Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Algorithmik

Wintersemester 2024/2025

Version 2024-11-14 17:26

Prof. Dr. Sebastian Wild
Dr. Nikolaus Glombiewski

Übungen zur Vorlesung

Effiziente Algorithmen

Präsenzübung 0

Aufgabe: Komplexität

a) Beweisen oder widerlegen Sie die folgenden Aussagen:

i) $\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n \in \mathcal{O}(n^3)$

$\lim_{n\to\infty} \frac{\frac{1}{6}n^3+\frac{1}{2}n^2+\frac{1}{3}n}{n^3} = \lim_{n\to\infty} \frac{1}{6} + \frac{1}{2n} + \frac{1}{3n^2} = \frac{1}{6}$

ii) $4n^3 + 2n + 1 \in \Theta(6n^3 + n + 12)$

$\lim_{n\to\infty} \frac{4n^3+2n+1}{6n^3+n+12} \overset{\text{L'Hôpital}}{=} \lim_{n\to\infty} \frac{12n^2+2}{18n^2+1} \overset{\text{L'Hôpital}}{=} \lim_{n\to\infty} \frac{24n}{36n} = \frac{2}{3}$
Kehrwert existiert 1P.

iii) $n\log_2(n) \in \mathcal{O}(n^2)$

$\lim_{n\to\infty} \frac{n\log_2(n)}{n^2} = \lim_{n\to\infty} \frac{\log_2(n)}{n} \overset{\text{L'Hôpital}}{=} \frac{1}{n} = 0$

iv) $n \cdot \sqrt{\log n} \in \Omega(n \cdot \log(n^2))$

$Falls\ \exists c \in R^+, n_0 \in N : \forall n > n_0 : f(n) \le c \cdot g(n)$, dann gilt f $\in \mathcal{O}(g)$

$\Rightarrow f(n) = n \cdot log\ (n^2), g(n) = n \cdot \sqrt{log\ n}$
$\Rightarrow n \cdot log\ (n^2) \le c \cdot n \cdot \sqrt{log\ n}$
$\Rightarrow 2n \cdot log\ n \le c \cdot n \cdot \sqrt{log\ n}$
$\Rightarrow 2 \cdot \sqrt{log\ n} \cdot \sqrt{log\ n} \le c \cdot \sqrt{log\ n}$
$\Rightarrow 2 \cdot \sqrt{log\ n} \le c$ Widerspruch
$\Rightarrow f \notin \mathcal{O}(g) \Rightarrow g \notin \Omega(f)$ 2P.

v) Aus $f(n) \in \Theta(n)$ folgt $2^{f(n)} \in \mathcal{O}(2^n)$

Aussage falsch:
Wähle z.B. $f(n) = 2n$. Dann ist $4^n \notin \mathcal{O}(2^n)$
$\lim_{n\to\infty} \frac{4^n}{2^n} = \lim_{n\to\infty} 2^n = \infty$

b) Implementieren Sie den in der Vorlesung vorgestellten Algorithmus *Bubblesort* in Java für ein ganzzahliges Array (`int[]`).

Testen Sie die Laufzeit Ihrer Implementierung, indem Sie ein zufälliges Array der Größe n generieren und anschließend den Algorithmus auf das Array anwenden.

Führen Sie das Experiment mehrfach aus und variieren Sie n, indem Sie die Größe schrittweise verdoppeln. Vergleichen Sie anschließend die Ergebnisse mit der Methode `Arrays.sort` der Java Standardbibliothek.