Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Jeremias Weihmann

Sommersemester 2014
Übungsblatt 6
26. Mai 2014

# Grundlagen: Algorithmen und Datenstrukturen

*Abgabetermin: Jeweilige Tutorübung in der Woche vom 2. bis 6. Juni*

## Tutoraufgabe 1

In dieser Aufgabe betrachten wir erneut die Laufzeitanalyse von MergeSort. In der Vorlesung wurde die Laufzeit rekursiv formuliert und das das Mastertheorem verwendet, um zu zeigen, dass die Laufzeit von MergeSort in $\mathcal{O}(n \log n)$ liegt.

Die rekursive Formulierung der Laufzeit war wie folgt:

$$\begin{aligned} T(n) &= T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + \Theta(n) \\ T(1) &= \Theta(1) \end{aligned} \tag{1}$$

Zeigen Sie ohne Verwendung des Master-Theorems, dass die Worst-Case-Laufzeit von MergeSort angewandt auf Zahlenfolgen, deren Länge Zweierpotenzen sind, in $\mathcal{O}(n \log n)$ liegt.

(a) Verwenden Sie dazu die Methode des iterativen Einsetzens.

(b) Verwenden Sie dazu (starke) vollständige Induktion.

## Tutoraufgabe 2

Wir betrachten eine Funktion $T(n)$, die für zwei Konstanten $a, c > 0$ die folgenden Eigenschaften besitzt:

$$\begin{aligned} T(n) &\leq T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn, \text{ falls } n > 1 \\ T(1) &= a \end{aligned}$$

In der ersten Tutoraufgabe wurde gezeigt, dass $T(n) \in \mathcal{O}(n \log n)$ für alle Zweierpotenzen $n$ gilt. Zeigen Sie, dass dies auch dann gilt, wenn alle natürlichen Zahlen, und nicht nur Zweierpotenzen, als Eingabe erlaubt sind. Was folgt daraus für die Worst-Case-Laufzeit von MergeSort?

## Zusatzaufgabe 1

In der Vorlesung lernen wir, Algorithmen aus theoretischer Sicht zu analysieren. Neben dieser Art der Algorithmen-Analyse gibt es die experimentelle Analyse. Hierbei testet man für Eingaben das Laufzeitverhalten eines Algorithmus.

In der Abbildung 1 sind für drei Algorithmen die gemessenen oberen Schranken für die Laufzeit $t$ in Abhängigkeit der Eingabegröße $n$ einer solchen experimentellen Analyse angegeben. Ordnen Sie die Algorithmen nach ihrer Laufzeit.

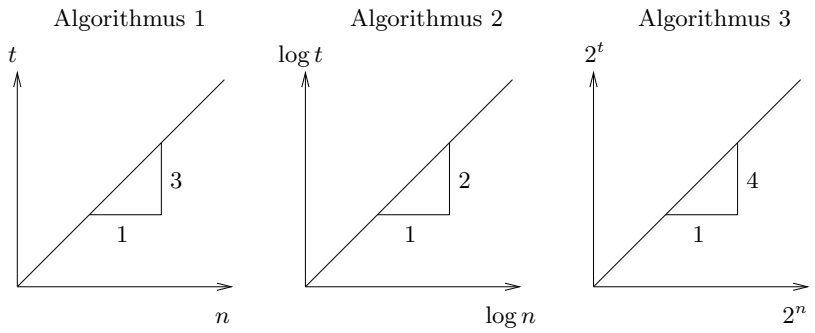


Abbildung 1: Graphen zu den Algorithmen in Aufgabe 1

## Hausaufgabe 1

Wir betrachten die wie folgt definierte Funktion:

$$T(n) \leq \begin{cases} T\left(\lfloor \frac{n}{2} \rfloor\right) + T\left(\lceil \frac{n}{2} \rceil\right) + 2n, & \text{falls } n > 1 \\ 1, & \text{falls } n = 1 \end{cases} \tag{2}$$

In der ersten Tutoraufgabe wurde mit (starker) vollständiger Induktion gezeigt, dass

$$T(n) \leq 2n \operatorname{ld}(2n) \tag{3}$$

für alle Zweierpotenzen $n$ ist. Daraus folgt natürlich automatisch, dass

$$T(n) \leq 2n \operatorname{ld}(2n) + 1/n \tag{4}$$

für alle Zweierpotenzen $n$ gilt.

Zeigen Sie am Beispiel der Funktion $T$ und den genannten oberen Schranken, dass es bei der Beweismethode über vollständige Induktion (analog wie in der ersten Tutoraufgabe durchgeführt) zu dem Phänomen kommen kann, dass der unmittelbare Beweis einer schwachen Schranke (4) fehlschlägt, während der unmittelbare Beweis einer starken Schranke (3) gelingt.

## Hausaufgabe 2

Zeigen Sie mit (starker) vollständiger Induktion die folgenden Gesetze:

(a) Für eine Funktion $T$ definiert durch

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + c, & \text{falls } n > 1 \\ a, & \text{falls } n = 1 \end{cases}$$

wobei $c, a$ positive Konstanten sind, gilt $T(n) \leq dn - \max\{c, a\}$ für eine geeignet gewählte Konstante $d$.

(b) Folgendes Gesetz ist eine Verallgemeinerung von (a). Sei $f : \mathbb{N} \to \mathbb{N}$ eine Funktion mit $1 \leq f(n) < n$. Für eine Funktion $T$ definiert durch

$$T(n) = \begin{cases} T(f(n)) + T(n - f(n)) + c, & \text{falls } n > 1 \\ a, & \text{falls } n = 1 \end{cases}$$

wobei $c, a$ positive Konstanten sind, gilt $T(n) \leq dn - \max\{c, a\}$ für eine geeignet gewählte Konstante $d$.

*Anmerkung:* Diese Gesetze implizieren im Wesentlichen, dass ein Divide-and-Conquer-Algorithmus, der eine Lösung in **konstanter Zeit** aus zwei Lösungen der beiden Teilprobleme zusammensetzt, lineare Laufzeit hat. Hierbei ist nur gefordert, dass die Eingabegröße der beiden Teilprobleme zusammen so groß (bzw. maximal so groß) wie die ursprüngliche Eingabe ist. Dies ermöglicht interessante Linearzeitalgorithmen, bei denen die Aufspaltung einer Eingabe der Länge $n$ in zwei Eingaben der Längen $n-1$ und 1 erfolgt. Oft können solche Algorithmen dann nichtrekursiv implementiert werden, wobei das Wissen ausgenutzt wird, dass bei einer rekursiven Implementation einer der beiden rekursiven Aufrufe stets der Basisfall ist.

## Hausaufgabe 3

Implementieren Sie in der Klasse `UIsmsArray` den MergeSort-Algorithmus, in der Funktion `sort`, der die Elemente in dem Feld `A` sortiert.

Verwenden Sie für Ihre Implementierung die auf der Übungswebseite bereitgestellten Klassen und verändern Sie für Ihre Implementierung *ausschließlich* die Klasse `UIsmsArray`.

Achten Sie bei der Abgabe Ihrer Aufgabe darauf, dass Ihre Klasse `UIsmsArray` heißt und auf den Rechnern der Linuxhalle (`lxhalle.informatik.tu-muenchen.de`) mit der bereitgestellten Datei `main_m` kompiliert werden kann. Anderenfalls kann eine Korrektur nicht garantiert werden. Achten Sie darauf, dass Ihr Quelltext ausreichend kommentiert ist.

Schicken Sie die Lösung per Email mit dem Betreff `[GAD] Gruppe <Gruppennummer>` an Ihren Tutor.