Kodutöö 5

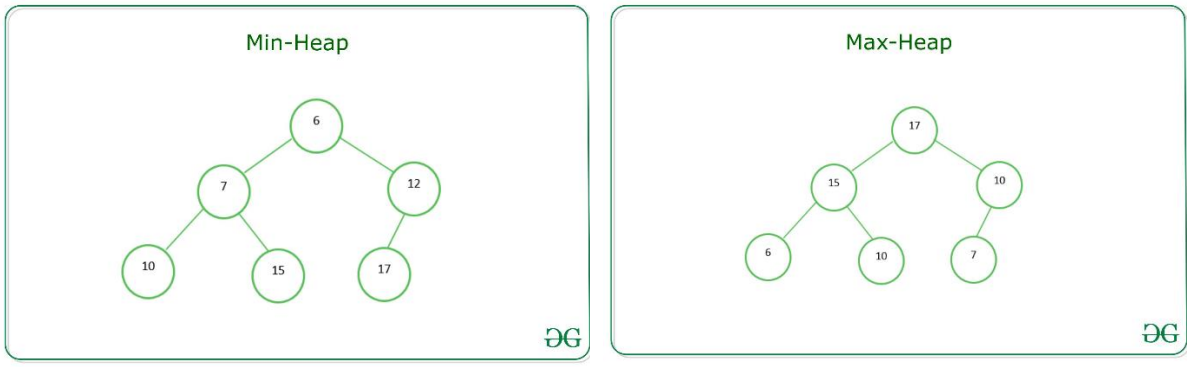
Ülesanne 1 Binaarpuu

Binaarpuu andmestruktuur koosneb kolmest osast: andmed ja viited oma kahele lehele.

```
struct node
{
 int data;
 struct node *left;
 struct node *right;
};
```

Ülesanne 2 Kuhja struktuuri analüüs

Kuhi on täielik binaarpuu, kus kõik tipud või lehed on täidetud. Min-Heap väärtused kasvavad tüvest ja Max-heap väärtused kahanevad tüvest.



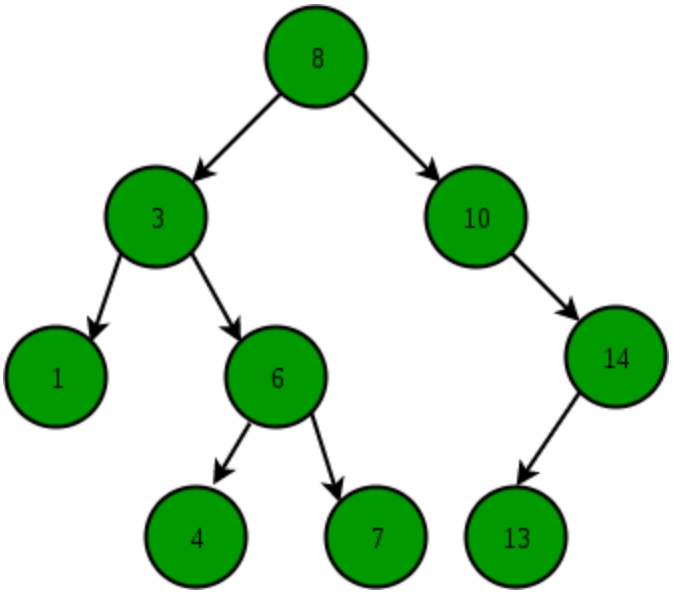
Joonis 1 Min-Heap ja Max-heap joonis https://www.geeksforgeeks.org/difference-between-min-heap-and-max-heap/

Kuhja elementide lisamise või kustutamise ajalinekompleksus on O(log N), kuna igal tsüklil me poolitame otsingut $1/2^N$. Puud on salvestatud massiivi ning ruumikompleksus on O(N).

Kuhja kasutatakse prioriteedi järjekorras, kuna andmete sisestamine ja kustutamine on keerukusega O(log N).

Ülesanne 3 Binaarse Otsingupuu analüüs

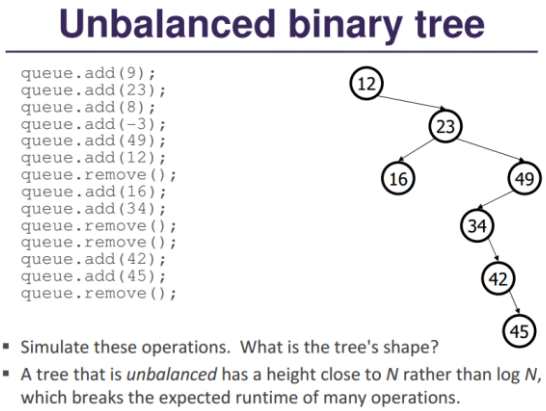
Binaarne Otsingupuu (BST) on binaarpuu, mille tüve vasaku poole elemendid peavad olema väiksemad kui tüvi ja tüve parema poole elemendid peavad olema suuremad kui tüvi. BST alampuud peavad samuti olema otsingupuud.



