Bergische Universität GH Wuppertal
Fachbereich Mathematik
Prof. Dr. A. Frommer, Holger Arndt

Informatik II: Algorithmen und Datenstrukturen

SS 2002

11. Übungsblatt

Aufgabe 32 (Heaps, 4T)

a) Zeigen Sie: Ein Heap mit n Knoten hat die Höhe

$$h = \lfloor \log_2 n \rfloor .$$

($\lfloor x \rfloor$ ist hierbei die größte ganze Zahl kleiner oder gleich x.)

b) Zeigen Sie: Wird ein Heap wie in der Vorlesung als Feld implementiert, so gilt:

Der linke Sohn des an Feldposition i stehenden Knotens befindet sich an Position $2 \cdot i + 1$ des Feldes.

c) Führen Sie folgende Heapoperationen nacheinander aus, und skizzieren Sie, was jeweils geschieht. Stellen Sie den Heap hierbei als Baum dar.

- Einfügen der Elemente mit den Schlüsseln 43, 28, 19, 50, 34 und 20 in einen zunächst leeren Heap
- Löschen des Elements mit maximalem Schlüssel
- Löschen des Elements mit maximalem Schlüssel

Aufgabe 33 (Umwandlung Feld in Heap, 6T)

Der folgende Algorithmus wandelt die ersten n Elemente des Feldes a in einen Heap um:

```
for i = ⌊n/2⌋ − 1, ..., 0 do
    k = i
    while a[k] hat noch einen Sohn und richt_pos_gef = false do
        bestimme Sohn mit maximalem Schlüssel
        (merke dessen Position j)
        if Schlüssel von a[k] < Schlüssel von a[j] then
            vertausche Datensätze a[k] und a[j] (sift down)
            k = j
        else
            richt_pos_gef = true
        end if
    end while
end for
```

a) Veranschaulichen Sie den Algorithmus an folgendem Beispiel: Das Feld a enthalte an den Positionen 0 bis $n = 11$ Datensätze mit den Schlüsseln 10, 15, 50, 41, 23, 7, 38, 3, 65, 19, 20 und 47.

b) Begründen Sie, warum der Algorithmus korrekt ist.

c) Zeigen Sie, dass der Algorithmus nur $\mathcal{O}(n)$ Schlüsselvergleiche und $\mathcal{O}(n)$ Datensatzbewegungen benötigt.

Aufgabe 34 (Filemergesort, 6P)

Programmieren Sie das Verfahren „Externes Mergesort“ zum Sortieren der Datensätze einer Datei `f`.

Hinweise:

- Die Header-Datei `datensatz.h` enthält die Definition des Typs `datensatz`.
- Die Programme in `CreateFile.c` und `PrintFile.c` dienen dazu, Datensätze im Binärformat in eine Datei zu schreiben und wieder auszugeben.
- Ihre Aufgabe ist es, die Datei `FileMergeSort.c` zu ergänzen, die bereits ein Hauptprogramm und die Funktion `void FileMergeSort(FILE *f)` enthält. Schreiben Sie eine Funktion

```
void Merge(int l, int n, FILE *ein1, FILE *ein2, FILE *aus1, FILE *aus2),
```

  die je eine `l`-elementige Folge von Datensätzen aus `ein1` und `ein2` mischt und die resultierende `2l`-elementige Folge abwechselnd nach `aus1` und `aus2` schreibt.
- Allen drei Programmen muss der Name der Datendatei als Parameter übergeben werden. Angommen, die unsortierten Datensätze stehen in `unsort.dat`. Dann empfiehlt sich (unter Unix) etwa folgender Programmaufruf:

```
cp unsort.dat tst.dat; FileMergeSort tst.dat; PrintFile tst.dat
```

- Mit der C-Funktion `FILE *tmpfile(void);` in `stdio.h` lassen sich temporäre Dateien erzeugen. Die Funktion `void rewind(FILE *f);` setzt die Dateiposition (z. B. zum erneuten Lesen der Daten) auf den Anfang der Datei zurück. Die Verwendung von `fread` und `fwrite` können Sie den vorhandenen Programmdateien entnehmen.
- Sie finden alle angegebenen Dateien auf der Webseite http://www.math.uni-wuppertal.de/~arndt/AuD/Uebungsblaetter.

Abgabe: Mi., 03.07.2002, 14 Uhr