technische universität
dortmund

Fakultät für Informatik
Lehrstühle 11, 13, 14

Jun.-Prof. Dr. Maike Buchin
Łukasz Czajka, N. Kriege, D. Kurz, M. Ludewig

Sommersemester 2019
16. April

DAP2 Praktikum – Blatt 3

Abgabe: 22.–26. April

Kurzaufgabe 3.1: BubbleSort (4 Punkte)

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 (in Sekunden, siehe Hinweis) Ihrer Methode auf einem Feld, das mit 50000 Elementen in absteigender Reihenfolge initialisiert ist. Das Befüllen des Feldes soll in eine zusätzliche Methode ausgelagert werden.

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

In dieser Aufgabe soll die Suchstrategie „Geometrische Suche“ erprobt werden. Gesucht wird die Länge eines Arrays, so dass das Sortieren des Arrays mittels BubbleSort in guter Näherung eine vorgegebene Zeit in Sekunden dauert. Dies soll vollständig automatisiert erfolgen, d.h. das zu entwickelnde Programm führt alle unten beschriebenen Schritte selbstständig aus. Die Felder sind hier der Einfachheit halber alle absteigend zu initialisieren. Gehen Sie 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.
- Beginnen Sie die Suche mit einer initialen Feldgröße von 1000 Elementen.
- Wenn BubbleSort auf diesem Feld schneller als die eingegebene Zeit ist, wird die Feldgröße verdoppelt. Dies wird so oft wiederholt, bis BubbleSort länger als die eingegebene Zeit benötigt.
- Zwischen 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 weniger als 0,1 Sekunden von der eingegebenen Laufzeit abweicht, wird die Suche terminiert und die Feldgröße ausgegeben.

Beachten Sie die Hinweise und Tipps auf den folgenden Seiten.

Hinweise und Tipps

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
```

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 Beispiel 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;
...
```

Zu Aufgabe 3.2: Diverse Effekte (wechselnde CPU-Auslastung der Poolrechner, wechselnde CPU-Taktungen bei Laptops und anderen Rechnern) können zu Schwankungen in der Laufzeit führen. Falls dies bei Ihnen auftreten sollte, können Sie bei Aufgabe 3.2 auch eine höhere Toleranzgrenze als 0,1 Sekunden wählen.

Best Practice (empfehlenswert, aber nicht Gegenstand der Punktwertung): Neben den oben genannten Effekten kann auch der Garbage-Collector einen Einfluss auf die Laufzeit haben. Bei einzelnen Messungen kann dies zu großen Laufzeitunterschieden führen. Daher ist es empfehlenswert, jede Messung mehrmals zu wiederholen und den Mittelwert der Laufzeiten zu bestimmen. Dem Garbage-Collector kann *zwischen* den Messungen mittels `System.gc()` empfohlen werden, den Speicher aufzuräumen. Vorher müssen die betreffenden Referenzen auf `null` gesetzt sein, damit der ehemals referenzierte Speicher freigegeben werden kann.