  Poistaa korkeimman prioriteetin alkion jonosta.

- int **PriorityQueue_insert** (**PriorityQueue** ∗queue, **Node** ∗node)

  Lisää alkio jonoon.

- **Node** ∗ **Node_new** (int priority, void ∗data)

  Alustaa uuden alkion.

- void **Node_free** (**Node** ∗node)

  Vapauttaa alkion varaaman muistin.

2.1.1 Määritysten dokumentointi

`#define` **left**`(` *index* `) 2 * index + 1`

Palauttaa vasemman lapsisolmun indeksin.

Parametrit

index	Solmun indeksi.

Palauttaa

Vasemman lapsisolmun indeksi.

`#define` **parent**`(` *index* `) index / 2`

Palauttaa isäsolmun indeksin.

Parametrit

index	Solmun indeksi.

Palauttaa

Isäsolmun indeksi.

`#define` **right**`(` *index* `) 2 * index + 2`

Palauttaa oikean lapsisolmun indeksin.

Parametrit

index	Solmun indeksi.

Palauttaa

Oikean lapsisolmun indeksi.

2.1.2 Funktioiden dokumentaatio

`void` **Node_free** `(` **Node** `*` *node* `)`

Vapauttaa alkion varaaman muistin.

Parametrit

node	Osoitin vapautettavaan alkioon.

Node `*` **Node_new** `(` `int` *priority*, `void` `*` *data* `)`

Alustaa uuden alkion.

Parametrit

priority	alkiolle annettava prioriteetti.
data	Osoitin alkioon talletettavaan dataan.

Palauttaa

Palauttaa osoittimen uuteen alkioon.

`void` **PriorityQueue_free** `(` **PriorityQueue** `*` *queue* `)`

Vapauta prioriteettijonon varaama muisti.

Vapauttaa kaikki prioriteettijonon jäsenet ja jonon itse.

Parametrit

queue	Osoitin vapautettavaan jonoon.

`int` **PriorityQueue_insert** `(` **PriorityQueue** `*` *queue*`,` **Node** `*` *node* `)`

Lisää alkio jonoon.

Parametrit

queue	Osoitin käsiteltävään jonoon.
node	Osoitin lisättävään alkioon.

Palauttaa

Palauttaa nollan onnistuessaan ja nollasta eroavan arvon virhetilanteessa.

PriorityQueue `*` **PriorityQueue_new** `(  )`

Alusta uusi prioriteettijono.

Palauttaa

Osoitin luotuun jonoon.

Node `*` **PriorityQueue_peekHighestPriority** `(` **PriorityQueue** `*` *queue* `)`

Palauttaa jonon korkeimman prioriteetin alkion, jättäen sen kuitenkin edelleen osaksi jonoa.

Parametrit

queue	Osoitin käsiteltävään jonoon.

Palauttaa

Osoitin korkeimman prioriteetin alkioon.

Node `*` **PriorityQueue_removeHighestPriority** `(` **PriorityQueue** `*` *queue* `)`

Poistaa korkeimman prioriteetin alkion jonosta.

Parametrit

queue	Osoitin käsiteltävään jonoon.

Palauttaa

Osoitin korkeimman prioriteetin alkioon.

2.2 Huffman-koodaus

Tietueet

- struct **_TreeNode**
  Solmu.

Määrittelyt

- #define **SYMBOL_TABLE_SIZE** (1 << CHAR_BIT)
  Symbolitaulukon koko.

Tyyppimäärittelyt

- typedef struct **_TreeNode TreeNode**
  Solmu.

Funktiot

- int * **analyzeFrequencies** (unsigned char *buffer, size_t length)
  Analysoi symbolien esiintymistiheyden.
- **TreeNode** * **buildHuffmanTree** (int *frequencies)
  Rakentaa huffman-puun symbolien tiheyksien pohjalta.
- char * **addCharToString** (char *str, char charToAdd, int size)
  Lisää merkkijonoon(str) merkin(charToAdd) ja uuden merkkijonon loppuun '\0'-merkin.
- int **printCodebook** (**TreeNode** *root, char *codeWord, int counter)
  Tulostaa koodikirjan syötteeksi annetusta Huffmanpuusta.
- **TreeNode** * **TreeNode_new** (unsigned char data)
  Alustaa uuden solmun.
- void **TreeNode_free** (**TreeNode** *node)
  Vapauttaa solmun varaaman muistin.

2.2.1 Määritysten dokumentointi

`#define` **SYMBOL_TABLE_SIZE** `(1 << CHAR_BIT)`

Symbolitaulukon koko.

2.2.2 Tyyppimääritysten dokumentaatio

`typedef struct` **_TreeNode TreeNode**

Solmu.

