# Algorithmische Mathematik I

Winter Semester 2015 / 2016
Prof. Dr. Sven Beuchler
Markus Siebenmorgen

universitätbonn

## Aufgabenblatt 5.

Abgabedatum: **25.11.2015**.

**Aufgabe 1.** (Mergesort)

Der benötigte Speicherplatz $S(n)$ für den Sortieralgorithmus Mergesort zum Sortieren von $n = 2^k$, $k \in \mathbb{N}$, Zahlen ist gegeben durch

$$S(n) = n + \frac{n}{2} + S\Big(\frac{n}{2}\Big), \qquad S(1) = 1.$$

Zeigen Sie, dass gilt

$$S(n) = 3n - 2.$$

**Hinweis.** Verwenden Sie Aufgabe 3a) von Blatt 0.

(5 Punkte)

**Aufgabe 2.** (Sortieren)

a) Wenden Sie den Sortieralgorithmus Bubble-Sort auf die Zahlensequenz $5, 2, 3, 4, 1$ an und schreiben Sie alle Vertauschungsschritte auf.

b) Geben Sie die Zahl der für Aufgabenteil a) notwendigen Vertauschungen und Vergleiche für das Bubble-Sort-Verfahren an.

c) Wenden Sie nun den Sortieralgorithmus Merge-Sort auf die Zahlensequenz $5, 2, 3, 4, 1$ an und schreiben Sie alle split- und merge-Schritte auf.

d) Geben Sie die Zahl der für Aufgabenteil c) notwendigen split- und merge-Schritte und Vergleiche für das Merge-Sort-Verfahren an.

(4 Punkte)

**Aufgabe 3.** (Asymptotische Laufzeitnotationen)

Zur Geschwindigkeits- und Speicherverbrauchsanalyse bei Algorithmen verwendet man die Notationen

$$\begin{aligned}
\mathcal{O}(g(n)) &:= \{f\colon \mathbb{N} \to \mathbb{R}_+ \,|\, \exists c > 0 \text{ und } \exists n_0 \in \mathbb{N} \text{ mit } f(n) \le c \cdot g(n)\ \forall n \ge n_0\},\\
o(g(n)) &:= \{f\colon \mathbb{N} \to \mathbb{R}_+ \,|\, \forall c > 0\, \exists n_0 \in \mathbb{N} \text{ mit } f(n) \le c \cdot g(n)\ \forall n \ge n_0\},\\
\Omega(g(n)) &:= \{f\colon \mathbb{N} \to \mathbb{R}_+ \,|\, \exists c > 0 \text{ und } \exists n_0 \in \mathbb{N} \text{ mit } f(n) \ge c \cdot g(n)\ \forall n \ge n_0\},\\
\Theta(g(n)) &:= \mathcal{O}(g(n)) \cap \Omega(g(n)).
\end{aligned}$$

Wenn bspw. die Anzahl der Rechenschritte eines Programms durch $f(n)$ in Abhängigkeit der Eingabegröße $n$ gegeben ist, bedeuten obige Definitonen

$f(n) = \mathcal{O}(g(n))$**:** auf *jedem* Rechner benötigt man *maximal* irgendeine Konstante mal $g$ Rechenschritte (spätestens ab einem gewissen $n$).

$f(n) = o(g(n))$**:** Die Laufzeit von $f$ hat *echt kleinere* Größenordnung als $g$ (man könnte schreiben $\mathcal{O}(f(n)) < \mathcal{O}(g(n)) \Leftrightarrow f(n) = o(g(n))$).

$f(n) = \Omega(g(n))$**:** man braucht *mindestens* $c \cdot g(n)$ Rechenoperationen (untere Schranke).

$f(n) = \Theta(g(n))$**:** wegen $f(n) = \mathcal{O}(g(n))$ braucht $f$ maximal $\bar{c} \cdot g$ Schritte und wegen $f(n) = \Omega(g(n))$ braucht $f$ minimal $\underline{c} \cdot g$ Schritte.

a) Definieren Sie die Notationen von $\mathcal{O}(g(n), o(g(n))$ und $\Omega(g(n))$ über den Grenzwert der Folge $\{f(n)/g(n)\}_{n\in\mathbb{N}}$, falls dieser existiert.

b) Beweisen oder wiederlegen Sie die folgenden Aussagen:

