Tietorakenteet, ensimmäinen kurssikoe 1.3.2010

Voit tehdä tehtävät 1 ja 2 samalle konseptille. **Tehtävään 3 tulee vastata erillisille konseptille. Tehtävät 1 ja 2 palautetaan eri pinoon kuin tehtävä 3**.

Kirjoita jokaiseen palauttamaasi konseptiin kurssin nimi, kokeen päivämäärä, nimi, nimikirjoitus ja opiskelijanumero.

Tehtävissä joissa pyydetään algoritmia, voit käyttää luentomonisteen pseudokoodityylin lisäksi muitakin "ymmärrettäviä" pseudokoodityylejä tai oikeista kielistä Javaa, Pythonia, C:tä, C++:aa tai jotain muuta selkeälukuista imperatiivista kieltä. Älä kuitenkaan käytä kielten eksoottisia piirteitä vaan pidä algoritmit pseudokoodimaisina.

Huomaa, että **Laskuharjoitus 7** pidetään 15.3 alkavalla viikolla. Tehtävät tulevat kurssisivulle viimeistään ma 8.3.

1. (6 p) Määritä seuraavien algoritmien aikavaativuus ja tilavaativuus. Ilmaise vaativuudet (niin tässä kuin kokeen muissakin tehtävissä) $\mathcal{O}$-merkinnän avulla ja perustele vastaukset lyhyesti.

   (a) Algoritmi `TutkiSumma` tarkistaa, onko taulukossa `T` kahta lukua, jotka muodostavat annetun summan. Taulukossa on `n` lukua.

   ```
   TutkiSumma(T, summa)
      for i = 1 to n-1
          for j = i + 1 to n
              if T[i] + T[j] == summa
                  return True
      return False
   ```

   (b) Algoritmi `KaannaTaulukko` kääntää taulukon `T` sisällön pinon avulla. Taulukossa on `n` lukua. Voit olettaa, että pino on toteutettu järkevällä tavalla.

   ```
   KaannaTaulukko(T)
      for i = 1 to n
          Push(pino, T[i])
      for i = 1 to n
          T[i] = Pop(pino)
   ```

   (c) Algoritmi `OnkoListassa` tarkistaa, onko listassa solmua, jolla on tietty arvo. Algoritmille annetaan viite listan ensimmäiseen solmuun sekä etsittävä arvo. Listassa on `n` solmua.

   ```
   OnkoListassa(solmu, arvo)
      if solmu == Nil
          return False
      if solmu.value == arvo
          return True
      else
          return OnkoListassa(solmu.next, arvo)
   ```

Käännä paperia

2. (8 p) Linkitetty lista

   (a) (6 p) Olkoon annettu linkitetty lista, jonka jokainen solmu sisältää yhden kokonaisluvun. Luvut ovat listassa muuten nousevassa suuruusjärjestyksessä, mutta etukäteen tiedetään, että yksi listan luvuista on väärässä paikassa. Esimerkiksi listan luvut voisivat olla järjestyksessä $1, 5, 7, 9, 13, 18, 14, 15$, jolloin luku 18 on väärässä paikassa.

   Kirjoita algoritmi, joka etsii väärässä paikassa olevan alkion, poistaa sen väärältä paikalta ja lisää alkion listaan oikealle paikalleen.

   Käytössäsi ei ole valmiina kurssilla esiteltyjä listaoperaatioita `search`, `insert`, `pred`, `succ` ja `delete`, vaan tarvittava koodi on kirjoitettava kokonaan itse.

   Huomaa, että listassa olevat luvut ja lukujen määrä voivat olla eri kuin yllä annetussa esimerkissä. Myöskään väärässä kohdassa olevan alkion paikkaa ei tiedetä etukäteen. Voit itse valita, käytätkö yhteen vai kahteen suuntaan linkitettyä listaa, rengaslistaa vai päättyvää listaa, tunnussolmulla varustettua listaa vai ilman tunnussolmua olevaa listaa. Kerro kuitenkin selvästi, millaista listavariaatiota käytät.

   (b) (2 p) Analysoi algoritmisi aika- ja tilavaativuus. Perustele vastauksesi.

3. (10 p) Binäärihakupuu

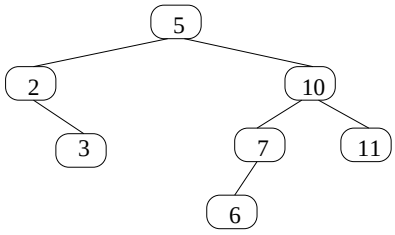
   (a) (2 p) Piirrä korkein mahdollinen AVL-puu, johon on talletettu avaimet 1-12.

   Perustele miksi piirtämäsi puu on AVL-puu. Perustele myös miksi puusi on binäärihakupuu.

   (b) (3 p) Tee algoritmi, joka tutkii syötteenä olevasta *normaalista* binäärihakupuusta täyttääkö se AVL-puun tasapainoehdon. Algoritmin syötteenä olevan puun solmuilla siis ei ole korkeutta kertovia `height`-attribuutteja.

   Selitä ensin algoritmisi toimintaperiaate tekstinä. Tämän jälkeen kirjoita algoritmi pseudokoodina. Analysoi myös algoritmisi aika- ja tilavaativuus.

   (c) (2 p) Näytä miten allaolevaan AVL-puuhun lisätään avaimet 9, 4 ja 8. Lisäykset tehdään luetellussa järjestyksessä.



   Näytä puun lopputilanne kunkin lisäyksen tultua suoritetuksi sekä välitilanteet, joissa tehdään jokin tasapainotusoperaatio.

   (d) (3 p) Perustele minkä takia AVL-puun tasapainon ylläpitäminen ei lisää oleellisesti avainten lisäyksen ja poistamisen vaativuutta.

   Selitä ensin ideatasolla miten lisäys ja poisto normaaliin binäärihakupuuhun tapahtuu, ja perustele mikä on operaatioiden vaativuus. Esittele sitten ideatasolla miten lisäys ja poisto AVL-puuhun tapahtuu ja vertaa operaatioiden vaativuutta tasapainoittamattoman puun lisäykseen ja poistoon. Tässä tehtävässä ei ole tarvetta esitellä operaatioita pseudokooditasolla.