58131 Tietorakenteet ja algoritmit (kevät 2015)

Harjoitus 5 (9.–13.2.2015)

1. Otetaan aluksi tyhjä hajautustaulu, jonka koko m on 11 ja hajautusfunktio $h(k) = k \bmod m$. Taulukkoon lisätään avaimet 23, 16, 82, 36, 13 ja 45 tässä järjestyksessä.

   (a) Piirrä ja selitä, miten taulukko täyttyy, kun yhteentörmäykset käsitellään ylivuotoketjuilla.

   (b) Piirrä ja selitä, miten taulukko täyttyy avoimessa hajautuksessa, kun yhteentörmäykset käsitellään lineaarisella kokeilulla ja $h' = h$.

   (c) Kuten edellä, mutta yhteentörmäykset käsitellään neliöisellä kokeilulla, missä $h' = h$, $c_1 = c_2 = 1$. Mitä huomioita sinulla on kokeilujonosta?

   (d) Kuten edellä, mutta yhteentörmäykset käsitellään kaksoishajautuksella $h'(k) = k \bmod m$ ja $h''(k) = 1 + (k \bmod (m-2))$.

2. Tavoitteena on luoda hajatustaulu yliopiston opiskelijoista avaimena opiskelijanumero. Helsingin yliopistossa on noin 40000 opiskelijaa.

   Yhteentörmäykset käsitellään ylivuotoketjuilla, joiden pituuden oletusarvon toivotaan olevan 1,0:n ja 1,5:n välillä. Ehdota perusteluineen sopivaa hajautustaulun kokoa m, kun käytetään jakojäännösmenetelmää.

3. Tiedostossa

   `http://www.cs.helsinki.fi/u/jkivinen/opetus/tira/k15/vocabulary.txt`

   on 61188 merkkijonoa, kukin omalla rivillään. (Merkkijonot ovat lähinnä englanninkielisiä sanoja, mutta mukana on myös lyhenteitä yms. verkkokeskusteluista löydettyjä merkkijonoja. Tiedosto on osa *20 Newsgroups* -aineistoa, jota käytetään yleisesti mm. tekstinluokittelualgoritmien testaamiseen. Aineiston sisältöä ei tässä tehtävässä tarvitse ymmärtää, mutta asiasta kiinnostuneet voivat tutustua siihen osoitteessa `http://qwone.com/~jason/20Newsgroups/`.) Käytämme sitä nyt hajautusfunktioiden tutkimiseen.

   (a) Ota selvää, miten Javan `String`-luokan `hashCode`-metodi toimii. Tee Java-ohjelma, joka tarkastelee, kuinka pitkiä ylivuotoketjuja syntyy, jos kyseiset $n = 61188$ merkkijonoa muunnetaan kokonaisluvuiksi tällä metodilla ja hajautetaan jakolaskumenetelmällä hajautustaulun kokona $m = 49157$, ja yhteentörmäykset siis käsitellään ylivuotoketjuilla. (Millä perusteella tämä saattaisi olla sopiva m?) Toista koe arvolla $m = 24593$.

   Huomaa, että tehtävässä ei pyydetä toteuttamaan hajautustaulua ylivuotolistoineen jne. Sinun tarvitsee vain laskea hajautusfunktion arvoja ja pitää kirjaa, kuinka monta kertaa kukin arvo esiintyy.

   Kiinnostavia mittoja tarkasteltavaksi ovat esim. pisimmän ylivuotoketjun pituus, pituusraja jonka ylittää vain 1% ylivuotoketjuista, ja pituusraja jonka ylittää vain 5% ylivuotoketjuista.

Laske myös, kuinka pitkään ylivuotoketjuun hajautustauluun talletettavat merkkijonot keskimäärin päätyvät. Jos merkkijonot ovat $x_1, \ldots, x_n$, merkkijonon x hajautusfunktion arvo on $h(x)$, ja $s(j)$ on kotiosoitetta j vastaavan ylivuotoketjun pituus kun $0 \leq j \leq m-1$, tämä suure on siis

$$\frac{1}{n} \sum_{i=1}^{n} s(h(x_i)).$$

Huomaa, että tämä ei ole sama kuin listojen pituuksien keskiarvo, koska pitkissä listoissa on enemmän alkioita ja ne saavat siten suuremman painon keskiarvon laskemisessa.

(b) Vertailun vuoksi tarkastele samaan tapaan kuin (a)-kohdassa tilastoja yhteentörmäysten määrästä, jos samoilla parametrien n ja m arvoilla sijoitettaisiin n alkiota m-paikkaiseen taulukkoon valiten sijainnit täysin satunnaisesti tasaisen jakauman mukaan. Tällainen arpominen ei tietenkään tule kysymykseen hajautustaulun toteutuksessa, mutta se antaa arvion siitä, millaisia tuloksia erittäin hyvä hajautusfunktio voisi ihannetilanteessa saavuttaa.

(c) Toista vielä (a)-kohdan koe niin, että `hashCode`-metodin sijaan muunnat merkkijonot kokonaisluvuiksi jollain vähemmän harkitulla tavalla, esim. luentojen sivulla 223 esitettyyn tapaan niin, että kustakin merkkijonosta lasketaan mukaan vain viisi merkkiä: kaksi ensimmäistä, kaksi viimeistä sekä keskimmäinen. (Jos merkkejä on alle viisi tai parillinen määrä, sovella oman harkintasi mukaan.) Onko tuloksissa merkittävää eroa verrattuna (a)-kohtaan?

4. Samassa hakemistossa tehtävän 3 tiedoston `vocabulary.txt` kanssa on myös `vocabulary2.txt`, joka on saatu siitä poistamalla yksi rivi ja vaihtamalla rivien järjestystä. Tehtävänä on laatia Java-ohjelma, joka ensin lukee kummankin tiedoston omaan merkkijonotaulukkoonsa ja selvittää sen jälkeen kolmella eri menetelmällä, mikä on poisjätetty merkkijono:

   (a) Käy `vocabulary`-taulukon merkkijonoja yksitellen läpi. Etsi merkkijono `vocabulary2`-taulukosta käymällä sitä alkio kerrallaan läpi. Jos ei löytynyt, tämä on haluttu merkkijono.

   (b) Talleta `vocabulary2`-taulukon alkiot hajautustauluun käyttäen Javan luokkaa `hashMap`. Käy `vocabulary`-taulukon merkkijonoja yksitellen läpi, kunnes löydät sellaisen, joka ei ole hajautustaulussa.

   (c) Järjestä kumpikin taulukko ja etsi ensimmäinen kohta, jossa ne poikkeavat.

   Vertaile ratkaisujen suoritusaikoja.

5. On olemassa tasan yksi binääripuu (ei hakupuu), jonka alkiot ovat

   - esijärjestyksessä 8, 15, 6, 9, 7, 1, 4, 5, 11, 13
   - sisäjärjestyksessä 6, 15, 7, 9, 1, 8, 11, 5, 13, 4.

   Piirrä tämä puu.

6. On annettu binääripuu, jonka kussakin solmussa on jokin kokonaisluku. Sama kokonaisluku voi esiintyä useita kertoja, eikä puu ole välttämättä hakupuu.

(a) Anna algoritmi, joka laskee, kuinka monta kertaa syötteenä annettu luku esiintyy puussa.

(b) Anna algoritmi, joka kertoo, millä syvyydellä annettu luku esiintyy puussa. Jos luku ei esiinny puussa lainkaan, palauta -1. Jos luku esiintyy useita kertoja, palauta pienin syvyys, jolla luku esiintyy.