

OHJ-2010 Tietorakenteiden käyttö

Tentti 16.3.2009

Tentin laatija: Mika Siikarla

- Tentti on mitoitettu 30 pisteen tentiksi. Tentissä on kysymyksiä $37\frac{1}{4}$ pisteen edestä. Kaikkiin tehtäviin ei tarvitse vastata. Kaikkiin tehtäviin **saa** vastata.
- Kirjoita vastauksesi lyijykynällä, luettavalla käsialalla lyhyesti.
- **Selitä ja perustele vastauksesi.**
- Tentissä ei saa käyttää kirjallista materiaalia, laskimia tai tietokoneita.

0. Jätä ensimmäisen sivun alareunaan viisi tyhjää riviä. ($\frac{1}{4}$ p)

1. Mille allaolevista tietorakenteista avainten lisäysjärjestys ei vaikuta lopputulokseen? Mille avaimen arvo ei vaikuta alkion sijaintiin tietorakenteessa? B-puu, binäärihakupuu, hyppylista, jono, järjestetty rengaslista, keko, prioriteettijono, rengaspushuri ja trie. (3 p)

2. Tästä tehtävästä pitää saada ainakin kolme ja puoli (3,5) pistettä.

Anna allaoleville funktioille samaan kertaluokkaan kuuluva yksinkertaisempi funktio. Sievennä myös kertaluokkien notaatiot kirjoittamalla ne yksinkertaisemman edustajansa avulla. Määritä funktioiden ja kertaluokkien väliset suhteet ja kertaluokkien keskinäiset suhteet taulukkona (Kuva 1 vasemmalla) tai listana (kuvassa oikealla).

Merkitse kukin suhde vain kerran. Toistuvat tai triviaalit suhteet on esimerkissä merkitty harmaalla. Käytä joukko-opin merkintöjä $\subseteq$, $\subset$, $\supseteq$, $\supset$, $=$, $\in$, $\ni$, $\notin$ ja $\nexists$ sekä $\cap$ (joukot leikkaavat, mutta eivät sisälly toisiinsa) ja $\cap$ (joukot eivät leikkaa). Jos suhteesta ei voi sanoa mitään, käytä kysymysmerkkiä (?). Tässä tehtävässä ei tarvita perusteluja.

Esimerkki:

x) -1, 5, parilliset luvut (PARILLISET), kokonaisluvut ($\mathbb{Z}$), luonnolliset luvut ($\mathbb{N}$)

Kuva 1: Käsitteiden väliset suhteet

	PARILLISET	$\mathbb{Z}$	$\mathbb{N}$
-1	$\notin$	$\in$	$\notin$
5	$\notin$	$\in$	$\in$
PARILLISET		$\subset$	$\cap$
$\mathbb{Z}$			$\supset$
$\mathbb{N}$			

-1 $\notin$ PARILLISET
 -1 $\in \mathbb{Z}$
 -1 $\notin \mathbb{N}$
 ... (kolme suhdetta)
 PARILLISET $\subset \mathbb{Z}$
 PARILLISET $\cap \mathbb{N}$
 $\mathbb{Z} \supset \mathbb{N}$

- a) $(n \log_2 n)$, $O(n^2)$, $\Theta(n)$, $\Omega(n)$. (2 p)
- b) $(n+9)(n-20)$, $(n \log n)$, $(\frac{1}{3}n^3 - \frac{2}{9}n + \frac{4}{27}n^7)$, $\Theta(n^{\frac{20}{9}})$, $O(n \log_2 n^2)$, $\Omega(n^2 \sqrt{n})$. (3 p)
- c) $O(3(\log_2 n) + 2^{\log_3 n})$, $\Theta(3^2 n + 2^3 n + 2 \cdot 3)$, $\Omega(2^n + n^2)$, $\Theta(2^{2n} + 2^{3n})$, $\Omega(13 + 7 \log_4 n)$, $\Theta(2^{n+2} - 2)$. (4 p)

3. a) Mitkä järjestämisalgoritmeista quicksort, heapsort, insertion sort ja bucket sort toimivat taulukoille ja mitkä linkitetuille listoille? Mitkä voi suoritusajan kertaluokkaa heikentämättä muuttaa käyttämään toista rakenteista? Miten? (2 p)
- b) Miksi quicksort-algoritmin perusversiosta ei ole tehty vakaata? Miten siitä voisi tehdä vakaan heikentämättä ajankäytön kertaluokkaa? Muistia uusi algoritmi saa tarvittaessa käyttää alkuperäistä enemmän. (2 p)
- c) Miksi tietyn avaimen etsiminen keosta ei ole kovin tehokasta? Miksi keon määritelmää ei muuteta hakujen nopeuttamiseksi? (2 p)
- d) Alla on keon rakentava funktio `Build-Heap()` prujusta. Miksi funktion `Heapify()` kutsuminen aloitetaan puolivälistä ja silmukka etenee laskevasti kohti yhtä? Toimisiko jokin muukin alkukohta tai järjestys? (3 p)

```
Build-Heap(A)
  A.heapsize := A.length
  for i := floor(A.length / 2) downto 1 do
    Heapify(A, i)
```

