Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Matthias Baumgart

Wintersemester 2007/2008
Übungsblatt 8
7. Dezember 2007

Effiziente Algorithmen und Datenstrukturen I

Abgabetermin: 14.12.2007 **vor** *der Vorlesung*

Aufgabe 1 (10 Punkte)

In einem Algorithmus soll ein großes Array verwendet werden. Der Algorithmus verlangt, dass zu Beginn alle Arraypositionen initialisiert werden (zum Beispiel mit NULL). Einerseits dauert uns das Initialisieren des Arrays aber zu lang, andererseits kann in einem nicht-initialisierten Speicher natürlich beliebiger Unsinn stehen. Beschreiben Sie eine Möglichkeit, wie man ohne die aufwendige Initialisierung auskommen kann. Genauer gesagt soll beim lesenden Zugriff auf eine Array-Position das Folgende passieren:

- Falls der Algorithmus vorher einen Eintrag in diese Position geschrieben hat, wird dieser Eintrag zurückgegeben.
- Falls in diese Position noch nicht geschrieben wurde (diese also „Bitmüll" enthält), wird NULL zurückgegeben.

Wie kann man ein solches Array so implementieren, dass ein (lesender oder schreibender) Zugriff auf ein Arrayelement Zeit $O(1)$ kostet und dass wir beim lesenden Zugriff immer die richtige Antwort zurückgeliefert bekommen? Das Zurücksetzen des Arrays (jeder lesende Zugriff liefert wieder NULL) soll ebenfalls in Zeit $O(1)$ möglich sein.

Hinweis: Benutzen Sie eine Extradatenstruktur, in der steht, welche Positionen des Arrays schon *echt* beschrieben wurden. (Dort werden sozusagen *Zeugen* der Echtheit verwaltet.) Benutzen Sie außerdem eine Extrainformation in jeder Arrayposition, die gegebenenfalls auf den Zeugen verweist.

Aufgabe 2 (10 Punkte)

Implementieren Sie ein Array, das die in Aufgabe 1 beschriebenen Eigenschaften hat. Der Wert x, mit dem jede Arrayposition implizit initialisiert ist, soll variabel sein.

Senden Sie Ihren Quelltext an `baumgart@in.tum.de`.

Aufgabe 3 (10 Punkte)

Stellen Sie die van Emde Boas-Priority Queue für die Menge

$$S = \{1, 2, 3, 4, 6, 7, 18, 21, 28, 31\} \subset \{0, 1, \ldots, 31\}$$

mit allen auftretenden k-Strukturen graphisch dar. Führen Sie auf dieser Priority Queue eine EXTRACTMIN-Operation und danach die Operation INSERT(5) aus und stellen Sie die wesentlichen Schritte und das jeweilige Ergebnis dieser Operationen wiederum graphisch dar.

Aufgabe 4 (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.