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

WS 1996/7
Übungsblatt 4
28. November 1996

Effiziente Algorithmen und Datenstrukturen

Abgabe am 4. Dezember 1996 bis 8:00 Uhr im Briefkasten

Aufgabe 13

In der Vorlesung haben wir für Fibonacci-Heaps nur die Prozeduren INSERT, DELETEMIN und DECREASEKEY betrachtet. Entwerfen Sie in Ergänzung hierzu eine Prozedur DELETE, die als Argument einen Zeiger auf ein Element im Fibonacci-Heap bekommt und dieses Element entfernen soll. Schätzen Sie die amortisierte Laufzeit dieser Prozedur ab. (Sie sollte $\mathcal{O}(\log n)$ nicht übersteigen.)

Aufgabe 14

Die amortisierte Laufzeit der Prozedur DELETEMIN ist in Fibonacci-Heaps $O(\log n)$, wobei n die Anzahl der in der Datenstruktur gespeicherten Elemente ist. Unter der Zusatzannahme, daß das Universum U, aus dem die Schlüssel kommen, „klein" ist, kann man diese Schranke noch verbessern.

Überlegen Sie sich eine Modifikation der Fibonacci-Heaps, bei der nach wie vor INSERT und DECREASEKEY konstante amortisierte Laufzeit haben und sich die Laufzeit von DELETEMIN durch $\mathcal{O}(\log(\min\{n, |U|\}))$ abschätzen läßt.

HINWEIS: Stellen Sie sicher, daß in dem F-Heap nie zwei Elemente mit dem gleichen Schlüssel enthalten sind. Verwenden Sie dazu ein zusätzliches Array der Größe $\mathcal{O}(|U|)$.

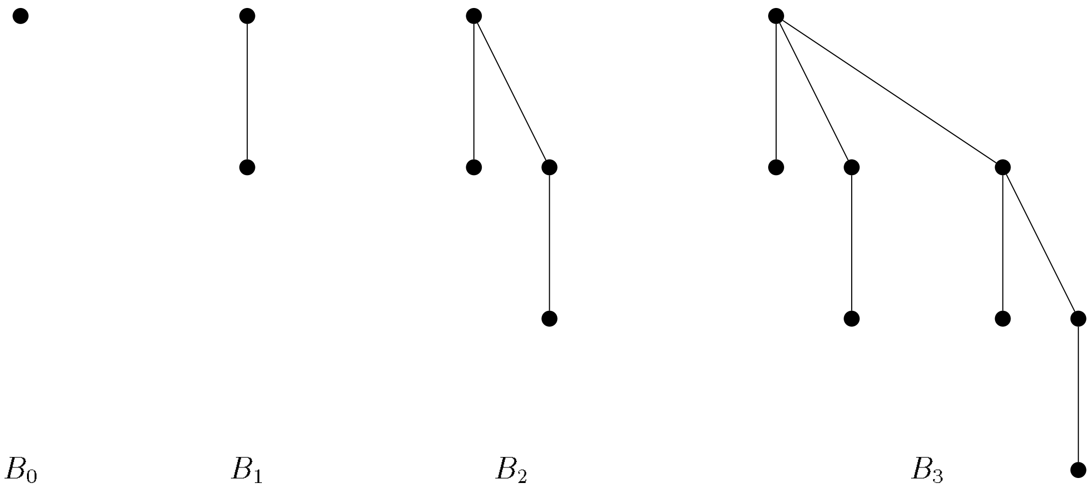
Aufgabe 15

a) Gibt es für einen (anfangs leeren) Fibonacci-Heap eine Folge von Operationen der Arten INSERT, DELETEMIN und DECREASEKEY, so daß die (tatsächliche) Laufzeit der letzten dieser Operationen $\Omega(n)$ ist? (n sei hierbei die Anzahl der gespeicherten Elemente vor dieser letzten Operation.) Begründen Sie Ihre Antwort.

b) Gibt es für einen (anfangs leeren) Fibonacci Heap eine Folge von Operationen der Arten INSERT, DELETEMIN und DECREASEKEY, so daß der F-Heap nach Ausführung der letzten dieser Operationen nur noch genau einen Baum enthält und dieser ein Binomialbaum B_n ist? Begründen Sie Ihre Antwort.

Binomialbäume B_n, $n \geq 0$, sind rekursiv wie folgt definiert:

- B_0 ist ein aus genau einem Knoten bestehender Baum.
- B_{n+1} entsteht aus zwei Exemplaren von B_n, in dem man die Wurzel des einen Baums zu einem (weiteren) Kind der Wurzel des anderen Baumes macht.



Aufgabe 16

Die Lösungen zu dieser wie auch allen folgenden Programmieraufgaben sind per e-mail an `kuehnle@informatik.tu-muenchen.de` zu schicken.

Implementieren Sie eine Priority-queue unter Verwendung des Datentypen `p_queue` aus der Bibliothek LEDA (Library of Efficient Data Types and Algorithms). Ihre Datenstruktur soll die in der Vorlesung besprochenen Operationen insert(<priority>,<identifier>), deletemin() und decreasekey(<new priority>,<identifier>) unterstützen. Ein Element der Datenstruktur enthalte neben der Verwaltungsinformation einen eindeutigen Identifikator (hier `unsigned`) und eine Priorität (ebenfalls `unsigned`). Das Programm soll analog zu dem Rahmen von Aufgabe 12 Befehle der Form i<priority><identifier>, m oder d<priority><identifier> von `stdin` lesen und entsprechende Aktionen ergreifen. Bei m (deletemin) soll außerdem der Identifikator des entfernten Elementes nach `stdout` ausgegeben werden. Sie dürfen davon ausgehen, daß nur Identifikatoren zwischen 1 und 10000 vorkommen.

Auf den hphalle-Maschinen steht Ihnen LEDA unter `/usr/proj/leda` zur Verfügung. Die Datei `MANUAL.ps` enthält die Dokumentation im PostScript-Format. Beachten Sie, daß beim Übersetzen und Binden zusätzliche Verweise auf die Bibliothek anzugeben sind. Um ein Programm `ledatest.c++` zu übersetzen, ist daher folgender Aufruf notwendig:

```
g++ ledatest.c++ -lL -L/usr/proj/leda -I/usr/proj/leda/incl
```

Außerdem ist ein `#include <LEDA/p_queue.h>` im Programm nötig.