58131 Tietorakenteet ja algoritmit (kevät 2015)

Harjoitus 2 (19.–23.1.2015)

1. Erään algoritmin suoritus vie 1 ms, kun syötteen koko on $n = 20$. Kuinka kauan suoritus kestää, kun syötteen koko on 5000 ja algoritmin aikavaativuus on suuruusluokkaa

   (a) $\Theta(\log n)$
   (b) $\Theta(n \log n)$
   (c) $\Theta(n^2)$
   (d) $\Theta(2^n)$?

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

2. Mitkä seuraavista väittämistä ovat tosia ja mitkä ovat epätosia? Todista.

   (a) $5n^3 + 7n + 5 = O(n^4)$
   (b) $5n^3 + 7n + 5 = \Theta(n^4)$
   (c) $5n^3 + 7n + 5 = \Omega(n^3)$
   (d) $3n \log n + 2n = O(n^2)$
   (e) $\log n = O(17)$
   (f) $\log(n^3) = \Theta(\log n)$
   (g) $\log(n^3) = \Omega((\log n)^3)$
   (h) $3^n = O(2^n)$

3. Tarkastelemme joidenkin luentomateriaalissa esitettyjen funktioden laskemista rekursiivisesti ja iteratiivisesti

   (a) Sivulla 24 on annettu palautuskaava kertomalle $n!$. Muodosta tästä rekursioon perustuva algoritmi. Määritä algoritmin aika- ja tilavaativuus.
   (b) Esitä kertoman laskemiseen iteratiivinen algoritmi. Mikä on siinä esiintyvän silmukan invariantti? Perustele invariantin avulla, että algoritmi toimii oikein. Määritä algoritmin aika- ja tilavaativuus.

4. Potenssi x^n, missä $x \geq 0$ ja $n \in \mathbb{N}$, voidaan laskea myös palautuskaavalla

$$x^n = \begin{cases} 1 & \text{jos } n = 0 \\ (x^{n/2})^2 & \text{jos } n > 0 \text{ ja } n \text{ on parillinen} \\ x \cdot x^{n-1} & \text{jos } n \text{ on pariton.} \end{cases}$$

   Esitä tähän perustuva rekursiivinen Java-metodi potenssin x^n laskemiseen. Mikä on algoritmin aika- ja tilavaativuus?

   Vapaaehtoinen jatkotehtävä (voit merkitä tehtävän tehdyksi, vaikka et olisi tehnyt tätä osaa): Muunna edellä saamasi rekursiivinen metodi iteratiiviseksi. Mikä nyt on sen aika- ja tilavaativuus?

5. Binomikerroin $\binom{n}{k}$ kertoo mm. kuinka monella tavalla n-alkioisesta joukosta voidaan valita k-alkioinen osajoukko. Binomikertoimet noudattavat tunnetusti palautuskaavaa

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \qquad \text{kun } n \in \mathbb{N} \text{ ja } k \in \mathbb{N} \text{ ja } 1 \leq k \leq n-1$$

ja reunaehtoja $\binom{n}{0} = \binom{n}{n} = 1$ kaikilla $n \in \mathbb{N}$. Näiden yhtälöiden perusteella rakentuu ns. Pascalin kolmio. (Jos Pascalin kolmio ei ole ennestään tuttu, perehdy siihen jonkin verkkomateriaalin tai oppikirjan avulla.)

   (a) Kirjoita rekursiivinen Java-funktio, joka laskee binomikertoimen $\binom{n}{k}$ arvon käyttämällä suoraan ylläolevaa palautuskaavaa.

   (b) Kirjoita Java-funktio, joka laskee binomikertoimen $\binom{n}{k}$ arvon ilman rekursiota. (Tämä tapahtunee helpoimmin muodostamalla tarvittava osa Pascalin kolmiosta.)

   Luulisitko, että rekursiivisen ja ei-rekursiivisen ratkaisun välillä on merkittävää nopeuseroa? Jos on, niin kumpi on nopeampi?

