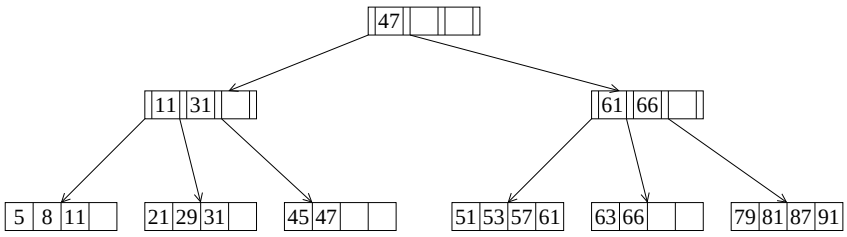
Tietorakenteet, laskuharjoitus 7, 15.-19.3.

Tämän viikon laskareissa on useita vaihtoehtoisia tehtäviä. Laskarien maksimirastimäärä on 6. Voit tehdä haluamasi tehtävät. Tehtävistä 1-6 kustakin yksi rasti

1. Lue osoitteesta `www.cs.helsinki.fi/u/mluukkai/tirak2010/b.pdf` löytyvä Kerttu Pollari-Malmin B+-puita esittelevä moniste

2. Esitä tärkeimmät välivaiheet, kun allaolevaan B^+-puuhun lisätään avaimet 33, 25, 19, 43 ja 49.



3. Aluksi tyhjään B^+-puuhun lisään avaimet 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 ja 12 (tässä järjestyksessä). Esitä lopputulos ja tärkeimmät välivaiheet, kun solmun maksimikoko on $2t = 4$.

4. Esitä tärkeimmät välivaiheet kun tehtävän 2 B^+-puusta poistetaan avaimet 5, 8 ja 47.

5. Poista vielä avain 45 edellisen tehtävän jälkeen tuloksena olevasta puusta. Näytä jälleen kaikki mielenkiintoiset välivaiheet

6. Pollari-Malmin monisteessa esitetyssä B^+-puun määritelmässä sisäsolmun lasten maksimilukumäärä on sama kuin lehden avainten maksimilukumäärä, mutta näin ei ole pakko menetellä. Vaihtoehtoisesti voitaisiin sijoittaa jokaiseen solmuun enimmillään niin paljon tietoa, kuin yhteen levymuistin lohkoon mahtuu. Ajatellaan tilannetta, jossa

   - lohkon koko on 4 kilotavua
   - avaimen koko on 8 tavua
   - lapsiosoittimen koko on 4 tavua
   - datakenttä vie 64 tavua.

   Mikä on 100 miljoonaa alkiota sisältävän B^+-puun suurin mahdollinen korkeus, jos solmujen koot valitaan tämän mukaisesti? Jos yksi levyhaku vie keskimäärin 10 millisekuntia, niin kuinka kauan yksittäisen avaimen haku keskimäärin kestää, jos oletataan juurisolmun sijaitsevan valmiiksi keskusmuistissa ja muuhun kuin levyhakuihin menevä aika jätetään huomioimatta? Mikä puolestaan on 100 miljoonaa alkiota sisältävän AVL-puun suurin mahdollinen korkeus, ja mikä olisi keskimääräinen hakuaika, jos edellä esitetyssä tilanteessa hakemisto toteutettaisiin levymuistissa AVL-puuna (jolloin levymuistin lohkorakennetta ei hyödynnetä)?

7. **Rotation-peli:** 3 rastia pseudokoodi, 6 rastia toteutus jollakin kielellä, toteutus voi olla luonnosmainen, esim. syötteiden virhetarkastuksia ei kannata tehdä. Voit myös rajoittaa toteutuksen käsittelemään pelilautoja joiden koko on esim. max 4×4

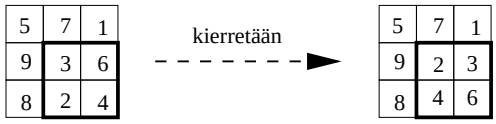
   - pelilautana on $n \times n$ -ruudukko, jokaisessa ruudussa on jokin numero väliltä $1 \ldots n \times n$ ja kukin luku esiintyy ruudukossa vain kerran
   - esim. jos pelilauta kokoa 3×3, niin ruudukossa luvut 1-9
   - pelin alussa luvut ovat ruudukossa satunnaisessa järjestyksessä, esim:

5	7	1
9	3	6
8	2	4

   - pelaajan on tarkoitus järjestää luvut suuruusjärjestykseen:

1	2	3
4	5	6
7	8	9

   - peli etenee *kiertoja* tekemällä, esim:



   - kierto-operaatio siis liikauttaa neljää lukua yhden askeleen "myötäpäivään"
   - kierto voidaan suorittaa missä kohtaan pelilautaa tahansa
   - kierto on ainoa tapa millä lukuja voidaan liikutella

   Suunnittele algoritmi joka etsii ratkaisun rotation-pelille

   - syötteenä pelilauta joka on annettuna $n \times n$-taulukkona
   - algoritmi kertoo mitkä kierrot suorittamalla peli ratkeaa
   - **huom:** onko varmaa että ratkaisu löytyy aina?

   Arvioi algoritmisi aika- ja tilavaativuutta.

   Löytääkö algoritmisi ratkaisun, joka sisältää pienimmän mahdollisen määrän kiertoja? Jos ei niin, miten algoritmiasi tulisi muuttaa jotta pienimmän kiertomäärän ratkaisu löytyisi?

   Huom: jos ratkaisusi sisältää hyvin syvän rekursion, joudut kasvattamaan ajoaikaisen pinon kokoa välttyäksesi `java.lang.StackOverflowError`:ilta. Pinoa voi kasvattaa antamalla käynnistyksen yhteydessä parametrin `-Xss`, jonka yhteydessä määritellään ajonaikaisen pinon koko. Esim. `java -Xss100m Rotation` määrittelee pinon kooksi 100 megatavua.

   Älä suorita ohjelmaasi koneissa melkki, melkinkari, melkinpaasi

8. **Taikaneliö:** 3 rastia pseudokoodi, 6 rastia toteutus jollakin kielellä, toteutus voi olla luonnosmainen, esim. mitään syötteiden virhetarkastuksia ei kannata tehdä

   Taikaneliö on $n \times n$ -ruudukko, joka sisältää kokonaisluvut $1...n^2$. Vaatimuksena on, että lukujen summa on sama jokaisella ruudukon pystyrivillä ja vaakarivillä sekä molemmilla lävistäjillä.

   Seuraavassa on yksi 4×4 -taikaneliö, jossa yhteinen summa on 34. Esimerkiksi taikaneliön toisella pystyrivillä 2+7+10+15 = 34, neljännellä vaakarivillä 1+15+4+14 = 34 ja vasemmasta yläkulmasta lähtevällä lävistäjällä 8+7+5+14 = 34.

   ```
   8  2  13 11
   9  7  12  6
   16 10  5  3
   1  15  4 14
   ```

   Tee algoritmi tai toteuta ohjelma, joka laskee, kuinka monta erilaista $n \times n$ -taikaneliötä on olemassa. Kuinka nopeaksi saat ohjelmasi tapauksessa n=4?

   Älä suorita ohjelmaasi koneissa melkki, melkinkari, melkinpaasi