Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Tobias Lieber

Wintersemester 2010/11
Übungsblatt 7
14. Dezember 2010

# Effiziente Algorithmen und Datenstrukturen

*Abgabetermin: 21. Dezember 2010 vor der Vorlesung*

## Aufgabe 1 (10 Punkte)

Verallgemeinern Sie die Idee der 2-Level-Buckets auf $k$-Level-Buckets. Die amortisierte Laufzeit von Insert und DecreaseKey soll wieder $O(1)$ sein, die von ExtractMin nur noch $O(\sqrt[k]{C})$, wobei $C$ wieder die obere Schranke für die Differenz zwischen dem größten und dem kleinsten Schlüssel sein soll.

*Hinweis:* Sie können zur Vereinfachung annehmen, dass nur Schlüssel aus der Menge $\{0, \ldots, C-1\}$ in den $k$-Level-Bucket eingefügt werden

## Aufgabe 2 (10 Punkte)

Geben Sie eine graphische Darstellung (d.h. eine Darstellung des Arrays $b$ mit seinen Listen sowie der Werte aus dem Array $u$) des Radix-Heaps an, der entsteht, wenn die Werte

```
100, 29, 50, 17, 72, 49, 18, 42, 31, 24, 117
```

in einen leeren Heap eingefügt werden. Führen Sie anschließend auf diesem Heap eine ExtractMin-Operation aus und geben Sie den resultierenden Heap an.

## Aufgabe 3 (10 Punkte)

Wie in der Vorlesung eingeführt, sei eine Union-Find Datenstruktur durch eine Menge von Intrees (d.h. Bäume mit zur Wurzel gerichteten Kanten) realisiert. Es wird keine Pfadkompression verwendet. Die Union-Operation sei wie folgt definiert:

> Die Wurzel des Baums mit geringerer Höhe wird als direktes Kind an die Wurzel des anderen Baums gehängt. Sind beide Bäume gleich hoch, wird einer der Bäume zufällig ausgewählt und wie beschrieben in den anderen Baum eingehängt.

Sei $\text{size}(T)$ bzw. $\text{height}(T)$ die Anzahl der Knoten bzw. die Höhe eines Intrees $T$. Zeigen Sie, dass für alle auftretenden Intrees $T$ in einer Folge von Operationen, die auf eine Menge von $n$ einelementigen Mengen angewendet wird, gilt:

$$\text{size}(T) \geq 2^{\text{height}(T)}.$$

## Aufgabe 4 (10 Punkte)

In der Vorlesung wurde die Laufzeit $O(k \log^* n)$ für eine Union-Find-Struktur mit *gewichteter Vereinigung* (engl. *weighted union*) und *Pfadkompression* (engl. *path compression*) bewiesen. Hierbei wurde für jeden Knoten der folgende Wert definiert:

$$rank(x) = \text{Höhe des Unterbaums mit Wurzel } x \text{ in der Datenstruktur } \textit{ohne} \text{ Pfadkompression.}$$

a) Zeigen Sie, dass für alle Knoten in der Datenstruktur mit Pfadkompression gilt, dass die Werte von *rank* auf allen Pfaden von den Blättern zur Wurzel strikt steigen.

b) Sei $rank_p(x)$ der Wert von *rank* des aktuellen Vaters des Knoten $x$. Zeigen Sie, dass $rank_p(x)$ bei einem Wechsel des Vaterknotens von $x$ nicht kleiner wird.