

## Datenstrukturen und Algorithmen – Informatik I

### 3. Übung

Abgabe der Lösungen: 17. Mai 2004

---

**Hinweis:** Die Implementierungsaufgaben sind in Java oder C/C++ zu lösen. Der Quellcode sowie ein dokumentierter Beispiellauf sind beim Betreuer der Kleingruppe abzugeben.

#### Aufgabe 11: Bäume

1+1+2 Punkte

- In welcher Reihenfolge müssen die Elemente 2, 42, 20, 6, 30, 56, 12 in einen leeren binären Suchbaum eingefügt werden, damit...
  - ...der Baum zu einer Liste degeneriert?
  - ...der Baum vollständig wird?
- Geben Sie jeweils zwei Binärbäume  $T_1 \neq T_2$  an, so dass  $preorder(T_1) = preorder(T_2)$  bzw.  $postorder(T_1) = postorder(T_2)$ .  
Gibt es zwei verschiedene Binärbäume mit  $preorder(T_1) = preorder(T_2)$  **und**  $postorder(T_1) = postorder(T_2)$  (mit Begründung, aber kein formaler Beweis)?
- Gegeben sei ein binärer Suchbaum, die Suche nach dem Schlüssel ende in einem Blatt. Sei  $L$  die Menge der Schlüssel links vom Suchpfad,  $R$  die Menge der Schlüssel rechts vom Suchpfad und  $M$  die Menge der Schlüssel auf dem Suchpfad. Zeigen oder widerlegen Sie: für alle  $l \in L, m \in M, r \in R$  gilt  $l \leq m \leq r$ .

#### Aufgabe 12: Implementierung: Binärer Suchbaum

2+2+2+2 Punkte

- Implementieren Sie *nichtrekursive* Funktionen, die auf einem Binärbaum einen
  - Preorder-Tiefendurchlauf
  - Postorder-Tiefendurchlauf
  - Inorder-Tiefendurchlauf
  - Breitendurchlauf

ausführen. Verwenden Sie für dazu entweder die Stack und Queue, die in Übung 1 implementiert wurden oder die vorgegebenen Java Klassen `Stack` bzw. als `Queue Vector` mit `remove(0)` als `get` und `add(Object o)` als `put`.

- Testen Sie Ihre Algorithmen auf den Beispieldaten aus der Übung 2.

#### Aufgabe 13: Implementierung: Matrixkettenprodukt

3+3+1+3 Punkte

Gegeben seien  $n$  Matrizen  $M_i$  der Dimensionen  $r_{i-1} \times r_i$ . Finden Sie eine Klammerung des Kettenprodukts  $M_1 \cdot M_2 \cdots M_n$ , die die Anzahl der (skalaren) Multiplikationen minimiert. Benutzen Sie die folgende Rekursionsgleichung für die Kostenfunktion:

$$m(i, j) = \begin{cases} 0 & \text{für } i = j \\ \min_{i \leq k < j} \{m(i, k) + m(k+1, j) + r_{i-1}r_kr_j\} & \text{für } i < j \end{cases}$$

1. Implementieren Sie eine *rekursive* Funktion, die die optimale Klammerung bestimmt. Erweitern Sie diese Funktion um Memoization (eine Variante der dynamischen Programmierung).
2. Verwenden Sie eine *iterative* Implementierung (mit expliziter Ablaufsteuerung), die die optimale Klammerung mittels dynamischer Programmierung bestimmt.
3. Testen Sie Ihre Programme an der Dimensionsfolge  $r = (5, 9, 3, 15, 7, 30, 6)$ . Wieviele (skalare) Multiplikationen sind nötig, wenn man die Matrizen von links nach rechts miteinander multipliziert? Wieviele im Fall der optimalen Klammerung?
4. Messen Sie die Laufzeit Ihrer Programme für unterschiedliche Werte von  $n$  und die Dimensionsfolge  $(r_1, \dots, r_n)$  mit  $r_i = \lceil 5 \sin(i/4) + 15 \rceil$ . Stellen Sie Ihre Ergebnisse graphisch dar, d.h. tragen Sie die Laufzeit  $T(n)$  für jeden der drei Algorithmen gegen  $n$  für  $1 \leq n \leq 100$  ab.

#### Aufgabe 14: ! Bonus: Softwarewettbewerb Matrixmultiplikation !

+6 Punkte

Schreiben Sie (basierend auf der Aufgabe 13) ein Programm für die Multiplikation von  $n$  reellwertigen Matrizen  $M_i$  der Dimensionen  $r_{i-1} \times r_i$ . Dieses Programm soll die für die effiziente Multiplikation optimale Klammerung des Kettenprodukts  $M_1 \cdot M_2 \cdots M_n$  bestimmen, und anschließend gemäß dieser Klammerung die Multiplikation durchführen. Am Ende des Programms soll die resultierende Matrix ausgegeben werden.

Verwenden Sie für diese Aufgabe die Klassen `Matrix` und `MatrixMultiplikation`, die die Ein- und Ausgabefunktionen für Matrizen bereitstellen und die Laufzeit von Ihrem Algorithmus messen. Die Klassen sowie Beispielmatrizen sind auf unserer Webseite verfügbar. Implementieren Sie die Klasse `Multipliiert`. Beachten Sie, dass die Laufzeit Ihres Programms auch von der effizienten Implementierung der eigentlichen Multiplikation und nicht nur von der optimalen Klammerung abhängt.

Senden Sie Ihre Lösung (die vollständig implementierte Methoden in `Multipliiert.java`) per E-Mail an Ihren Übungsgruppenleiter. Die Laufzeit der eingereichten Programme wird an einem Lehrstuhlrechner auf neuen Testmatrizen gemessen. Die Autoren des schnellsten korrekt funktionierenden Programms bekommen für diese Aufgabe **25 Bonuspunkte**; die Autoren anderer korrekt funktionierender Programme bekommen jeweils 6 Bonuspunkte.

**Achtung: Der Einsendeschluß für das Softwarewettbewerb ist Mo., 24.05.2004, 00:00 Uhr!**  
Die Gewinner werden in der Vorlesung am 28. Mai vorgestellt.

#### Aufgabe 15: Master-Theorem

3 Punkte

Zwei Algorithmen  $A_1$  und  $A_2$  besitzen die Rekursionsgleichungen

$$T_{A_1} = a_1 \cdot T_{A_1}(n/2) + n \quad \text{bzw.} \quad T_{A_2} = a_2 \cdot T_{A_2}(n/4) + n$$

für natürliche Zahlen  $a_1 \geq 2$  und  $a_2 \geq 4$ . Bestimmen Sie auf der Grundlage des Master-Theorems die Beziehung zwischen  $a_1$  und  $a_2$  unter der Annahme, dass beide Algorithmen gleich schnell sind.