Prof. G. Zachmann
David Mainzer

TU Clausthal
Institut für Informatik
9. Mai 2010

Sommersemester 2010

Übungen zu Informatik II - Blatt 5

Abgabe in der Übung am 11. 05 / 12. 05. 2010

Bitte beachten Sie, dass die Programmieraufgaben von Ihnen in der Übung vorgeführt und erklärt werden müssen. Zusätzlich senden Sie die Lösung, unter Angabe ihres Namens, an **dm@tu-clausthal.de**.

Aufgabe 1 (Anzahl Blätter im Baum, *2 Punkte*)

Sei l = die Anzahl der Blätter eines binären Baumes und i = die Anzahl innerer Knoten.

Zeigen Sie: in jedem binären Baum (nicht notwendigerweise vollständigen), in dem jeder *innere Knoten* genau 2 Kinder hat, gilt $l = i + 1$.

Tipp: Beweis durch vollständige Induktion.

Aufgabe 2 (Binärbaum, *2 Punkte*)

Für jeden Binärbaum gilt: $h + 1 \leq i \leq 2^h$, wobei h = Höhe des Baumes und i = die Anzahl innerer Knoten.

Geben Sie je ein Beispiel an, wo die untere Schranke bzw. die obere Schranke angenommen wird.

Aufgabe 3 (Komplexität Fibonacci, *3 Punkte*)

Es ist möglich, Fibonacci-Zahlen mit Hilfe von Matrizen zu berechnen.

Im ersten Schritt wird die Gleichung $F_1 = F_1$ und $F_2 = F_0 + F_1$ in Matrixschreibweise dargestellt:

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}, \quad F_0 = 0,\ F_1 = 1$$

Gleichermaßen gilt,

$$\begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^2 \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

Somit kann allgemein gesagt werden:

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \cdot \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

Wenn eine Matrix M gegeben ist, so kann M^n folgendermaßen berechnet werden:

$$M^n = \begin{cases} (M^2)^{n/2} & , \quad n \text{ gerade} \\ (M^2)^{\lfloor n/2 \rfloor} \cdot M & , \quad n \text{ ungerade} \end{cases}$$

Ihre Aufgabe besteht nun darin, die Komplexität des folgenden Algorithmus, welcher die Fibonacci-Zahl auf die oben beschriebene Art und Weise berechnet, zu bestimmen.

```
1  M = ( 0  1 )
       ( 1  1 )
2  R = ( 1  0 )          # Enthält am Ende M^n
       ( 0  1 )
3  while n > 0:
4    if n is even:
5      M = M^2
6    else:
       R = M · R
7      M = M^2
8    n = n/2              # ganzzahlige Division (mit Abrunden)
9  F_n = M_n[0][1]
10 F_{n+1} = M_n[1][1]
```

Gehen Sie davon aus, dass die Multiplikation von zwei Integers $\in \mathcal{O}(1)$ ist.

Bestimmen Sie die Komplexität dieses Algorithmus für $n = 2^k$.

Aufgabe 4 (Anwendung QuickSort, *1+2 Punkte*)

Sortieren Sie die Folge: `13, 19, 9, 5, 12, 8, 21` mit dem Quicksort-Algorithmus unter Verwendung der folgenden Partitions-Methode. Notieren Sie die Zwischenergebnisse.

a) Mittelwert:
Eine weitere Möglichkeit ist die Verwendung des Mittelwerts des ersten und des letzten Elements als Pivot-Wert: $pivot := \lfloor (a[l].key + a[r].key)/2 \rfloor$. Sortieren Sie die oben angegebene Folge mit der Mittelwert-Variante. Beachten Sie, dass bei dieser Variante kein Pivot-Element benutzt wird, sondern ein Pivot-Wert. Das bedeutet, dass das Pivot-Element in der Regel nicht in der Folge vorkommt.

b) Realisieren Sie die beschriebene Mittelwert-Variante von `partition(A, l, r)` in Pseudocode.

Aufgabe 5 (MaxHeap, *3 Punkte*)

Erstellen Sie einen Max-Heap (Baum) aus der folgenden Eingabesequenz: `12, 78, 23, 14, 86, 36, 2`. Beginnen Sie mit einem leeren Heap und fügen Sie der Reihe nach die Elemente ein. Geben Sie dabei **jeden** Zwischenschritt an. Die Zwischenschritte sind: das Einfügen eines neuen Elementes und die schrittweise Wiederherstellung der Heap-Eigenschaft.