Aalto-yliopiston perustieteiden korkeakoulu
Tietotekniikan laitos

TENTTI
7.12.2015

CSE-A1140 Tietorakenteet ja algoritmit

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 + 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 (`bs1` ja `bs2`), jotka molemmat etsivät puolitushaulla järjestetystä taulukosta `table` alkiota `x`. 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 bs1(int table[], int x) {
2      int low = 0;
3      int high = table.length - 1;
4      int mid;
5
6      while( low <= high )
7      {
8          mid = (low + high) / 2;
9
10         if (table[mid] < x)
11                 low = mid + 1;
12         else if (table[mid] > x)
13                 high = mid - 1;
14         else return mid;
15     }
16     return -1;
17 }
```

```
18 def bs2(table, x, low, high):
19     if low > high:
20         return -1
21     mid = (low + high) / 2
22     item = table[mid]
23     if item == x:
24         return mid
25     elif x < item:
26         return bs2(table,x,low,mid-1)
27     else:
28         return bs2(table,x,mid+1,high)
29
30
31
32
33
34
```

a) *Selitä* algoritmin `bs1` toiminta sanallisesti (ilman esimerkkiä). Huom! Pyri selittämään *miten* algoritmi ratkaisee ongelman. Älä selitä koodia rivi-riviltä.

b) Selitä nyt algoritmin `bs2` toiminta sanallisesti. Miten toiminta eroaa edellisestä?

c) Anna esimerkki hausta, jossa algoritmilla `bs1` etsitään alkiota `x` = 512 taulukosta, jossa on alkiot 1, 2, 4, 8, 16, 32, 64, 128, 256 ja 512. Vihje: Taulukoi mitä arvoja muuttujat `low`, `high` ja `mid` saavat ohjelman suorituksen edetessä. Mitä ohjelma palauttaa ja mitä laskennan tuloksena saadaan?

d) Anna esimerkki hausta, jossa algoritmilla `bs2` etsitään em. taulukosta arvoa `x` = 100. Vihje: taulukoi tässä muuttujat `low`, `high`, `mid` ja `item` kuten edellä. Mitä ohjelma palauttaa ja mikä on laskennan tulos tässä tapauksessa?

e) Määrittele algoritmien 1 ja 2 ns. "syötteen koko", ts. mistä muuttujista ja miten algoritmien suoritusajat riippuvat?

f) *Analysoi* algoritmin 1 suoritusaika sen saaman syötteen koon *n* funktiona.

g) *Analysoi* algoritmin 2 suoritusaika sen saaman syötteen koon *n* funktiona.

h) Algoritmia 1 testattiin suurella aineistolla (haettiin aineiston pienintä alkiota), jolloin sen suoritusajaksi saatiin noin 1 millisekunti. Tämän jälkeen aineiston koko kaksinkertaistettiin ja suoritusajaksi saatiin noin 2 millisekuntia. Jos aineisto jälleen kaksinkertaistettaisiin, niin kuinka pitkään arvioisit laskennan tällä kertaa kestävän? Perustele.

i) Millaisia oletuksia ja reunaehtoja algoritmin 2 virheetön ja tehokas suoritus asettaa taulukolle `table` ja taulukon sisältämille alkioille?
j) *Perustele* pitääkö väite paikkansa vai ei: algoritmi 1 on tehokkaampi kuin algoritmi 2.
k) Bonustehtävä: *Pohdi ja vertaile* algoritmien 1 ja 2 muistinkäyttöä.

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

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

a) Kekoehto (Heap property)
b) Hajautusfunktio (Hash function)
c) Lineaarinen kokeilu (Linear probing)
d) Valikointi-ongelma (Selection-problem)

3) Puiden läpikäyntialgoritmit (2p + 2p)

Binääripuu voidaan määrittää yksikäsitteisesti, jos tiedetään sen läpikäyntijärjestys esijärjestyksessä (*preorder*) ja sisäjärjestyksessä (*inorder*). Erään binääripuun esijärjestys oli K-I-B-A-M-H-L-P-Q-F ja sisäjärjestys oli B-I-M-A-H-K-P-L-Q-F. *Piirrä* puu ja *anna* sen vastaava *jälkijärjestys (postorder)*.

4) Järjestämismenetelmät (3p + 3p + 2p)

Joudut valitsemaan algoritmin tehtävään, jossa tulee järjestää annettu aineisto. Mitä asioita (kriteereitä) huomioit tehdessäsi valintaa? Lue ensin koko tehtävänanto.

a) *Valitse kolme* (3) keskeistä *kriteeriä* joiden valossa tarkastelet tilannetta. *Perustele* miksi tai miten valitsemasi kriteerit liittyvät järjestämisongelmaan.
b) *Nimeä* jokaisen kriteerin kohdalla erikseen ainakin yksi *algoritmi*, joka täyttää ko. kriteerin ja yksi joka ei täytä (algoritmien toimintaperiaatteita ei tarvitse selittää). Anna vastauksesi matriisimuodossa 3x2, jossa sarakkeilla (3) on kriteerit ja niiden alapuolella riveillä (2) algoritmien nimiä, jotka *täyttävät* ja *eivät täytä* ko. kriteeriä.
c) *Nimeä algoritmi*, joka täyttää kaikki kriteereistäsi. *Nimeä* myös jokin *algoritmi*, joka ei täytä ainakaan kahta kriteereistäsi.

5) Verkot (4p + 4p + 4p)

Määrittele seuraavat käsitteet. *Anna* jokaisesta keskenään erilainen *esimerkki*. Käytä esimerkissä yhtenäistä verkkoa (*connected graph*), jossa on vähintään 7 solmua ja vähintään 10 kaarta. *Selitä* lyhyesti miten kyseessä oleva laskennallinen ongelma voidaan ratkaista (käytä em. esimerkkejäsi ja kuvaile sen avulla lähtötilanne, mainitse jokin algoritmi, jolle se voidaan antaa syötteenä ja kerro lyhyesti miten ko. algoritmi tuottaa halutun lopputuloksen).

a) Virityspuu (Spanning tree)
b) Minimaalinen virityspuu (Minimum spanning tree)
c) Lyhimmin poluin virittävä puu (Shortest paths spanning tree)