Algorithmen und Datenstrukturen (WS 13/14)

Prof. Dr. Michael Köhler-Bußmeier

Hausaufgaben 2 : Datentypen

Version vom: 11. Oktober 2013

Wir wiederholen zunächst ein wenig Mathematik: Sie kennen $m \times n$ Matrizen A als Anordnungen von $m \cdot n$ Elementen $a_{i,j}$ in m Zeilen und n Spalten.

Für den Zeilenindex i gilt also $1 \leq i \leq m$ und für den Spaltenindex j gilt $1 \leq j \leq n$.

Die Matrixelemente sind dabei Elemente irgendeines Körpers $\mathbb{K}$, z.B. den der reellen Zahlen.

Besonders einfach ist der Fall der quadratischen Matrizen, bei denen die Anzahl der Zeilen gleich der Anzahl der Spalten ist. Auf diesen wollen wir uns im folgenden beschränken.

Die *Matrixmultiplikation* $A \cdot B$ funktioniert so: Man multipliziert zwei Matrizen $A = (a_{i,j})$ und $B = (b_{i,j})$ zu $C = (c_{i,j})$, indem man die i-te Zeile von A mit der j-ten Spalte von B multipliziert:

$$c_{i,j} = a_{i,1} \cdot b_{1,j} + \cdots + a_{i,n} \cdot b_{n,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j}$$

Übungsaufgabe 2.1:

1. Die Grundoperationen einer quadratischen Matrix sind die Initialisierung mit einer Dimension n, die Addition $A+B$, die skalare Multiplikation $\lambda \cdot A$, die (Matrix-)Multiplikation $A \cdot B$ und die Potenzierung A^k. Auch der Zugriff auf das Element $a_{i,j}$ ist wichtig. Gelegentlich muss man auch wissen, ob zwei Matrizen gleich sind.

   Entwerfen Sie eine geeignet Schnittstelle. Verwenden Sie Fließkommazahlen für die Elemente. Wenn Sie weitere Methoden für sinnvoll halten, so begrüden Sie dies kurz.

2. Schicken Sie das Interface per E-Mail vorab an mich. Die Gruppenmitglieder sollen als Autoren in der Datei benannt sein.

3. Beschreiben Sie Testfälle. Beispielsweise ist die Multiplikation assoziativ, d.h. $A \cdot (B \cdot C) = (A \cdot B) \cdot C$; die skalare Multiplikation $1 \cdot A$ ergibt A; die Multiplikation $A \cdot A$ ist identisch mit A^2 usw.

4. Implementieren Sie die Matrixschnittstelle auf drei Varianten:

   (a) durch ein zweidimensionales Array;

   (b) durch ein Array $z[1..m]$ von Listen (d.h. für jede Zeile der Matrix eine Liste), bei der Sie nur diejenigen Paare $(j, a_{i,j})$ in der Liste $z[i]$ speichern, bei denen $a_{i,j} \neq 0$ gilt;
   (Dies ist insbesondere bei sehr dünn besetzten Matrizen, bei denenen aber fast keine Zeile nur Nullen enthält, eine interessante Alternative. Beispiel: stochastische Matrizen.)

   (c) durch eine Liste, bei der Sie nur diejenigen Tripel $(i, j, a_{i,j})$ speichern, bei denen $a_{i.j} \neq 0$ gilt.
   (Dies ist insbesondere bei sehr dünn besetzten Matrizen eine interessante Alternative.)

Anmerkung: Addition, Multiplikationn und Exponentierung können Sie generisch (d.h. unabhängig von der Datenstruktur) formulieren. Voraussetzung ist natürlich, dass Sie nicht direkt auf die Datenstruktur zuzugreifen, sondern ($\rightarrow$ information hiding) nur Zugriffsmethoden wie `get(i,j)` und `set(i,j)` verwenden, um auf die Matrixelemente zuzugreifen.

5. Entwickeln Sie ein Generatormodul, mit dem Sie zufallsgesteuert $n \times n$ Matrizen initialisieren können. Die Dimension n soll dabei ein Parameter sein.

   Die Wahrscheinlichkeit, dass ein Matrixelement $a_{i,j}$ ungleich Null ist, soll dabei als Parameter eingestellt werden können.

   Ein weiterer Parameter soll im Falle "ungleich Null" für eine zufallsgesteuerte Wahl eines Elements sorgen.

6. Erzeugen Sie zwei zufällige $n \times n$ Matrizen A und B und speichern Sie diese jeweils in allen drei Implementationsvarianten. Dies ergibt also die Matrizenpaare A_1, B_1, $A_,B_2$ und A_3, B_3.

   Multiplieren Sie für jedes Paar A_i und B_i die beiden Matrizen. Vergewissern Sie sich, dass in allen Implementationsvarianten das gleiche Endergebnis produizieren.

7. "Wieiviel Platz sparen wir mit den Listenimplementationen tatsächlich ein?"

   Sei n ein festes Argument. (Nehmen Sie die letzten 4 Ziffern Ihrer Matrikelnummer.)

   (a) Erzeugen Sie t zufällige $n \times n$ Matrizen A_t und messen Sie in den Implementationsvarianten (ii) und (iii) den Platzbedarf! (Der Platzbedarf ist die Anzahl der in den Listen gespeicherten Werte.)

   Den Wert von t bestimmen wir der Einfachheit halber so, dass sich Mittelwert bei den ersten 5 Stellen "stabil" bleibt.

   Variieren Sie die Wahrscheinlichkeit p, dass ein Matrixelement $a_{i,j}$ ungleich Null ist. Verwenden Sie die Werte: 1%, 5% und 10%.

   (b) Stellen Sie den Zusammenhang von Platzbedarf $platz_p(t)$ für diese Werte von p geeignet graphisch dar!

   (c) Bestimmen Sie für alle drei Implementationen zu den obigen Werten von p, welche Größe n sie im Mittel speichern können!

   (d) Bestimmen Sie aus Ihren Experimenten den Overhead der Listenimplementationen!

8. "Wieviel Zeit-Overhead haben die Listenimplementationen?"

   Sei n ein festes Argument. (Nehmen Sie die letzten 4 Ziffern Ihrer Matrikelnummer.)

   (a) Messen Sie den Zeitaufwand für die Variationen von p!

   Der Zeitaufwand wird dabei wieder in der Anzahl der Dereferenzierungen gemessen. (Die Anzahl der Multiplikationen und Addition ist ja bei allen Verfahren identisch.) Diese können Sie ja bei Ihrer Listenimplementation abfragen.

   (b) Wiederholen Sie dieses Experiment t-mal (solange, bis wieder 5 Stellen stabil sind). Geben Sie den Mittelwert und Streuung von $zeit_p(t)$ für diese Werte von p an und illustrieren Sie die Messreihe graphisch!

   (c) Bestimmen Sie aus Ihren Experimenten den Overhead der Listenimplementationen!