Claus/Hagenah/Lewandowski Sommersemester 1999
Aufgabenblatt 2

Übungen zur Vorlesung Entwurf und Analyse von Algorithmen

---

1. Gegeben ist eine RAM (Befehle wie auf der Vorlesungsfolie EA 14). Schreiben Sie ein effizientes Programm für diese RAM, das $\lfloor R_0/R_1 \rfloor$ berechnet und das Ergebnis in R_2 speichert.

2. Die Fibonacci-Zahlen $fib(0), fib(1), fib(2), \ldots$ sind durch die Rekursion

$$fib(0) = fib(1) = 1$$
$$fib(n+2) = fib(n+1) + fib(n), \quad (n \in \mathcal{N})$$

definiert.

a) Entwerfen Sie einen rekursiven Algorithmus mit linearer Zeitkomplexität, der $fib(n)$ für die Eingabe n berechnet. Wieviel Platz braucht der Algorithmus?

b) Zeigen Sie durch Induktion über $1 \leq k < n$:

$$fib(n) = fib(k)fib(n-k) + fib(k-1)fib(n-k-1).$$

Entwickeln Sie mit Hilfe dieses Ergebnisses ein Programm, das $fib(n)$ in $O(\log n)$ Zeit berechnet. Berechnen Sie dazu jeweils Paare aufeinanderfolgender Zahlen der halben Größe und verwenden Sie einen Stapel. Wieviel Platz braucht der Algorithmus?

3. Die Catalansche Zahl $T(n) = \frac{1}{n}\binom{2n-2}{n-1}$ $(n \geq 1)$ stellt die Anzahl der Möglichkeiten dar, ein Produkt von n Faktoren vollständig und korrekt zu klammern. Sie erfüllt die Rekursionsgleichung

$$\begin{aligned} T(1) &= 1 \\ T(n) &= \sum_{i=1}^{n-1} T(i) \cdot T(n-i) \quad (n > 1) \end{aligned}$$

a) Zeigen Sie $\binom{2n}{n} \in \mathcal{O}(4^n)$. *Hinweis:* Betrachten Sie $\binom{2(n+1)}{n+1} \cdot \binom{2n}{n}^{-1}$.

b) Ermitteln Sie mittels Stirlingscher Formel $f(n)$, so dass $\binom{2n}{n} \in \Theta(f(n))$.

c) Berechnen Sie für $n \geq 1$ die Anzahl der Elemente der Menge $B(n)$ der verschiedenen geordneten Binärbäume mit n Knoten mit Hilfe der Catalanschen Zahlen (verschieden bedeutet z.B. $|B(2)| = 2$, da das einzige Blatt entweder als linker oder als rechter Unterbaum auftreten kann.)

4. Sei $T_{mult}(n)$ der Zeitaufwand um zwei $n \times n$-Matrizen zu multiplizieren. Sei $T_{squ}(n)$ der Zeitaufwand um eine $n \times n$-Matrizen zu quadrieren. Zeigen Sie, daß $T_{mult}(n) \in \Theta(T_{squ}(n))$ gilt, d.h. das Quadrieren einer Matrix ist genauso schwer wie Matrixmultiplikation.

5. Wie ist die Platzkomplexität des in der Vorlesung vorgestellten Quicksort-Algorithmus? Entwerfen Sie eine Quicksort-Variante mit einer besseren Platzkomplexität.