4. a) Piirrä trie, johon on lisätty merkkijonot "Jouko", "joko", "joogaa", "tai", "jodlaa", "hän", "taitaakin", "olla", "taitava", "ja", "tasapainoinen" ja "taiteilija" tässä järjestyksessä. Voit halutessasi käyttää tiivistettyä esitysmuotoa, kunhan selität sen. (2 p)
- b) Järjestä merge sort -algoritmia käyttäen alkiot 76, 22₁, 100, 22₂, 44, 99₁, 26, 66, 99₂. Alkiot ovat alkujaan annetussa järjestyksessä. Käytä alaindeksiä erottamaan samanarvoiset alkiot toisistaan. (2 p)
- c) Komponentissa, jonka parissa työskentelet, pitää tehdä hakuja puurakenteesta. Ensimmäisenä mieleesi tulee käyttää punamustaa puuta. Huomaat kuitenkin, että heikompiakin rajoite puun tasapainolle riittäisi. Työpäivän jälkeen rentoudut työtoverisi ja ystäväsi Adttu Pedkuleen ja hänen veljensä Henrwin seurassa. Pulmasi tulee puheeksi ja Henrwi saa kertakaikkisen oikeen älyvväläyksen ja kuningasidean luoda uusi tietorakenne, punavihermusta puu.
- "Yksinkertaisesti vain lisätään vihreä sallituksi väriksi punaisen ja mustan lisäksi. Kopiaoidaan invariantin säännöt, joissa mainitaan sana "punainen" ja korvataan kopiassa sana "punainen" sanalla "vihreä" ja homma on siinä! Pieniä teknisiä yksityiskohtia ehkä pitää hiukan viilata."*, Henrwi selittää. Epätasapainoa sallitaan kuulemma 1,5-kertaisesti (ja samalla haut hidastuvat *"kertaluokaltaan puolitoistakertaiseksi"*) punamustaan puuhun verrattuna. Tuollaista juuri etsitkin!
- Maanantaina töissä löydät taskustasi tuopin alusen, johon kirjoitit punavihermustan puun perusajatuksen. Perjantaina niin hyvältä kuulostanut idea alkaa epäilyttää.
- Mikä on Henrwin ohjeiden mukaan määritelty punavihermustan puun invariantti (sen säännöt eli rajoitteet)? Salliiko invariantti todella punamustaa puuta epätasapainoisemman rakenteen kuitenkin rajoittaen epätasapainon enintään 1,5-kertaiseksi punamustaan puuhun verrattuna? Jos ei, voiko rajoitteiden *"pienet tekniset yksityiskohdat"* *"viilata"* niin, että ratkaisu toimii? Mitä voit sanoa Henrwin kommentista hakujen suorituskyvyn kertaluokasta? (2 p)

5. Funktio `flood_fill()` toimii kaksiulotteisessa ruudukossa. Mitä se tekee? Funktio muistuttaa toiminnaltaan erästä prujun algoritmia ja onkin sen erikoistus. Mikä algoritmi? Anna funktioiden `flood_fill()` ja `lisää_jonoon()` ajankäyttö kertaluokkamerkinnöin. Olisiko mahdollista tehdä kertaluokaltaan nopeampi ratkaisu? Jokin funktion toiminnassa tekee siitä kertaluokassaan verrattain hitaan. Mikä? (4 p)

```
flood_fill( Taso, alku_x, alku_y, vanha, uusi )
    if vanha == uusi or Taso[alku_y][alku_x] != vanha
        return

    Taso[alku_y][alku_x] = uusi
    jono = new Jono
    jono.push( <x,y> ) # tallenna pari (tuple) jonoon
    while not jono.empty()
        <nyk_x, nyk_y> = jono.pop() # lue pari jonosta
        lisää_jonoon( Taso, jono, nyk_x, nyk_y -1)
        lisää_jonoon( Taso, jono, nyk_x, nyk_y +1)
        lisää_jonoon( Taso, jono, nyk_x -1, nyk_y)
        lisää_jonoon( Taso, jono, nyk_x +1, nyk_y)

lisää_jonoon( Taso, jono, x, y, vanha, uusi )
    if vasen_reuna <= x <= oikea_reuna and alareuna <= y <= yläreuna
        if Taso[y][x] == vanha
            Taso[y][x] = uusi
            jono.push( <x,y> )
```

