СОФИЙСКИ УНИВЕРСИТЕТ
„СВ. КЛИМЕНТ ОХРИДСКИ“

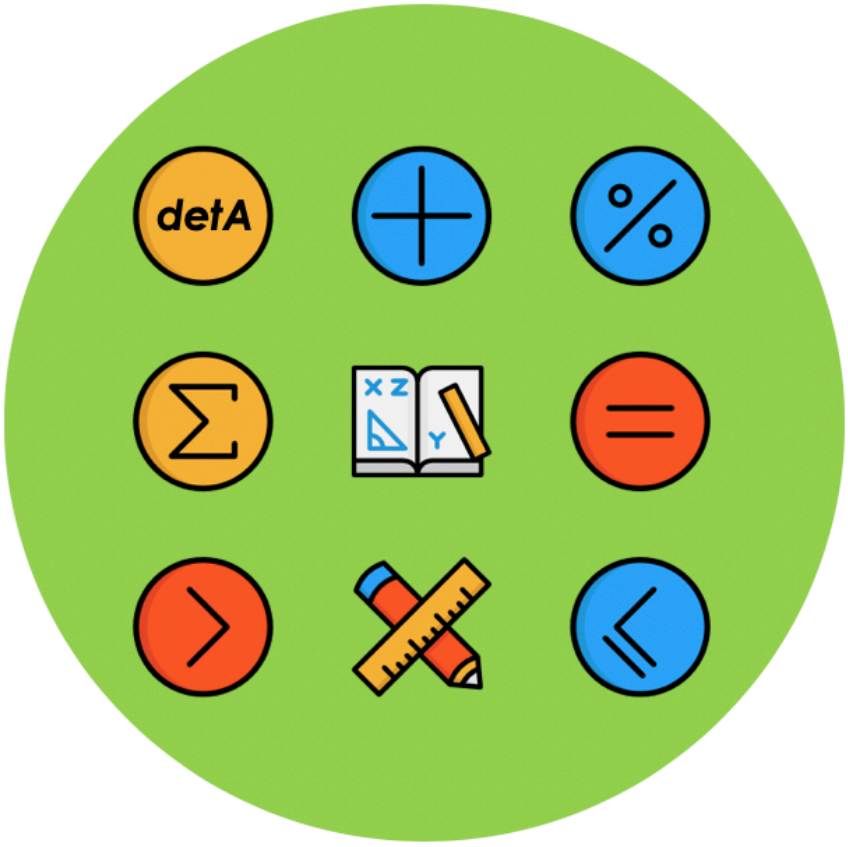
SOFIA UNIVERSITY
ST. KLIMENT OHRIDSKI

Курсова работа

по

Разпределени софтуерни архитектури(РСА)

Зад. 7. Детерминанта на матрица



Автор: Стефка Москова, 61979
Факултет: ФМИ
Специалност: Софтуерно инженерство
Курс: III

СЪДЪРЖАНИЕ

1. Условие

*** Приложението е реализирано с технологията – Java.**

Зад. 7 (Детерминанта на матрица)

Разглеждаме матрицата A с размерност (n, n). Да се напише програма която пресмята det A. Работата на програмата по пресмятането на детерминантата да се раздели по подходящ начин на две или повече нишки (задачи).

Изискванията към програмата са следните:

(o) Размерността на матрицата се задава от подходящо избран команден параметър – например **"-n 13"**; Елементите на матрицата генерираме произволно с помощта на **Math.random()** (класа **java.util.Random)** или **java.util.concurrent.ThreadLocalRandom**; (Тоест матрицата може да има float или double елементи)

Разликата между двата начина - **Math.random()** е достъпен във всички версии на Java и ужасно бавен. Не е проектиран за работа в много-нишкова (multi-threaded) среда; В Java 7 и 8, разполагаме с **ThreadLocalRandom**, който е специално проектиран за работа в рамките на отделен thread (респективно multi-threaded среда);

Ще бъде много интересно да реализирате програма, използваща и двата начина и съответно получим два вида резултати от работата на програмата;

(o) Команден параметър указващ входен текстов файл, съдържащ матрицата, чийто детерминанта ще пресмятаме – например **„-i m3x-data.in"**. Параметрите **„-n"** и **„-i"** са взаимно-изключващи се; Ако все пак бъдат зададени и двата решението как да реагира програмата е Ваше.

