Tietorakenteet, laskuharjoitus 5, 16.-20.2.

Muista TRAKLA:n toisen kierrosten tehtävien deadline 14.2!

Tehtävässä 2-5 riittää pseudokoodi, mutta voit myös tehdä esim. Python-toteutuksen, jolloin pohjaksi voi ottaa osoitteessa `www.cs.helsinki.fi/u/mluukkai/tirak2010/python/` olevan puukoodin. Tehtävät 2-4 kannattanee ratkaista rekursion avulla.

1. Alunperin tyhjään binäärihakupuuhun lisätään monisteen insert-operaatiota käyttäen avaimia seuraavassa järjestyksessä: 2, 0, 3, 7, 9, 1, 5, 6 ja 8. Näytä miten puu kasvaa lisäysten seurauksena. Piirrä siis kuva tilanteesta jokaisen (tai ainakin melkein jokaisen) lisäyksen jälkeen.

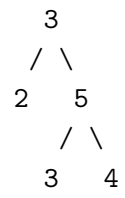
   Simuloi delete-operaation toimintaa tuloksena olevevalle puulle parametrina solmu, jossa avain 3. Poista tämän jälkeen avain 9. Tämän jälkeen avain 2. Ja lopuksi avain 5. Piirrä jokaisen operaation jälkeen tuloksena oleva puu.

2. Esitä algoritmi joka saa syötteekseen binääripuun juurisolmun ja laskee solmujen lukumäärän. Seuraavan tehtävän esimerkkipuulla algoritmi palauttaisi 5.

   Mikä on algoritmisi aika- ja tilavaativuus?

3. Esitä algoritmi, joka laskee, kuinka monta kertaa jokin alkio esiintyy binääripuussa (puun siis ei tarvitse olla hakupuu). Algoritmille annetaan syötteksi viite binääripuun juurisolmuun sekä etsittävä alkio.

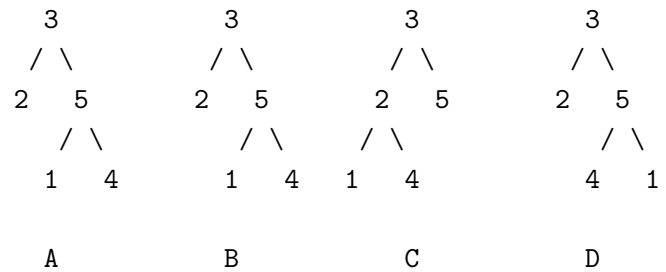
   Esimerkiksi jos puu on seuraava ja etsittävä alkio on 3, algoritmin vastaus on 2.



   Mikä on algoritmisi aika- ja tilavaativuus?

4. Esitä algoritmi, joka tarkistaa, ovatko kaksi binääripuuta samanlaiset. Algoritmille annetaan viite kummankin binääripuun juurisolmuun.

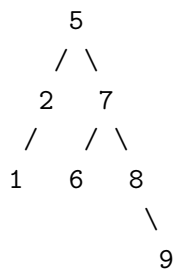
   Esimerkiksi seuraavassa kuvassa binääripuut A ja B ovat samanlaiset. Binääripuut A ja C eivät ole samanlaiset, koska niiden rakenne on erilainen. Binääripuut A ja D eivät ole samanlaiset, koska solmuissa olevat luvut eivät ole samat.



5. Esitä algoritmi, joka saa syötteekseen binääripuun ja tulostaa solmut tasojärjestyksessä, eli ensin tason 0 solmut, sitten tason 1 solmut, tason 2. solmut ...

   Mikä on algoritmisi aika- ja tilavaativuus?

   Lisäviritys edelliseen (ei vaadita laskarirastiin): Laajenna algoritmiasi siten, että se tekee puulle yksinkertaisen visualisoinnin. Periaate on seuraava, jos puu on esim.



   Tulostaa algoritmi

```
_5_
_2  _7_
1  6  8_
9
```

   Solmut tulostataan tasoittain. Solmun yhteyteen tulostetaan tieto siitä onko solmulla lapsia. Merkki _ solmun vasemmalla puolella tarkoittaa että solmulla on vasen lapsi ja merkki _ solmun oikealla puolella tarkoittaa, että solmulla on oikea lapsi. Esim. puussa 2:lla on vain vasen lapsi, joten tulostukseen tulee _2.

6. 2-3-puu on yksi variaatio tasapainotetusta puusta. Puu on 2-3-puu jos seuraavat ehdot ovat voimassa:

   - Jokaisella solmulla lehtisolmuja lukuunottamatta on 2 tai 3 lasta.
   - Kaikki lehtisolmut ovat samalla tasolla

   Tällä kertaa emme välitä puuhun talletetuista avaimista, vaan päähuomio on puun muodolla ja erityisesti sen korkeuden ja solmumäärän suhteella.

   (a) Piirrä suurin ja pienin mahdollinen 2-3-puu, jonka korkeus on 1, 2 ja 3 (eli siis kaikille korkeuksille omaa suurin ja pienin puunsa).
   (b) Oletetaan, että 2-3-puun korkeus on h. Näytä mikä on solmumäärän ala- ja yläraja korkeuden suhteen.
   (c) Oletetaan, että 2-3-puussa on n solmua. Näytä mikä on puun korkeuden ala- ja yläraja solmumäärän suhteen.

   Vihje: b- ja c-kohtien päättelyt voidaan tehdä täysin samalla tekniikalla kuin Lauseiden 1 ja 2 todistukset monisteen sivuilla 133 ja 137. Hyödyllistä on myös muistaa geometrisen sarjan summakaava
   $\sum_{i=0}^{k} a^k = \frac{a^{k+1}-1}{a-1}$ missä $a \neq 1$.