1. Aufgabenblatt zur Vorlesung

Höhere Algorithmik — WiSe 2011/12

Wolfgang Mulzer, Claudia Dieckmann, Romain Grunert, Tillmann Miltzow, Lena Schlipf

Abgabe am 28. Oktober 2011 vor der Vorlesung in die Mitarbeiterschachteln

Aufgabe 1 O-Notation *10 Punkte*

Ordnen Sie die folgenden Funktionen derart, dass $g_i \in \Omega(g_{i+1})$ gilt. Geben Sie auch an, wenn sogar $g_i = \Theta(g_{i+1})$ ist. Begründen Sie jeweils kurz, warum dies so ist ($\log n := \log_2 n$).

$$\begin{array}{lllll} (\sqrt{2})^{\log n}, & n^2, & (\lceil \log n \rceil)!, & \log^2 n, & 2^{(2^n)}, \\ n^{\frac{1}{\log n}}, & n^{\log\log n}, & \ln n, & 2^n, & 4^{\log n}. \end{array}$$

Aufgabe 2 Sammelbilder *10 Punkte*

Die Europameisterschaft steht kurz bevor. Daher gibt es in jeder Bircher-Müslipackung einen Aufkleber mit dem Bild eines Fußballspielers. Wir nehmen an, dass es insgesamt n verschiedene Sammelbilder gibt und dass jede neue Müslipackung ein zufällig gleichverteiltes Bild enthält. Uns interessiert, wie viele Müslipackungen wir kaufen müssen, bis unsere Sammlung vollständig ist.

Sei also X die Zufallsvariable, welche die Anzahl der benötigten Müslipackungen darstellt. Wir wollen den Erwartungswert $\mathrm{E}[X]$ berechnen.

(a) Wir unterteilen den Sammelprozess in *Runden*. Dabei dauert jede Runde so lange, bis wir ein Sammelbild erhalten, welches wir vorher noch nicht besessen haben. Am Ende der ersten Runde haben wir also ein Sammelbild, nach der zweiten Runde sind es zwei verschiedene Sammelbilder, usw. Die *Länge* einer Runde ist die Anzahl der Packungen, die wir im Laufe der Runde kaufen müssen.

Sei X_i die Zufallsvariable, welche die Länge der i-ten Runde darstellt. Zeigen Sie, dass $\mathrm{E}[X] = \sum_{i=1}^{n} \mathrm{E}[X_i]$ ist.

(b) Berechnen Sie $\mathrm{E}[X_i]$.

Hinweis: Verwenden Sie die geometrische Verteilung.

(c) Zeigen Sie, dass $\mathrm{E}[X] = O(n \log n)$ ist.

Hinweis: $\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$.

Bitte wenden

Aufgabe 3 Varianten von Mergesort *10 Punkte*

(a) Beschreiben Sie in jeweils einem Satz, wie Sortieren durch Auswählen (*Selection Sort*) und Sortieren durch Verschmelzen (*Mergesort*) funktionieren. Bestimmen Sie jeweils exakt die Anzahl der benötigten Vergleiche im schlimmsten Fall. Dabei dürfen Sie annehmen, dass die Eingabegröße n eine Zweierpotenz ist.

(b) Für kleine n ist $n^2/2 - n/2 < n \log n$. Daher braucht Sortieren durch Auswählen in diesen Fällen weniger Vergleiche als Mergesort. Wir betrachten im Folgenden eine Variante von Mergesort namens M-Mergesort. Hierbei ist $M \in \mathbb{N}$ eine natürliche Zahl. M-Mergesort versucht, den Vorteil von Selection Sort für kleine n auszunutzen, indem es Folgen der Länge kleiner M nicht mehr rekursiv, sondern mittels Sortieren durch Auswählen sortiert.

 (i) Bestimmen Sie die Anzahl der von M-Mergesort benötigten Vergleiche exakt. Sie dürfen wieder annehmen, dass n und M Zweierpotenzen sind.

 (ii) Für welche Werte von M ist M-Mergesort besser als Mergesort? Welches M ist optimal?

(c) Implementieren Sie Mergesort und M-Mergesort in Java. Vergleichen Sie Ihre beiden Implementierungen bezüglich der Anzahl der Vergleiche sowie der Gesamtlaufzeit für verschiedene Werte von M und n. Können Ihre theoretischen Ergebnisse in der Praxis bestätigt werden? Was ist die Konstante (z.B. in Mikrosekunden) vor dem $n \log n$-Term bei der wirklichen Laufzeit?