Universität Konstanz
Fachbereich Informatik & Informationswissenschaft
Prof. Dr. Ulrik Brandes / Dr. Andreas Karrenbauer / Dr. Jürgen Lerner / Martin Mader

Algorithmen und Datenstrukturen
WS 2010/2011

4. Übungsblatt

Ausgabe: 10. November 2010 **Abgabe:** 17. November 2010, 10 Uhr
Die Bearbeitung in Zweiergruppen ist ausdrücklich erwünscht.

Aufgabe 1: CountingSort — 3 Punkte

(a) Seien $k = 4$ und ein Array M der Länge 5 von Paaren der Form $(key, value)$ gegeben durch

$$M = [(1, v_1), (4, v_2), (1, v_3), (3, v_4), (3, v_5)] \ .$$

Die Paare werden beim Sortieren ausschliesslich durch ihre *key*s eingeordnet. Ein Ausdruck der Form $C[M[i]]$ in Algorithmus 14 muss durch $C[M[i].\texttt{key}()]$ ersetzt werden (also C an der Stelle die durch den key von $M[i]$ gegeben ist).

Sortieren Sie M mit CountingSort (Algorithmus 14) und dokumentieren Sie Ihre Schritte ausführlich.

(b) Die vierte for-Schleife in Algorithmus 14 wird geändert zu

for $i = 1, \ldots, n$ **do**

(Inhalt der Schleife bleibt gleich). Führen Sie den modifizierten Algorithmus auf der Eingabe M (wie oben) aus. Dokumentieren Sie Ihre Schritte ausführlich (hier natürlich nur noch die beim Ausführen der vierten Schleife).

Arbeitet der Algorithmus im Allgemeinen immer noch richtig? Welche Eigenschaft geht verloren?

Aufgabe 2: BucketSort — 2 Punkte

Führen Sie BucketSort (Algorithmus 12) auf der Eingabe

$$M = [0.68, 0.72, 0.12, 0.39, 0.65, 0.17, 0.13, 0.84, 0.41, 0.96]$$

aus und dokumentieren Sie Ihre Schritte ausführlich.

Aufgabe 3: RadixExchangeSort — 3 Punkte

Führen Sie RadixExchangeSort (Algorithmus 16) auf der Eingabe

$$M = [0.01001, 0.11001, 0.01001, 0.10101, 0.01100, 0.01000]$$

aus und dokumentieren Sie Ihre Schritte ausführlich.

Aufgabe 4: Sortieralgorithmen – Vergleich — 2 Punkte

Nehmen Sie an, Sie müssen n ganze Zahlen aus der Menge $\{1, \ldots, m\}$ sortieren. Geben Sie die asymptotischen Laufzeiten (Θ-Notation, in Abhängigkeit von n und m) für die folgenden Strategien an.

(a) Sortieren mit MergeSort.

(b) Sortieren mit CountingSort.

(c) Umwandeln in Binärzahlen und dann mit RadixExchangeSort sortieren.

Aufgabe 5: Sortieralgorithmen – empirische Studie — 10 Punkte

(a) Implementieren Sie ein generisches SelectionSort-Verfahren unter Benutzung des Heaps aus der letzten Übung. Implementieren Sie dazu das Interface `material.u02.ISort` in einer Klasse `u04.name.SelectionHeapSort`. (2 Punkte)

(b) Realisieren Sie BucketSort in einer Klasse `u04.name.BucketSort`, die `material.u02.ISort<Double>` implementiert. Sie können davon ausgehen, dass der gültige Bereich für die Eingabewerte im Intervall $(0, 1]$ liegt. Zur Umsetzung der Listen in BucketSort können Sie die entsprechenden Datentypen in `java.util` verwenden, etwa `LinkedList`, und zur Sortierung der Listen entsprechende Methoden aus `java.util.Collections`. (2 Punkte)

(c) Erzeugen Sie `Double`-Arrays verschiedener Größen, Verteilungen und Anordnungen im Wertebereich $(0, 1]$, z.B. mit Hilfe der Klasse `material.u04.DoubleArrayGenerator`. Sortieren Sie diese mit den drei QuickSort-Varianten aus Übung 2, und mit den zwei Verfahren

aus dieser Aufgabe. Messen Sie Laufzeiten und erstellen Sie aussagekräftige Diagramme; achten Sie insbesondere auch auf Titel, Achsenbeschriftung und Legende der Diagramme. Fassen Sie die Diagramme in einer Datei `u04_name_studie.pdf` zusammen, erläutern Sie Ihre Vorgehensweise und interpretieren Sie die Ergebnisse kurz (max. eine Seite Text). (6 Punkte)

Hinweise:

- Sie dürfen die auf der WWW-Seite zur Übung vorgestellten Lösungsvorschläge verwenden.
- Mitteln Sie Ihre Messwerte über mehrere Wiederholungen, so dass Seiteneffekte reduziert werden.