Programmschritte zählen

Um eine Größenordnung für die Laufzeit von Programmen anzugeben, kann man zählen, wie viele Schritte das Programm durchschnittlich ausführt. Bei einem Programm mit Arrays ist die Anzahl der Schritte abhängig von der Länge des Arrays. Man kann zunächst die Schritte für eine bestimmte Länge des Arrays zählen. Das Ziel ist aber, die Schritte allgemein für eine Liste der Länge n anzugeben.

Programmschritte

Ein Schritt wird im Programm benötigt für

- eine Zuweisung (z.B. `temp = zahlen[i];`)
- eine Rechnung (z.B. `i - 1`)
- einen Vergleich (z.B. `if (zahlen[i] > zahlen[min])` …)

Eine Zeile kann durchaus mehrere Schritte benötigen: `zahlen[i] = zahlen[j+1];` enthält eine Rechnung und eine Zuweisung, also insgesamt zwei Schritte.

Aufgabe

Auf Seite 2 ist der Java-Code für das Verschieben aller Einträge um eine Position nach links sowie für drei Suchverfahren gegeben (nur die Schleifen ohne Methoden-Köpfe).

a) Zähle, wie viele Schritte jedes Verfahren benötigt, für eine Liste mit 5 bzw. 10 Einträgen.
Die Anweisungen, die in den Zeilen mit `for (...)` stehen, kannst du dabei vernachlässigen..

b) Versuche nun, die Zählung zu verallgemeinern für eine Liste mit n Einträgen.

Anzahl der Schritte für verschiedene Suchverfahren

Länge der Liste	Verschieben aller Elemente	Bubble Sort einfach	Bubble Sort optimiert	Selection Sort
5	1 (i = 0) + 4 * 3 (Schleife) + 4 * 2 (Zuweis.) = 1 + 4*5	1 + 4 * 3 + 4 * (4 * 7,5) = 4 * (3 + 4 * 7,5)	4 * 3 + 4 * 8,5 + 3 * 8,5 + 2 * 8,5 + 1* 8,5	4 * 3 (for) + 4 * 4 (Tausch) + 4 * 1,5 + … + 1 * 1,5
10	1 + 9 * 3 + 9 * 2 = 1 + 9 * 5	9 * (3 + 9 * 7,5)	9 * 3 + 9 * 8,5 + 8 * 8,5 + … + 1 * 8,5	9 * 3 (for) + 9 * 3 (Tausch) + 9 * 1,5 + … + 1 * 1,5
n (allgemein)	1 + n * 5	n * (3 + n * 7,5)	$n \cdot 3 + \sum_{i=1}^{n} i \cdot 8{,}5$ $= n \cdot 3 + \frac{n^2 + n}{2} \cdot 8{,}5$	$n \cdot 6 + \sum_{i=1}^{n} i \cdot 3{,}5$ $= n \cdot 6 + \frac{n^2 + n}{2} \cdot 3{,}5$

```
// 1. Alle Einträge eins nach links schieben
for (i = 0; i < zahlen.length - 1; i++) {
    zahlen[i] = zahlen[i + 1];
}
```

2 Schritte

```
// 2. Bubble Sort: einfache Version
for (i = 0; i < zahlen.length - 1; i++) {
    for (j = 0; j < zahlen.length - 1; j++) {
        if (zahlen[j] > zahlen[j+1]) {
            temp      = zahlen[j];
            zahlen[j]   = zahlen[j+1];
            zahlen[j+1] = temp;
        }
    }
}
```

durchschnittlich: 2,5 Schritte

```
// 3. Bubble Sort: optimierte Version
for (i = 0; i < zahlen.length - 1; i++) {
    for (j = 0; j < zahlen.length - i - 1; j++) {
        if (zahlen[j] > zahlen[j+1]) {
            temp      = zahlen[j];
            zahlen[j]   = zahlen[j+1];
            zahlen[j+1] = temp;
        }
    }
}
```

durchschnittlich: 2,5 Schrtte

```
// 4. Selection Sort
for (i = 0; i < zahlen.length - 1; i++) {
    int = i;
    for (j = i + 1; j < zahlen.length; j++) {
        if (zahlen[j] < zahlen[min]) {
            min = j;
        }
    }
    temp      = zahlen[i];
    zahlen[i]   = zahlen[min];
    zahlen[min] = temp;
}
```

durchschnittlich: 1,5 Schritte

3 Schritte