Aalto-yliopiston perustieteiden korkeakoulu
Tietotekniikan laitos
CSE-A1141 Tietorakenteet ja algoritmit Y

TENTTI
16.2.2016

Kirjoita jokaiseen paperiin oma nimi, oppilasnumero, tutkinto-ohjelma, 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)

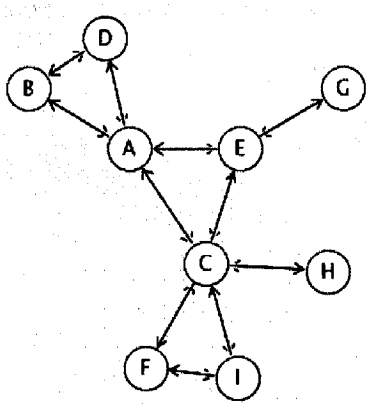
Tämä tehtävä on *tentin pakollinen osa*, josta on saatava vähintään 5p/10p, jotta loput tentistä tarkistetaan. Tämä tehtävä ei kuitenkaan yksistään riitä tentin läpäisyyn. Toisaalta viiteen pisteeseen ei edellytetä "täysin oikeaa vastausta" vaan oleellista on, että pystyt osoittamaan *ymmärtäneesi* tehtävän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

Alla on annettu kaksi verkon läpikäyntialgoritmia (`DFS1` ja `DFS2`). Lue ensin kaikki kysymyskohdat vastaamatta niihin ja sen jälkeen tutustu annettuihin koodinpätkiin erittäin huolella. Vastaa tämän jälkeen kaikkiin kysymyksiin. Huomaa, että kaikissa kysymyksissä viitataan alla oleviin algoritmeihin ja, että vastaukset tulee perustella hyvin, eli *pisteet tulevat vain perusteluista*!

```
1  Algorithm DFS1(G, u)
2      visited[u] ← true
3      for each v ∈ Adj[u] do
4          if visited[v] == false then
5              DFS1(G, v)
6      finished[u] ← true
7
8
9
10
```

```
11 Algorithm DFS2(G, u)
12     Stack.push(u)
13     visited[u] ← true
14     while ( Stack not empty )
15         u = Stack.pop()
16         for each v ∈ Adj[u] do
17             if visited[v] == false then
18                 visited[v] ← true
19                 Stack.push(v)
20         finished[u] ← true
```

a) Koodiriveillä 3 ja 16: `each v ∈ Adj[u]` tarkoittaa kaikkia niitä solmuja v, jotka ovat solmun u naapureita. *Miten toteuttaisit* tietorakenteen `Adj`? Miksi?
b) Anna esimerkki em. tietorakenteesta kuvan verkon tapauksessa. Esitä tietorakenne siten, että siinä näkyy kaikkien solmujen vierussolmut ja niiden järjestys.
c) *Selitä* `DFS1`:n toiminta sanallisesti (ilman esimerkkiä). Huom! Pyri selittämään *miten* algoritmi toimii yleisesti. Älä selitä koodia rivi-riviltä.
d) *Selitä* nyt `DFS2`:n toiminta sanallisesti. Miten se eroaa edellisestä?
e) *Anna esimerkki*, kun `DFS1` vierailee kuvan verkon solmuissa lähtien solmusta A. Ilmoita missä järjestyksessä solmu merkitään sekä `visited` että `finished` –lipuilla algoritmin suorituksen edetessä.



f) *Anna esimerkki*, kun `DFS2` vierailee kuvan verkon solmuissa lähtien solmusta A. Ilmoita pinon `Stack` sisältö aina kun algoritmissa tullaan riville 14. Alleviivaa lisäksi solmu, joka poistetaan pinosta rivillä 15.
g) *Perustele* pitääkö väite paikkansa vai ei: `DFS1` tuottaa saman läpikäyntijärjestyksen kuin `DFS2`.
h) *Perustele* pitääkö väite paikkansa vai ei: `DFS1` on tehokkaampi kuin `DFS2`.
i) Algoritmi `DFS1` käyttää *for*-silmukkaa. Voitaisiinko se korvata jollakin toisella silmukalla? *Perustele* joko miksi ei tai anna esimerkki miten (kirjoita algoritmi uusiksi).
j) Algoritmi `DFS2` käyttää *pinoa* (stack) apurakenteenaan. Mikä olisi solmujen läpikäyntijärjestys, jos se korvattaisiin *jonolla*?
Bonustehtävä:
k) *Pohdi ja vertaile* algoritmien `DFS1` ja `DFS2` muistinkäyttöä.

2) Hajautus (2p + 4p + 2p)

a) *Selitä* hajautuksen (*hashing*) *perusperiaatteet.*

b) Arvioi yleisesti hakurakenteisiin liittyvien operaatioiden aikakompleksisuutta hajautusmenetelmien yhteydessä. *Mitkä operaatiot ovat tehokkaampia* hajautusmenetelmissä kuin esim. hakupuissa? Entä *mitkä operaatiot ovat tehottomampia* hajautusmenetelmissä kuin esim. hakupuissa? Pohdi sekä keskimääräistä että pahinta tapausta.

c) *Arvioi* hajautusmenetelmien *hyviä ja huonoja puolia. Minkälaisiin sovelluksiin* kurssilla esitetyt *perushajautusmenetelmät soveltuvat* tai eivät sovellu hyvin?

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

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 alkiot 7, 2, 25, 1, 10, 23, 14, 20, 3, 5, 6, 15 ja 13 voidaan lisätä yksi kerrallaan alun perin tyhjään binäärikekoon (*binary heap*). Kekoehto on "isä pienempi kuin lapsensa".

d) *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".

4) Järjestämismenetelmät (1p + 3p + 3p + 3p)

a) *Määrittele* järjestämisongelma (*sorting problem*).

b) *Selitä* jonkin järjestämismenetelmän toiminta *sanallisesti.*

c) *Kirjoita* edellä kuvaamasi algoritmi *ohjelmakoodiksi.* Voit käyttää jotakin tuntemaasi ohjelmointikieltä tai esittää algoritmin vapaammin pseudo-koodina.

d) *Analysoi* algoritmisi aikavaatimus *iso O –notaatiossa* (Huom! perustelut mukaan). Kuinka tehokas algoritmisi on suhteessa tehokkaimpiin järjestämismenetelmiin?