HuffmanCoding-projektin dokumentaatio

Määrittelydokumentti

Ohjelmani on tiedostonpakkausohjelma, joka on toteutettu Huffman koodauksella. Tietorakenteena käytetään minimikekoa, kun muodostetaan Huffmanin puu, jossa lehtinä ovat tiedostossa esiintyvät erilaiset tavut.

Uusien koodien muodostamisen aikavaativuus on O(n * log n), missä n on tiedoston erilaisten tavujen määrä. Jokainen tavu lisätään solmuna minimikekoon ja poistetaan sieltä. Alkuperäisen (pakatun) tiedoston lukeminen tapahtuu ajassa O(m), missä m on tiedoston tavujen määrä. Pakkauksessa tiedoston kirjoittaminen tapahtuu ajassa O(m * k), missä k on pisimmän uuden bittiesityksen pituus (eli puun korkeus). Purkamisen aikavaativuudet ovat samat kuin pakkauksen (tavujen etsiminen tapahtuu siis ajassa O(m * k)). Lisäksi purettaessa käytetään aikavaativuutta O(8 * h), missä h on purettavien tavujen määrä.

Pakattava tiedosto tallennetaan kokonaisuudessaan boolean[][]-taulukkoon, jolloin tilavaativuus on O(2 * m), sillä taulukkoon luodaan ylimääräistä tilaa, jottei jouduta kopioimaan taulukkoa uudestaan isompaan taulukkoon ylivuodon sattuessa. Purkamisen tilavaativuus on O(8 * h), sillä bittitaulukko on tallennettu boolean[]-taulukkoon kokonaisuudessaan ennen muuntamista tavuiksi.

Tarkka lähdeluettelo on dokumentin lopussa.

Toteutusdokumentti

Toteutukseni ratkaisee ongelman määrittelyssä esitetyllä tehokkuudella. Pseudokoodia perusteluiksi (tilavaativus perusteltu määrittelyssä):

```
pakkaaminen(){
    luetaanTiedostopolkuJaTiedosto()
    lasketaanFrekvenssit() //aikavaativuus O(m)
    muodostetaanMinimikekoJaPuu() //aikavaativuus O(n * log n)
    muodostetaanUudetKoodit() //puun läpikäynti O(v), v solmujen määrä
    kirjoitetaanUudetTavut() //boolean[][]-taulukko, aikavaativuus O(m * k)
    kirjoitetaanLopullinenPakattuTiedosto() }
```

```
purkaminen(){
    luetaanTiedostopolkuJaPakattuTiedosto()
    muodostetaanMinimikekoJaPuu() // aikavaativuus O(n * log n)
    muodostetaanBittiesitys() //aikavaativuus O(8 * h)
    muodostetaanTavutUudestaan() //aikavaativuus O(m *k)
    kirjoitetaanUudestaanAlkuperäinenTiedosto() }
```

Puutteita toteutuksessani on liian suuri tilavaativuus, joka seuraa siitä, että uudet tiedostot tallennetaan ensin kokonaisuudessaan taulukkoon ennen kirjoittamista. Puutteen voisi korjata tallentamalla tiedostoja yksi tavu kerrallaan, jolloin vain 8 bitin kokoinen taulukko olisi muistissa. Puute on myös Windowsin command promptista .jar-tiedostoa ajettaessa se, ettei ääkkösiä sisältäviä tiedostopolkuja tai tiedostonimiä voida avata. Ubuntussa tämän pystyy kiertämään terminaalissa, mutta Netbeansin kautta ääkkösnimet eivät Ubuntussa taas toimi.

Testausdokumentti

Ohjelmassani on automaattinen testaus toteutettuna JUnit testeillä. Testejä ajettiin erilaisilla tapauskohtaisilla testisyötteillä, jotka vastaisivat oikean tiedoston antamaa tietoa laadullisesti. Testisyötteet eivät vastanneet pituudessaan ”aitoja” tiedostoja, sillä testeissä ei ole käytetty kaikkia tavuja sisältäviä syötteitä. Ohjelmaa on paras testata käytännössä esimerkkitiedostoilla, joilla pakatun ja alkuperäisen tiedoston kokoa pääsee vertailemaan. Lisäksi purkamisen jälkeen on hyvä testata avaamalla purettu tiedosto, ettei se ole korruptoitunut. Pakkaaminen ja purkaminen korvaavat kaikki samannimiset alkuperäiset tiedostot siinä kansiossa, johon pakkaaminen ja purkaminen suoritetaan.

README

Ohjelman voi ajaa Windowsissa .jar-tiedostosta tiedoston sisältävästä kansiosta komennolla *java -jar HuffmanCoding.jar* (tällöin ei kuitenkaan pysty käyttämään ääkkösiä sisältäviä tiedostopolkuja). Ubuntussa tiedoston voi ajaa komennolla *java -jar -Dfile.encoding=UTF-8 HuffmanCoding.jar* (tällöin toimivat myös ääkköset).

Ohjelma kysyy ensiksi, halutaanko tiedostoja pakata vai purkaa. Sen jälkeen ohjelma pyytää antamaan pakattavan tai purettavan tiedoston tiedostopolun. Ohjelma ilmoittaa onnistuneesta pakkaamisesta tai purkamisesta tai mahdollisista virhetilanteista.

Esimerkkisyöte pakkaukseen voisi olla (oletetaan että testi.txt -tiedosto on olemassa): ”pakkaus[enter]C:\Users\TestiTaavo\Tirala\Dokumentointikansio\testi.txt[enter]”.

Esimerkkisyöte purkamiseen voisi olla (oletetaan että pakattu tiedosto pakattutesti.txt.ep on olemassa):
”purku[enter]C:\Users\TestiTaavo\Tirala\Dokumentointikansio\pakattutesti.txt.ep[enter]”.

Lähteitä

http://www.cs.helsinki.fi/u/ejunttil/opetus/tiraharjoitus/ Idealähde tehtävään.

http://www.cs.helsinki.fi/u/ejunttil/opetus/tiraharjoitus/bittiohje.txt Bittikäsittelyohje, lainattu koodia byteToBits()- ja bitsToByte()-metodeihin.

http://en.wikipedia.org/wiki/Huffman_coding Tietoja Huffman koodauksesta.

http://www.programcreek.com/2009/02/java-convert-a-file-to-byte-array-then-convert-byte-array-to-a-file/ Tiedoston lukeminen byte array:ksi.

http://people.cs.nctu.edu.tw/~cjtsai/courses/imc/classnotes/imc12_03_Huffman.pdf Tietoja Huffman koodauksesta.

http://www.mkyong.com/java/how-to-convert-array-of-bytes-into-file/ Tiedoston kirjoittaminen.

http://www.sfu.ca/~jiel/courses/861/pdf/03_Huffman.pdf Tietoja Huffman koodauksesta.

http://dzone.com/snippets/convert-int-byte-array Int-muunnos byte[]:ksi, lainattu koodia intToByteArray()- ja byteArrayToInt()-metodeista.