Joonis 2 Binaar Otsingupuu joonis

https://www.geeksforgeeks.org/introduction-to-binary-search-tree-data-structure-and-algorithm-tutorials/

Kuna BST reegel on et vasakul väiksemad elemendid ja paremal suuremad elemendid tüvest, siis võib tekkida olukord, kus puu kasvab ainult ühe tipu juures.



Joonis 3 Tasakaalust väljas Binaar Otsingupuu

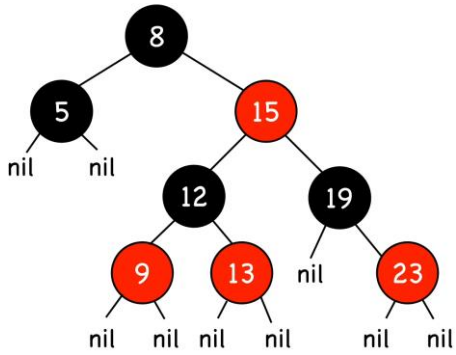
https://stackoverflow.com/questions/28772080/why-is-add-in-unbalanced-binary-search-tree-on

Sel juhul muutub optimiseeritud BST ajakompleksus O(log N) hoopis O(N), kuna enam me ei jaga iga operatsiooniga otsingut pooleks. Üks võimalus optimiseerimiseks on tasakaalustada puud. Tasakaalustada tuleb, siis kui iga sõlme kahe alampuu kõrgus erineb rohkem kui ühe võrra.

Ülesanne 4 Punase-Musta puu analüüs

Punane-Must puu on samade omadustega, mis on BST. Puu sõlmed värvitakse punaseks või mustaks.

Mustad sõlmed on tüvi ja lehed ehk sõlmed millel väärtust ei ole. Kui sõlm on punane, siis selle alamsõlmed või lapsed on mustad. Värvid peavad olema vaheldumisi, punasel sõlmel ei saa olla punaseid lapsi näiteks.



Joonis 4 Punase-Musta Puu

https://www.youtube.com/watch?v=qvZGUFHWChY

Puu operatsioonid on otsing, sisestamine ja kustutamine. Iga operatsiooni ajakompleksus on O(log N).

BST ja Punase-Musta puu ülesehitus on sarnane, vasakul on väiksemad elemendid ja paremal suuremad, aga Punase-Musta puu tagab, et see oleks alati tasakaalus. Igal operatsioonil Punase-Musta puul toimub uuesti sõlmede värvimine ja rotatsioon.

Ülesanne 5 AVL Puu ja Punase-Musta võrdlus

AVL Puu on BST, mis on tasakaalustub igal operatsioonil. Iga taseme sõlme vasaku ja parema alampuu kõrguste erinevus ei saa olla suurem kui üks. Oluline omadus AVL puus on balansifaktor mis on vasaku alampuu kõrgus lahutatud parema alampuu kõrgusest. Võimalikud väärtused balansifaktori jaoks on {-1, 0, 1}, kuna kõrgused ei tohi erineda rohkem kui 1. Kui balansifaktor on suurem kui 1 toimub pööramine, kus tõstetakse sõlmed ümber et puu säilitaks tasakaalu.

Üldiselt tunduvad AVL puu ja Punase-Musta puu sarnased, kuna mõlemad tasakaalustavad ennast pärast operatsioone. Tasakaalustamine on kahel erinev, sest AVL puus kasutatakse balansifaktorit ja Punase-Musta puus kasutame värve. Leian, et Punase-Musta puu sisestamis ja kustutamis operatsioonid on kiiremad, kuna ei ole keerulisi sõlmede ümber pööramisi nagu teeb AVL puu.

Punase-Musta puud saab rakendada kohtades, kus on vaja kiiresti sisestada ja kustutada andmeid. Näiteks MySQL kasutab seda indekseerimisel.

AVL puud saab kasutada suurte andmete talletamisel, kus tihti sisestamist ja kustutamist ei toimu, kuna meil ei ole vaja talletada, mis värvi iga sõlm puus on, ei tee tihti sõlmede pööramist ja andmed on talletatud efektiivselt.