Technische Universität München
Institut für Informatik
Prof. Dr. J. Esparza/Dr. S. Gerke

WS 2000/01
Übungsblatt 6
28.November 2000

Effiziente Algorithmen und Datenstrukturen

(Abgabetermin: Dienstag, den 5.Dezember, in der Vorlesung)

Aufgabe 1

a) Geben Sie eine Sequenz von ACCESS-, INSERT- und DELETE-Operationen an, die einen zu einer linearen Liste entarteten Splay-Tree mit n Knoten erzeugt. (D.h., jeder Knoten darf höchstens einen Nachfolger haben).

b) Es sei T ein solcher entarteter Splay-Tree mit $n = 2^k - 1$ Elementen, und zwar so, daß das größte Element an der Wurzel steht und jeder (außer dem kleinsten) Knoten nur ein linkes Kind (nämlich das nächst kleinere Element) hat. Nennen Sie eine Reihe von k ACCESS-Operationen, die bewirken, daß der Baum anschließend Höhe $k-1$ hat, also ein vollständiger binärer Baum ist.

Aufgabe 2

Zeigen Sie: Für jede beliebige Folge von k Splay-Tree-Operationen der Arten FIND, INSERT, DELETE, JOIN, SPLIT, die mit einer Kollektion leerer Splay-Trees beginnen, ist die *worst case* Laufzeit beschränkt durch

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

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

Aufgabe 3

Wir betrachten folgende alternative Implementierung der DELETE-Operation bei Splay-Trees. Für DELETE(i, T) mit $T \neq \emptyset$ und $i \in T$ suchen wir zuerst nach dem Element i. Sei v der Vater von i, falls i nicht die Wurzel ist. Wir ersetzen i als Kind von v durch den JOIN des linken und rechten Unterbaums von i und führen dann eine SPLAY-Operation auf v durch. Falls i die Wurzel ist, so führen wir die unmodifizierte Delete-Operation aus. Zeigen Sie, daß die amortisierten Kosten dieser Implementierung

$$3 \log \left(\frac{W - w(i)}{\min\{w(i-), w(i)\}} \right) + O(1)$$

sind.
(Dabei bezeichnet $i-$ das bezüglich i nächstkleinere Element.)