2.2.3 Funktioiden dokumentaatio

`char *` **addCharToString** `( char * str, char charToAdd, int size )`

Lisää merkkijonoon(str) merkin(charToAdd) ja uuden merkkijonon loppuun '\0'-merkin.

Parametrit

str	Viittaus merkkijonoon jonka loppuun merkki halutaan lisätä.
charToAdd	Merkki joka halutaan lisätä merkkijonoon.
size	Sen merkkijonon koko johon uusi merkki lisätään. (Merkkien määrä merkkijonossa. Myös '\0'-merkki lasketaan mukaan!)

Palauttaa

Viittaus uuteen merkkijonoon joka sisältää vanhan merkkijonon + uuden merkin + '\0'-merkin.

`int *` **analyzeFrequencies** `( unsigned char * buffer, size_t length )`

Analysoi symbolien esiintymistiheyden.

Parametrit

buffer	Osoitin analysoitavaan dataan.
length	Analysoitavan datan pituus tavuina.

Palauttaa

Osoitin symbolien tiheyksiin.

TreeNode * **buildHuffmanTree** (`int` * *frequencies*)

Rakentaa huffman-puun symbolien tiheyksien pohjalta.

Parametrit

frequencies	Osoitin symbolien tiheystaulukkoon.

Palauttaa

Osoitin puun juureen.

`int` **printCodebook** (**TreeNode** * *root*, `char` * *codeWord*, `int` *counter*)

Tulostaa koodikirjan syötteeksi annetusta Huffmanpuusta.

Parametrit

root	Viittaus Huffmanpuun juurisolmuun.
codeWord	Viittaus edeltäviin "suuntiin"(ts.koodisanan alkuosaan) matkalla juuresta lehteen (= koodattava symboli).
counter	counter. Monesko metodin kutsukerta on meneillään? [default=1]

Palauttaa

0 jos kaikki on mennyt hyvin.

`void` **TreeNode_free** (**TreeNode** * *node*)

Vapauttaa solmun varaaman muistin.

Parametrit

node	Vapautettava solmu.

TreeNode ∗ **TreeNode_new** (`unsigned char` *`data`*)

Alustaa uuden solmun.

Parametrit

data	Solmuun talletettava data.

Palauttaa

Osoitin alustettuun solmuun.

2.3 Tiedostonkäsittely

Funktiot

- unsigned char ∗ **readFile** (char ∗path, size_t ∗file_length, long int startPos)

  Lukee tiedoston muistiin aloittaen määritellystä kohdasta.
- int **encode** (**CodeWord** ∗∗codebook, unsigned char ∗readBuffer, size_t readBSize, char ∗writePath)

  Enkoodaa puskurin sisällön ja tallettaa enkoodatun datan tiedostoon.
- int **decode** (unsigned char ∗readBuffer, size_t readBSize, char ∗writePath, **TreeNode** ∗root)

  Dekoodaa puskurin sisällön ja tallettaa dekoodatun datan tiedostoon.
- int **writeFreqsToFile** (char ∗path, int ∗freqs)

  Kirjoittaa symbolien esiintymistiheydet tiedostoon.
- int ∗ **readFreqsFromFile** (char ∗path, long int ∗endpos)

  Lukee tiedostosta symbolien esiintymistiheydet.

Muuttujat

- const unsigned char **BITANDLIST** [] = { 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01 }

  Apulista jota käytetään tietyn bitin "korostamisessa". (Eli muut bitit nollataan).

2.3.1 Funktioiden dokumentaatio

`int` **decode** `(` `unsigned char *` *readBuffer*, `size_t` *readBSize*, `char *` *writePath*, **TreeNode** `*` *root* `)`

Dekoodaa puskurin sisällön ja tallettaa dekoodatun datan tiedostoon.

Parametrit

readBuffer	Osoitin puskuriin jonka sisältö dekoodataan.
readBSize	Dekoodattavan puskurin koko tavuina.
writePath	Osoitin merkkijonoon joka sisältää sen tiedoston nimen
root	Osoitin huffman-koodipuun juurisolmuun.

Palauttaa

0 jos kaikki menee hyvin. < 0 jos virheitä.

`int` **encode** `(` **CodeWord** `**` *codebook*, `unsigned char *` *readBuffer*, `size_t` *readBSize*, `char *` *writePath* `)`

Enkoodaa puskurin sisällön ja tallettaa enkoodatun datan tiedostoon.

Parametrit

codebook	Koodikirja
readBuffer	Osoitin bufferiin jonka sisältö encoodataan.
readBSize	Enkoodattavan puskurin koko tavuina.
writePath	Osoitin merkkijonoon joka sisältää sen tiedoston nimen.

Palauttaa

0 jos kaikki menee hyvin. < 0 jos virheitä.

`unsigned char *` **readFile** `(` `char *` *path*, `size_t *` *file_length*, `long int` *startPos* `)`

Lukee tiedoston muistiin aloittaen määritellystä kohdasta.

