Tietorakenteet 2018
Harjoitukset 13, ratkaisut (Viikko 49)

1. Puista yleisesti:
   - Puu on tietorakenne, jossa alkioiden suhde on hierarkkinen: Puu T on solmujen joukko, joka tallettaa alkiot vanhempi-lapsi -suhteessa
   - Jokaisella solmulla on vanhempi ja mahdollisesti lapsia
   - Päällimmäisin solmu on juuri
   - Tärkeitä metodeja ovat esimerkiksi: root(), parent(v), children(v), isInternal(v), isExternal(v), isRoot(v), size(), isEmpty()

   Binääripuu:
   - Binääripuu on järjestetty puu, jossa solmuilla on joko 2 tai ei yhtään lasta (lasten välillä on järjestys, vasen ja oikea)
   - Tätä kutsutaan myös aidoksi binääripuuksi; epäaidossa binääripuussa solmuilla voi olla 0, 1 tai 2 lasta.
   - Metodeja: leftChild(), rightChild()

   Binäärihakupuu:
   - Binäärihakupuu on binääripuu, jossa lehtisolmut toimivat vain paikanpitäjinä (Jos ei haluta käyttää paikanpitäjälehtiä, voitaisiin vaihtoehtoisesti myös määritellä, että binäärihakupuu on järjestetty puu, jossa solmuilla voi olla 0, 1 tai 2 lasta)
   - Binäärihakupuussa pätee lisäksi: jokainen sisäsolmu tallettaa tietoyksikön (k,e), ja vasempaan alipuuhun talletetut avaimet ovat pienempiä kuin k ja oikeaan alipuuhun talletetut avaimet ovat suurempia tai yhtäsuuria kuin k.

2. a) Valintalajittelu
   - Tapa lajitella sekvenssi niin, että jokaisella kierroksella valitaan lajittelemattoman sekvenssin pienin alkio, ja viedään se lajitellun sekvenssin loppuun
   - Aikaa vievä osuus on siis hakea sekvenssistä kierroksen pienintä alkiota
   - O(n^2)
   - Prioriteettijonoa käyttäen valintalajittelu voidaan tehdä seuraavasti (käytetään lajittelematonta prioriteettijonoa): siirretään alkiot lajiteltavasta sekvenssistä S prioriteettijonoon P, poistetaan yksi kerrallaan pienin alkio *removeMinElement*-operaation avulla prioriteettijonosta P ja viedään takaisin alkuperäiseen sekvenssiin S:

$$S = (4, 1, 6, 7, 2),\ P = ()$$

...siirretään alkiot...

$$S = (),\ P = (4, 1, 6, 7, 2)$$
$$S = (1),\ P = (4, 6, 7, 2)$$
$$S = (1, 2),\ P = (4, 6, 7)$$
$$S = (1, 2, 4),\ P = (6, 7)$$
$$S = (1, 2, 4, 6),\ P = (7)$$
$$S = (1, 2, 4, 6, 7),\ P = ()$$

Lisäyslajittelu

- Tapa lajitella sekvenssi niin, että jokaisella kierroksella valitaan lajittelemattoman sekvenssin ensimmäinen alkio, ja viedään se oikealle paikalle lajitellussa sekvenssissä
- Aikaa vievä osuus on siis etsiä alkiolle oikea paikka lajitellussa sekvenssissä
- O(n^2)
- Prioriteettijonoa käyttäen (käytetään lajiteltua prioriteettijonoa): poistetaan alkiot yksi kerrallaan lajiteltavasta sekvenssistä S ja lisätään prioriteettijonoon P oikeille paikoille, lopuksi tyhjennetään prioriteettijono P *removeMinElement*-operaation avulla eli poistetaan yksi kerrallaan jonon ensimmäinen alkio ja viedään se takaisin sekvenssiin S:

