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. Ziel ist aber, die Schritte allgemein für eine Liste der Länge n anzugeben.

Programmschritte

Ein einzelner „Schritt“ (also eine elementare Prozessor-Anweisung) wird benötigt für

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

Eine Zeile benötigt of mehrere Schritte: Die Zeile `liste[i] = liste[j+1];` enthält z.B. eine Rechnung und eine Zuweisung, also insgesamt zwei Schritte[1].

Aufgabe

Auf Seite 2 ist der Quellcode 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 Trage deine Ergebnisse in die Tabelle ein.

b) Verallgemeinere die Zählung nun für eine Liste mit n Einträgen.

Anzahl Elemente	Verschieben	Bubble Sort einfach	Bubble Sort optimiert	Selection Sort
5				
10				
n (allgemein)				

1 Die Indizierung liste[i] benötigt tatsächlich auch einen Prozessorbefehl (eine Addition), aber das vernachlässigen wir an dieser Stelle.

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


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


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


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

Hinweis: wenn im Quellcode eine Verzweigung steht, z.B. if (liste[j] < liste[min]) { min = j }, dann findet der Vergleich (liste[j] < liste[min]) in jedem Fall statt.
Die Anweisung (min = j) findet nicht jedesmal statt, sondern (geschätzt) nur jedes zweite Mal.