Aalto-yliopiston teknillinen korkeakoulu 11.5.2010 TENTIN ARVIOINTIOHJE
Tietotekniikan laitos 12.5.2010
T-106.1220/1223 Tietorakenteet ja algoritmit T/Y

Tämä arviointiohje pyrkii *luonnehtimaan* millaisista vastauksista saa pisteitä. Se ei ole oikeiden vastausten luettelo (tehtäviin ei tyypillisesti ole yhtä ainoaa oikeaa vastausta), vaan ennemminkin joukko esimerkkejä perusteluista, joilla pisteitä saa.

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.

Kuten edellä todetaan, tämän tehtävän tavoite on mitata sitä, onko opiskelija ymmärtänyt tehtävän koodin toiminnasta *riittävästi*. Seuraavassa arviointiperusteena ei ole ”oikea vastaus” vaan kyky osoittaa pystyvänsä seuraamaan annettua esimerkkiohjelmaa ja tietorakenteen koodausta. Luonnollisesti ”oikea vastaus” tulkitaan osoitukseksi edellisestä, mutta perustelluista ”vääristä vastauksista” on myös mahdollista saada piste, jos tulkinta on mahdollista ymmärtää annetun esimerkkiohjelman valossa ja erityisesti, jos perustelut on kytketty koodiin esim. viittaamalla rivinumeroihin.

Alla on annettu kaksi algoritmia (fact_1 ja fact_2), jotka laskevat kertoma-funktion (*factorial*). 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 väittämät voi perustella yhtä hyvin vastaamalla kyllä tai ei, joten pisteet tulevat vain *perusteluista*!

```
1 int fact_1(int n)
2 {
3    if (n<2)
4        return 1;
5    else
6        return fact_1(n-1)*n;
7 }
```

```
 8 int fact_2(int n)
 9 {
10    int i,fact;
11    fact = 1;
12    for (i=1; i<=n; i++)
13        fact = fact*i;
14    return fact;
15 }
```

a) *Selitä* algoritmin 1 (ts. fact_1) toiminta *esimerkin avulla*.
b) *Selitä* algoritmin 2 (ts. fact_2) toiminta *esimerkin avulla*.
c) *Missä järjestyksessä ja kuinka monta* kertolaskua fact_1 suorittaa? *Anna esimerkki,* kun algoritmia kutsutaan parametrillä n=3.
d) *Missä järjestyksessä ja kuinka monta* kertolaskua fact_2 suorittaa? *Anna esimerkki,* kun algoritmia kutsutaan parametrillä n=3.
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 6 muutettaisiin muotoon ”return n*fact_1(n-1);”? *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.

Kyky seurata esimerkkiohjelmia näkyy mm. siinä, että kohdassa (1p/kohta)
a) havaitsee ohjelman rekursiivisuuden
b) havaitsee ohjelman ei-rekursiivisuuden (ts. tarvitaan silmukka)
c) ja d) antaa oikeat esimerkit (1*2 ja 2*3 sekä 1*1, 1*2 ja 2*3)
Kohdat a-d arvioidaan yhdessä: pisteet selityksistä (2*1p) ja esimerkeistä (2*1p)
e) ja f) päätyy molemmissa tapauksissa lineaariseen suoritusaikaan, esim. O(n)
g) ja h) osaa perustella väitteet puoleen tai toiseen viitaten ohjelmien ajonaikaisiin suorituksiin. Perusteissa voi viitata esim. kohtiin c) ja d) tai vaikkapa siihen, että fact_1:n laskemiseksi tarvitaan rekursiopino tai siihen, että kääntäjä saattaa optimoida rekursion pois, jolloin oleellisesti molemmat koodit tuottavat (lähes) saman ohjelman. Sen havaitseminen, että fact_2 tekee yhden ”turhan” kertolaskun (1*1) verrattuna fact_1:een tulkitaan myös koodin toiminnan ymmärtämiseksi riippumatta tehdystä johtopäätöksestä.
i) osoittaa, että suoritusjärjestys on sama kuin c-kohdassa mutta vaihdannaisesti (2*1,3*2).
j) osaa muuntaa for-silmukan esim. while-silmukaksi
k) pohdiskelee esim. rekursiopinoa (ks. g-kohta edellä). Tehtävästä 1 max. 11p

