Algorithmen und Berechnungskomplexität I, WS 12/13
Aufgabenblatt 4
Universität Bonn, Institut für Informatik, Abteilung I

- *Die Lösungen können bis Dienstag, 13.11., 12:15 Uhr in den Postkasten im AVZ III eingeworfen werden.*

Aufgabe 13: Pseudocode Matrizenmultiplikation (4 Punkte)

In der Vorlesung haben Sie ein Verfahren kennengelernt, das mittels Dynamischer Programmierung eine optimale Klammerung für die Multiplikation mehrerer reellwertiger Matrizen bestimmt. Formulieren Sie einen Pseudocode, der dieses Verfahren beschreibt. Die Ausgabe soll sowohl die minimalen Kosten enthalten, als auch die Klammerung, durch die sie erreicht werden.

Aufgabe 14: Dynamische Programmierung (4 Punkte)

Seien $n, k \in \mathbb{N}_0$ mit $0 \leq k \leq n$ gegeben. Der Binomialkoeffizient $\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$ kann auch unter Zuhilfenahme folgender Gleichung berechnet werden:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

Entwickeln Sie einen Algorithmus, der diese Rechenvorschrift mit einem Speicherbedarf in $O(k)$ mit Dynamischer Programmierung implementiert.

Bitte wenden!

Aufgabe 15: Das Rucksackproblem (4 Punkte)

Wir betrachten das Rucksackproblem. Gegeben sind 5 Gegenstände

$$\{a_1 = (1, 4), a_2 = (3, 5), a_3 = (4, 6), a_4 = (6, 9), a_5 = (7, 10)\}.$$

Der erste Wert bezeichnet das Gewicht, der Zweite den Wert der Gegenstände; das Gewicht von a_1 ist beispielsweise 1 und dessen Wert ist 4.

Bestimmen Sie für einen Rucksack mit Kapazität 10 eine optimale Lösung für das Rucksackproblem. Geben Sie an, welche Gegenstände den Rucksack optimal füllen. Stellen Sie hierzu, wie in der Vorlesung besprochen, eine Tabelle von Teillösungen auf um dieses Rucksackproblem mit Hilfe der Dynamischen Programmierung zu lösen. Die Tabelle soll sowohl den für die Teilergebnisse jeweils den erzielten Maximalwert als auch die verwendeten Gegenstände beinhalten.

Aufgabe 16: Maximum einer monotonen Matrix (4 Punkte)

Eine $m \times n$ Matrix $M[.][.]$ heißt *monoton*, wenn für alle $1 \leq k < \ell \leq m$ und $1 \leq i < j \leq n$ gilt, dass

$$M[k][i] < M[k][j] \Rightarrow M[\ell][i] < M[\ell][j].$$

Das bedeutet wenn in der k-ten Zeile das i-te Element kleiner ist als das j-te Element, dann ist in jeder Zeile unterhalb der k-ten Zeile das i-te Element ebenfalls kleiner als das j-te.

Geben Sie einen Divide- and Conquer Algorithmus an, der in jeder Zeile die Position des linkesten Zeilenmaximums bestimmt, und geben Sie dessen Laufzeit in O-Notation an. Die Laufzeit des Algorithmus soll in

$$O(n + m + n \log m)$$

liegen.

Es darf angenommen werden, dass die Matrixeinträge schon initialisiert sind. Für die Laufzeit relevant sind lediglich die Rechenoperationen des Algorithmus, wie zum Beispiel Vergleiche der Form „Gilt $M[a][b] < M[c][d]$?“.

Hinweis: Beweisen Sie zunächst, dass der Wert „Spaltenindex des linkesten Maximums in Zeile k“ monoton in k wächst.