Algorithmen und Datenstrukturen – Effiziente Algorithmen I

(Hausaufgaben-Abgabetermin: 8. Dezember 1994, 10:00Uhr)

Aufgabe 1

Sei $d \in \mathbb{N}$. Ein *d-Heap* ist definiert als ein vollständiger d-ärer Baum, der die Heapbedingung erfüllt. Beschreiben Sie eine Implementierung von d-Heaps, die INSERT-Operationen in Zeit $O(\log_d n)$, FIND_MIN-Operationen in konstanter und DELETE-Operationen in Zeit $O(d \cdot \log_d n)$ ermöglicht.

Aufgabe 2

Zeigen Sie, daß ein n-elementiger d-Heap in Zeit $O(n)$ aufgebaut werden kann.

Aufgabe 3

Bei der Standard-Implementierung von (a, b)-Bäumen wird bei INSERT- und DELETE-Operationen zuerst ein Suchpfad von der Wurzel zu einem Blatt durchlaufen und dann gegebenenfalls der Baum bottom-up rebalanziert. Seien $a \geq 2$ und $b \geq 2a$. Zeigen Sie, wie (a, b)-Bäume bereits während des Durchlaufens des Suchpfades, also top-down, rebalanziert werden können, so daß nach dem Einfügen bzw. Löschen des gefundenen Blattes keine bottom-up Balanzierung mehr nötig ist.

Aufgabe 4

Ersetzen Sie Ihre Implementierung von (2,4)-Bäumen aus Aufgabe 4, Übungsblatt 3 einmal durch den Datentyp `sortseq`, das andere mal durch den Datentyp `priority_queue` aus der Bibliothek LEDA (Library of Efficient Data types and Algorithms). Sortierte Sequenzen wurden in LEDA durch (2,4)-Bäume implementiert, Vorrangwarteschlangen durch Fibonacci-Heaps. Führen Sie Testläufe durch und vergleichen Sie die Ausführungszeiten der beiden Datentypen für unterschiedlich lange Operationsfolgen.
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:

```
gcc ledatest.C -liostream -lL -L/usr/local/dist/DIR/GNU/libg++-2.5.3/lib
-L/usr/proj/leda -I/usr/proj/leda/incl
```