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

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

Oikea selitys 1p
Esimerkki 1p

a) Stabiili (*stable*) järjestämismenetelmä

…säilyttää samanarvoisten alkioiden keskinäisen järjestyksen
Esim. mergesort TAI vaikkapa selitys miten ”stabiili järjestämismenetelmä mahdollistaa sähköpostien järjestämisen kahden attribuutin mukaan”

b) Paikoillaan tapahtuva (*in place*) järjestäminen

…ei tarvitse lisätilaa.
Esim. lisäysjärjestäminen TAI selitys miten järjestäminen tapahtuu syötteenä saadussa taulukossa.

c) Inversio (*inversion*)

…on tapaus, jossa kaksi listan alkiota on väärässä järjestyksessä, ts. a[i] > a[j] ja i < j.
Esim. taulukko kokonaislukuja, joka epäjärjestyksessä ja osoitettu i ja j, joille em. pätee.

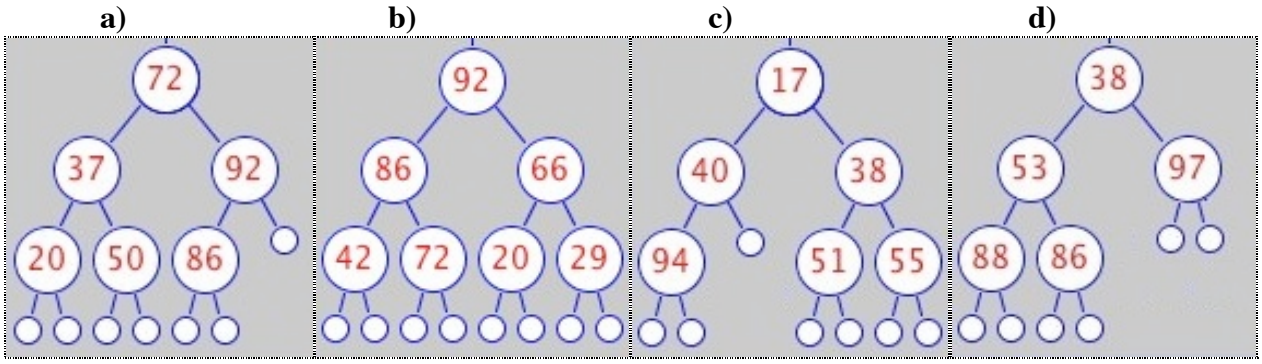
d) Valikointi (*selection problem*)

…etsii mielivaltaisessa järjestyksessä olevasta aineistosta k:nneksi suurimman alkion.
Esim. quickselect-algoritmin toiminnan selittäminen TAI ”mediaanin haku (k = n/2)”.

3) Prioriteettijonot (2p + 2p + 2p + 2p)

Seuraavissa kuvissa on neljä binääripuuta (*binary tree*) a, b, c ja d. *Vastaa jokaisen kohdalla erikseen* onko kuvan tietorakenne *minimikeko* (*MinHeap*) (kyllä/ei, 1p) ja *perustele* miksi (1p).

Oikea vastaus 1p (ei, ei, ei ja kyllä = 4p)
Perustelu 1p/kohta
a- ja b-kohdissa kekoehto ei täyty (esim. koska juuri on suurempi kuin vasen lapsi)
c-kohdassa puu ei ole täydellinen binääripuu
d-kohta täyttää kekoehdon eli kaikille alkioille isä pienempi kuin lapsi



4) Tasapainotetut hakupuut (2p + 6p + 4p)

a) *Määrittele* käsite *tasapainotettu hakupuu* (*balanced search tree*).

…on **hakurakenne** (operaatiot lisäys, haku ja poisto tehokkaita) 0,5p
…on hakupuu, jossa **haraudutaan isäsolmun perusteella** johonkin alipuuhun, esim. binäärisissä hakupuissa joko vasemmalle (etsittävä alkio pienempi kuin juuri) tai oikealle (etsittävä alkio suurempi kuin juuri) 0,5p
…**säilyttää** jonkin **tasapainoehdon**, esim. AVL-puussa alipuiden korkeusero saa olla korkeintaan yksi. 1p
Pisteet pyöristetään ylöspäin seuraavaan kokonaiseen pisteeseen.

b) Millaisia erilaisia tasapainotettuja hakupuita on olemassa? Miten ne eroavat toisistaan? *Käsittele* tässä *kolmea eri tietorakennetta*.