$$
\begin{aligned}
S = (4,1,6,7,2),\ P &= () \\
S = (1,6,7,2),\ P &= (4) \\
S = (6,7,2),\ P &= (1,4) \\
S = (7,2),\ P &= (1,4,6) \\
S = (2),\ P &= (1,4,6,7) \\
S = (),\ P &= (1,2,4,6,7) \\
&\text{...siirretään alkiot...} \\
S = (1,2,4,6,7),\ P &= ()
\end{aligned}
$$

b) Lomituslajittelu
   - Perustuu hajoita ja hallitse -menetelmään (ratkaistaan osaongelmat)
   - Lajittelu on luontevaa toteuttaa rekursiivisena
   - Jos lajiteltavassa sekvenssissä S on vähintään 2 alkiota, jaetaan se kahteen sekvenssiin S_1 ja S_2
   - Lajitellaan rekursiivisesti sekvenssit S_1 ja S_2
   - Sijoitetaan alkiot takaisin sekvenssiin S lomittamalla lajitellut sekvenssit S_1 ja S_2
   - O(n log(n)) ajassa toimiva lajittelu, kun sekvenssi kooltaan n
   - Esimerkki: katso harjoitusten 9 tehtävän 7 ratkaisu

   Pikalajittelu
   - Perustuu hajoita ja hallitse -menetelmään
   - Lajittelu on luontevaa toteuttaa rekursiivisena
   - Jos lajiteltavassa sekvenssissä S on vähintään 2 alkiota, valitaan pivot-alkio x, jonka perusteella sekvenssi jaetaan kolmeen osaan: L - alkiota x pienemmät alkiot, E - alkion x kanssa yhtä suuret alkiot ja G - alkiota x suuremmat alkiot
   - Lajitellaan rekursiivisesti sekvenssit L, E ja G
   - Sijoitetaan alkiot takaisin sekvenssiin S järjestyksessä ensin sekvenssin L alkiot, sitten E ja sitten sekvenssin G alkiot
   - Keskimäärin ajassa O(n log(n)) toimiva lajittelu, pahin tapaus kuitenkin O(n^2), jos sekvenssi on jo nousevassa järjestyksessä ja pivot-alkioksi valitaan aina viimeinen alkio.
   - Esimerkki: katso harjoitusten 9 tehtävän 7 ratkaisu

