Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Michal Mnuk

SS 1998
Übungsblatt 1
13. Mai 1998

Grundlegende Algorithmen

Abgabe bis 20. Mai 1998 in der Übung

Aufgabe 1

a) Stellen Sie sich einen Computer vor, der in der Sekunde 1000 komplexe Operationen ausführen kann. Für ein bestimmtes Problem seien fünf verschieden effiziente Algorithmen gegeben. Die Zeitkomplexität ist jeweils

$$\begin{aligned} T_1(n) &= n \\ T_2(n) &= n \cdot \log_2(n) \\ T_3(n) &= n^2 \\ T_4(n) &= n^3 \\ T_5(n) &= 2^n. \end{aligned}$$

Geben Sie in einer Tabelle an, für welche Werte der Eingabelänge n die Laufzeit T_i von Algorithmus i ($i = 1, \cdots, 5$) eine Sekunde, eine Minute, einen Tag beträgt.
b) Stellen Sie sich weiter vor, daß ein neues Modell des Computers aus Teil a) gebaut wird, das um den Faktor 10 schneller ist, also 10000 komplexe Operationen pro Sekunde bewältigt. Wie verändert sich die Tabelle? Bei welchen Algorithmen ermöglicht die Verzehnfachung der Rechengeschwindigkeit, wesentlich größere Eingabegrößen zu bearbeiten?

Aufgabe 2

Zur Landau'schen O–Notation.
a) Seien $f(n) := 100000000 \cdot n$ und $g(n) := 0.000001 \cdot n^2$.
(Fragen Sie sich zunächst intuitiv: Ist f „größer" als g oder umgekehrt?)
Gilt $f \in O(g)$ oder $g \in O(f)$?
b) Seien $f(n) := n^2$ und $g(n) := 100 \cdot (n^2 - \log(n))$.
(Intuitiv: Wachsen f und g „gleich schnell"?)
Gilt $f \in \Theta(g)$? Gilt $g \in \Theta(f)$?
c) Seien $f(n) := \frac{1}{17} \cdot n^2$ und $g(n) := 2345.7 \cdot n \cdot \log(n)$.
(Intuitiv: Wächst f - bis auf Konstanten - „mindestens so schnell" wie g?)
Gilt $f \in \Omega(g)$?

Aufgabe 3

Will man die Effizienz eines Algorithmus quantifizieren, ist die Wahl dessen, was man als Kosten bezeichnen will, durchaus willkürlich. In der Praxis wird man natürlich Größen

wählen, die sinnvoll wiedergeben, was man intuitiv als Komplexität versteht. Typische Größen für die Kosten sind etwa Laufzeit oder Speicherplatzbedarf. Auch die Eingabegröße kann verschieden festgelegt werden, wie diese Aufgabe illustrieren soll. Wenn man die Komplexität eines Algorithmus diskutiert, muß man also annehmen, daß aus dem Kontext klar wird, wie Kosten und Eingabegröße festgelegt sind, oder dies explizit klarstellen.
Bei arithmetischen Aufgabenstellungen, beispielsweise, kann man entweder die Anzahl Bits (oder allgemeiner Stellen) der eingegebenen Zahlen als Eingabegröße betrachten, oder den Wert der Zahlen selbst. Betrachten Sie $\mathcal{A}_1$, den in der Grundschule gelehrten Algorithmus zur Multiplikation zweier Zahlen a und b, d.h. die Zahl a mit der ersten Ziffer von b multiplizieren, dann mit der zweiten Ziffer, usw. (nur im binären System). Alternativ könnte man auch einen RAM–Algorithmus $\mathcal{A}_2$ verwenden, der a b–mal zu sich selbst addiert.

Geben Sie die logarithmische Laufzeitkomplexität (in der O–Notation) von $\mathcal{A}_1$ und $\mathcal{A}_1$ an für den Fall,

a) daß man logarithmische Eingabegröße wählt, d.h. die Eingabe ist in binärer Darstellung gegeben;

b) daß man als Eingabegröße den Wert der Zahlen selbst wählt ($\max\{a, b\}$), d.h. die Eingabe ist in unärer Darstellung (durch Striche) gegeben.

Für Interessierte: Da man hier immer genau zwei Zahlen multipliziert, macht es natürlich keinen Sinn, die Anzahl der zu multipizierenden Zahlen als Eingabegröße zu verwenden – jede denkbare Komplexitätsfunktion wäre für alle Eingaben konstant. Multipliziert man aber m Zahlen, etwa durch iteriertes Multiplizieren mit $\mathcal{A}_i$, so wird man sinnvollerweise m in der Definition der Eingabegröße berücksichtigen. Wie sieht eine brauchbare Definition aus, gibt es ähnlich wie oben auch hier zwei sinnvolle Möglichkeiten?