Aalto-yliopiston perustieteiden korkeakoulu
Tietotekniikan laitos
CSE-A1140 Tietorakenteet ja algoritmit

TENTTI
17.8.2015

Kirjoita jokaiseen paperiin oma nimesi, oppilasnumerosi, tutkinto-ohjelmasi, kurssikoodi ja kurssin nimi, päivämäärä, sali, palauttamiesi paperien lukumäärä sekä *allekirjoituksesi*. Numeroi palauttamasi paperit juoksevalla numeroinnilla. Tentissä ei saa käyttää mitään ylimääräisiä apuvälineitä.

1) Kymmenen kysymystä (10 x 1p)

Tämä tehtävä on *tentin pakollinen osa*, josta on saatava vähintään 5p/10p, jotta loput tentistä tarkistetaan. Tämä tehtävä ei kuitenkaan yksistään riitä tentin läpäisyyn. Toisaalta viiteen pisteeseen ei edellytetä "täysin oikeaa vastausta" vaan oleellista on, että pystyt osoittamaan *ymmärtäneesi* tehtävän *koodin* toiminnan. Käytä siis aikaa perustelujen miettimiseen ja esittämiseen. Viittaa perusteluissa ohjelmakoodin rivinumeroihin, jos mahdollista.

Alla on annettu kaksi algoritmia (pow1 ja pow2), jotka molemmat laskevat potenssifunktion (x^n) kokonaisluvuille x ja n. Lue ensin kaikki kysymyskohdat vastaamatta niihin ja sen jälkeen tutustu annettuihin koodinpätkiin erittäin huolella. Vastaa tämän jälkeen kaikkiin kysymyksiin ja käytä aikaa perustelujen pohtimiseen ja muotoilemiseen. Huomaa, että kaikissa kysymyksissä viitataan alla oleviin algoritmeihin ja, että vastaukset tulee perustella hyvin, tai siis *pisteet tulevat vain perusteluista*!

```
1 int pow1(int x, int n) {
2     if (n == 0)
3         return 1; else
4     if (n == 1)
5         return x; else
6     if ("n on pariton")
7         return pow1(x², ⌊n/2⌋)) *
x; else
8     if ("n on parillinen")
9         return pow1(x², ⌊n/2⌋));
10 }
```

```
11 int pow2(int x, int n) {
12         p = 1;
13   for (int i=1; i<=n; i++)
14         p = p * x;
15   return p;
16 }
17
18
19
20
```

a) *Selitä* algoritmin `pow1` toiminta sanallisesti.
b) *Selitä* algoritmin `pow2` toiminta sanallisesti.
c) *Millä ohjelmariveillä ja kuinka monta* kertolaskua `pow1` suorittaa? *Anna esimerkki*, kun algoritmia kutsutaan parametrillä $n=9$.
d) *Millä ohjelmariveillä ja kuinka monta* kertolaskua `pow2` suorittaa? *Anna esimerkki*, kun algoritmia kutsutaan parametrillä $n=9$.
e) *Analysoi* algoritmin 1 suoritusaika sen saaman syötteen koon n funktiona.
f) *Analysoi* algoritmin 2 suoritusaika sen saaman syötteen koon n funktiona.
g) *Perustele* pitääkö väite paikkansa vai ei: algoritmi 1 on tehokkaampi kuin algoritmi 2.
h) *Perustele* pitääkö väite paikkansa vai ei: algoritmi 1 laskee saman funktion kuin algoritmi 2.
i) *Mikä* olisi algoritmin 1 *kertolaskujen suoritusjärjestys*, jos riviä 7 muutettaisiin muotoon "`return x * pow1(x*x, n/2); else`"? *Anna esimerkki*.
j) Algoritmi 2 käyttää *for*-silmukkaa. Voitaisiinko se korvata jollakin toisella silmukalla? Perustele joko miksi ei tai anna esimerkki miten (kirjoita algoritmi uusiksi).

Bonustehtävä:
k) *Pohdi ja vertaile* algoritmien 1 ja 2 muistinkäyttöä.

2) Terminologiaa (2p + 2p + 2p + 2p)

Määrittele lyhyesti seuraavat *käsitteet* (4 x 1p). *Anna* jokaisesta myös *esimerkki* (4 x 1p).

a) Abstrakti tietotyyppi (ADT)
b) Pino
c) Valikointi (Selection)
d) Stabiili järjestämismenetelmä

3) Hakurakenteet (2p + 2p + 2p + 2p +2p)

Vertaile tasapainotettuja hakupuita (balanced search trees) ja hajautusrakenteita (hashing) keskenään. Jäsennä vastauksesi seuraavasti:

a) *Määrittele* abstrakti tietotyyppi *hakurakenne* (dictionary).
b) *Nimeä* ainakin *yksi tasapainotettu hakupuu* ja *yksi hajautusrakenne*.
c) *Mitkä operaatiot* (vähintään 2) *voidaan toteuttaa* tasapainotetuissa hakupuissa tehokkaammin kuin hajautusrakenteissa? Perustele vastauksesi.
d) *Mitkä operaatiot* (vähintään 2) *voidaan toteuttaa* hajautusrakenteissa tehokkaammin kuin tasapainotetuissa hakupuissa? Perustele vastauksesi.
e) *Miten* jokin tasapainotettu hakupuu ja jokin hajautusrakenne voitaisiin *yhdistää* siten, että saataisiin aikaan vielä tehokkaampi hakurakenne?

Voit valita teetkö tehtävän 4.1 vai 4.2. Jos teet molemmat, niin saat kuitenkin pisteet vain toisesta (vain paremmat pisteet huomioidaan).

4.1) Prioriteettijonot (2p + 2p + 4p)

a) *Määrittele* käsite *prioriteettijono (priority queue)*.

b) *Mainitse nimeltä* jokin algoritmi, joka käyttää prioriteettijonoa aputietorakenteenaan. *Selitä lyhyesti* mihin ko. algoritmi sitä tarvitsee.

c) *Esitä välivaiheittain* miten lineaarisessa ajassa toimiva *BuildHeap*-algoritmi (joka tunnetaan myös nimillä *FixHeap* ja *Bottom-Up Heap Construction*) toimii, jos se saa syötteenään taulukon, jossa on alkiot 7, 2, 25, 1, 10, 23, 14, 20, 3, 5, 6, 15 ja 13. Kekoehto on "isä pienempi kuin lapsensa". **Huom!** Kyseessä ei ole tapaus, jossa alkiot lisätään rakenteeseen yksi kerrallaan.

4.2) Pienimmän virityspuun etsiminen (4p + 4p)

a) Esitä jonkin algoritmin toiminta, jolla voidaan tuottaa verkon pienin virityspuu. Mikä on esittämäsi algoritmin aikakompleksisuus, jos graafissa on V solmua ja E kaarta? Perustele tuloksesi.

b) Tarkastellaan painotettua suuntaamatonta verkkoa, johon kuuluvat solmut A-F ja särmät AB(2), AD(2), AE(6), BC(4), BE(3), CE(1), CF(4), DE(1), EF(3). Särmien painot on esitetty särmän jälkeen suluissa. Esitä miten annetulle verkolle voidaan löytää pienin virityspuu soveltamalla a-kohdassa kuvaamaasi algoritmia.