Übungsaufgaben Algorithmen und Datenstrukturen I
Theory Übungsblatt 1

Marco Möller <Marco.Moeller@gmxpro.de>
Matrikel Nummer: 23220320

20. Dezember 2004

1 Laufzeitanalyse eines veränderten Mergesort

Um eine einfachere Formel zu erlangen gehe ich von der Vereinfachung aus, das $n = 4^k$ ($k \in \mathbb{N}$) (bzw. $k = \log_4 n$) die Anzahl der zu sortierenden Elemente ist. Die Laufzeitfunktion $T(n)$ lässt sich nun wie folgt (rekursiv) definieren.

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ 4T\left(\frac{n}{4}\right) + \Theta(n) & n > 1 \end{cases}$$

Diese Formel kommt zustande, indem man sich überlegt, das die Zeit die für das Sortieren eines Datensatzes mit 1nem Element genau $\Theta(1)$ dauert. Um einen Datensatz mit n Elementen zu sortieren, muss man insgesamt 4 Datensätze mit je $\frac{n}{4}$ Elementen mischen was mit $\Theta(n)$ abgeschätzt wird (jedes der n Elemente muss ca. 4 mal angesehen werden, und 1mal kopiert, also cn). Diese Datensätze müssen allerdings auch ersteinmal erstellt werden, daher $4T\left(\frac{n}{4}\right)$ ($\frac{n}{4}$ ist die Länge der Unterdatensätze).

Nun muss man versuchen, die Rekursion aus der Formel "herauszuschummeln":

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{4}\right) + cn \\ &= 4\left[4T\left(\frac{\frac{n}{4}}{4}\right) + c\frac{n}{4}\right] + cn \\ &= 4^2 T\left(\frac{n}{4^2}\right) + 2cn \\ &= 4^3 T\left(\frac{n}{4^3}\right) + 3cn \\ &= \dots \\ &= 4^k T\left(\frac{n}{4^k}\right) + kcn \\ &= 4^k T(1) + \log_4(n)\, cn \\ &= cn + cn \log_4 n \\ &= cn(1 + \log_4 n) \\ &= O(n \log n) \end{aligned}$$

Da Konstanten (wie c, 1 und 4) keine Rolle für die Abschätzung spielen, werden sie bei der O Notation weggelassen, und wir erhalten für unseren speziellen Mergesort Algorithmus eine Laufzeitkomplexität von $O(n \log n)$ (identisch mit der vom orginal Mergesort Algorithmus).

2 Laufzeitanalyse eines Sortieralgorithmus

Bei diesem Algoritmus gebe ich die Laufzeit mit T_g (g wie Gesamt) an. T_i steht für die Laufzeit eines Insertion Sort Algorithmus. Für $T_g(n)$ gilt:

$$T_g(n) = \sqrt{n}T_i\left(\sqrt{n}\right) + n\sqrt{n}c$$

Diese Formel kommt wie folgt zustande:

- Jedes der $\sqrt{n}$ Teilfelder wird mit Insertion Sort Sortiert. Da jedes Feld die Länge $\sqrt{n}$ hat, benötigt der Algorithmus für ein einzelnes Feld $T_i(\sqrt{n})$ und für alle zusammen $\sqrt{n}$ mal solange, also $\sqrt{n}T_i(\sqrt{n})$
- Diese Felder müssen anschließend noch gemischt werden. Hierbei muss sich der Algoritmus für jeden der n Werte in $\sqrt{n}$ Feldern den ersten Wert ansehen, um das Minimum finden zu können. Das bedeutet, das dies $n\sqrt{n}c$ Zeit in Anspruch nimmt.

Die Laufzeit von Insertion Sort ist bekannt und kann mit $T(n) = cn^2$ abgeschätzt werden. Dadurch ergibt sich:

$$\begin{aligned} T_g(n) &= \sqrt{n}c\left(\sqrt{n}\right)^2 + n\sqrt{n}c \\ &= \sqrt{n}cn + n\sqrt{n}c \\ &= 2cn\sqrt{n} \\ &= O\left(n\sqrt{n}\right) \end{aligned}$$

Die Laufzeitkomplexität dieses Algorithmus ist also $O(n\sqrt{n})$ und liegt damit zwischen der von Insertion Sort ($O\left(n^2\right)$) und Merge Sort ($O(n\log n)$).

3 Verifikation einer veränderten Quicksort Partitionierung