Praktikum 1

Ajalise keerukuse empiiriline hindamine (I)

Algoritmi ajaline keerukus

Harjutusülesanded

- **AKE_1. Fibonacci arvude leidmise meetodid**
- **AKE_2. Meetodi tööaeg ja ajaline keerukus empiiriliselt**
- **AKE_3. Hanoi tornide ülesande lahendamise ajaline keerukus**

Algoritmi ajaline keerukus

Mõiste: vt *Õpik*, ptk 1; vt ka *AjalineKeerukus.pdf*.

Harjutusülesanded

AKE_1. Fibonacci arvude leidmise meetodid

Kirjutada (keeles Java) ja testida *n*-nda Fibonacci arvu leidmise

- rekursiivne meetod *fiboRek*;
- iteratiivne meetod *fiboIter*.

Vt ka [*Ülesannete kogu* 2.30 (*b*) ja (*c*)].

AKE_2. Meetodi tööaeg ja ajaline keerukus empiiriliselt

AKE_2_A. Koostada programm, mis väljastab neljaveerulise tabeli päisega

```
 n       Aeg(rek)          Aeg(iter)           F(n)
---      --------          --------          -------
```

mille esimeses veerus on Fibonacci arvu järjekorranumber *n* (nt sammuga 5). Teises ja kolmandas veerus on *n*-nda Fibonacci arvu leidmiseks kulunud aeg millisekundites (vastavalt rekursiivse ja iteratiivse meetodi korral) ning viimases veerus Fibonacci arv *F*(*n*).

```
long t0 = System.nanoTime(); // alustamise ajahetk
  ...<meetodi väljakutse> ...
long t1 = System.nanoTime(); // lõpetamise ajahetk
  // meetodi töö kestus:
long nt = t1 - t0;                // nanosekundites
double t = (t1 - t0)/1000000.0; // millisekundites
```

AKE_2_B. Koostada programm, mis väljastab neljaveerulise tabeli päisega

```
 n       Liitmisi(rek)     Liitmisi(iter)    F(n)
---      --------          --------          -------
```

mille esimeses veerus on Fibonacci arvu järjekorranumber *n* (nt sammuga 5). Teises ja kolmandas veerus on *n*-nda Fibonacci arvu leidmisel sooritatud liitmistehete (põhitehete) arv, vastavalt rekursiivse ja iteratiivse meetodi korral, ning viimases veerus Fibonacci arv *F*(*n*).

AKE_3. Hanoi tornide ülesande lahendamise ajaline keerukus

1. Laadida programm *Hanoi.java* oma töökausta. Avada toimetiga *Notepad++.*
   Kontrollida (vajadusel tagada), et see programm oleks ANSI kodeeringus.
2. Kompileerida ja käivitada töökaustas, käsurealt:

```
javac Hanoi.java
java Hanoi
```

3. Uurida meetodi *tõsta* ajalist keerukust, kui põhitehteks on meetodi *tõsta* väljakutse.

3.1 Täiendada programmi *Hanoi.java* selliselt, et väljastataks ka sooritatud põhitehete arv rakenduse *tõsta*(*n*, 1, 2, 3) korral.

NB! Programmi toimetamiseks on soovitatav kasutada editori *Notepad++*.

Täita tabel programmi abil saadud arvudega:

n	Leitud tõstmiskäskude arv	Meetodi *tõsta* väljakutsete arv, st ajalise keerukuse väärtus *f*(*n*)
1		
2		
…		
12		

3.2. Esitada ajaline keerukus analüütilisel kujul (valemina) *f*(*n*) = … .

3.3. Leida funktsiooni *f(n)* Θ-hinnang: *f*(*n*) on Θ(…).