Форматът на файла **m3x-data.in** е следният:

=== цитат ===
n
a11 a12 a13 … a1n
a21 a22 a23 … a2n
…
an1 an2 an3 … ann
=== цитат ===

Тоест:

1вият ред съдържа единствено число, указващо размерността на матрицата.

На оставащите **n** реда във файла са разположени редовете от матрицата чиято детерминанта ще пресмятаме. Елементите на всеки ред от матрицата са разделени със интервали.

(o) Команден параметър указващ изходен файл, съдържащ резултата от пресмятането – например **„-o m3x-data.out"**. Форматът на изходният файл можете да определите сами, стига файлът да е текстов; При липса на този команден параметър **не се** записва във файл резултата от пресмятането на детерминантата;

(o) Друг команден параметър задава максималния брой нишки (задачи) на които разделяме работата по пресмятането на det A – например **"–t 1"** или **"–tasks 3"**;

(o) Програмата извежда подходящи съобщения на различните етапи от работата си, както и времето отделено за изчисление и резултата от изчислението;

Примери за подходящи съобщения:

„**Thread-<num> started.**",
„**Thread-<num> stopped.**",
„**Thread-<num> execution time was (millis): <num>**",
„**Threads used in current run: <num>**",
„**Total execution time for current run (millis): <num>**" и т.н.

(o) Да се осигури възможност за **„quiet"** режим на работа на програмата, при който се извежда само времето отделено за изчисление на det A, отново чрез подходящо избран друг команден параметър – например **"-q";**

(o) Изчислението на det A може да бъде извършено с помощта на адюнгирани количества и развитието на детерминантата по ред или стълб;

ЗАБЕЛЕЖКА:

Зад. 7, 1/2 (3.4)

(о) При желание за направата на подходящ графичен потребителски интерфейс (**GUI**) с помощта на класовете от пакета **javax.swing** задачата може да се изпълни от **двама души**; Разработването на графичен интерфейс не отменя изискването Вашата програма да поддържа изредените командни параметри. В този случай към функцията на параметъра параметъра **„-q“** се добавя изискването **да не пуска** графичният интерфейс. Причината за това е, че Вашата програма трябва да позволява отдалечено тестване, а то ще се извършва в **terminal**.

Уточнения (hints) към задачата:

(о) В условието на задачата се говори за разделянето на работата на две или повече нишки. Работата върху съответната задача, в случаят в който е зададен **„-t 1“** (т.е. цялата задача се решава от една нишка) ще служи за еталон, по който да измерваме евентуално ускорение (т.е. това е **T1**). В кода реализиращ решението на задачата трябва да се предвиди и тази възможност – задачата да бъде решавана от единствена нишка (процес); Пускайки програмата да работи върху задачата с помощта на единствена нишка, ще считаме че използваме серийното решение на задачата; Измервайки времето за работа на програмата при използването на **„p“** нишки – намираме **Tp** и съответно можем да изчислим **Sp**. Представените на защитата данни за работата на програмата, трябва да отразят и ефективността от работата и, тоест да се изчисли и покаже **Ep**.

Като обобщение - данните събрани при тестването на програмата Ви, трябва да отразяват **Tp**, **Sp** и **Ep**. Желателно е освен табличен вид, да добавите и графичен вид на **Tp**, **Sp**, **Ep**, в три отделни графики.

(о) Не се очаква от Вас да реализирате библиотека, осигуряваща математически операции със комплексни числа. Подходяща за тази цел е например **Apache Commons Math3 (http://commons.apache.org/proper/commons-math/userguide/complex.html).** При изчисленията, свързани с генерирането на множеството на Манделброт (задачите за фрактали), определено ще имате нужда от нея.

(о) Не се очаква от вас да реализирате библиотека, осигуряваща математически операции със голяма точност. Подходяща за тази цел библиотека е например **Apfloat (http://www.apfloat.org)**. Ако програмата Ви има нужда от работа с големи числа, можете да използвате нея.