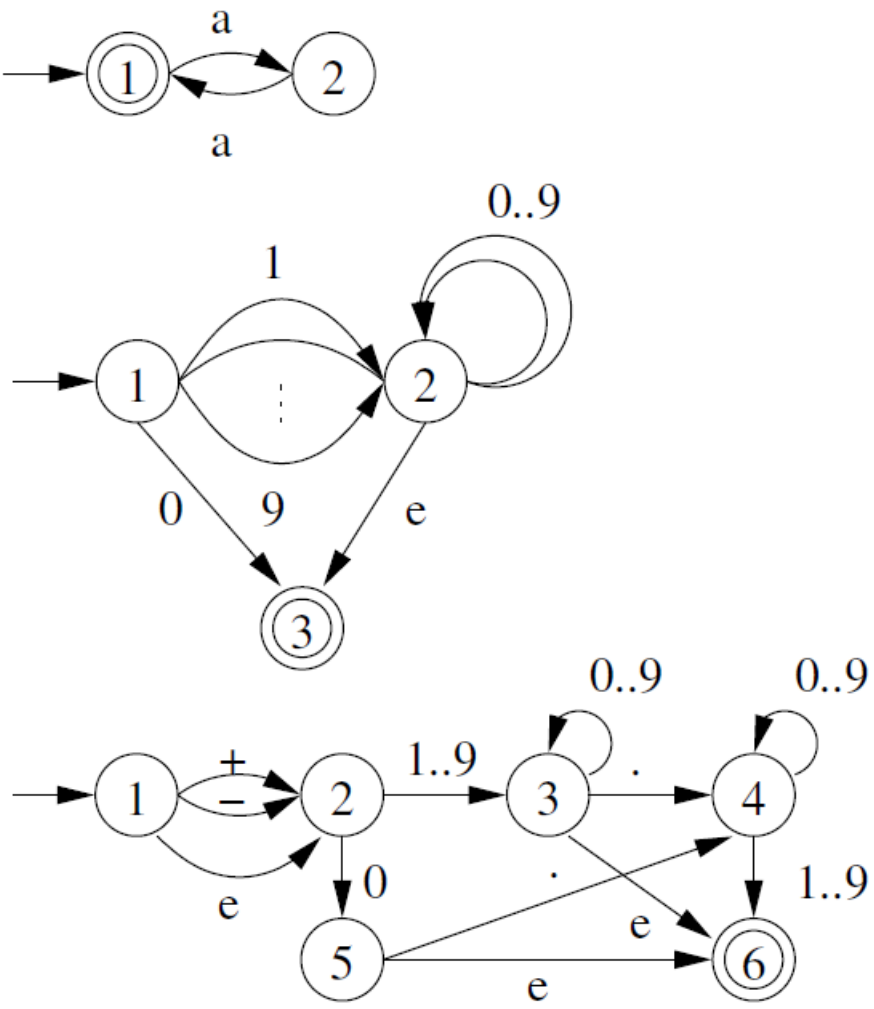
3. a) $(aa)^*$

b) $0 + (1 + 2 + \cdots + 9)(0 + 1 + \cdots + 9)^*$

c)

$$
\begin{aligned}
0 + (\text{'+'} + \text{'-'})0 + (\text{'+'} + \text{'-'})\big(&0.(0 + \cdots + 9)^*(1 + \cdots + 9) \\
&+ (1 + \cdots + 9)(0 + \cdots + 9)^* \\
&+ (1 + \cdots + 9)(0 + \cdots + 9)^* \\
&\quad .(0 + \cdots + 9)^*(1 + \cdots + 9)\big) \\
+\big(&0.(0 + \cdots + 9)^*(1 + \cdots + 9) \\
&+ (1 + \cdots + 9)(0 + \cdots + 9)^* \\
&+ (1 + \cdots + 9)(0 + \cdots + 9)^* \\
&\quad .(0 + \cdots + 9)^*(1 + \cdots + 9)\big)
\end{aligned}
$$

4. Oheisessa kuvassa.



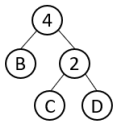
5. DATA STRUCTURES AND ALGORITHMS IN JAVA

6. Vastaus voi vaihdella riippuen samoja avaimia sisältävien alkioiden käsittelystä prioriteettijonossa. Merkkien frekvenssit:

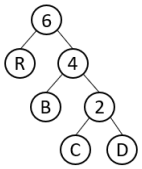
A	B	C	D	R
5	2	1	1	2

Vaihe 1. Prioriteettijono $Q = ((A, 5), (B, 2), (C, 1), (D, 1), (R, 2))$. Poistetaan kaksi pienimmän avaimen alkiota ja ne muodostavat puun, joka puolestaan lisätään prioriteettijonoon. Siis, solmut, joissa ovat merkit C ja D, muodostavat puun:

Vaihe 2. Prioriteettijono $Q = ((A, 5), (B, 2), (\text{"C, D"}, 2), (R, 2))$. Nyt edellä saatu puu ja solmu B muodostavat puun, jossa B on vasempana lapsena ja edellisen kohdan puu oikeana lapsena.



Vaihe 3. Prioriteettijono $Q = ((A, 5), (\text{"B, C, D"}, 4), (R, 2))$. Edellä saatu puu ja solmu R muodostavat puun, jossa R on vasempana lapsena ja edellisen kohdan puu oikeana lapsena.



Vaihe 4. Prioriteettijono $Q = ((A, 5), (\text{"R, B, C, D"}, 6))$. Edellä saatu puu ja solmu A muodostavat puun, jossa A on vasempana lapsena ja edellisen kohdan puu oikeana lapsena.