technische universität dortmund

Fakultät für Informatik
Lehrstühle 2, 11

Prof. Dr. C. Sohler, A. Krivošija
H. Blom, N. Kriege, H. Timm, B. Zey
http://tiny.cc/dap2p

Sommersemester 2014

DAP2 Praktikum – Blatt 3

Ausgabe: 22. April — Abgabe: 28. April – 2. Mai

Kurzaufgabe 3.1 **(4 Punkte)**
Lernziel: BubbleSort

Implementieren Sie den BubbleSort-Algorithmus (siehe Hinweis) in einer Methode namens `public static void bubbleSort(int[] array)`, welcher ein Feld aufsteigend sortiert. Ermitteln Sie die Laufzeit Ihrer Methode auf einem Feld, das mit 50000 Elementen in absteigender Reihenfolge initialisiert ist. Das Befüllen des Feldes soll in einer zusätzlichen Methode ausgelagert sein.

Kurzaufgabe 3.2 **(4 Punkte)**
Lernziel: Binäre Suche / Geometrische Suche

In dieser Aufgabe soll die Feldgröße für einen BubbleSort-Aufruf bestimmt werden, so dass ziemlich genau eine vorgegebene Zeit in Sekunden zum Sortieren gebraucht wird. Dies soll das Programm komplett automatisch erreichen, d.h., dass das Programm beim Ausführen alle unten beschriebenen Schritte selbstständig ausführt. Die Felder sind hier der Einfachheit halber alle absteigend zu initialisieren. Gehen Sie dazu wie folgt vor.

- Erweitern Sie den Quellcode von Aufgabe 3.1, indem Sie Ihre `main`-Methode so anpassen, dass ein `float` als Parameter auf der Kommandozeile übergeben werden kann. Dieser entspricht der vorgegebenen Zeit in Sekunden.
- Fangen Sie mit einer Feldgröße von 1000 Elementen an.
- Wenn BubbleSort auf diesem Feld schneller als die gesuchte Zeit ist, wird die Feldgröße verdoppelt. Dies wird so oft wiederholt, bis BubbleSort länger als die gesuchte Zeit braucht.
- Mit den letzten beiden Feldgrößen (eine mit kürzerer und eine mit längerer Laufzeit) wird eine *binäre Suche* gestartet.
- Geben Sie die Feldgrößen und Zeiten der Zwischenschritte aus, damit man den Verlauf verfolgen kann.
- Wenn die Laufzeit bis auf 0,1 Sekunden an der gegebenen Laufzeit ist, wird die Suche abgebrochen und die Feldgröße ausgegeben.

Beachten Sie die Hinweise und Tipps auf den folgenden Seiten.

Hinweise und Tipps

3.1 Messung von Laufzeiten, Laufzeitschwankungen

Um Laufzeiten in Programmen zu messen, kann man die Methode `System.currentTimeMillis()` benutzen. Diese gibt die Systemzeit in Millisekunden zurück, und hat den Rückgabetyp `long` Ein Beispielcode zur Laufzeitmessung sieht dann wie folgt aus:

```
...
long tStart, tEnd, msecs;
...
// Beginn der Messung
tStart = System.currentTimeMillis();

// Hier wird der Code ausgeführt, dessen Laufzeit gemessen werden soll

// Ende der Messung
tEnd = System.currentTimeMillis();

// Die Vergangene Zeit ist die Differenz von tStart und tEnd
msecs = tEnd - tStart;
...
```

Beachten Sie, dass der Garbage-Collector einen Einfluss auf die Laufzeit haben kann. Bei einzelnen Messungen kann dies zu recht großen Laufzeitunterschieden führen. Daher ist es empfehlenswert, jede Messung mehrmals zu wiederholen und den Mittelwert der Laufzeiten zu bestimmen. Dem Garbage-Collector kann auch zwischen den Messungen mittels `System.gc()` empfohlen werden den Speicher aufzuräumen. Vorher müssen natürlich die betreffenden Referenzen auf `null` gesetzt sein, damit der Speicher auch wirklich freigegeben werden kann.

Außerdem kann es aufgrund verschiedener Effekte (wechselnde CPU-Auslastung der Poolrechner, wechselnde CPU-Taktungen bei Laptops und anderen Rechnern) zu Schwankungen in der Laufzeit kommen. Falls dies bei Ihnen auftreten sollte, können Sie auch eine höhere Toleranzgrenze als 0,1 Sekunden wählen.

3.2 BubbleSort-Pseudocode

Procedure BUBBLESORT(Array A)

```
n ← LENGTH[A]
for i ← 1 to n do
    for j ← n downto i + 1 do
        if A[j − 1] > A[j] then
            tmp ← A[j]
            A[j] ← A[j − 1]
            A[j − 1] ← tmp
```