Aalto-yliopiston perustieteiden korkeakoulu
Tietotekniikan laitos
T-106.1223 Tietorakenteet ja algoritmit Y

TENTTI
11.4.2014

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

1) Kymmenen kysymystä (10 x 1p)

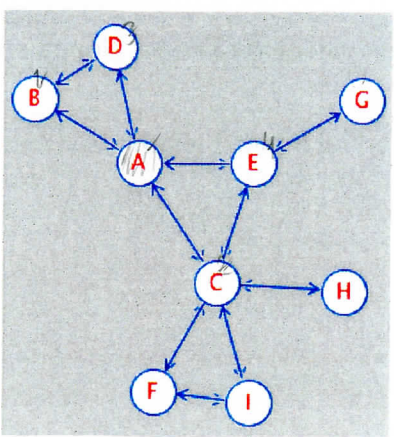
Tämä tehtävä on *tentin pakollinen osa*, josta on saatava vähintään 5p/10p, jotta loput tentistä tarkistetaan. Tämä tehtävä ei kuitenkaan yksistään riitä tentin läpäisyyn. Toisaalta viiteen pisteeseen ei edellytetä "täysin oikeaa vastausta" vaan oleellista on, että pystyt osoittamaan *ymmärtäneesi* tehtävän *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 kaikissa 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 kaikissa 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) Miten täydentäisit algoritmia DFS1, jotta algoritmi rakentaisi DFS-virityspuun, joka talletetaan johonkin tietorakenteeseen? 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) Prioriteettijonot (2p + 2p + 4p + 2p)

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 suurempi kuin lapsensa". Poista sen jälkeen keosta yksitellen 2 suurinta avainta ja piirrä keko kummankin poiston jälkeen.

d) Mikä on binäärikekoon (kekoehtona "isä suurempi kuin lapsensa") liittyvien operaatioiden DeleteMax ja Insert suoritusaika, kun keossa on N alkiota? *Perustele* vastauksesi.

3) Järjestämismenetelmät (8p)

Vertaile seuraavia neljää järjestämisalgoritmia: Lisäysjärjestäminen (insertion sort), lomitusjärjestäminen (mergesort), pikajärjestäminen (quicksort) ja suora digitaalinen järjestäminen (straigt radix sort) vähintään neljän erilaisen kriteerin valossa. Perustele valitsemasi kriteerit ja vertailun tulokset lyhyesti.

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

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 valitsemasi tasapainotetun hakupuun toiminnasta, kun alun perin tyhjään puuhun lisätään alkiot `11,5,2,18,9,4,3,8,10,20,7,1` tässä järjestyksessä.

5) Palaute (2p)

Anna palautetta kurssin Noppa-sivulta löytyvän lomakkeen kautta 30.4.2014 mennessä.

6) Arvioi lopuksi tenttiin vastaamiseen käyttämäsi aika noin 15 minuutin tarkkuudella.