

NitscheSahoo - AD Praktikum für 17.10.13 – Aufgabe 2 - Matrix

Das Interface:

```
public interface Matrix {  
  
    void insert(int i, int j, double value);  
  
    int getM();  
  
    int getN();  
  
    double get(int i, int j);  
  
    void copyFrom(Matrix source);  
  
    int memoryUsage();  
  
    int accessCount();  
  
    void resetAccessCount();  
  
    Matrix add(Matrix m);  
  
    Matrix mul(double skalar);  
  
    Matrix mul(Matrix factor);  
  
    Matrix pow(int exponent);  
  
}
```

Begründungen:

Insert: man braucht eine Methode um ein Element an eine entsprechende Position zu packen.

GetM/getN: Simple getter für Matrixdimensionen.

CopyFrom: eine exakte Kopie von einer Matrix im Speicher, da wir nicht nur eine Referenz wollen.

Aufgabe 7 – Platzaufwand der Listenimplementationen

Siehe Datei „ab2_test/Aufgabe7.java“.

Siehe Excel Tabelle Sheet „Aufg2“.

Graphische Darstellung: Siehe Excel Tabelle. Da beide Implementierungen mit den gleichen zufälligen Matrizen arbeiten, haben beide daher den gleichen Platzaufwand in den Messdaten.

Platzbedarf: Füllt man die beiden Listenimplementierungen von Matrizen mithilfe der Generatormodus, dann können wir feststellen, dass bei geringem p (also geringer Chance auf eine Zahl ungleich Null) beide Listenimplementierungen tatsächlich Platz sparen. Erhöht man p um einen Faktor, so wird auch der Platzbedarf um ca. den gleichen Faktor höher.

Overhead: Wir haben die Kosten eines einzigen Elements als Overhead interpretiert.

Aufgabe 8 – Platzaufwand der Listenimplementationen

Siehe Datei „ab2_test/Aufgabe7.java“.
Siehe Excel Tabelle Sheet „Aufgb2“.

Graphische Darstellung: Siehe Excel Tabelle. ListAddAverage und ArrayListMultAverage haben sehr ähnliche Werte. Deswegen sind nicht beide Linien zu erkennen. Das Diagramm ist außerdem logarithmisch.

Zeitbedarf(In Anzahl der Dereferenzierungen): Der Zeitbedarf vervielfacht sich bei beiden Implementationen und beiden Operationen proportional zu p. Dabei ist die ArrayListImplementation ungefähr 100x effizienter.

Overhead: ?

Aufwandsabschätzung:

Arbeitszeit: 15h gemeinsam

Zeitverteilung: 60% MatrixImplementations - 40% Tests/Experimente

Vorschau der Javadoc:

The screenshot displays a Javadoc preview for the package `ab2_adts`. On the left, a sidebar titled "All Classes" lists the following classes: `AbstractMatrix`, `GeneratorModule`, `Matrix`, `MatrixArray`, `MatrixArrayList`, and `MatrixList`. The main content area shows the package header `Package ab2_adts` with navigation links: `Prev Package`, `Next Package`, `Frames`, and `No Frames`. Below the header, there are two summary tables. The first is the "Interface Summary" table, which contains one entry: `Matrix`. The second is the "Class Summary" table, which lists the following classes: `AbstractMatrix`, `GeneratorModule`, `MatrixArray`, `MatrixArrayList`, and `MatrixList`. The bottom of the screenshot shows a second instance of the package header and navigation links.