Sortierverfahren

1. SelectionSort

- SelectionSort basiert auf der Minimumsuche
- Es wird das Minimum der Daten gesucht und dieses ans Ende der bereits sortierten Daten gespeichert
- Minimumsuche braucht n Schritte/Vergleiche
- Um alles zu sortieren, muss man die Minimumsuche n mal ausführen
- Insgesamt als immer ein Auswand von n^2

2. InsertionSort

- Ist vergleichbar mit Karten sortieren beim Kartenspiel
- die jeweils nächste unsortierte Karte wird genommen und schrittweise vom rechten Ende nach links an die richtige Position verschoben
- Im besten Fall ist die Liste schon sortiert. Dann müssen keine Zahlen/Einträge vertauscht werden. Für jeden Eintrag muss deshalb nur ein Vergleich gemacht werden, bei n Einträgen. Der Aufwand ist also n
- Im schlechtesten Fall ist die Liste umgekehrt sortiert. Dann muss jeder Eintrag komplett an den Anfang verschoben werden. Für jeden Eintrag müssen im Durchschnitt $\frac{n}{2}$ Vergleiche gemacht werden, bei n Einträgen. Der Aufwand ist also $\frac{n^2}{2}$

3. BubbleSort

- Es werden immer zwei Zahlen miteinander verglichen
- Die größere Zahl wandert Schrittweise ans Ende
- Nach dem ersten Durchlauf ist nach n Vergleichen die größte Zahl am Ende angekommen
- Insgesamt n Durchläufe mit durchschnittlich $\frac{n}{2}$ Vergleichen
- Gesamter Aufwand $\frac{n^2}{2}$

4. MergeSort

- „divide-and-conquer“-Verfahren
- gesamte Liste wird halbiert
- die beiden Teillisten werden getrennt voneinander sortiert ($\rightarrow$ rekursiv!)
- anschließend werden die sortierten Teillisten wieder zu einer Liste zusammengefügt
- (Da bei jedem Schritt die Anzahl der Elemente halbiert wird, und dieses n mal machen müssen ist der Aufwand $n \cdot \log(n)$)

5. Quicksort

- „divide-and-conquer“-Verfahren
- ein Element wird als *Pivot*-Element verwendet
- Anhand dieses Elementes wird der Rest der Liste in zwei Teillisten aufgeteilt: alle Elemente kleiner und alle Elemente größer als dieses Pivot-Element
- Diese beiden Listen werden getrennt voneinander sortiert ($\rightarrow$ rekursiv!)
- Anschließend können die sortierte Liste mit den kleineren Elementen, das Pivot-Element und die sortierte Liste mit den größeren Elementen zusammengesetzt werden
- (Da bei jedem Schritt die Anzahl der Elemente halbiert wird, und dieses n mal machen müssen ist der Aufwand $n \cdot \log(n)$)

6. in-place vs. out-of-place

- bei einem in-place-Algorithmus wird (bis auf einzelne Variablen) kein zusätzlicher Speicherplatz verwendet
- bei einem out-of-place-Algorithmus wird ein zweites Array angelegt, in welches die sortierten Elemente eingefügt werden

7. stabil

- Ein Sortierverfahren ist dann stabil, wenn es die Reihenfolge „identischer“ Werte aus dem ursprünglichen Array auch im Ergebnis beibehält

8. Visualisierung

`https://www.toptal.com/developers/sorting-algorithms`