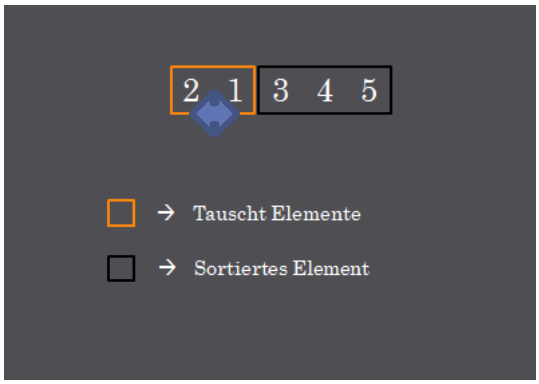
14 Juni 2018

Bubblesort

1.) Verfahren:



- Arbeitet in-Place und stabil
- Laufzeit:
  - Bester Fall: O(n)
  - Ungünstigster Fall: $O(n^2)$

- Es werden immer 2 Zahlen einer Reihe betrachtet
  - Ist die 2. Zahl kleiner werden die Zahlen ausgetauscht

2.) Verfahren im Vergleich:

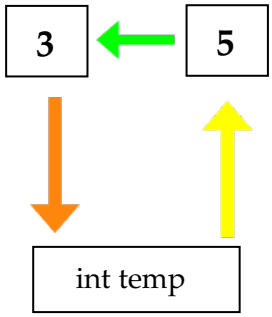
Sortierverfahren	Best-Case	Average-Case	Worst-Case	Stabil
Bubblesort	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	ja
Heapsort	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n \cdot \log(n))$	nein
Insertionsort	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	ja
Mergesort	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n \cdot \log(n))$	ja
Quicksort	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^2)$	nein
Selectionsort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	nein
Swap-Sort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	–
Timsort	$\mathcal{O}(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n \cdot \log(n))$	ja

3.) Vor – und Nachteile:

Vorteile	Nachteile
▪ In-Place ▪ Stabilität ▪ Einfaches Verfahren ▪ Minimumsuche nicht benötigt	▪ Langsames Verfahren: $O(n^2)$ ▪ Wird in der Praxis kaum eingesetzt

Leon Wenzler
Johannes Bergmann

4.) Java:




```
for(int i=1; i<array.length; i++) {
                for(int j=0; j<array.length-i; j++) {

        if(array[j]>array[j+1]) {
        int temp = array[j];
        array[j] = array[j+1];
        array[j+1] = temp;
} } }
```

5.) Quellen:

- https://www.geeksforgeeks.org/bubble-sort/
- https://de.wikipedia.org/wiki/Bubblesort
- http://www.java-programmieren.com/bubblesort-java.php
- http://www.inf.fh-flensburg.de/lang/algorithmen/sortieren/networks/bubble.htm