Parametrit

path	Luettavan tiedoston polku

file_ length	Osoitin muuttujaan johon talletetaan luetun tiedoston pituus tavuina.
startPos	Sijainti josta lukeminen aloitetaan.

Palauttaa

Osoitin luettuun dataan.

`int * **readFreqsFromFile** ( char * path, long int * endpos )`

Lukee tiedostosta symbolien esiintymistiheydet.

Parametrit

path	Osoitin merkkijonoon joka sisältää luettavan tiedoston nimen (ja polun).
endpos	Osoitin kokonaislukumuuttujaan johon talletetaan tieto siitä missä kohtaa tiedostoa symbolien esiintymistiheydet on luettu ja varsinainen koodattu osuus alkaa.

Palauttaa

Osoitin symbolien tiheyksiin. NULL jos virheitä luvun aikana.

`int **writeFreqsToFile** ( char * path, int * freqs )`

Kirjoittaa symbolien esiintymistiheydet tiedostoon.

Parametrit

path	Osoitin merkkijonoon joka sisältää sen tiedoston nimen (ja polun) johon kirjoitetaan.
freqs	Osoitin symbolien tiheyksiin.

Palauttaa

>= 0 tallennettujen symbol + esiintymistiheys parien lukumäärä. < 0 Jos virhe.

2.3.2 Muuttujien dokumentaatio

`const unsigned char` **BITANDLIST**`[] = { 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01 }`

Apulista jota käytetään tietyn bitin "korostamisessa". (Eli muut bitit nollataan).

2.4 Koodikirjan käsittely

Tietueet

- struct **CodeWord**

  Koodisana.

Määrittelyt

- #define **byteAsBit**(bytes) bytes ∗ CHAR_BIT

  Montako bittiä 'bytes' tavua on.
- #define **bitAsByte**(bits) ((bits - 1) / CHAR_BIT) + 1

  Montako tavua 'bits' bittiä on.
- #define **bitAsIndex**(bits) (bits - 1) / CHAR_BIT

  bit muutettuna tavu-indeksiksi.

Funktiot

- **CodeWord** ∗∗ **Codebook_new** ()

  Alustaa uuden koodikirjan.
- void **Codebook_free** (**CodeWord** ∗∗codebook)

  Vapauttaa koodikirjan varamaan muistin.
- void **fixBitsToHighest** (unsigned char ∗byteToFix, size_t size)

  Korjaa koodisanan viimmeisen tavun bitit alkamaan most sig bitistä.
- **CodeWord** ∗ **CodeWord_new** (**CodeWord** ∗∗codebook, unsigned char symbol, void ∗codeword, size_t size)

  Luo uuden structCodeWord:in.

- **CodeWord** ∗ **CodeWord_copy** (**CodeWord** ∗codeword)

  Kopioi structCodeWord:iksi.
- int **removeBitsFromCodeWord** (**CodeWord** ∗codeword, size_t number)

  Poistaa structCodeWord:in koodisanan alusta(most significant bit) 'number' määrän bittejä.
- unsigned char ∗ **addBitToCodeWord** (unsigned char ∗codeWord, unsigned char bit, int size)

  Lisää koodisanan loppuun(least significant bit) yhden uuden bitin.
- int **Codebook_create** (**CodeWord** ∗∗codebook, **TreeNode** ∗root, unsigned char ∗codeWord, int counter)

  Luo koodikirjan syötteeksi annetusta Huffmanpuusta.

2.4.1 Määritysten dokumentointi

`#define` **bitAsByte**`(` *bits* `) ((bits - 1) / CHAR_BIT) + 1`

Montako tavua 'bits' bittiä on.

`#define` **bitAsIndex**`(` *bits* `) (bits - 1) / CHAR_BIT`

bit muutettuna tavu-indeksiksi.

`#define` **byteAsBit**`(` *bytes* `) bytes ∗ CHAR_BIT`

Montako bittiä 'bytes' tavua on.

2.4.2 Funktioiden dokumentaatio

`unsigned char ∗` **addBitToCodeWord** `( unsigned char ∗` *codeWord,* `unsigned char` *bit,* `int` *size* `)`

Lisää koodisanan loppuun(least significant bit) yhden uuden bitin.

Käytännössä kopioi koodisanan uudeksi koodisanaksi ja lisää uuden kopion loppuun bitin. Palauttaa uuden koodisanan.

Parametrit

codeWord	Osoitin koodisanaan johon bitti lisätään.
bit	Bitin arvo joka koodisanaan lisätään.
size	Koodisanan pituus bitteinä ennen bitin lisäystä.

Palauttaa

Osoitin kopioon 'codeWord':sta, jonka loppuun on lisätty bitti 'bit'.

