

Praktikum 2

Ajalise keerukuse empiiriline hindamine (II)

Harjutusülesanded

- **AKE_3. Graafikud**
- **AKE_4. Kahe sorteerimismeetodi ajagraafikud**

Iseseisev töö nr 1

AKE_3. Graafikud

AKE_3.1. Koostada Java-programm funktsioonide

$$f = 0,1(n \log n),$$

$$g = 0,1 n^2$$

graafikute joonistamiseks vastavalt kõrvalolevale skeemile.

AKE_3.2. Lisada veel kahe funktsiooni, $h1$ ja $h2$ graafikud:

$$h1 = f(n) / g(n),$$

$$h2 = g(n) / f(n).$$

Märkus. Käsurealt kompileerimine ja käitamine: vt järgmine slaid.

```
Amadeus - <noname7>.*
Fail Toimeti Lisada Minna Teisendus Raam Erisoovid Abi

import ee.ut.kiho.aa.util.GraafikuteJoonistaja;»
»»
String autor = "Ülli Õpilane";»
int N = 2; graafikute (funktsioonide) arv
String[] legend = new String[N]; »
legend[0] = "f(n) = 0.1*n*log(n)";»
legend[1] = "g(n) = 0.1*n*n";»
»»
int m = 100; punktide arv (abstsissteljel)
double[][] x = new double[N][m]; koht abstsissidele
double[][] y = new double[N][m]; koht ordinaatidele
»»
double n = 1.0; esimene argumenti väärtus
double samm = 1.0; samm abstsissteljel

* i = 0 .. m-1»
»»
double f = 0.1 * n * Math.log(n);»
double g = 0.1 * n * n;»
x[0][i] = n;»
y[0][i] = f; »
x[1][i] = n;»
y[1][i] = g; »
n += samm;»

new GraafikuteJoonistaja(autor, x, y, legend);»
```

Java-programmi (AKE_3_1) kompileerimine ja käivitamine käsurealt

Kui jooksvas kaustas on:

- 1) Java lähtekood failis *AKE_3_1.java* *
- 2) fail *ee_ut_kiho_aa.zip* -- pakettide [*ee.ut.kiho.aa.*] *graaf* ja *util* klassifailid (alla laaditud lingilt <http://kodu.ut.ee/~kiho/ads/fall15/>)

siis selles kaustas

 kompileerimine:

```
...>javac -cp ee_ut_kiho_aa_.zip AKE_3_1.java
```

 käivitamine:

```
...>java -cp .\;ee_ut_kiho_aa_.zip AKE_3_1
```

* Kus peaklassi nimeks on *AKE_3_1*:

```
import ee.ut.kiho.aa.util.GraafikuteJoonistaja;
public class AKE_3_1 {
    public static void main(String[] args){
    ...
```

AKE_4. Kahe sorteerimismeetodi ajagraafikud

Valida välja, realiseerida Java-meetodina ja testida üks aeglane, ruutkeerukusega (keskmise ajalise keerukuse hinnanguga $O(n^2)$) algoritm järjendi elementide ümberpaigutamiseks mittekahanevasse järjekorda. Sellisteks algoritmideks on näiteks mullimeetod (*bubble sort*), valikumeetod (*selection sort*), pistemeetod* (*insertion sort*) jt. Testida ka nn äärejuhte (järjendid pikkusega 0, 1 või 2).

Koostada programm järgmise kahe sorteerimismeetodi tööaegade graafikute kujutamiseks (ühel joonisel):

- Valitud aeglane meetod
- Süsteemne sorteerimismeetod, nt (Java korral) funktsioon *Arrays.sort*.

Sorteeritavateks olgu juhujärjendid.

Ekspirimendi selgituseks

Järgmisel slaidil on ühe eksperimendi tulemus.

Järjendite pikkuste massiiv N , nt:

$N = \{500, 1000, 1500, 2000, 2500, \dots, 25000\}$

Vastav sorteerimisele võetavate järjendite massiiv M :

{ juhujärjend pikkusega 500,
juhujärjend pikkusega 1000,
...
juhujärjend pikkusega 25000 }

Graafiku abstsissid: 0, 1, 2, ..., 49

Graafiku ordinaat kohal n on aeg, mis kulus $(500(n+1))$ -elemendilise järjendi sorteerimiseks

