Institut für Informatik
Ludwig-Maximilians-Universität München

Prof. Dr. Hans-Peter Kriegel
Thomas Bernecker, Tobias Emrich

SS 2008
02.06.2008
Übungsblatt 8

Übungen zur Vorlesung

Effiziente Algorithmen

Zur Wiederholung:

Eine Sortierung von Objekten basiert auf der Definition einer vollständigen Ordnung '<' auf dem Wertebereich der Objekte. Eine vollständige Ordnung ist unter anderem dadurch charaterisiert, daß für zwei beliebige Objekte a und b genau eine der folgenden Beziehungen gilt:

1.) $a < b$
2.) $b < a$
3.) $a = b$

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, $K_{i,\ i = 1,\dots,n}$ erwartet man eine Permutation $p(1)...p(n)$ der Indizes, so daß die Schlüssel in nicht-absteigender Ordnung angeordnet sind:

$$K_{p(1)} \le K_{p(2)} \le \dots \le K_{p(n)}$$

Ein Sortierverfahren kann zusätzlich *stabil* sein. Dann ist von der Permutation p verlangt, daß:

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

Aufgabe 8.1: *Stabilität von Sortierverfahren* **(2+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* **(3+2 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)...p(n)*, die man als Ergebnis des Sortierverfahrens erhält, *eindeutig bestimmt* ist.

b) Nun sei eine Ordung '<' auf $K_1,...,K_n$ 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: *Quick-Sort* **(8+1 Punkte)**

a) Geben Sie eine Auswertung von Quick-Sort mit dem Array *a = {4, 6, 2, 8, 3, 1, 7, 9, 5, 1}* an. Die Auswertung soll für alle Aufrufe der Methode *sort* (siehe Skript) die aktuellen Werte der Variablen *li*, *r*, *i*, *j*, sowie die Werte im Array verfolgen. Fertigen Sie dazu eine Tabelle an, in der eine Zeile die jeweils neuen Werte enthält, nach folgendem Muster:

<table>
<tr><th>Aufruf
sort</th><th>li</th><th>r</th><th>i</th><th>j</th><th>a1</th><th>a2</th><th>a3</th><th>a4</th><th>a5</th><th>a6</th><th>a7</th><th>a8</th><th>a9</th><th>a10</th></tr>
<tr><td>1</td><td>1</td><td>10</td><td>2</td><td>10</td><td>4</td><td>6</td><td>2</td><td>8</td><td>3</td><td>1</td><td>7</td><td>9</td><td>5</td><td>1</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>6</td></tr>
<tr><td colspan="15">...</td></tr>
<tr><td>1.1</td><td>1</td><td colspan="13">...</td></tr>
<tr><td colspan="15">...</td></tr>
</table>

b) Ist Quick-Sort stabil? Begründen Sie Ihre Antwort!