1. Nimeämisestä 0,5p/rakenne (esim. AVL-puu, punamusta puu, B-puu) = 1,5p
2. - AVL-puu ja punamusta muu ovat binäärisiä hakupuita, joissa haarautumisaste on kaksi 0,5p + 0,5p = 1p
   - B-puussa haarautumisaste m on suuri 0,5p

TAI
   - Tarkemmista määritelmistä 1p/oikein määritelty rakenne

3. Erojen toteamisista 0,5p/kahden rakenteen vertailu
- AVL-puu vs. Punamusta puu; erilainen tasapainoehto (0,5p)
- B-puussa haarautumisaste on suurempi kuin binäärisissä hakupuissa, esim. 2-3-4-puussa m=4 (1p)

Pisteet pyöristetään ylöspäin seuraavaan kokonaiseen pisteeseen.

c) *Anna esimerkki* (*piirrä ja merkitse* (nimeä) välivaiheet ennen ja jälkeen *jokaisen tasapainotusoperaation*) jonkin tasapainotetun hakupuun toiminnasta, kun siihen lisätään alkiot `11,5,2,18,7,4,3,6,10,5,6,5` tässä järjestyksessä. Samanarvoiset alkiot sijoitetaan aina oikeanpuoleiseen haaraan.

AVL-puussa 1p/oikein suoritettu rotaatio; punamustassa puussa 0,5p/tasapainotus; 2-3-4-puussa 1p/split; Vähennetään 2 pistettä, jos jokin tila ei ole tasapainotettu hakupuu. Pyöristys ylöspäin seuraavaan kokonaiseen pisteeseen. Min. 0p ja Max. 4p

5) Verkkoalgoritmit (7p + 3p)

a) *Kirjoita algoritmi*, joka etsii annetun suuntaamattoman verkon yhtenäiset osaverkot (*connected sub-graphs*). Voit esittää algoritmin jollakin tunnetulla ohjelmointikielellä tai käyttää vapaampaa pseudokieliesitystä. Selitä algoritmin toiminta kuitenkin myös sanallisesti.

- DFS-algoritmin mainitsemisesta 1p
DFS-algoritmin kuvauksessa (pseudo-koodi tai sanallinen kuvaus) tulee löytyä seuraavat elementit:
- alustustoimenpiteet (mm. visited-taulukon nollaaminen) 1p
- solmujen merkitseminen (visited) 1p
- naapureiden läpikäynti (silmukka) 1p
- reunuksen toteutus (pino tai rekursiokutsujen myötä rekursiopino) 1p
- DFS:n kutsuminen (ja esim. osaverkon solmujen merkitseminen yhteisellä tunnuksella) erillisille yhtenäisille komponenteille (silmukka) 1p
- toimiva algoritmi/riittävän selkeä esitys, jotta algoritmi on mahdollista analysoida 1p
Muut ratkaisut arvioidaan tapauskohtaisesti, edellistä arviointiohjetta soveltaen.

b) *Analysoi* algoritmisi suoritusaika.

Algoritmi suoritusaika on O(V+E) 1p
Perusteluista saa 2p, jos vetoaa esim. siihen, että algoritmi
- merkitsee jokaisen solmun V vierailluksi (visited) täsmälleen kerran 1p tai
- jokainen solmu viedään reunukseen ja otetaan sieltä pois täsmälleen kerran 1p ja
- jokainen kaari E tutkitaan molempiin suuntiin eli 2*E tai O(E) kertaa 1p

6) Palaute (2p)

Anna asiallista *kurssipalautetta* täyttämällä kurssin kotisivun kautta löytyvä palautelomake 18.5. mennessä, niin saat tenttiin 2 lisäpistettä. Lisäpisteillä ei voi kuitenkaan nostaa hylättyä arvosanaa hyväksytyksi.

Aalto University School of Science and Technology — EXAMINATION
Department of Computer Science and Engineering — 11.5.2010
T-106.1227 Data Structures and Algorithms

Write on each paper your name, student number *with the control letter*, degree programme, and the course code and name. Also write the date, hall, the number of papers you return, and your *signature*.

1) Ten Questions (10 x 1p)

This is a *compulsory part* of the final exam. You need to get at least 5p out of the maximum 10p so that the rest of the exam will be checked. However, this part alone is not enough to pass the whole exam. On the other hand, in order to get 5p, you are not required to give "the exactly correct answer", but more or less show that *you have understood the functionality of the code fragments* related to this part. Thus, pay attention to the reasoning. Refer to the code line numbers if possible.

