Vergleich zweier Algorithmen

(*Algorithmen-Vergleich*)

Stichwörter: Algorithmische Komplexität (O-Notation)

Seien **A** und **B** zwei Algorithmen, die dasselbe Problem lösen. Zur Lösung von Problemen der Eingabegröße n benötigt Algorithmus **A** $500 \cdot n^2 - 16 \cdot n$ Elementoperationen und **B** $\frac{1}{2} \cdot n^3 + \frac{11}{2} \cdot n + 7$ Elementoperationen.

$$A(n) = 500 \cdot n^2 - 16 \cdot n$$
$$B(n) = \frac{1}{2} \cdot n^3 + \frac{11}{2} \cdot n + 7$$

(a) Wenn Sie ein Problem für die Eingabegröße 256 lösen wollen, welchen Algorithmus würden Sie dann wählen?

Lösungsvorschlag

Wir können die Aufgabe durch Einsetzen lösen, d. h. wir berechnen explizit die Anzahl benötigter Elementoperationen und vergleichen. Algorithmus A benötigt $500 \cdot n2 - 16 \cdot n$ Operationen bei einer Eingabe der Größe n, also bei $n = 256$ genau

$$\begin{aligned} A(256) &= 500 \cdot 256^2 - 16 \cdot 256 \\ &= 32763904 \end{aligned}$$

In der gleichen Art können wir den Aufwand von Algorithmus B berechnen:

$$\begin{aligned} B(256) &= \frac{1}{2} \cdot 256^3 + \frac{11}{2} \cdot 256 + 7 \\ &= 8390023 \end{aligned}$$

In diesem Fall benötigt Algorithmus B also deutlich weniger Elementoperationen.

(b) Wenn Sie ein Problem lösen wollen, deren Eingabegröße immer mindestens 1024 ist, welchen Algorithmus würden Sie wählen? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Da in der Aufgabenstellung von Eingaben der Größe mindestens 1024 die Rede ist, stellen wir uns die Frage, für welche n Algorithmus A schneller als B ist, also für welche n

$$500 \cdot n^2 - 16 \cdot n < \frac{1}{2} \cdot n^3 + \frac{11}{2} \cdot n + 7$$

gilt. Dies lässt sich äquivalent umformen zu

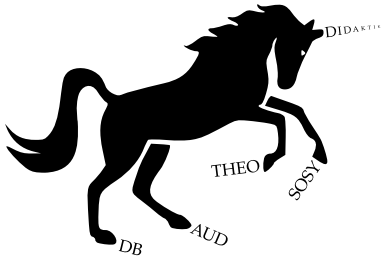
$$\frac{1}{2} \cdot n^3 - 500 \cdot n^2 + \frac{43}{2} \cdot n + 7 > 0$$

Diese Ungleichung ist erfüllt, wenn allein $\frac{1}{2} \cdot n^3 - 500 \cdot n^2 > 0$ gilt, denn es gilt $\frac{43}{2} \cdot n + 7 > 0$. Das Problem reduziert sich also zu

$$\frac{1}{2} \cdot n^3 - 500 \cdot n^2 > 0 \Leftrightarrow n > 1000$$

Auf jeden Fall ist A schneller als B für $n > 1000$ (man erinnere sich, dass das bei n = 256 noch andersherum war). Wie ist dieses Verhalten zu erklären?

Obwohl der Aufwand von A im O-Kalkül $\mathcal{O}(n^2)$ und der von B $\mathcal{O}(n^3)$ ist, man also geneigt sein könnte zu sagen, A ist immer schneller als B, stimmt das nicht immer. Im Einzelfall können es durchaus große Konstanten (wie in diesem Fall die 500) sein, die dafür sorgen, dass n erst einmal sehr groß werden muss, damit sich die Laufzeiten tatsächlich so verhalten, wie erwartet. Wenn nur kleine Eingaben verarbeitet werden sollen, kann es manchmal also durchaus lohnenswert sein, einen $\mathcal{O}(n^3)$-Algorithmus anstatt eines $\mathcal{O}(n^3)$-Algorithmus zu verwenden, wenn die im O-Kalkül unterschlagenen Konstanten zuungunsten des eigentlich langsameren Algorithmus sprechen.



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/50_Algorithmische-Komplexitaet/Aufgabe_Algorithmen-Vergleich.tex`