## Selection

Beim Selection-Sort wird das gesamte Array in einer verschachtelten for-Loop durchgegangen. Es gibt zwei Zähler, der eine zeigt auf das zu untersuchende Element und der andere geht alle Elemente durch und schaut, ob das zu untersuchende kleiner ist. Gegebenenfalls vertauscht er die beiden. Er endet, wenn Zähler 1 die array Länge erreicht.

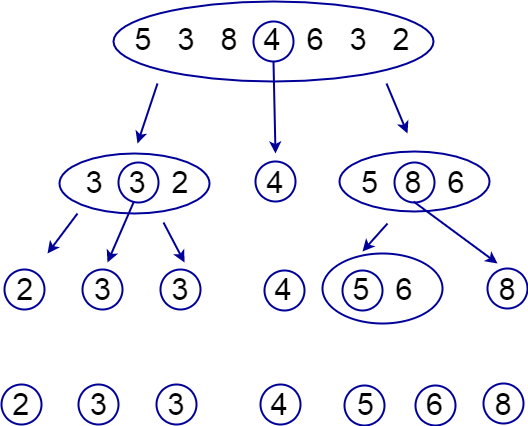
O(n^2)

## Insertion

Ähnlich wie beim Selection gibt es auch zwei Zähler, einer zählt hoch und der andere schaut an welcher Stelle er den Wert einsetzen kann, er zählt also vom ersten Wert bis nach null runter. Er endet, wenn Zähler 1 die array Länge erreicht.

O(n^2)

## Quick

Zuerst wird ein Element als Pivot ausgewählt (Egal welches, in der Regel das letzte).

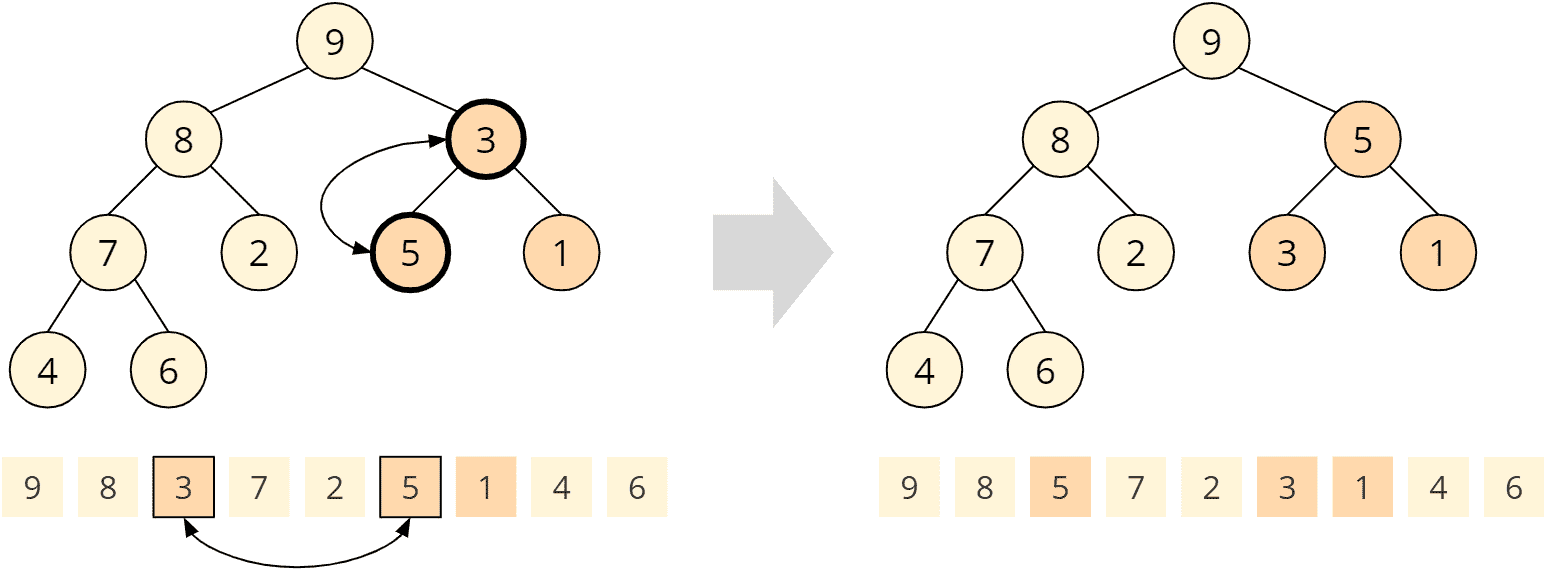
Dieses wird dann an die richtige Stelle im Array gelegt.

Anschließend werden alle kleineren Teile als das Pivot auf die linke Seite gelegt und alle größeren Rechts.

Danach werden diese Schritte rekursiv für die weiteren beiden Teilarrays durchgezogen (links und rechts von Pivot). Diese werden dann direkt verarbeitet, bis keine weitere Spaltung mehr erfolgen kann (in der Funktion dieselbe Funktion aufrufen). Dann wird wieder nach oben zurückgezogen. Man kann sich das als Baum vorstellen.

O(n\*log(n))

## Heap

Es wird das Array in ein sogenanntes Heap eingeteilt. Es gibt somit Parent Elementen, die jeweils 2 Children haben. Dieses Heap sind einfach die Array Elemente von links nach rechts eingetragen. Es muss aber ein Max-Heap gebildet werden, d.h. dass in jedem Parent-Child-Child-Dreieck muss der Parent größer als die Children sein. Wenn das vorhanden ist, muss das erste und das letzte Element des Heaps vertauscht werden. Dadurch ist das letzte Element an der richtigen Stelle und es kann aus dem weiteren Heaping ausgeschieden werden. Nun muss wieder ein Max-Heap gebildet werden. …

## Radix (LSD)

Hierbei wird nach den einzelnen Stellen der Werte sortiert. Es gibt ein temporäres Array von 0-9. Im ersten Durchlauf wird die hinterste/rechte Ziffer des Werts betrachtet. Diese wird in das temporäre Array eingeordnet, je nachdem wie hoch die Ziffer ist. Wenn alle eingeordnet sind, werden sie wieder zurück ins Array geschrieben (0 zuerst, 9 zuletzt). Dann wird nach der zweiten Ziffernstelle, dann nach der dritten, …

## Cocktail

Hierbei läuft ein Zähler von unten nach oben und dann wieder von oben zurück nach unten. Bei jeder De-/Inkrementierung. Wird überprüft, ob der aktuelle Wert größer ist als der n+1, wenn nein werden diese vertauscht. Sieht cool aus, ist aber sehr langsam.

## Merge

Der Merge-Sort vermischt bestimmte Teilarrays zu einem. D.h. wir starten mit dem Array und sortieren das erste und zweite Element. Das Ergebnis kann direkt wieder in das Array geschrieben werden. Das wird mit allen im ersten Schritt gemacht. Anschließend werden die zusammengefügten Teile wieder mit dem Nachbar sortiert. Und dass, bis das Array am Ende nur noch eines ist. Am Anfang ist es jeweils 1 Wert pro Array, dann 2, 4, 8, 16, n\*2, …

Das Aufteilen muss bei einem Array nicht beachtet werden, da die einzelnen Teile ja zugänglich sind.

O(n\*log(n))