1. $n^6 = \mathcal{O}(2^n)$
2. $42n^3 + 13n^2 + 2n + 500 = \mathcal{O}(n^3)$
3. $42n^3 + 13n^2 + 2n + 500 = o(n^2)$
4. $g(n) = o(f(n))$ impliziert $f(n) + g(n) = \Theta(f(n))$
5. $f(n) = \Theta(g(n))$ impliziert $g(n) = \Theta(f(n))$

c) Sortieren Sie die folgenden Funktionen bezüglich ihrer Größenordnung, d.h. geben Sie eine Reihenfolge $g_1, \ldots, g_k$ an mit $g_i = \Omega(g_{i+1})$. Geben Sie dabei an, wenn mehrere $g_i$ dieselbe Größenordnung haben (d.h. wenn $g_i(n) = \Theta(g_j(n))$).

$$(\sqrt{2})^{\log_2(n)}, \quad n^2, \quad \left(\frac{3}{2}\right)^n, \quad n^3, \quad 2^n, \quad \log_2^2(n), \quad 2^{\log_2(n)}, \quad \sqrt{n},$$

$$10^4, \quad e^n, \quad n\log_2(n), \quad n!, \quad 2^{(2^n)}$$

**Bemerkung:** Aufgabenteil c) dient dazu ein Gefühl für Komplexitäten zu erhalten. Die Einordnung der letzten beiden Terme $n!, 2^{(2^n)}$ darf ohne Begründung angegeben werden und diese beiden Terme zählen nicht in die Bewertung. Zudem darf in der gesamten Aufgabe vorausgesetzt werden, dass z.B. $1/n^k$ oder $n^k/e^n$ eine Nullfolge ist.

(8 Punkte)

**Aufgabe 4.** (Aufwand elementarer Matrix-Vektoroperationen)

Gegeben seien zwei Vektoren $x, y \in \mathbb{R}^n$ und eine Matrix $A \in \mathbb{R}^{n\times n}$. Bestimmen Sie den Aufwand

a) des Skalarproduktes der Vektoren $x$ und $y$.

b) der Matrix-Vektormultiplikation von $A$ und $x$.

Zählen Sie hierfür jeweils die Anzahl der Additionen und Multiplikationen und geben Sie zudem den Aufwand in der $\mathcal{O}$-Notation an.

(3 Punkte)

**Aufgabe 5.** (Präsenzübung: Mergesort)

Überlegen Sie sich wie man den Mergesort aus der Vorlesung ohne Rekursion implementieren kann und schreiben Sie dies in der Pseudocode-Notation auf.

Die Präsenzübung wird in der Woche 23.-27.11 in den Programmiertutorien besprochen.

**Programmieraufgabe 1.** (Elementare Vektoroperation)

Gegeben seien Vektoren $x, y \in \mathbb{R}^n$ sowie eine reelle Zahl $\alpha$. Implementieren Sie in C/C++ eine Funktion `VDAXPY(x,y,alpha,n)` zur Berechnung von

$$z_i = x_i + \alpha y_i \qquad \text{für } i = 1, \ldots, n.$$

Dabei soll das Ergebnis $z$ auf dem Wert von $y$ gespeichert werden.

(2 Punkte)

**Programmieraufgabe 2.** (Mergesort)

Schreiben Sie ein C/C++ Programm, dass die **Nicht**-rekursive Version des Mergesorts umsetzt. Implementieren Sie also den Pseudocode aus der Präsenzaufgabe.

(6 Punkte)

Die Programmieraufgabe wird in der Woche vom 07.12-11.12 im Cip-Pool Endenicher Allee oder im Cip-Pool Wegelerstraße abgegeben/vorgestellt. In der Woche vom 30.11-04.12 werden in den Cip-Pools Listen für die Abgabe aushängen.

**Programmieraufgabe 3.** (Präsenzübung: Quicksort ohne Doppelvergleiche)

In der Vorlesung wurde in Algorithmus 1.9 quicksort vorgestellt.

a) Ist der angehängte Quelltext eine korrekte Implementation von quicksort?

b) Sortiert dieser Algorithmus $n$ Zahlen?

c) Entwickeln Sie eine korrrekte Implementation von quicksort ohne zusätzlichen Speicher mit möglichst wenig Vergleichen und setzen Sie diese in C/C++ um.

**Die Fachschaft Mathematik feiert am 26.11. ihre Matheparty in der N8schicht. Der VVK findet am Mo. 23.11., Di. 24.11. und Mi. 25.11. vor der Mensa Poppelsdorf statt. Alle weiteren Infos findet Ihr auf fsmath.uni-bonn.de.**