In the following, you can see two algorithms computing the factorial. Read through all the questions below without answering them and after that familiarize yourself with the code throughout. After this, answer all the questions and take time for pondering and explaining your reasoning. Note, however, that all the questions refer to the given algorithms. In addition, the claims in the questions can be justified to be either true or false, thus the *argumentation* is the only thing that matters for the points!

```
1 int fact_1(int n)
2 {
3    if (n<2)
4        return 1;
5    else
6        return fact_1(n-1)*n;
7 }
```

```
 8 int fact_2(int n)
 9 {
10    int i,fact;
11    fact = 1;
12    for (i=1; i<=n; i++)
13        fact = fact*i;
14    return fact;
```

a) *Describe* how Algorithm 1 (i.e., fact_1) works by using an appropriate *example*.
b) *Describe* how Algorithm 2 (i.e., fact_2) works by using an appropriate *example*.
c) *In which order and how many* multiplications fact_1 does? *Give an example* in case the algorithm is called with parameter n=3.
d) *In which order and how many* multiplications fact_2 does? *Give an example* in case the algorithm is called with parameter n=3.
e) *Analyse* the time complexity of Algorithm 1 in terms of the input size n.
f) *Analyse* the time complexity of Algorithm 2 in terms of the input size n.
g) *Argue* whether it is true or false: Algorithm 1 is more efficient than Algorithm 2.
h) *Argue* whether it is true or false: Algorithm 1 computes the same function than Algorithm 2.
i) *What is the order of multiplications* in Algorithm 1 if the line 6 would be changed to "return n*fact_1(n-1);"? *Give an example*.
j) Is it possible to replace the for-loop in Algorithm 2 with another loop? Argue either why not or give an example how to replace it (write the algorithm anew).

Bonustehtävä:
k) *Ponder and compare* the memory consumption of Algorithm 1 and 2.

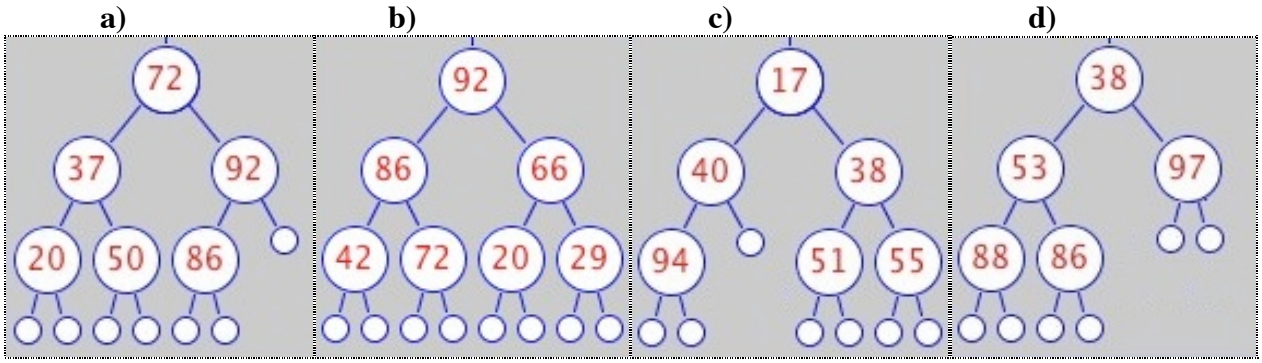
2) Terminology (2p + 2p + 2p + 2p)

Define the following *concepts* (4 x 1p). In addition, *give an example* of each (4 x 1p).

a) *Stable* sorting method
b) *In place* sorting
c) *Inversion*
d) *Selection problem*

3) Priority Queues (2p + 2p + 2p + 2p)

Consider the following four binary trees in Figures a, b, c, and d. *For each case*, argue whether the figure depicts a MinHeap (yes/no, 1p) and *justify your answer* (1p).



3) Balanced Search Trees (2p + 6p + 4p)

a) *Define* the concept Balanced Search Tree.
b) What kind of balanced search trees you know? How do they differ from each other? *Deal with three different data structures*.
c) *Give an example* (*draw and mark* (give a name) the intermediate states before and after *each balancing operation*) of a balanced search tree in case the items `11,5,2,18,7,4,3,6,10,5,6,5` will be inserted into the tree in this order. You can assume the duplicates will be inserted into the right branch of the tree.

4) Graph algorithms (7p + 3p)

a) *Write an algorithm* to find the connected sub-graphs of a given undirected graph.

b) *Analyze* the running time of your algorithm.

You can represent the algorithm by using some known programming language or pseudo-code. However, explain the working of the algorithm also verbally.

5) Feedback (2p)

Give objective *feedback* about this course by filling in the feedback questionnaire linked from the course web page before May 18, and you will get two extra points for this exam.