

Aufgabe 6

Gegeben sei ein einfacher Sortieralgorithmus, der ein gegebenes Feld A dadurch sortiert, dass er das *Minimum* m von A findet, dann das Minimum von A ohne das Element m usw.

- (a) Geben Sie den Algorithmus in Java an. Implementieren Sie den Algorithmus *in situ*, d. h. so, dass er außer dem Eingabefeld nur konstanten Extra-speicher benötigt. Es steht eine Testklasse zur Verfügung.

```
3 public class SortierungDurchAuswaehlen {
4     static void vertausche(int[] zahlen, int index1, int index2) {
5         int tmp = zahlen[index1];
6         zahlen[index1] = zahlen[index2];
7         zahlen[index2] = tmp;
8     }
9
10    static void sortiereDurchAuswählen(int[] zahlen) {
11        // Am Anfang ist die Markierung das erste Element im Zahlen-Array.
12        int markierung = 0;
13        while (markierung < zahlen.length) {
14            // Bestimme das kleinste Element.
15            // 'min' ist der Index des kleinsten Elements.
16            // Am Anfang auf das letzte Element setzen.
17            int min = zahlen.length - 1;
18            // Wir müssen nicht bis letzten Index gehen, da wir 'min' auf
19            // ↳ das letzte Element
20            // setzen.
21            for (int i = markierung; i < zahlen.length - 1; i++) {
22                if (zahlen[i] < zahlen[min]) {
23                    min = i;
24                }
25            }
26            // Tausche zahlen[markierung] mit gefundenem Element.
27            vertausche(zahlen, markierung, min);
28            // Die Markierung um eins nach hinten verlegen.
29            markierung++;
30        }
31    }
32
33    public static void main(String[] args) {
34        int[] zahlen = { 5, 2, 7, 1, 6, 3, 4 };
35        sortiereDurchAuswählen(zahlen);
36        for (int i = 0; i < zahlen.length; i++) {
37            System.out.print(zahlen[i] + " ");
38        }
39    }
40 }
```

Code-Beispiel auf Github ansehen:
[src/main/java/org/bachelorjava/examen_66115/jahr_2014/herbst/SortierungDurchAuswaehlen.java](https://github.com/bachelorjava/examen_66115/jahr_2014/herbst/SortierungDurchAuswaehlen.java)

- (b) Analysieren Sie die Laufzeit Ihres Algorithmus.

Beim ersten Durchlauf des *Selectionsort*-Algorithmus muss $n - 1$ mal das Minimum durch Vergleich ermittelt werden, beim zweiten Mal $n - 2$. Mit Hilfe der *Gaußschen Summenformel* kann die Komplexität

gerechnet werden:

$$(n-1) + (n-2) + \dots + 3 + 2 + 1 = \frac{(n-1) \cdot n}{2} = \frac{n^2}{2} - \frac{n}{2} \approx \frac{n^2}{2} \approx n^2$$

Da es bei der Berechnung der Komplexität um die Berechnung der asymptotischen oberen Grenze geht, können Konstanten und die Addition, Subtraktion, Multiplikation und Division mit Konstanten z. B. $\frac{n^2}{2}$ vernachlässigt werden.

Der *Selectionsort*-Algorithmus hat deshalb die Komplexität $\mathcal{O}(n^2)$, er ist von der Ordnung $\mathcal{O}(n^2)$.

- (c) Geben Sie eine Datenstruktur an, mit der Sie Ihren Algorithmus beschleunigen können.

Der *Selectionsort*-Algorithmus kann mit einer Min- (in diesem Fall) bzw. einer Max-Heap beschleunigt werden. Mit Hilfe dieser Datenstruktur kann sehr schnell das Minimum gefunden werden. So kann auf die vielen Vergleiche verzichtet werden. Die Komplexität ist dann $\mathcal{O}(n \log n)$.