`int` **Codebook_create** `(` **CodeWord** `**` *codebook*, **TreeNode** `*` *root*, `unsigned char *` *codeWord*, `int` *counter* `)`

Luo koodikirjan syötteeksi annetusta Huffmanpuusta.

Parametrit

codebook	koodikirja
root	Osoitin Huffmanpuun juurisolmuun.
codeWord	Osoitin edeltäviin "suuntiin"(ts.koodisanan alkuosaan) matkalla juuresta lehteen (= koodattava symboli).
counter	counter. Monesko metodin kutsukerta on meneillään? [default=1]

Palauttaa

0 jos kaikki on mennyt hyvin. -1 jos juurisolmua ei löydy.

`void` **Codebook_free** `(` **CodeWord** `**` *codebook* `)`

Vapauttaa koodikirjan varamaan muistin.

Parametrit

codebook	koodikirja

CodeWord `**` **Codebook_new** `( )`

Alustaa uuden koodikirjan.

Palauttaa

Osoitin alustettuun koodikirjaan.

CodeWord `*` **CodeWord_copy** `(` **CodeWord** `*` *codeword* `)`

Kopioi structCodeWord:iksi.

(Voisi olla nimeltään myös clone)

Parametrit

codeword	Osoitin structCodeWord:iksi.

Palauttaa

Osoitin structCodeWord:in kopioon.

CodeWord * **CodeWord_new** (**CodeWord** ** *codebook*, unsigned char *symbol*, void * *codeword*, size_t *size*)

Luo uuden structCodeWord:in.

Asettaa CodeWordin codebook:iin symbol:in structCodeWord:iksi. codebook koodikirja

Parametrit

symbol	Symboli jonka structCodeWord on kyseessä.
codeword	Osoitin uuden structCodeWord:in varsinaiseen koodisanaan.
size	koodisanan pituus bitteinä.

Palauttaa

Osoitin uuteen luotuun structCodeWord:iin.

void **fixBitsToHighest** (unsigned char * *byteToFix*, size_t *size*)

Korjaa koodisanan viimmeisen tavun bitit alkamaan most sig bitistä.

Parametrit

byteToFix	Osoitin koodisanaan, jonka viimeinen tavu korjataan.
size	Korjattavan koodisanan koko bitteinä.

int **removeBitsFromCodeWord** (**CodeWord** * *codeword*, size_t *number*)

Poistaa structCodeWord:in koodisanan alusta(most significant bit) 'number' määrän bittejä.

Muutokset tehdään suoraan parametrinä saatuun structCodeWord * muut-

tujaan. Päivittää myös koodisanan koon oikeaksi.

Parametrit

codeword	Osoitin structCodeWord:iin jonka koodisanan alusta bitit poistetaan.
number	Lukumäärä montako bittiä poistetaan.

Palauttaa

Poistettujen bittien lukumäärän jos kaikki menee hyvin. -1 Jos virhe.

Luku 3

Testausjärjestely

Testiohjelma löytyy tiedostosta `test.c`. Testiohjelma testaa itse Huffman-puun generointia kuin prioriteettijononkin toimintaa muutamilla testialkioilla. Prioriteettijonon tapauksessa tulostetaan suurimman prioriteetin omaava alkio jonka jälkeen sitä pienemmän ja niin edelleen. Näitä arvoja täytyy verrata annettuihin.

Huffman-puun tapauksessa lause ”This is an example sentence. Hello world.” Ajetaan puun läpi ja tämän jälkeen tulostetaan koodikirja. Huffman-puun tuottamaa koodikirjaa pystyy myös tarkastelemaan omalla tiedostosyötteellä vivun -p avulla, josta lisää käyttöohjeosiossa.

Liite A

Ohjelmalistaus

Ohjelmakoodi löytyy samasta paketista tämän dokumentaation kanssa, joten sitä ei sisällytetty tähän dokumenttiin.

Liite B

Käyttöohje

Mukana tuleva käännös on tehty 64-bittiselle Linux-järjestelmälle, mutta koodin kääntöä on myös testattu Windowsin alla MinGW:n avulla. Käännösskripti on hoidettu Make:n avulla ja testikäännökset gcc:llä, joten jotakin Unix-pohjaista järjestelmää suositellaan. Tehdäksesi oman käännöksen ohjelmasta testiohjelman kera, komenna

```
make all test
```

ohjelman pääkansiossa.

```
make clean
```

puhdistaa.

Ohjelmalle voi syöttää tiedoston -i (input) -vivulla ja ulostulotiedoston nimen voi määrittää -o (output) -vivulla. Vivut -e (encode) ja -d (decode) kertovat ollaanko tiedostoa koodaamassa vai purkamassa. Lisäksi demonstraatiotarkoituksia varten ohjelmassa on tulostusvipu -p (print), jolla voidaan käskeä ohjelmaa tulostamaan merkkien koodatut binääriesitykset näytölle.