Institut für Informatik
der Universität München
Prof. M. Hofmann
Dr. J. Johannsen

SS 2002

26. April 2002

Übungen zur Vorlesung
Effiziente Algorithmen

Blatt 2

Aufgabe P-5: Führen Sie die Prozedur BUILD-HEAP(A) von Hand aus, mit folgendem Array A als Eingabe:

$$[40, 38, 8, 50, 45, 29, 16, 6, 8, 2, 13, 48, 39, 43, 49, 28, 15, 23]$$

Aufgabe P-6: Zeigen Sie, dass $\lfloor \text{heap-size}[A]/2 \rfloor$ der größte Index ist, der noch Nachfolger im Heap hat.

Aufgabe P-7: Implementieren Sie die Datenstrukturen *FIFO-Warteschlange* und *Keller* unter Verwendung der Operationen einer Prioritätsschlange, wie sie in der Vorlesung gezeigt wurden.

Aufgabe P-8: Entwerfen Sie eine Prozedur HEAP-DELETE(A, i), die das Element $A[i]$ aus dem Heap A entfernt und die Heap-Eigenschaft erhält. Zeigen Sie, dass Ihre Prozedur eine Laufzeit von $\Theta(\log n)$ bei einem Heap der Grösse n hat.

Aufgabe P-9: Zeigen Sie, dass QUICKSORT im besten Fall $\Omega(n \log n)$ Operationen auf einem Array der Größe n ausführt.

Aufgabe P-10: Zeigen Sie sorgfältig, dass die in der Vorlesung gezeigte Prozedur PARTITION korrekt ist.

Hausaufgaben:

Aufgabe H-5: Entwerfen Sie eine Prozedur HEAP-INCREASE-KEY(A, i, k), die im Heap A das Element $A[i]$ durch $\max(A[i], k)$ ersetzt und die Heap-Eigenschaft erhält, und eine Laufzeit von $\Theta(\log n)$ bei einem Heap der Größe n hat. (4 Punkte)

Aufgabe H-6: Betrachten Sie die folgende alternative Implementierung von BUILD-HEAP:

```
BUILD-HEAP(A)
    heap-size[A] ← 1
    for i ← 2 to length[A]
        do  HEAP-INSERT(A, A[i])
```

- Erzeugt diese Prozedur bei jeder Eingabe die selbe Ausgabe wie die in der Vorlesung vorgestellte Prozedur BUILD-HEAP? Beweisen Sie dies, oder geben Sie ein Gegenbeispiel an.
- Zeigen Sie, dass auch die obige Prozedur BUILD-HEAP im *worst-case* eine Laufzeit von $\Theta(n \log n)$ hat.

(6 Punkte)

Aufgabe H-7: Zeigen Sie, dass QUICKSORT bei Eingabe eines absteigend sortierten Arrays der Größe n eine Laufzeit von $\Theta(n^2)$ benötigt. (2 Punkte)

Aufgabe H-8: N. Lomuto hat die folgende alternative Partitionierungs-Prozedur für QUICKSORT angegeben:

```
LOMUTO-PARTITION(A, p, r)
    x ← A[r]
    i ← p − 1
    for j ← p to r
        do if A[j] ≤ x
              then i ← i + 1
                   exchange A[i] ↔ A[j]
    if i < r
      then return i
      else  return i − 1
```

- Vergleichen Sie, wie häufig bei PARTITION und LOMUTO-PARTITION ein Array-Element maximal seine Position ändern kann.
- Zeigen Sie, dass die Prozedur LOMUTO-PARTITION korrekt ist, und bei einer Eingabe der Größe n eine Laufzeit von $\Theta(n)$ hat.

(8 Punkte)

Abgabe der Hausaufgaben: Mittwoch, 8. 5. 2002, 10^{15} Uhr.