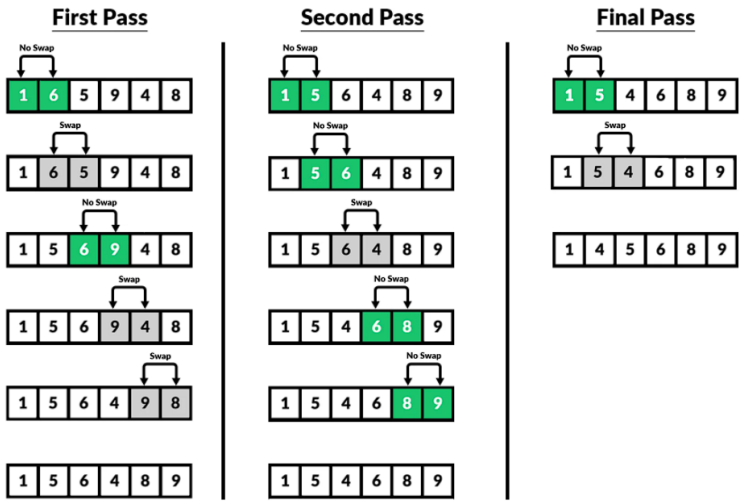
[A7] Bubble Sort

Grafische Beschreibung



Textuelle Beschreibung

Beim Bubblesort handelt es sich um einen iterativen Algorithmus, der in einer endlichen Anzahl an Durchläufen x Daten in aufsteigender oder absteigender Reihenfolge sortiert. Pro Durchlauf wird jedes Element mit seinem Nachbarn verglichen und bei Bedarf (bei aufsteigender Sortierung: wenn das Element größer ist als sein Nachbar) vertauscht. Daraus folgt, dass nach jedem Durchlauf das größte Element am Ende des Arrays steht. Somit müssen im nächsten Durchlauf nur mehr x-1 Elemente miteinander verglichen werden.

Der Algorithmus endet, wenn während eines Durchlaufs keine Vertauschungen mehr stattgefunden haben, oder die Anzahl der Durchläufe der Anzahl der Elemente entspricht.

Laufzeit

public void sort(Person[] personen) : $O(n^2)$

Begründung: Es werden zwei ineinander verschachtelte Schleifen benötigt, um die Elemente miteinander zu vergleichen.

Wichtige Elemente

```
/** Big O notation -> O(n²) */
public void sort(Person[] personen) {

  Person tmpPerson;          // temporary helper for sorting process
  boolean changed = true;  // termination variable for while loop
  int perfCount = 1;          // counter to reduce loop iterations // start with 1 for initial offset due .length

  while (changed){
    changed = false;    // set termination condition always to false, to terminate in case we do not enter IF

    for (int i = 0; i < (personen.length - perfCount); i++) {  //
      if (personen[i].compareTo(personen[i+1]) > 0){     // compare two elements which are next to each other
and check if they must be exchanged
        tmpPerson = personen[i];          // store element in helper variable
        personen[i] = personen[i+1];      // overwrite element with next element next to it (smaller one)
        personen[i+1] = tmpPerson;        // write on 2nd element the content of the helper
        changed = true;                   // set changed flag to to true to ensure another iteration (=while loop)
      }
    }
    perfCount++;  // increase count of performance counter, with each iteration of the for loop the largest unsorted
                  // element is already in the right position
                 // therefore the for loop does not need to check it again
  }
}
```