Ludwig-Maximilians-Universität München
Institut für Informatik
PD Dr. Matthias Schubert
Dr. Andreas Züfle, Dr. Erich Schubert

München, 03.06.2014

Algorithmen und Datenstrukturen

SS 2014

Übungsblatt 8: Sortieren

Besprechung: 20.6.–26.6.

Abgabe aller mit **Punkten** versehenen Aufgaben bis 18.6.

Zur Wiederholung:

Eine Sortierung von Objekten basiert auf der Definition einer vollständigen Ordnung "$<$" auf dem Wertebereich der Objekte. Eine vollständige Ordnung ist u.a. dadurch charakterisiert, dass für zwei beliebige Objekte a und b genau eine der folgenden Beziehungen gilt:

$$a < b \qquad \text{oder} \qquad b < a \qquad \text{oder} \qquad a = b$$

Diese Eigenschaft ist auch bekannt als "*Gesetz der Trichotomie*" (obwohl "*Tritomie*" eigentlich der korrektere Name wäre).

Üblicherweise nimmt man auch an, dass für drei beliebige Objekte a, b und c gilt: Wenn $a < b$ und $b < c$, dann $a < c$. Diese Eigenschaft ist bekannt als "*Transitivität*". Als Ergebnis eines Sortierverfahrens auf n Schlüsseln K_i, $i = 1, \ldots, n$ erwartet man eine Permutation $p(1), \ldots, p(n)$ der Indizes, so dass die Schlüssel in nicht-absteigender Ordnung angeordnet sind:

$$K_{p(1)} \leq K_{p(2)} \leq \ldots \leq K_{p(n)}$$

Ein Sortierverfahren kann zusätzlich *stabil* sein. Dann wird von der Permutation p verlangt, dass gilt:

$$(K_{p(i)} = K_{p(j)} \wedge i < j) \Rightarrow p(i) < p(j)$$

Aufgabe 8-1 Stabilität von Sortierverfahren — 2 Punkte

Entscheiden Sie, ob folgende Aussagen zutreffen und begründen Sie Ihre Entscheidung:

(a) "Sortieren durch Abzählen ist stabil."

(b) "Sortieren durch direktes Einfügen (Insertion-Sort) ist stabil."

Aufgabe 8-2 Eindeutigkeit von Sortierverfahren — 4 Punkte

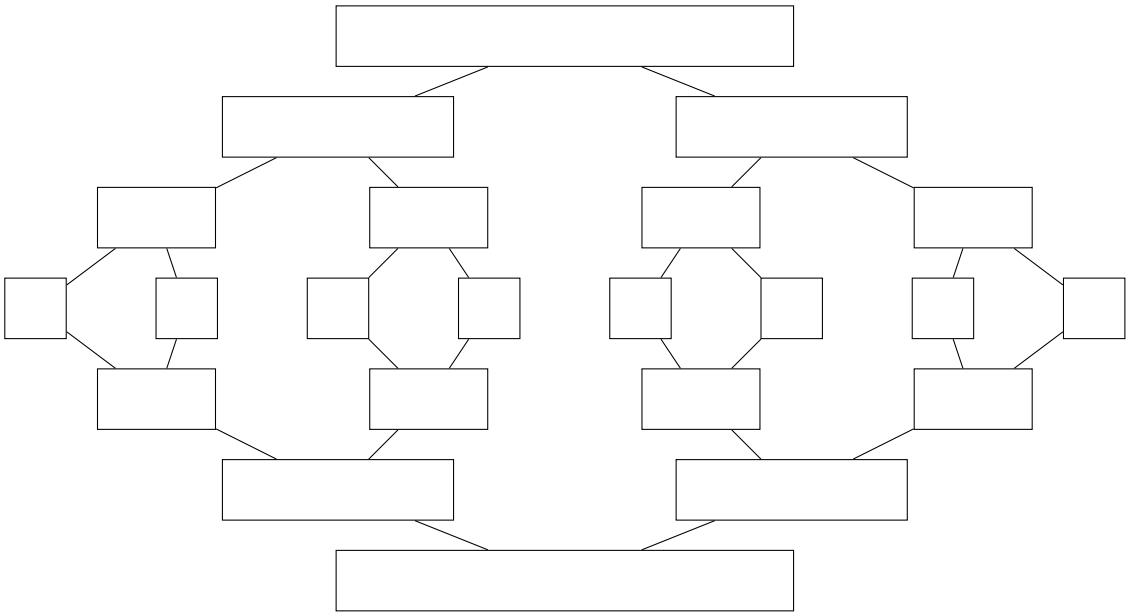
(a) Gegeben sei ein *stabiles* Sortierverfahren, basierend auf einer vollständigen Ordnung, die den Gesetzen der *Trichotomie* und *Transitivität* genügt. Beweisen Sie, dass die Permutation $p(1), p(2), \ldots, p(n)$, die man als Ergebnis des Sortierverfahrens erhält, *eindeutig bestimmt* ist.

(b) Nun sei eine Ordnung "$<$" auf $K_1, \ldots, K_n$ gegeben, die dem Gesetz der *Trichotomie* genügt, aber *nicht transitiv* ist. Begründen Sie, warum es auch möglich ist, die Schlüssel *stabil* zu sortieren, wenn die vorausgesetzte Ordnung *nicht transitiv* ist. Nennen Sie zwei Beispiele für Sortierverfahren aus der Vorlesung, die im nicht-transitiven Fall eine stabile Sortierung ermöglichen.

Aufgabe 8-3 Merge-Sort **2 Punkte**

Geben Sie die Zwischenergebnisse des Merge-Sort-Algorithmus in dem untenstehenden Graphen an, der die in der Vorlesung verwendete Struktur besitzt. Die obere Hälfte des Graphen stellt die verschiedenen Rekursionsstufen des Divide-Schritts dar, der obere Knoten des Graphen enthält das unsortierte Array. In die untere Hälfte des Graphen soll der Merge-Schritt eingetragen werden.

Zu sortieren ist die folgende Liste von Zahlen: 14, 84, 75, 25, 49, 45, 64, 78



Aufgabe 8-4 Quick-Sort **4 Punkte**

Sortieren Sie das folgende Array mit dem Quick-Sort-Verfahren *exakt wie es in der Vorlesung besprochen wurde*: Wählen Sie stets das erste Element als Pivot-Element. Verwenden Sie eine Tabelle mit der folgenden Struktur:

Anfang:	61	91	76	87	32	1	2	22	69	9	55	72
Vor Pivot:												
Nach Pivot:												
Vor Pivot:												
Nach Pivot:												

⋮

Markieren Sie für jeden rekursiven Durchlauf des Verfahrens das betroffene Intervall *vor und nach* nach dem Verschieben des Pivot-Elements. Es reicht, wenn Sie das Pivot-Element und diejenigen Elemente eintragen, deren Position sich ändert (um die Schreibarbeit zu reduzieren)!

Streichen Sie *nie* richtige Elemente durch: Durchgestrichenes können wir nicht bewerten, die Lösung muss für uns eindeutig erkennbar sein! Eigene Markierungen (Zeiger, Vertauschungspfeile etc.) können Sie zwischen den Zeilen eintragen. Diese Tabellenform ist darauf ausgelegt, dass Sie nichts durchstreichen müssen.