3. Übungsblatt zu Grundzüge von Algorithmen und Datenstrukturen, WS 12/13

Prof. Dr. Markus Bläser, Radu Curticapean, Christian Engels
http://www-cc.cs.uni-sb.de/course/38/

Abgabe: Never

Aufgabe 3.1 Es seien $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$ fast überall positive Funktionen. Zeigen Sie die folgenden Äquivalenzen:

- $f \in O(g) \Leftrightarrow \exists C \geq 0 : \lim_{n\to\infty} \frac{f(n)}{g(n)} = C$
- $f \in o(g) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
- $f \in \Omega(g) \Leftrightarrow \exists C \in \mathbb{R}^{>0} \cup \{+\infty\} : \lim_{n\to\infty} \frac{f(n)}{g(n)} = C$
- $f \in \omega(g) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$
- $f \in \Theta(g) \Leftrightarrow \exists C > 0 : \lim_{n\to\infty} \frac{f(n)}{g(n)} = C$

Aufgabe 3.2 Geben Sie einen Algorithmus an, der als Eingabe eine einfach verkettete Liste mit n Elementen erhält und ausgibt, ob diese Liste eine Schleife hat. Ihr Algorithmus soll eine Laufzeit von $O(n)$ und einen Speicherbedarf von $O(1)$ haben. Zeigen Sie, dass ihr Algorithmus korrekt ist und beweisen Sie die Laufzeit.

Hinweis: Mit $O(1)$ Speicher können Sie (nur) eine konstante Anzahl von Zeigern in die Liste verwalten. Zwei Zeiger reichen aus.

Aufgabe 3.3 Zeigen Sie, dass für jede ganze Zahl $n > 0$ und jede beliebige Zahl m, $1 \leq m \leq \lfloor \log n \rfloor + 1$, es immer eine Eingabe mit n Elementen gibt, für die "Binary-Search" Algorithmus genau m Vergleiche macht. (Hinweis: Hierbei zählen die Vergleiche in Zeile 4 und 6 im Skript nur als ein Vergleich.)

Aufgabe 3.4 Der folgende Algorithmus ist bekannt unter dem Titel "Bubble Sort":

Input: array $a[1..n]$

```
1: procedure BUBBLE SORT
2:     Finished ← FALSE
3:     while Finished = FALSE do
4:         Finished ← TRUE
5:         for i ← 1 to n − 1 do
6:             if a[i] > a[i + 1] then
7:                 swap a[i], a[i + 1]
8:                 Finished ← FALSE
```

Zeigen Sie, dass dieser Algorithmus immer nach einer endlichen Anzahl von Iterationen das Array sortiert hinterlässt. Benutzen Sie dazu eine Invariante. Wie viele Schritte benötigt der Algorithmus im schlimmsten Fall?

Aufgabe 3.5 Die Fibonacci-Zahlen $\{F_n\}$, $n \geq 0$, werden definiert als $F_0 = 0$, $F_1 = 1$ und $F_{n+1} = F_n + F_{n-1}$ für $n \geq 1$. Überlegen Sie sich, wie man mittels $O(\log n)$ arithmetischen Operationen die Fibonacci-Zahl F_n ausrechnen kann. Erlaubt sind in dieser Aufgabe nur Operationen mit *ganzen* Zahlen. Die Variablen Ihres Algorithmus dürfen Werte bis zu $2^{O(n)}$ beinhalten. (Insbesondere passt also die n-te Fibonacci-Zahl in eine einzelne Variable.)