Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Christian Scheideler
Stefan Eckhardt

Wintersemester 2005/06
Übungsblatt 6
26. November 2005

# Effiziente Algorithmen und Datenstrukturen I

*Abgabetermin: 02.12.2004* **vor** *der Zentralübung*

## Aufgabe 1 (10 Punkte)

In dieser Aufgabe wollen wir die erwartete Anzahl von Vergleichen der MFR bestimmen, wenn wir eine Liste $a, b, c$ von drei Elementen haben und nur ACCESS-Operationen ausgeführt werden. Wir nehmen an, dass das Element $a$ mit Wahrscheinlichkeit $p$, das Element mit Wahrscheinlichkeit $q$ und das Element $c$ mit Wahrscheinlichkeit $r$ gesucht wird, wobei $0 \leq p, q, r < 1$ und $p + q + r = 1$ gelten. Das MFR beschreibt eine Markovkette mit in diesem Fall 3! Zuständen, wobei jeder Zustand eine der 6 Anordnungen der drei Elemente $a, b, c$ beschreibt.

a) Verifizieren Sie, dass die folgenden Wahrscheinlichkeiten die stationäre Verteilung beschreiben:

$$\begin{aligned}
P(abc) &= pq/(1-p) \\
P(acb) &= pr/(1-p) \\
P(bac) &= pq/(1-q) \\
P(bca) &= qr/(1-q) \\
P(cba) &= qr/(1-r) \\
P(cab) &= pr/(1-r)
\end{aligned}$$

Beachten Sie, dass die Markovkette irreduzibel und aperiodisch ist und somit die stationäre Verteilung existiert.

b) Berechnen Sie die erwartete Anzahl von Vergleichen, indem Sie die stationäre Verteilung zu Grunde legen.

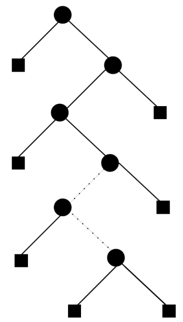
c) Wie groß ist die erwartete Anzahl von Vergleichen, wenn $p = 3/4$ und $q = r = 1/8$?

## Aufgabe 2 (10 Punkte)

Seien $T$ und $T_0$ zwei unterschiedliche binäre Suchbäume auf den gleichen $n$ Schlüsseln. Zeigen Sie, dass man mit $O(n)$ Rotationen $T$ in $T_0$ umwandeln kann.

## Aufgabe 3 (10 Punkte)

Sei $T$ ein Splay-Tree auf $n$ Knoten, $n$ ungerade. Geben Sie eine Sequenz von Zugriffen auf $T$ an, die $T$ durch die entsprechenden Splayoperationen in einen Baum umwandelt, der die im Bild angegebene Form hat. (Der Pfad der inneren Knoten alterniert zwischen rechten und linken Kinder.)



## Aufgabe 4 (10 Punkte)

Zeigen Sie: Für jede beliebige Folge von Splay-Tree-Operationen der Art Find, Insert, Delete, Join und Split, die mit einer Kollektion leerer Splay-Trees beginnen, ist die *worstcase* Laufzeit beschränkt durch

$$O\left(k + \sum_{i=1}^{k} \log n_i\right),$$

wobei $n_i$ die Anzahl Knoten der in der $i$-ten Operation involvierten Bäume sei.