Beadandó feladat

Határidő 2023. máj 28 23:59 -ig **Pont** 20 **Beküldés...** egy fájlfeltöltés
Fájltípusok zip **Elérhető** 2023. máj 28, 23:59 -ig

Ez a feladat zárolva lett ekkor: 2023. máj 28, 23:59 .

Egyenletmegoldó

Feltételek

A feladat önállóan oldandó meg.

Amennyiben régi Java verzió van fent a gépeden, a legújabb kiadás letölthető innen: https://www.azul.com/downloads/?package=jdk

Töltsd le (majd nevezd át `junit5all.jar`-ra): https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/1.9.2/junit-platform-console-standalone-1.9.2.jar

Töltsd le a `junit5-demo.zip` fájlt a Canvasról, és csomagold ki a `checkthat.jar`-t. A `zip` fájl mutatja, hogyan használandó a tesztelő.

JUnit és checkthat teszteléshez az `EquationSolver-Tests.zip` fájlból ki kell csomagolni ezeket a fájlokat: `GaussianEliminationTest.java` és `GaussianEliminationStructureTest.java`, valamint a `GaussianEliminationTestSuite.java`, amely az előző két tesztet fogja össze. A teljes pontszám megszerzéséhez minden tesztet teljesíteni kell.

Feladat

Implementálnod kell egy lineáris algebrai egyenletmegoldó programot. Egy mátrixot a lépcsős alakjára kell hoznod Gauss-elimináció segítségével. Ebben a feladatban elég lesz csak a redukált lépcsős alakjára hozni a mátrixot, hogy egyszerűbb legyen a végső lépés implementációja, amely a visszahelyettesítés lesz. A redukált lépcsős alak egy olyan lépcsős alak amiben minden vezéregyes az egyetlen nemnulla elem az oszlopában.

A lépések a következők.

1. feladat: (4 pont)

Hozd létre a `linear.algebra.GaussianElimination` osztályt, amely egy mátrix másolatát `double` típusú tömbök tömbjeként tárolja.

- Hozz létre gettereket az oszlopok (`cols`), a sorok (`rows`) és maga a mátrix (`matrix`) számára.
- Csak a mátrix (`matrix`) számára készíts settert. Új mátrix beállítása esetén, ha az új mátrix sor- vagy oszlopszáma nem egyezik meg a jelenlegivel, akkor dobj `IllegalArgumentException`-t.
- Készíts az osztálynak konstruktort, amely a `sorok-` és `oszlopok számát` valamint a `double` típusú `mátrixot` kapja bemenetként. A konstruktor legyen látható az osztályon kívülről is.
- Készítsd el a `rowEchelonForm` metódust, amely kiszámítja a redukált lépcsős formát az itt elérhető pszeudokód alapján .
  - Ügyelj arra, hogy ez a pszeudokód 1 alapú indexelést használ, míg a Java 0 alapú indexelést!

2. feladat: (6 pont)

Mivel a lépcsős forma kiszámítása során 3 művelet ismétlődik, ezért készítsd el a következő metódusokat.

- `argMax`, amely egy sorindexet és egy oszlopindexet kap paraméterül. Amely visszaadja a sorindex után következő olyan sor indexét, amely a megadott oszlopban a legnagyobb abszolút értékű értéket tartalmazza (`Math.abs`).
- `swapRows`, amely két sorindexet kap paraméterként, és megcseréli a két sorban lévő értékeket.
- `multiplyAndAddRow`, hatékonyabb ha a sorokat megszorozzuk és összeadjuk egy lépésben, ahogy a pszeudokódban is van. A metódus egy `addRow` indexet, egy `mulRow` indexet és egy `colIndex`-et kap paraméterül. Kivonja az `addRow` sorindexekből, az `addRow`-nak és `mulRow`-nak a `colIndex` beli hányadosának a `mulRow`-val vett szorzatát, ahogy azt az pszeudokód belső ciklusában láthatod.
- `multiplyRow`, egy sorindexet és egy oszlopindexet kap paraméterül, majd elosztja a sorban lévő értékeket az adott sor- és oszlopindexnél lévő értékkel.
- Amikor a `multiplyRow`-t meghívod, a pszeudokóddal ellentétben, azelőtt hívd meg a `multiplyRow` metódust, mielőtt megnövelnéd a sor és oszlop pivot indexeit. Ez fogja elősegíteni azt, hogy a megoldáshoz a mátrix redukált lépcsős alakját kapd.

3. feladat: (4 pont)

- Implementáld a `backSubstitution` metódust, amely visszafelé megy végig a sorokon. Az aktuális sor előtti minden további sor esetében a `multiplyAndAddRow` metódussal kivonja a már megoldott változókat. A belső ciklusba lépés előtt ellenőrizd, hogy a diagonális elem nem nulla-e, különben `IllegalArgumentException`-t dobj, mivel ez akkor egy megoldás nélküli egyenletrendszer.
- Implementáld a `print` metódust, amely a lineáris egyenlethalmazt ember számára érthető módon írja ki a konzolra. Például:

```
+2.0x+1.0y-1.0z=8.0
-3.0x-1.0y+2.0z=-11.0
-2.0x+1.0y+2.0z=-3.0
```

4. feladat: (6 pont)

Implementáld a `linear.EquationSolver` osztályt, amelyben a `main` parancssori argumentumainak a formája a következő: `2,1,-1,8 -3,-1,2,-11 -2,1,2,-3`. Ez a bemenet az előző példát írja le.

- Készítsd el a `stringsToDoubles` osztályszintű segédmetódust, amely egy szöveges tömböt `double` tömbbé alakít.
- A `main` metódusban alakítsuk át a bemeneteinket, majd hozzunk létre egy `GaussianElimination` objektumot, hívjuk meg a megadott metódusokat ebben a sorrendben: `print`, `rowEchelonForm`, `print`, `backSubstition`, `print`. Az előző példa esetében a redukált lépcsős alakja és a végső megoldás így fog kinézni:

```
+1.0x+0.3333333333333333y-0.6666666666666666z=3.6666666666666665
+0.0x+1.0y+0.4000000000000001z=2.6
+0.0x+0.0y+1.0z=-0.9999999999999999

+1.0x+0.0y+0.0z=2.0
+0.0x+1.0y+0.0z=3.0
+0.0x+0.0y+1.0z=-0.9999999999999999
```

Ne felejtsd el lefuttatni a megadott teszteket, hogy meggyőződj a programod megfelelő működéséről!