Разбира се **BigInteger** и **BigDecimal** класовете в **java.math** са също възможно решение – въпрос на избор и вкус.

Преди да направите избора, проверете дали избраната библиотека не използва също нишки – това може да доведе до неочаквани и доста интересни резултати; ;)

(о) Не се очаква от Вас да търсите (пишете) библиотека за генериране на **.png** изображения. Java има прекрасна за нашите цели вградена библиотека, която може да се ползва. Примерен проект, показващ генерирането на чернобялата и цветната версия на фрактала на Манделброт /множество на Манделброт за формула (2)/, цитирани в задачите за фрактали, е качена на **http://rmi.yaht.net/docs/example.projects/ - pfg.zip**.

(о) Командните аргументи (параметри) на терминална Java програма, получаваме във масива **String args[]** на **main()** метода, намиращ се в стартовият клас. За „разбирането“ им (анализирането им) може да ползвате и външни библиотеки писани специално за тази цел . Един добър пример за това е: **Apache Commons CLI (http://commons.apache.org/cli/)**.

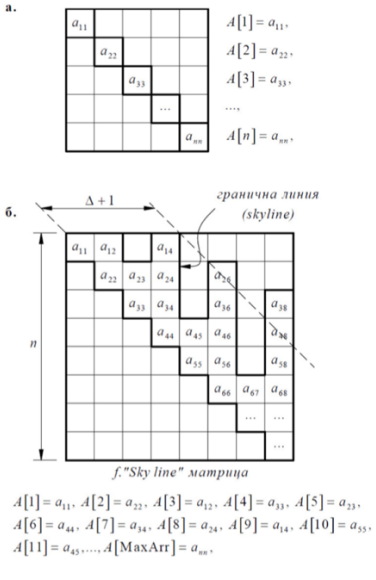
2. Обща информация за пресмятане на *detA*

Определение: Матрицата е правоъгълна таблица (масив) от числа. Нека размерът на масива е ***mxn***. Това означава, че матрицата има ***m реда*** и ***n колони***. Матриците се означават по следния начин (1.1):

$$\mathbf{A} \equiv \left[A\right] \equiv \mathbf{A}_{m\times n} \equiv \left[a_{ij}\right] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}. \qquad (1.1)$$

Ако **m=n**, матрицата се нарича ***квадратна*** и ще казваме, че тя е от ред n. С a_{ij} ще означаваме произволен елемент от матрицата , който принадлежи на ***i***-тия ред и на ***j***-тата колона. Елементите a_{ij} на матрицата, за които ***i=j***, тоест a_{ii} се наричат главен диагонал на матрицата.

Много често в математиката обект, който се състои от много компоненти, се охарактеризира(оценява) с помощта на една величина – скалар. Пример за такава величина е ***детерминанта***. Изчисляване на детерминанта е възможно само за квадратни матрици. Тази величина е функция на компонентите на матрицата. В литературата съществуват два подхода при дефиниране на детерминанта. Ще се обърне внимание само на по-известния и използван подход – индукция по размера n на матрицата, тоест детерминантата на матрицата $\mathbf{A_{nxn}}$ ще изчисляваме чрез детерминантите на матриците от по-нисък ред. На фигурата по-долу (1.3) се вижда съхраняване на елементите на матрица в едномерен масив на диагонална матрица (а.) и съответно на симетрична ивична матрица „skyline“:



(1.3)

Определение:

i. Детерминантата на матрица от първи ред (***n=1***) е равна на самия елемент. Ако **A=[a_{11}]**, то **detA=a_{11}**.

ii. Ако i е произволен елемент от матрицата $\mathbf{A_{nxn}}$, то детерминантата ***detA*** се изчислява по формулата:

$$\det \mathbf{A} = \sum_{j=1}^{n} (-1)^{i+j} \det \mathbf{A}_{ij}, \qquad (1.4)$$

където A_{ij} е матрица от ред ***(n-1)x(n-1)***, която се получава от A чрез отстраняване на i-тия ред и j-тата колона. Формулата (1.4) е известна като ***Разлагане на Лаплас по i-тия ред на матрицата***. Аналогично е ***Разлагането на Лаплас по j-тата колона****(1.5)*:

$$\det \mathbf{A} = \sum_{i=1}^{n} (-1)^{i+j} \det \mathbf{A}_{ij} \ . \qquad (1.5)$$

3.Приложение

Матриците се използват много повече в ежедневието, отколкото хората биха си помислили. Матриците са пред нас всеки ден - когато ходим на работа, на университет, вкъщи и дори в цялата заобикалящата ни среда:

i. Графичните софтуери използват матрици за обработка на линейни трансформации, за да визуализират изображения. Квадратната матрица може да представлява линейна трансформация на геометричен обект.

ii. В приложения, свързани с физиката, матриците се използват при изучаването на електрически вериги, квантовата механика и оптика. Инженерите използват матрици за моделиране на физически системи и извършват точни изчисления, необходими за работата на сложната механика.

iii. В програмирането матриците се използват за код и кодиране на съобщения. Съобщение се прави като поредица от числа в двоичен формат за комуникация.

iv. В роботиката и автоматизацията матриците са основните компоненти за движенията на роботите. Входовете за управление на роботи се получават въз основа на изчисленията от матрици и това са много точни движения.

4.Решение

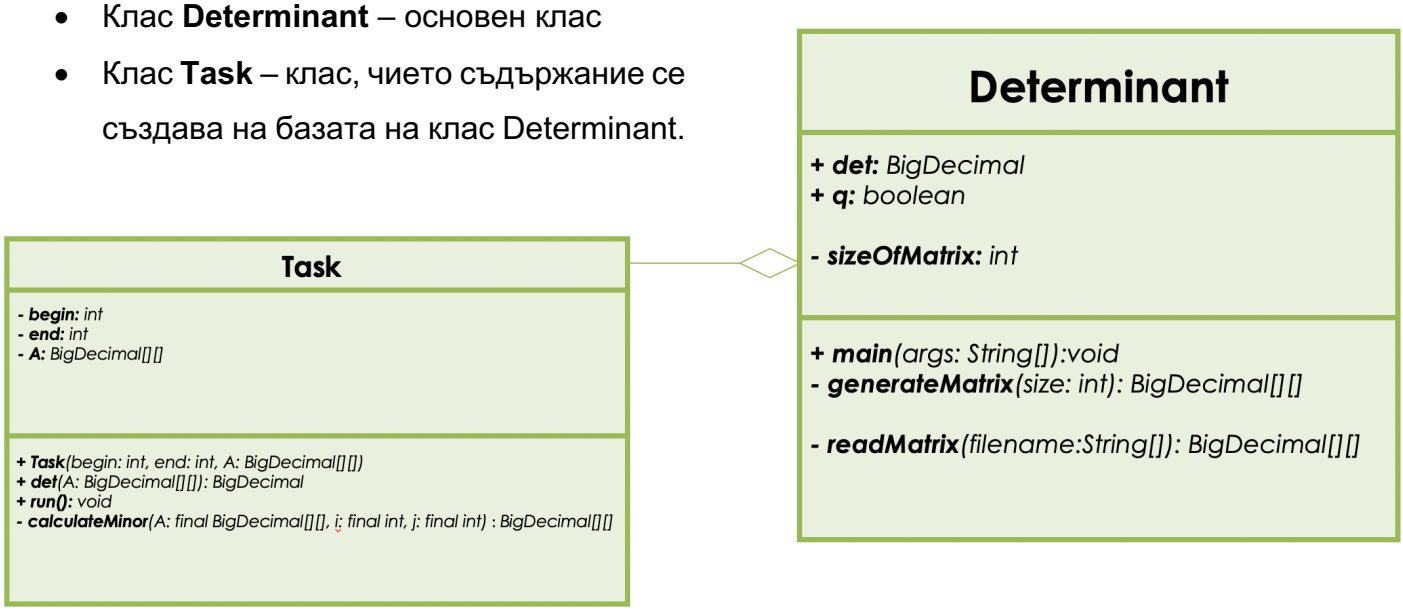
Решението бе организирано, структурирано и реализирано в средата за разработка ***Eclipse 4.11.0*** на операционна система ***macOS Mojave 10.14.5***.
Програмата, реализираща алгоритъма за пресмятане на детерминанта, дава възможност на потребителя да изчисли детерминантата на матрица по зададени:

