Tietorakenteet, laskuharjoitus 2, 25.-29.1

Huom: Ensimmäisen kierroksen TRAKLA2-tehtävien deadline on 31.1. Deadlinen jälkeen vastatuista tehtävistä saa ainoastaan puolet pisteistä.

1. Erään algoritmin suoritus vie 1 ms, kun syötteen koko on $n = 200$. Kuinka kauan suoritus kestää, kun syötteen koko on 5000 ja algoritmin aikavaativuus on suuruusluokkaa

(a) $O(\log n)$

(b) $O(n)$

(c) $O(n \log n)$

(d) $O(n^2)$

(e) $O(2^n)$?

Kussakin tapauksessa oleta, että alemman kertaluvun termit aikavaativuudessa voidaan jättää huomiotta; ts. esim. kohdassa (d) oleta, että aikavaativuus on cn^2 jollain vakiolla c.

2. Kuten edellisessä tehtävässä, algoritmin suoritus vie 1 ms, kun syötteen koko on 200, ja tarkastellaan samoja mahdollisia aikavaativuusluokkia (a)–(e). Mikä on kussakin tapauksessa suurin syötteen koko, jolla algoritmin suoritus vie korkeintaan yhden minuutin?

3. Mitkä seuraavista väitteistä ovat tosia, mitkä eivät? Perustele vastauksesi.

(a) $2n^2 + 7n + 3 = O(n^3)$

(b) $2n^3 + 7n + 3 = O(n^2)$

(c) $\log n = O(10^6)$

(d) $10^6 = O(1)$

(e) $\log(n^2) = O(\log n)$

Huom: väittämän $f \neq O(g)$ todistaminen kannattanee tehdä laskarista 1 tutulla vastaväitetekniikalla.

Vastaväitetodistus eteni seuraavasti. Kun halutaan todistaa, että väittämästä A seuraa B, toimitaan siten, että näytetään että A:n ja negaatio B:n yhtäaikainen olemassaolo johtaa ristiriitaan.

Haluamme todistaa, että $f \neq O(g)$. Nyt osaa A, eli oletusta ei ole, eli todistettavana on, että "olemattomasta" oletuksesta seuraa $f \neq O(g)$. Vastaväitetodistus tehdään siten, että oletetaan että "olematon" oletus ja väittämän negaatio, eli $f = O(g)$ ovat voimassa yhtäaikaa ja näytetään että tämä johtaa ristiriitaan. Todistuksen tekemisen kannalta on tietysti olennaista käyttää hyväkseen O:n määritelmää.

4. Seuraava algoritmi laskee summan $1 + 2 + 3 + \ldots + n$:

```
summa = 0
for i = 1 to n
    summa = summa + i
```

Määritä algoritmin aikavaativuus ja tilavaativuus.

Mikä on for-lauseen invariantti? Perustele invariantin avulla, että algoritmi toimii oikein.

5. Seuraava algoritmi laskee polynomifunktion $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_kx^k$ arvon, kun kertoimet $a_0, \ldots, a_k$ on talletettu taulukkoon $A[0 \ldots k]$:

Evaluate-Polynomial(A,x)

```
1  s = 0
2  for i = 0 to k
3      z = 1
4      for j = 1 to i
5          z = z * x
6      s = s+A[i] * z
7  return s
```

Simuloi algoritmin toimintaa funktiolla $p(x) = 2x^3 - 3x^2 - 7$ kun $x = 2$

Määritä algoritmin aikavaativuus. Oleta aritmeettisten operaatioiden sujuvan vakioajassa.

Mikä on ulomman for-silmukan invariantti?

6. Tee vähintään toinen seuraavista. Laskareissa käsitellään molemmat, mutta rastiin riittää joko (a) tai (b).

(a) Taulukko, jonka koko on $n - 1$ sisältää kaikki kokonaisluvut väliltä $1 \ldots n$ yhtä lukuun ottamatta. Jokainen luku esiintyy taulukossa vain kerran. Esitä algoritmi, joka selvittää taulukosta puuttuvan luvun. Algoritmin täytyy toimia lineaarisessa ajassa, ja se saa käyttää vain vakiomäärän muistia taulukon lisäksi. Lineaarisessa ajassa toimiva algoritmi siis ei saa sisältää kahta sisäkkäistä looppia. Peräkkäiset loopit ovat sallittuja.

Esimerkiksi seuraavassa taulukossa puuttuva luku on 6.

2	3	7	1	4	8	5

(b) Tehtävänä on suunnitella tietorakenne, joka sisältää n peräkkäin asetettua lukua (ks. alla oleva esimerkki). Tietorakenteen muodostuksen jälkeen ainoa tarvittava operaatio on $osasumma(a, b)$, joka ilmoittaa peräkkäin olevien lukujen summan kohdasta a kohtaan b. Yksi ratkaisu on tallentaa luvut tavallisesti taulukkoon,

mutta tällöin summan laskussa täytyy käydä läpi kaikki summaan kuuluvat luvut eli operaation *osasumma* aikavaativuus on $O(n)$. Suunnittele tietorakenne, jossa operaation *osasumma* aikavaativuus on vain $O(1)$. Operaation käyttämien komentojen määrän siis täytyy pysyä samana taulukon pituudesta riippumatta. Esimerkiksi seuraavassa taulukossa $osasumma(3, 6)$ antaa tuloksen 4, koska $-1 + 3 + 0 + 2 = 4$.

2	5	-1	3	0	2	8