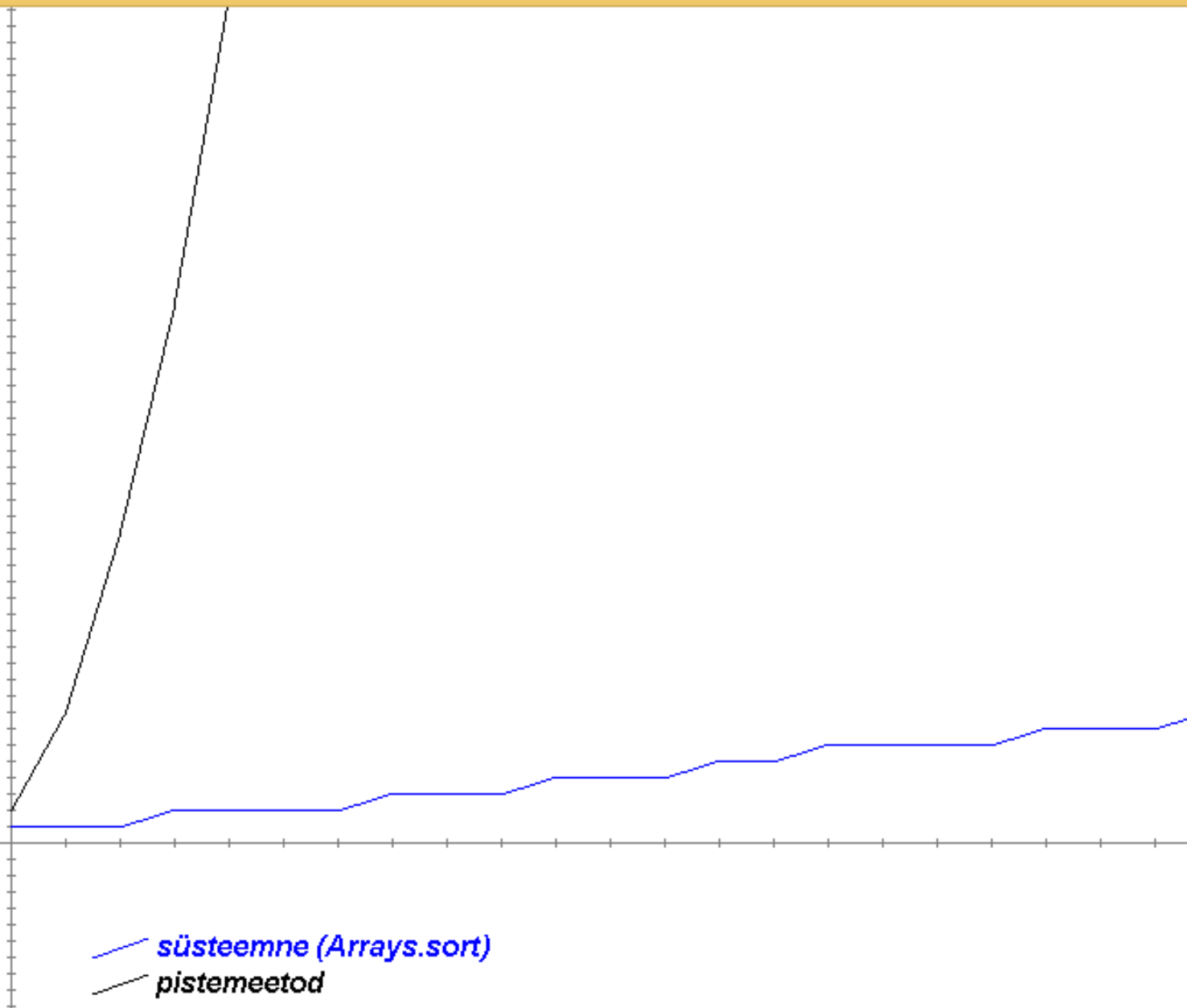
* [Kiho 2003, lk 60] või otsida nt “Java insertion sort”



FUNKTSIOONIDE GRAAFIKUD




ÜLLI ÕPILANE
10.07.2015



süsteemne (Arrays.sort)
pistemeetod

Sorteerimismeetodi (nr m)
tööaja mõõtmise meetodi skeem.

```
Amadeus - <noname13>*
Fail Toimeti Lisada Minna Teisendus Raam Erisoovid Abi
»
» long ajavõtt(int m, int[] a)»
»
» Antud: sorteerimismeetod m (numbri abil), järjend a
» Tulemus: tagastatakse aeg (nanosek),
» . mis kulub a sorteerimiseks meetodiga m
long min = Long.MAX_VALUE;»
* k = 0 .. 15 korduvalt, välistamaks "kõrvaltegevuste" mõju
»
» int[] b = a.clone();»
» long nt0 = System.nanoTime();»
»  sorteerida järjend b meetodiga m
» long nt1 = System.nanoTime();»
» long naeg = nt1-nt0;»
»
» if (naeg < min)»
»     min = naeg;»
return min; parim aeg katsetest
```

Iseseisev töö nr 1

Esitamise tähtaeg rühmal J. Liivi 2-206, E12-14: **26. september 2015, kell 12.15**
Esitamise tähtaeg rühmal J. Liivi 2-207, T14-16: **20. september 2015, kell 14.15**

Tulemuse näidis:

