Kodutöö esitamise tähtaeg: 19. november, 23:59

Kuhi

Kompaktse kahendpuu esitamiseks kasutatakse tihti massiivi, kuhu on kirjed salvestatud tippude tasemete kaupa. Siis on ajaga $O(1)$ kättesaadavad nii antud tipu alluvad kui ka ülemus: kui positsioone nummerdatakse 0-st ja tipu v kirje paikneb positsioonis k, siis

- vasaku ja parema alluva kirjed, kui nad eksisteerivad, paiknevad vastavalt positsioonides $2k + 1$ ja $2k + 2$;
- ülemuse kirje, kui ta eksisteerib, paikneb positsioonis $[(k - 1)/2]$.

Kahendkuhi (ingl *binary heap*) on kompaktset kahendpuud kasutav andmestruktuur, kus iga tipu v korral kehtib tingimus, et tipu v võti on väiksem või võrdne tema alluvate võtmetest.

Kahendkuhja saab kasutada eelistusjärjekorra realiseerimiseks. Eelistusjärjekord võimaldab (lisaks muudele operatsioonidele) minimaalse võtme väärtusega elemendile kiiret ligipääsu.

Ülesanne 1
Kasutades kompaktset kahendpuud implementeerida liides *MinBinaryHeap*.

Liidesed on kättesaadavad aaddressil `https://github.com/ut-aa/aa2016-lab6`.

Nüüd vaatame ühte rakendust eelistusjärjekorrale.

Tekstide kompaktsemaks esitamiseks ehk pakkimiseks arvuti välismälus võetakse kasutusele muutuva pikkusega sümbolikoodid. Üks võimalus niiviisi teksti pakkida on kasutades prefikskoode. Parimad sellised koodid leitakse kasutades Huffmani algoritmi. Algoritmi kirjelduse võib leida näiteks Jüri Kiho, Algoritmid ja andmestruktuurid, 2003, lehekülgedel 83-86. (kättesaadav moodlest)

Boonusülesanne (2p)
Realiseerida Huffmani algoritm vastavalt liidesele *HuffmanAlgorithm*.

Boonusülesande liides on kättesaadav aaddressil `https://github.com/ut-aa/aa2016-huffman`.