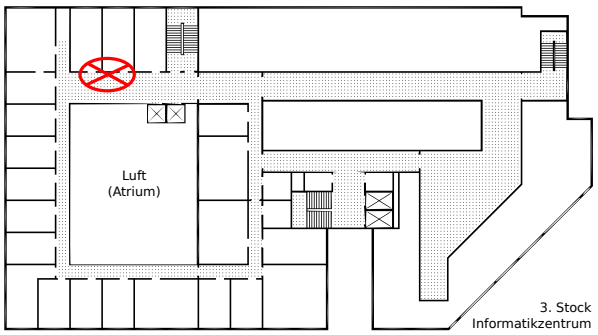
Abteilung Algorithmik **Winter 2016/17**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

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

Algorithmen und Datenstrukturen
Übung 3 vom 28. 11. 2016

Abgabe der Lösungen bis zum Montag, den 12. 12. 2016 um 13:15 im Hausaufgabenrückgabeschrank bei Raum IZ 337.

Bitte die Blätter zusammenheften und vorne deutlich mit eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!



In der Vorlesung wurde die $\mathcal{O}$-Notation zur Abschätzung des Wachstums von Funktionen eingeführt. Verwende für die folgenden Aufgaben ausschließlich Definitionen und Aussagen, die in der Vorlesung oder der großen Übung präsentiert wurden.

Aufgabe 1 ($\mathcal{O}$-Notation und Logarithmen): Zeige oder widerlege:

a) Für alle $a, c, d \in \mathbb{R}$ mit $a, c, d > 1$ gilt:

$$\log_a(c^n) \in \Theta(\log_a(d^n)).$$

b) Für alle $a, b \in \mathbb{R}$ mit $a, b > 1$ und alle Funktionen $f : \mathbb{N} \to \mathbb{R}$ mit $f(n) > 1$ gilt:

$$\log_a(f(n)) \in \Theta(\log_b(f(n))).$$

c) Für alle $c, d \in \mathbb{R}$ mit $c, d > 1$ gilt:

$$c^n \in \Theta(d^n).$$

d) Für alle $a, b, c \in \mathbb{R}$ mit $a, b, c > 1$ gilt:

$$c^{\log_a(n)} \in \Theta(c^{\log_b(n)}).$$

e) Seien $a \in \mathbb{R}$, $a > 1$ und $r \in \mathbb{N}$, $r \geq 2$. Sei $l_r(n)$ die *Länge* von n bei Darstellung zur Basis r, d.h. die Anzahl von Ziffern, die n besitzt, wenn man n ohne führende Nullen als Zahl zur Basis r aufschreibt[1]. Dann ist $l_r(n) \in \Theta(\log_a(n))$.

(2+3+3+3+3 Punkte)

[1] Also zum Beispiel $l_{10}(1234) = 4$.

Aufgabe 2 ($\mathcal{O}$-Notation):

a) Zeige durch Angabe geeigneter Konstanten c (bzw. c_1, c_2) sowie n_0, dass die folgenden Funktionen tatsächlich in der angegebenen Klasse liegen. Beweise, dass deine Konstanten korrekt gewählt sind.

$$\begin{array}{rcrcl} f_1(n) & = & \frac{1}{7}n - 4\sqrt{n} & \in & \Omega(n) \\ f_2(n) & = & n^2 + n\log_2 n & \in & \Theta(n^2) \\ f_3(n) & = & \frac{1}{\sqrt{n}} + 4242 & \in & \mathcal{O}(1) \\ f_4(n) & = & \log_2(n^{17}) & \in & \Theta(\log_3 n) \\ f_5(n) & = & 5 \cdot 2^n & \in & \mathcal{O}(e^n) \end{array}$$

b) Kreuze an, in welchen Klassen die jeweilige Funktion liegt (ohne Begründung).

$f(n)$	$\mathcal{O}(1)$	$\Omega(1)$	$\Theta(1)$	$\mathcal{O}(n)$	$\Omega(n)$	$\Theta(n)$	$\mathcal{O}(n^2)$	$\Omega(n^2)$	$\Theta(n^2)$
$\frac{n^{1.99}\log n}{0.1n}$									
$\log_2(n!)$									
$3^{3\log_3(n)}$									
$\sqrt{n\log_2^2(n^2)}$									
$2^{2^{100}}n$									
$1/n^2$									

c) In welcher Beziehung stehen die folgenden Klassen zueinander? Schreibe $\subseteq$ in das Feld, wenn Klasse A in Klasse B enthalten ist, $\supseteq$, wenn Klasse B in Klasse A enthalten ist, $=$, wenn die Klassen A und B übereinstimmen und $\times$, wenn dies alles nicht zutrifft. Eine Begründung ist nicht notwendig.

$A \setminus B$	$\mathcal{O}(\log_2 n)$	$\mathcal{O}(n)$	$\Omega(n^2)$	$\Theta(\sqrt{n})$	$\Omega(2+\cos(n))$
$\Omega(1)$					
$\mathcal{O}(\log_3 n)$					
$\Omega(n^3)$					
$\mathcal{O}(n^2)$					
$\mathcal{O}(e^n)$					
$\Omega(n!)$					

(10+6+6 Punkte)

Aufgabe 3 ($\mathcal{O}$-Notation): Mit der $\mathcal{O}$-Notation lassen sich Funktionen nach ihrem asymptotischen Wachstum ordnen. Seien $f, g, h : \mathbb{N} \to \mathbb{R}$ Funktionen. Wir definieren folgende Relationen:

$$f \preceq g \Leftrightarrow f \in \mathcal{O}(g)$$
$$f \simeq g \Leftrightarrow f \in \Theta(g)$$

Beweise folgende Aussagen[2]:

a) $f \preceq f$,

b) $f \preceq g$ und $g \preceq h \Rightarrow f \preceq h$,

[2]Für diejenigen unter euch, die Spaß an Mathematik haben: Wir wollen darauf hinaus, dass $\preceq$ eine Halbordnung auf den durch $\simeq$ gegebenen Äquivalenzklassen ist.

c) $f \preceq g$ und $g \preceq f \Rightarrow f \simeq g$.

(1+4+4 Punkte)

Aufgabe 4 (Laufzeit von Algorithmen): Du hast einen Input von n Elementen und einen Algorithmus, der einzelne Elemente bearbeitet. Das kostet $n^2 + 7 \log n$ Operationen pro Element. Alle Elemente müssen bearbeitet werden.

a) Wie lange dauert das (mit und ohne Θ-Notation)?

b) Die Bearbeitung zweier Elemente ist unabhängig voneinander, man kann sie also parallelisieren. Dazu stehen 8 Kerne zur Verfügung. Wie wirkt sich dies auf die Gesamtlaufzeit aus (mit und ohne Θ-Notation)?

c) Alternativ kannst du ein Preprocessing verwenden, welches in $3n \log n + 4n - 7$ läuft und danach die Bearbeitungszeit pro Element auf $\log n$ reduziert, allerdings ist die Bearbeitung der Elemente dann nicht mehr parallelisierbar. Wie wirkt sich das auf die Gesamtlaufzeit aus (mit und ohne Θ-Notation)?

d) Für welche Möglichkeit (ursprünglicher Algorithmus, b), oder c)) entscheidest du dich, um die Leistungsfähigkeit des Programms für große Inputmengen zu maximieren? Begründe deine Wahl.

(3+4+5+3 Punkte)