6. (a) Tee metodi, joka tulostaa yhden rivin, jolla on parametrina annettu määrä tähtimerkkejä `*`. Sen lisäksi funktio kutsuu itseään rekursiivisesti, jos parametrin arvo on positiivinen.

   Metodin runko näyttää seuraavalta:

```
private static void tahtia(int lkm) {
   // tulosta lkm tahtea
   // kutsu funktiota rekursiivisesti tulostamaan lkm-1 tahteä
}
```

   Käyttöesimerkki:

```
public static void main(String[] args) {
    tahtia(3);
}
```

   Ruudulle pitäisi tulostua

```
***
**
*
```

   Huom: yksittäinen metodin `tahtia`-komento-osan suoritus siis tulostaa ainoastaan yhden tähtirivin. Eli edellisessä esimerkissä metodi tulee kutsutuksi yhteensä 3 kertaa, vaikka main tekeekin ainoastaan yhden kutsun.

   (b) Pienellä muutoksella ohjelma saadaan tulostamaan tähdet päinvastaisessa järjestyksessä, eli esim. kutsulla `tahtia(3)` tulostuu

```
*
**
***
```

Kokeile, mikä muutos saa tämän aikaan. Muutos liittyy rekursiivisen metodikutsun paikkaan ja onnistuu yhtä koodiriviä siirtämällä. Et tarvitse esim. mitään apumuuttujia.

(c) Yksittäinen metodi voi kutsua itseään rekursiivisesti useampaan kertaan. Kirjoita metodin runkoon kaksi rekursiivista kutsua. Laittamalla kutsut sopiviin paikkoihin saadaan pääohjelmassa kutsumalla tahtia(2) aikaan kuvio

```
*
**
*
```

Ja kutsumalla tahtia(3) kuvio

```
*
**
*
***
*
**
*
```

Kokeile, mikä muutos saa tämän aikaan.

(d) Muokkaa edellistä ohjelmaa siten, että mukaan tulee staattinen muuttuja, jonka avulla voidaan laskea, kuinka monennesta rekursiivisesta kutsusta on kysymys. Tulosta jokaisen tähtirivin perään sen aiheuttaneen rekursiivisen kutsun järjestysnumero.

Seuraavassa hahmotelma:

```
static int kutsut;

public static void main(String[] args) {
    kutsut = 1;
    tahtia(3);
}

private static void tahtia(int lkm) {
    // otetaan talteen monesko kutsu itse ollaan ja kasvatetaan
    // kutsujen yhteenlaskettua lukumäärää
    int kutsunNumero = kutsut++;

    // ...
    // tulosta lkm tahtea ja tulosta kutsunNumero
    // ...
}
```

Edellisen kohdan kutsun tahtia(3) pitäisi näyttää suunilleen seuraavalta (huom. jos kutsut koodissasi rekursiivisesti myös tahtia(0) voi numerointi mennä hieman eri tavalla):

```
* 3
** 2
* 4
*** 1
* 6
** 5
* 7
```

(e) Kun alat hallita rekursion toimintaperiaatteen, saat pienellä muokkauksella (lisäämällä kolmannen rekursiokutsun) saat ohjelmasi toimimaan esim. seuraavasi:

```
* 3
* 4
* 5
** 2
* 7
* 8
* 9
** 6
* 11
* 12
* 13
** 10
*** 1
```

Tee tämä muutos.

7. **Ylimääräinen lisätehtävä**

Tarkastellaan johonkin tilaisuuteen osallistuvia henkilöitä ja määritellään *julkkis* henkilöksi, jonka kaikki läsnäolijat tuntevat, mutta joka itse ei tunne ketään muuta. Tehtävänä on löytää tilaisuudesta julkkis, tai todeta julkkisten puuttuminen, esittämällä kysymymyksiä "Tunteeko henkilö x henkilön y?" Esitä tehtävään ratkaisumenetelmä, joka n läsnäolijan tapauksessa tekee $O(n)$ kysymymystä. Perustele, että menetelmälläsi on haluttu ominaisuus.

Huom: Tämä tehtävä on lisätehtävä, eli tehtävää ei lasketa mukaan maksimilukumäärään tehtäviä.