Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Angelika Steger

WS 1996/7
Übungsblatt 3
20. November 1996

Effiziente Algorithmen und Datenstrukturen

Abgabe am 27. November 1996 bis 8:00 Uhr im Briefkasten

Aufgabe 9

a) Es sei T ein Splay-Tree mit n Elementen, der durch eine beliebige Folge von FIND, INSERT und DELETE Operationen entstanden ist. Kann T die Form einer linearen Liste haben? (D.h., kann es sein, daß jeder Knoten höchstens einen Nachfolger hat?) Wenn ja, wie kann man T erzeugen? Wenn nein, warum nicht?

b) Es sei T ein zur linearen Liste 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 FIND-Operationen, die bewirken, daß der Baum anschließend Höhe $k-1$ hat, also ein vollständiger binärer Baum ist.

Aufgabe 10

Es sei $w : V(T) \to \mathbb{R}_{>0}$ eine Funktion, die jedem Knoten $v \in T$ eines Splay-Trees ein Gewicht $w(v)$ zuordnet.
Betrachten Sie die folgende Modifikation der Potentialfunktion:

$$\varphi'(x) = \log_2\left(\sum_{v \in T_x} w(v)\right) \qquad \text{für alle } x \in T,$$

wobei T_x den Unterbaum von T mit Wurzel x bezeichne. Das Potential von T definieren wir wie bisher durch $\Phi'(T) = \sum_{x\in T} \varphi'(x)$.
Zeigen Sie: Bezüglich der so definierten Potentialfunktionen φ' und Φ' ist die amortisierte Laufzeit der Prozedur SPLAY(x, T) beschränkt durch $1 + 3(\log_2(\sum_{x\in T} w(x)) - \varphi'(x))$.
Hinweis: Modifizieren Sie die Beweise von Lemma 2.3 und Korollar 2.4 geeignet.

Aufgabe 11

a) 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

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

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

b) Es sei T ein Splay-Tree mit n Elementen. Zeigen Sie, daß für jede beliebige Folge von k FIND Operationen, die auf jeden Knoten mindestens einmal zugreifen ($k \geq n$), die *amortisierte* Laufzeit durch

$$\mathcal{O}\left(k + \sum_{i=1}^{n} q(i) \log(\frac{k}{q(i)})\right)$$

beschränkt ist. Dabei gebe $q(i)$ an, wie oft der i-te Knoten das Ziel einer FIND Operation ist.

Aufgabe 12

Implementieren Sie einen `splaytree` als Datentypen in C++. Sie können dafür den Rahmen, der auf der hphalle unter `~kuehnle/effalg/a12/frame.c++` liegt, verwenden. Wie auch beim Programmierwettbewerb, müssen natürlich die relevanten Teile erst ergänzt werden. Zur Vereinfachung enthalte ein Element außer Verwaltungsinformation (Zeiger auf Kinder o.ä.) nur einen Schlüssel vom Typen `unsigned`. Es sollen die Operationen `find`, `insert` und `remove` (`delete` ist in C++ schon anderweitig belegt) zur Verfügung stehen. Ihr Programm soll von `stdin` Befehle der Form f<key>, i<key> oder d<key> lesen und den angegebenen Schlüssel finden, einfügen oder entfernen. `find` soll die entsprechende Umordnung des Baumes vornehmen und 0 oder 1 zurückgeben, abhängig davon, ob der Schlüssel im Baum vorhanden ist oder nicht. `insert` soll das entsprechende Element einfügen, falls es nicht schon im Baum ist, und zurückgeben, ob es eingefügt wurde oder schon vorhanden war (0 oder 1). `remove` soll das entsprechende Element entfernen und 0 zurückgeben, falls es vorhanden war, ansonsten 1 zurückgeben. Alle drei Operationen dürfen dabei auch dann den Baum umstrukturieren, wenn eigentlich nichts, außer 1 zurückgeben, zu tun wäre. Zu Testzwecken können Sie zufällige Folgen solcher Befehle mit dem Programm `~kuehnle/effalg/a12/testgen.x` erzeugen.