# Zusammenfassung für Minitest 1

## Komplexitätsklassen:

O(1) C O(ln n) C O(log(n^r) C O(log^r n) C O(n^ε) für alle ε > 0

O(n^a) C O(n^b) für a < b

O(n^a) C O(b^n) für alle a,b in R

O(b^n) C O(n^n)

## Fundamentalsatz:

## Bubblesort:

* Sortiert auftseigend
* Größtes Element wandert wie eine Blase nach hinten
* Best Case: O(n), wenn alle Elemente (Schlüssel) gleich sind 🡪 keine Vertauschung notwendig
* Average Case: O(n^2)
* Worst Case: O(n^2), wenn Array rückwärts sortiert ist, denn: Bewegung des k-ten Elements an die Stelle n erfordert n-k Vertauschungen. Aufsummiert ergibt das ½ (n^2 – n)
* Abbruch wenn nach einem Durchlauf keine Vertauschung erfolgt ist

## Shakersort:

* Funktioniert wie Bubblesort, allerdings sortiert es einmal von links und einmal von rechts
* Gleiche Komplexität wie bei Bubblesort

## Selectionsort:

* Voraussetzung dass der vordere Teil des Arrays bereits sortiert ist.
* Es wird das kleinste Element der unsortierten Liste gesucht und an den Anfang des Arrays gesetzt
* Es sind immer Vergleiche notwendig 🡪 Best Case = Average Case = Worst Case = O(n^2)

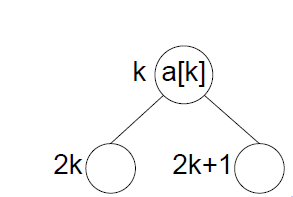
## Insertionsort:

* Beginnt mit dem Vergleichen der ersten beiden Elemente 🡪 sind sortiert und ergeben sortierte Teilliste
* Das aktuelle Element a wird in die bereits sortierte Teilliste am Beginn eingeordnet
* Best Case: O(n), Arrays bereits sortiert, muss durch durchlaufen werden, keine Vertauschungen
* Average Case: O(n^2) mit den Schieberoperationen
* Worst Case: O(n^2), wenn Liste absteigend sortiert ist 🡪 jedes Element von Ursprungsposition nach ganz vorne, Schiebeoperationen

## Quicksort:

* Aus dem Array wird ein Pivotelement gewählt 🡪 Array wird aufgeteilt in zwei Teillisten
* Element a wird mit Pivotelement verglichen. Ist es kleiner (compareTo =-1) wird es links eingeordnet, ist es größer (compareTo = 1) wird es rechts eingeordnet.
* Best Case: O(n\*(log n)), wenn Pivotelement so gewählt dass entstehende Teillisten ca. gleich groß; oder auch wenn Liste bereits sortiert und Pivotelement in der Mitte 🡪 kein Vertauschen
* Average Case: O(n\*(log n))
* Worst Case: O(n^2), wenn Liste bereits sortiert ist **und** Pivotelement ganz rechts oder ganz links gewählt wird.

## Heapsort:

Heap-Baum wird folgend aufgebaut:

Die Nummerierung der Elemente erfolgt immer von links nach rechts. Alle Elemente stehen von oben nach unten in einer <= Beziehung.

Beim Überprüfen der Heap-Eigenschaft wird von unten rechts begonnen. Abarbeitung von rechts nach links und anschließend immer eine Reihe weiter nach oben.

