Universität Karlsruhe (TH)
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders, Dr. Rob van Stee
G. Veit Batz, Johannes Singler

0. Übungsblatt zur Algorithmentechnik WS 2007/08

http://algo2.iti.uni-karlsruhe.de/algotech.php
{sanders|batz|singler}@ira.uka.de

Aufgabe 1 (*O-Notation, Schwierigkeit* $1+1+1$)

a) Zeigen Sie: $g(n) = O(f(n)) \implies O(f(n)+g(n)) = O(f(n))$.

b) Zeigen Sie: $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$.

c) Für $o(f(n))$ wird auch die folgende, alternative Definition verwendet:

$$g(n) = o(f(n)) \quad :\Longleftrightarrow \quad \lim_{n\to\infty} |g(n)/f(n)| = 0\,.$$

Es soll nun bewiesen werden, dass beide Definitionen äquivalent sind für $g : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0}$ und $f : \mathbb{N}_{\geq 0} \to \mathbb{R}_{>0}$. Zeigen Sie also:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0 \quad \Longleftrightarrow \quad \forall c \in \mathbb{R}_{>0} \colon \exists n_0 \in \mathbb{N}_{>0} \colon \forall n \geq n_0 \colon g(n) \leq cf(n)$$

Aufgabe 2 (*Schleifeninvariante, Schwierigkeit* $2+3$)

a) Was ist eine Schleifeninvariante? Wie beweist man die Korrektheit von Schleifen mittels Schleifeninvarianten (Beweisidee)?

b) Beweisen Sie die Korrektheit der binären Suche mittels Schleifeninvariante (Algorithmus umseitig).

Aufgabe 3 (*Master-Theorem, Schwierigkeit* $4+1$)

a) Beweisen Sie die umseitig abgedruckte, vereinfachte Version des Master-Theorems.

b) Die Laufzeit des Strassen-Algorithmus zur Multiplikation von Matrizen lässt sich durch die Rekurrenz $T(n) = 7T(n/2) + O(n^2)$ beschreiben. Zeigen Sie mit Hilfe des vereinfachten Master-Theorems, dass $T(n) = O(n^{\log_2 7})$ gilt.

Hinweise. Die **Lösungen** werden jeweils in der nächsten Übung vorgestellt, im Falle dieses Übungsblattes also am **Donnerstag, den 8.11.2007 ab 15.45 Uhr im HMU.** Eine Abgabe zur Korrektur ist **nicht** möglich. Alle Aufgaben sind mit einem Schwierigkeitsgrad von 1 bis 5 versehen, wobei 1 am einfachsten und 5 am schwersten ist. Die 5 wird allerdings nur an ungelöste Probleme aus der Forschung vergeben, d. h. normalerweise ist 4 die höchste Bewertung.

Definition (O-Notation). Folgende Mengen bilden die Grundlage für die *O-Notation*: Gegeben sei eine Funktionen $f : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Dann definieren wir wie folgt:

$$\begin{aligned} O(f(n)) &:= \{g : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0} \mid \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N}_{>0} \colon \forall n \geq n_0 \colon g(n) \leq cf(n)\} \\ \Omega(f(n)) &:= \{g : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0} \mid \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N}_{>0} \colon \forall n \geq n_0 \colon g(n) \geq cf(n)\} \\ o(f(n)) &:= \{g : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0} \mid \forall c \in \mathbb{R}_{>0} \colon \exists n_0 \in \mathbb{N}_{>0} \colon \forall n \geq n_0 \colon g(n) \leq cf(n)\} \\ \omega(f(n)) &:= \{g : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0} \mid \forall c \in \mathbb{R}_{>0} \colon \exists n_0 \in \mathbb{N}_{>0} \colon \forall n \geq n_0 \colon g(n) \geq cf(n)\} \end{aligned}$$

Weiter sei $\Theta(f(n)) := O(f(n)) \cap \Omega(f(n))$. □

Satz (vereinfachtes Master-Theorem). Seien $a, b \in \mathbb{N}$ mit $a \geq 1, b \geq 2$ und $c, d, k \in \mathbb{R}_{>0}$. Weiter sei $T(n)$ durch die Rekurrenzbeziehung

$$T(n) = \begin{cases} d & \text{für } n = 1 \\ aT(n/b) + cn^k & \text{für } n > 1 \end{cases}$$

gegeben. Dann gilt für $T(n)$ die asymptotische Abschätzung

$$T(n) = \begin{cases} O(n^{\log_b a}) & \text{für } a > b^k \\ O(n^k \log n) & \text{für } a = b^k \\ O(n^k) & \text{für } a < b^k \end{cases} .$$

□

Algorithmus (Binäre Suche). Für ein sortiertes Array a der Länge n von paarweise verschiedenen reellen Zahlen (d. h. $a[1..n]$ mit $a[1] < a[2] < \cdots < a[n]$), sowie für ein $x \in \mathbb{R}$ berechnet der folgende Algorithmus den Index i mit $a[i-1] < x \leq a[i]$ (wobei $a[0] = -\infty$ und $a[n+1] = \infty$ angenommen werden).

$(\ell, r) := (0, n+1)$
while *true* **do**

  if $\ell + 1 = r$ **then return** $\ell + 1$

  $m := \lfloor (r + \ell)/2 \rfloor$

  if $x = a[m]$ **then return** m
  else if $x < a[m]$ **then** $r := m$
  else $\ell := m$

end

□