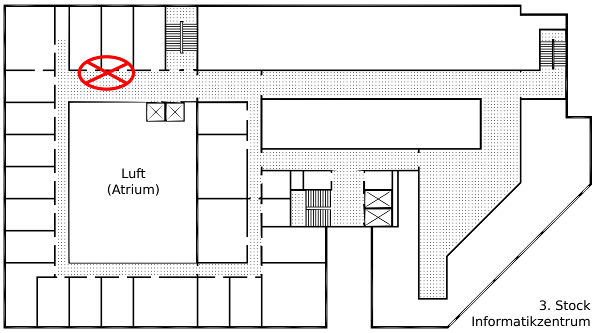
**Abteilung Algorithmik** **Winter 2016/17**
**Institut für Betriebssysteme und Rechnerverbund**
**TU Braunschweig**

Prof. Dr. Sándor P. Fekete
Phillip Keldenich
Arne Schmidt

# Algorithmen und Datenstrukturen
# Übung 6 vom 23. 1. 2017

Abgabe der Lösungen bis zum Montag, den 6. 2. 2017 um 13:15 im Hausaufgabenrückgabeschrank bei Raum IZ 337. Wir geben **keine** Punkte auf Lösungen mit Bleistift. Bitte einen nicht-löschbaren Stift verwenden!
**Bitte die Blätter zusammenheften und vorne deutlich mit eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!**




Das Blatt wird am 9. 2. 2017 in der großen Übung besprochen.

**Aufgabe 1 (Klausurvorbereitung):** Gib deinen Namen (Format: Nachname, Vorname), Matrikelnummer, Studiengang und angestrebten Abschluss *leserlich* (in Druckschrift) an. Diese Angaben brauchen wir für die Weiterleitung der Klausurergebnisse, also gebt euch bitte Mühe ;-) . **(2 Punkte)**

**Aufgabe 2 (Mastertheorem):** Bestimme die Laufzeiten der folgenden Funktionen mithilfe des in der Vorlesung vorgestellten Mastertheorems. Gib dabei alle im Mastertheorem auftretenden Parameter an.

a) $T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + 17n + T(\frac{n}{6}) + 57$

b) $U(n) = 26 \cdot U(\frac{n}{3}) + 4n^3 + 2 \cdot U(\frac{n}{5}) + 7n + 13$

c) $V(n) = 3 \cdot V(\frac{n}{2}) + 4 \cdot V(\frac{n}{4}) + 23n + 2$

d) $W(n) = 243 \cdot W(\frac{n}{3}) + 3n^2$

**(5+5+5+5 Punkte)**

**Aufgabe 3 (Erzeugende Funktionen):** Gegeben sei die Reihe $a_0 = 1$, $a_n = -a_{n-1}$. Finde eine geschlossene Form für $a_n$ mithilfe einer erzeugenden Funktion. **(5 Punkte)**

**Aufgabe 4 (Bubblesort):**

a) Sortiere das in Abbildung 1 stehende Array $A$ mit Hilfe von Bubblesort. Gib dabei nach jeder Tauschoperation das getauschte Paar, sowie das Array nach jeder Iteration der äußeren For-Schleife an.

b) Zeige, dass jeder Sortieralgorithmus, der nur benachbarte Elemente vertauscht, im Worst Case die Laufzeit $\Omega(n^2)$ besitzt.

(*Hinweis:* Betrachte die Summe der Distanzen der Elemente zu ihrer Zielposition.)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 2 | 5 | 17 | 7 | 35 | 12 | 1 | 3 | 4 | 51 | 64 |

**Abbildung 1:** Das Array $A$

**(15+5 Punkte)**

**Aufgabe 5 (Mediane):** Ziel dieser Aufgabe ist es, einen effizienten Algorithmus zu entwickeln, der den Median eines unsortierten Arrays bestimmt. Der Median, das *mittlere Element* eines Arrays, ist ein Element $x \in A$, für das Folgendes gilt.

- Es gibt höchstens $\frac{n}{2}$ Elemente, die kleiner sind als $x$: $|\{y \in A \,|\, y < x\}| \leq \frac{n}{2}$.
- Es gibt höchstens $\frac{n}{2}$ Elemente, die größer sind als $x$: $|\{y \in A \,|\, y > x\}| \leq \frac{n}{2}$.

Bei Arrays mit einer ungeraden Anzahl von unterschiedlichen Elementen ist der Median eindeutig bestimmt — so ist z.B. 3 der Median von $\{2, 3, 4\}$. Bei Arrays mit gerader Länge gibt es mehr als einen Median — so sind z.B. sowohl 3 als auch 4 Mediane von $\{2, 3, 4, 5\}$.

Um einen Median eines Arrays zu bestimmen, kann man zunächst das Array sortieren und das mittlere Element zurückgeben. Mit einem effizienten Sortieralgorithmus wie z.B. Mergesort braucht dies $\Omega(n \log n)$ Operationen. Es geht aber noch schneller, was auch in der Vorlesung behandelt wird (siehe Vorlesung vom 01.02.17).

Alternativ zum in der Vorlesung vorgestellten Verfahren kann man das Problem mit einem auf der Idee von Quicksort basierenden Algorithmus mit durchschnittlich $O(n)$ Operationen lösen. Dabei kann man in jedem Zwischenschritt annehmen, dass im Durchschnitt ein Pivotelement gefunden wird, welches das Array im Verhältnis 3:1 teilt; dies bedeutet, dass das größere Teilarray $\frac{3}{4}n$ Elemente enthält.

a) Gib einen Algorithmus (in Pseudocode) mit durchschnittlicher Laufzeit $O(n)$ an, der auf der Pivotisierungsidee von Quicksort basiert und einen Median des gegebenen Arrays bestimmt.

Hinweise:
(1) Der Median befindet sich nach der Pivotisierung in höchstens einem Teilarray.
(2) Für ein Teilarray muss bekannt sein, wieviele Elemente links bzw. rechts liegen.

b) Beweise mithilfe des Mastertheorems, dass dein Algorithmus die durchschnittliche Laufzeitschranke $O(n)$ einhält.

**(10+3 Punkte)**