- Команден параметър -i/-n, оказващ входен текстов файл, съдържащ матрицата, чийто детерминанта ще пресмятаме – например „-i m3x-data.in“. Параметрите „-n“ и „-i“ са взаимноизключващи се;
- Команден параметър -o, указващ изходен файл, съдържащ резултата от пресмятането – например „-o m3x-data.out“, като форматът на файла потребителят може да определени сам (текстов файл). При липса на този команден параметър не се записва във файл резултата от пресмятането на детерминантата.
- Команден параметър -t, който задава максималния брой нишки, на които разделяме работата по пресмятането на det A – например “–t 1”;
- Команден параметър -q, който да се осигури възможност за „quiet“ режим на работа на програмата, при който се извежда само времето отделено за изчисление на det A.

При реализирането на алгоритъма бяха необходими следните класове, включващи съответните член-данни и функции:

Програмата съдържа 2 класа:

- Клас **Determinant** – основен клас
- Клас **Task** – клас, чието съдържание се създава на базата на клас Determinant.



5.Получени диаграми и резултати

Получените резултати се оформиха благодарение на проведените тестове на РСА-сървъра: ***t5600.rmi.yaht.net***. На този сървър се изчисли детерминантата на квадратна матрица с размерност 10x10 – матрицата има 10 реда и 10 стълба. В таблицата по-долу са посочени стойностите – резултат от проведените тестове:

Данните от таблицата са представени графично в следните графики:

Брой процеси	Време в ms	Ускорение	Ефективност
1	*8226*	*1*	*1*
2	*4523*	*1,8187044*	*0,9093522*
3	*2953*	*2,78564172*	*0,92854724*
4	*2331*	*3,528957529*	*0,882239382*
5	*1807*	*4,552296624*	*0,910459325*
6	*1433*	*5,740404745*	*0,956734124*
7	*1369*	*6,008765522*	*0,858395075*
8	*1203*	*6,837905237*	*0,854738155*
9	*697*	*6,489239598*	*0,721026622*
10	*488*	*6,051229508*	*0,605122951*

- ***Време в милисекунди(ms) Tp*** – времето за пресмятане на детерминантата на квадратна матрица с размерност 10x10 – матрицата има 10 реда и 10 стълба, при употребата на определен брой нишки.

- ***Ускорение Sp*** – ускорението при пресмятане на детерминантата на квадратна матрица с размерност 10x10 – матрицата има 10 реда и 10 стълба, при употребата на определен брой нишки.

- ***Ефективност Ep*** – ефективността при пресмятане на детерминантата на квадратна матрица с размерност 10x10 – матрицата има 10 реда и 10 стълба, при употребата на определен брой нишки.

6.Използвана литература

Списък източници	Атрибути на източниците
http://rmi.yaht.net/projects/zad7-determinant.pdf	*От линка бе изведено условието на заданието **„Детерминанта на матрица“**.*
https://uacg.bg/filebank/att_7603.pdf	*От линка бе изведена обща информация за матриците: определение, примери, и формули.*
https://www.matematika.bg/visha-matematika/lineina-algebra-matrici/determinanta-index.html	*От линка бе изведена обща информация за матриците – работа с матрици, примери и подходящи изображения.*
http://web.uni-plovdiv.bg/marta/tema-6.pdf	*От линка бе изведена обща информация за матриците и тяхното приложение.*
https://www.ukessays.com/essays/mathematics/application-of-matrices-in-real-life-problems.php	*От линка бе изведена информация за приложението на матриците в заобикалящата ни среда.*
https://www.w3schools.com/java/	*От линка бе изведена информация за кода към заданието – основни Java практики.*
https://www.w3schools.com/java/	*От линка бе изведена информация за кода към заданието – Java tutorial.*

7.Речник на новите думи