6. Lex Tirakan myötä kaikilla oppilaitoksilla on velvollisuus tarkkailla opiskelijoiden viestintää kouluruoasta purnauksen varalta. Sinulla on suuri kunnia palvella maatasi kehittämällä automaattista purnauksen tunnistavaa valvontajärjestelmää SPuTunIK II:ta (Suuri Purnauksen Tunnistava Isänmaallinen Konetoveri II). Eri välineistä (sähköposti, puhelin, savumerkit, ruokajonon valvontakameran kuva) kaapatuista ja tekstimuotoon muutetuista viesteistä tunnistetaan sanoja. Erittäin monimutkainen ja huippusalainen algoritmi laskee sanojen esiintymistiheydestä, läheisyydestä ja opiskelijan seurantahistoriasta viestin purnauksellisuustodennäköisyyden, jonka perusteella poliisi voi toimia. Ensimmäisen, perillä luodun suoraa sanavertailua tekevän SPuTunIK I:n opiskelijat päihittivät siirtymällä vastarintaviestinnässään anagrammien käyttöön. SPuTunIK II:n pitää tunnistaa sana, vaikka kirjainten järjestystä olisi muutettu. Sanoja tunnistavalla osajärjestelmällä on lista sanoista. Jokaiseen sanaan liittyy tietoa (mm. sanan vaarallisuus, vallankumouksellisuus, jne.), jota salainen algoritmi käyttää. Sanantunnistajan pitää palauttaa annettua merkkijonoa vastaavaan sanaan liittyvä tieto. Merkkijono voi olla sanan muunnos, mikä pitää ottaa huomioon. Osajärjestelmän ei tarvitse ymmärtää tai käsitellä sanaan liittyvää tietoa. Sanalistoja päivitetään harvoin.

- Miten suunnittelet sanantunnistusjärjestelmän? Sanojen hakuun liittyy usein esiintyvä ongelmatilanne, jonka oikeaa ratkaisua ei ole määritelty. Mikä? (3 p)
- Opiskelijat ovat päihittäneet SPuTunIK II:n!! He lisäävät (anagrammimuotoiseen) sanaan ylimääräisiä merkkejä, esim. "surkea" + "xzg" = "xrsuzkega". Miten suunnittelet SPuTunIK III:n sanantunnistusjärjestelmän siten, että se toimii kirjainten järjestyksen muuttamisesta ja kirjainten lisäämisestä huolimatta. (3 p)

Palaute

Ole hyvä, ja anna kurssista palautetta. Tämä ei ole tenttitehtävä. Palautetta voit lisäksi antaa tentin jälkeen lähettämällä sähköpostia osoitteeseen tiraka@cs.tut.fi.

• Jos tämä et ollut ensimmäisessä tentissä:

- Monesko kerta tämä on, kun käyt tirakan tai UDS:n tentissä?
- Mitä mieltä ole kurssin järjestelyistä?
- Kuinka usein kävit luennoilla? Miksi kävit niin usein/harvoin?
- Kuinka usein kävit opintopiirissä? Miksi kävit niin usein/harvoin?
- Oliko luennoista tai opintopiiristä hyötyä?
- Oli Trakla hyödyllinen?
- Mitä pitäisi parantaa kurssin järjestelyissä tai mikä toimi hyvin?

• Jos olit ensimmäisessä tentissä:

- Monesko kerta tämä on, kun käyt tirakan tai UDS:n tentissä?
- Mikä oli tentin paras kysymys? Miksi?
- Mikä oli tentin huonoin kysymys? Miksi?
- Mikä oli tentin olennaisin kysymys? Miksi?
- Mikä oli tentin epäolennaisin kysymys? Miksi?
- Mikä oli tentin helpoin kysymys? Miksi?
- Mikä oli tentin vaikein kysymys? Miksi?
- Onko sinulla muita kommentteja tentistä tai kurssista? Miksi edellisen virkeen perässä ei lue "Miksi?"?

Kerro mielipiteesi kurssista ja sen järjestelyistä. Kuinka usein kävit luennoilla ja opintopiirissä? Miksi kävit niin usein/harvoin? Oliko niistä hyötyä? Oliko Trakla mielestäsi hyödyllinen? Mitä pitäisi parantaa kurssin järjestelyissä tai mikä toimi hyvin?