Aalto-yliopiston perustieteiden korkeakoulu
Tietotekniikan laitos
CS-A1141/1143 Tietorakenteet ja algoritmit Y

TENTTI
20.2.2024

Kirjoita jokaiseen paperiin oma nimesi, oppilasnumerosi, tutkinto- tai koulutusohjelmasi, kurssikoodi ja kurssin nimi, päivämäärä, sali, palauttamiesi paperien lukumäärä sekä *allekirjoituksesi*. Tentissä ei saa käyttää mitään ylimääräisiä apuvälineitä.

1) Kertomafunktiot (2 p + 2 p + 1 p + 1 p + 1 p + 1 p + 2 p)

Tämä tehtävä on *tentin pakollinen osa*, josta on saatava vähintään 5 p / 10 p, 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. 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. Esimerkiksi analyysiksi ei riitä pelkkä lopputulos. Huomaa, että kaikissa kysymyksissä viitataan alla oleviin algoritmeihin ja väittämät voi perustella yhtä hyvin vastaamalla kyllä tai ei, joten pisteet tulevat vain *perusteluista*!

Seuraavassa on annettu kaksi algoritmia, jotka laskevat kertomafunktion (*factorial*). Ne on toteutettu Python-funktioina, joita Aimo Assari testasi lisäämällä riveille 7 (fact_1) ja 4 (fact_2) print-lauseet.

```
1 def fact_1(n):
2     if (n < 2):
3         return 1
4     else:
5         a = fact_1(n-1)
6         b = n
7         print(a,b,a*b)
8         return a*b
```

```
1 def fact_2(n):
2     f = 1
3     for i in range(2, n+1):
4         print(f,i,f*i)
5         f = f * i
6     return f
7
8
```

a) *Mitä* fact_1(4) tulosti ruudulle, kun Assari ajoi ohjelman?
b) *Mitä* fact_2(4) tulosti ruudulle, kun Assari ajoi ohjelman?
c) *Analysoi* algoritmin 1 ts. fact_1(n) aikakompleksisuus sen saaman syötteen n funktiona.
d) *Analysoi* algoritmin 2 ts. fact_2(n) aikakompleksisuus sen saaman syötteen n funktiona.
e) *Perustele* pitääkö väite paikkansa vai ei: algoritmi 1 on yhtä tehokas kuin algoritmi 2.
f) *Perustele* pitääkö väite paikkansa vai ei: algoritmi 1 laskee saman funktion kuin algoritmi 2.
g) *Pohdi ja vertaile* algoritmien 1 ja 2 muistinkäyttöä.

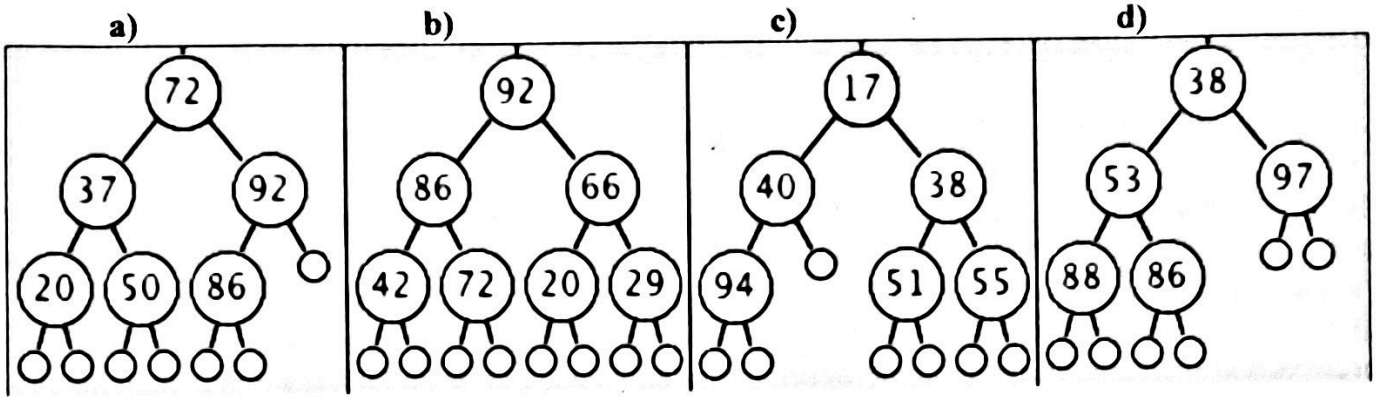
2) Terminologiaa (2 p + 2 p + 2 p + 2 p)

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

a) Stabiili (*stable*) järjestämismenetelmä
b) Paikoillaan tapahtuva (*in place*) järjestäminen
c) Inversio (*inversion*)
d) Valikointi (*selection problem*)

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

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



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

a) *Määrittele* käsite *tasapainotettu hakupuu* (*balanced search tree*).
b) Millaisia erilaisia tasapainotettuja hakupuita on olemassa? Miten ne eroavat toisistaan? *Käsittele* tässä *kolmea eri tietorakennetta.*
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.

5) Verkkoalgoritmit (7 p + 3 p)

a) *Kirjoita algoritmi*, joka etsii annetun suuntaamattoman verkon yhtenäiset osaverkot (*connected sub-graphs*).
b) *Analysoi* algoritmisi suoritusaika.

Voit esittää algoritmin jollakin tunnetulla ohjelmointikielellä tai käyttää vapaampaa pseudokieliesitystä. Selitä algoritmin toiminta kuitenkin myös sanallisesti.