Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Tobias Lieber

Sommersemester 2011
Übungsblatt 5
8. Juni 2011

Grundlagen: Algorithmen und Datenstrukturen

Abgabetermin: In der jeweiligen Tutorübung

Hausaufgabe 1

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 Rayhalle (`rayhalle.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.

Hausaufgabe 2

Geben Sie ein Verfahren an, um 5 Elemente mit 7 Vergleichen zu sortieren.

Aufgabe 1

Veranschaulichen Sie das Vorgehen des RadixSort-Algorithmus anhand der folgenden Wörter:

```
alt, hort, kalt, bar, sport, spalt, ort, stall, spart, speer, bart
```

Benutzen Sie die lexikographische Sortierung (Dabei sei das Leerzeichen ␣ kleiner als jeder andere Buchstabe). Es reicht aus, wenn sie als Alphabet die Buchstaben ␣, a, b, e, h, k, l, o, p, r, s, t verwenden und daher die Schlüssel 0,...,11 verwenden.

Aufgabe 2

Wir haben uns in der Vorlesung beim Beweis, dass die Laufzeit von MergeSort in $\mathcal{O}(n \log n)$ liegt, auf das Mastertheorem berufen.
Zeigen Sie ohne Benutzung des Mastertheorems, dass die geschlossende Darstellung der Rekursionsformel

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

in $\mathcal{O}(n \log n)$ liegt. Sie dürfen vereinfachend annehmen, dass n eine Zweierpotenz ist.

Aufgabe 3 (9 Punkte)

Wir wollen in dieser Aufgabe den `QuickSort` Algorithmus näher analysieren.

a) Geben Sie eine Familie von Eingabearrays an, so dass der `QuickSort` Algorithmus für jedes n Laufzeit $\Omega(n^2)$ benötigt.

b) Zeigen Sie: Wird in dem Algorithmus als Pivot-Element das k-größte Element ausgewählt (wobei $k \in \mathbb{N}$ eine beliebige aber feste Konstante ist), so liegt die Worst-Case Laufzeit des modifizierten Algorithmus immer noch bei $\Omega(n^2)$.

   Ist $|A| < k$, so wird das Pivot-Element wie im Algorithmus der Vorlesung gewählt.

c) Beweisen Sie: Wählen wir als Pivot-Element das $\lfloor \frac{9}{10} n \rfloor$-größte Element aus `A` ($|A| = n$), so ist die Worst-Case Laufzeit $\mathcal{O}(n \log n)$ (wenn die Selektion des Pivot Elements in $\mathcal{O}(n)$ realisiert ist).