Übungsblatt 2

Abgabe: 27.04.2009

Aufgabe 1 Computer Ticket Service

Die Firma ODDTIM verkauft Tickets für Veranstaltungen aller Art. Auf der Startseite ihres Webshops soll eine stets aktuelle Rangliste mit Künstlern zu finden sein, sortiert nach den meistverkauften Tickets. Dazu wird ein Array von Künstlern mit ihren aktuell verkauften Tickets sortiert. Üblicherweise ist der vorhandene Sortieralgorithmus schnell genug, doch bei Vorverkaufstarts sehr beliebter Künstler (z.B. Peter Fux oder EySeeDeeSee) kann es zu einer rasanten Änderung der verkauften Tickets pro Künstler kommen.

Aufgabe 1.1 Schnell, schneller, ...

Der Abteilungsleiter schlägt daher den folgenden Algorithmus zur Sortierung vor:

```
static void sortiere(Kuenstler[] kuenstler) {
  for (int index = 0; index < kuenstler.length-1; index++) {
    for (int j = index; j>=0; j--) {
      if (soldMoreTickets(kuenstler[j+1], kuenstler[j])) {
        Kuenstler tmp = kuenstler[j];
        kuenstler[j] = kuenstler[j+1];
        kuenstler[j+1] = tmp;
      }
    }
  }
}
```

Hinweis: Die Funktion `soldMoreTickets` liefert genau dann *true*, wenn der erste Künstler **mehr** Tickets verkauft hat als der zweite.

Die neue Mitarbeiterin, die soeben ihr Studium der Informatik an der Universität Bremen erfolgreich abgeschlossen hat, hat einen alternativen Vorschlag:

```
private static void sortiereAnders(Kuenstler[] kuenstler) {
  int left = 0;
  int right = kuenstler.length;
  boolean tausch = false;

  while (left < right) {
    for (int i=left; i < right-1; i++) {
      if (soldMoreTickets(kuenstler[i+1], kuenstler[i])) {
        Kuenstler tmp = kuenstler[i];
        kuenstler[i] = kuenstler[i+1];
        kuenstler[i+1] = tmp;
        tausch = true;
      }
    }
    if (!tausch) {
      return;
    }
    right--;
    tausch = false;
```

```
    for (int i=right-1; i >= left; i--) {
      if (soldMoreTickets(kuenstler[i+1], kuenstler[i])) {
        Kuenstler tmp = kuenstler[i];
        kuenstler[i] = kuenstler[i+1];
        kuenstler[i+1] = tmp;
        tausch = true;
      }
    }
    if (!tausch) {
      return;
    }
    left++;
  }
}
```

Wenig überraschend wird der erste Algorithmus eingesetzt. Ist der zweite Algorithmus wirklich langsamer? Bestimmt den Aufwand der beiden Algorithmen jeweils für den schlechtesten und den besten Fall.

Nehmt dabei an, dass $O(soldMoreTickets()) = O(1)$ ist. Was für eine Empfehlung würdet Ihr aufgrund der Aufwandsabschätzung abgeben?

Aufgabe 1.2 ... genauso langsam?

Der Geschäftsführer ist von beiden Algorithmen nicht wirklich überzeugt. Deshalb bittet er die ohnehin eingesetzte und teuer bezahlte Unternehmensberatung um eine Empfehlung. Nach einigen Monaten erhält er folgenden Vorschlag:

```
void consulting(Kuenstler[] kuenstler, int left, int right) {
  if (left < right) {
    int index = consult(kuenstler, left, right);
    consulting(kuenstler, left, index);
    consulting(kuenstler, index+1, right);
  }
}

int consult(Kuenstler[] kuenstler, int left, int right) {
  Kuenstler merker = kuenstler[left];
  int i = left-1;
  int j = right+1;
  while (true) {
    do {
      j--;
    } while (soldMoreTickets(merker, kuenstler[j]));
    do {
      i++;
    } while (soldMoreTickets(kuenstler[i], merker));
    if (i < j) {
      Kuenstler tmp = kuenstler[i];
      kuenstler[i] = kuenstler[j];
      kuenstler[j] = tmp;
    } else {
      return j;
    }
  }
}
```

Dieser Algorithmus muss mit

```
consulting(kuenstler, 0, kuenstler.length-1);
```

aufgerufen werden.

Hat sich die Wartezeit gelohnt und die Unternehmensberatung tatsächlich einen schnelleren Algorithmus geliefert? Bestimmt dazu wieder den Aufwand für den schlechtesten Fall.

Hinweise: Da der Algorithmus rekursiv aufgebaut ist, ist zunächst der Aufwand für die Funktion *consult* zu bestimmen. Überlegt Euch dann, wie häufig der Grundalgorithmus ausgeführt wird.

Der schlechteste Fall tritt dann ein, wenn die *while*-Schleifen in *consult* maximal oft ausgeführt werden.