Kodutöö esitamise tähtaeg: 1. oktoober, 23:59

Variantide läbivaatamine ja kombinatoorika

Järgnevad ülesanded tuleks lahendada rekursiivsete funktsioonide abil, taandades ülesande teatavaks arvuks esialgse ülesandega sarnasteks alamülesanneteks. Proovige ülesanne jaotada osadeks (alamülesanneteks) nii, et iga osa jaoks saaks rakendada rekursiivselt sama funktsiooni.

Kombinatsioone, permutatsioone ja muid valikuid teostavaid standardfunktsioone kasutada ei tohi.

Ülesanne 1

1. Kirjutada rekursiivne meetod, mis väljastab ekraanile kõik bitivektorid, mille pikkus on võrdne etteantud arvuga. Modifitseerida seda meetodit nii, et ekraanile väljastamise asemel meetod tagastaks kõikide sobivate bitivektorite hulga. Modifitseeritud meetod peaks implementeerima liidest *BitVectorGenerator*.

Ülesanne 2
Implementeerida liides *SubSetGenerator*.

1. Implementeerida rekursiivne meetod *allSubsets*, mis tagastab määratud hulga kõik alamhulgad.
2. Implementeerida rekursiivne meetod *allSubsetsOfSizeK*, mis tagastab määratud hulga kõik alamhulgad, kus on täpselt k elementi.

Ülesanne 3
Implementeerida liides *PermutationGenerator*.

1. Implementeerida rekursiivne meetod, mis võtab parameetriks listi A ja tagastab listi, mis koosneb kõikidest listi A permutatsioonidest.

Ülesanne 4
Järgmistest ülesannetest valida üks ja lahendada see.

a) *Bobimeeskonna koostamine.* Realiseerida liides *BobsledTeam*. n-liikmelisest treeningrühmast ($n = 10 \pm 2$) tuleb eelolevaks võistluseks välja valida neljase bobi meeskond. Treeningrühma iga liikme kohta on antud nimi, kaal ja naturaalarvuline reiting (mis näitab treenituse taset). Implementeerida meetod *selectTeam*, mis leiab sellise bobimeeskonna, mille kogukaal ei ületa etteantud väärtust ja mille reitingute summa on võimalikult suur.

b) *Sobivaima töörühma väljavalimine.* Implementeerida liides *HarmonicWorkgroup*. Antud on töötajate nimekiri ja sobivusmaatriks (s_{ij}), mille element s_{ij} näitab töötajate i ja j vastastikuse sobivuse taset reaalarvuna lõigul $[0; 1]$. Implementeerida meetod *selectWorkgroup*, mis leiab võimalikult suure rühma töötajaid, mille korral vastastikuste sobivuste keskmine on suurem kui 0,5.

c) *Praktikumijuhendajaga kokkuleppel* valida ise üks ülesanne, mille lahendusalgoritm vajab kõikide variantide läbivaatamist (kas siis eksponentsiaal- või faktoriaalkeerukusega algoritmi abil). Nõutav on, et see ülesanne teeks midagi „kasulikku“.

Lihtsamad puud

Puu on hierarhiline andmestruktuur, mis koosneb **tippudest** (ehk sõlmedest; inglise keeles *nodes* või *vertices*), ja nende vahelistest **servadest** *(edges)*. Servad ühendavad tippu tema **vanemaga** *(parent)* ning **lastega** *(children)*. Igas mittetriviaalses[1] puus võib eristada üht tippu, mida nimetatakse **juureks** *(root)*. Igal tipul on täpselt üks vanem v.a juurtipp, millel ei ole vanemat. Tippe, millel ei ole lapsi, nimetatakse **lehtedeks** (*leafs*). Täpsemalt puudest ja nende tüüpidest räägitakse hilisemates loengutes.

Selles praktikumis me vaatleme üht lihtsamat puutüüpi — **kahendpuud**. Kahendpuu tippudel võib olla maksimaalselt kaks last, mille järjekord on tavaliselt tähtis (öeldakse *vasak* ja *parem* laps). Niisuguse puu tipu realisatsioon võib olla järgmine (täielik realisatsioon on toodud lab-2 projektis):

```
public class BinaryTreeVertex<Data>{
   private BinaryTreeVertex<Data> parent = null;
   private BinaryTreeVertex<Data> leftVertex = null;
   private BinaryTreeVertex<Data> rightVertex = null;

   private Data data = null;
}
```

Järgmised ülesanded eeldavad seda, et etteantud puu on esitatud, kasutades tippude jaoks projektis antud *BinaryTreeVertex* realisatsiooni.

Ülesanne 5
Implementeerida liides *BinaryTreeMeasurer*.

1. Implementeerida meetod *measureTreeHeight*, mis tagastab etteantud puu kõrguse. Arvestada, et ühetipulise puu kõrgus on 0.
2. Implementeerida meetod *countNodes*, mis tagastab etteantud puu tippude arvu.
3. Implementeerida meetod *countLeaves*, mis tagastab etteantud puu lehtede arvu.

Ülesanne 6
Implementeerida liides *BinaryTreeDataCounter*. Selleks implementeerida meetod *countData*, mis tagastab mitu tippu sisaldab meetodi argumendina saadud väärtust.

`https://github.com/ut-aa/aa2016-lab2`

[1] Triviaalne puu on niisugune puu, mis ei sisalda ühtegi tippu ega kaart ehk tühi puu.