# Quicksort

Um den Quicksort Algorithmus verstehen und implementieren zu können, sollte man die Abschnitte Rekursion und das Teile-und-Herrsche-Prinzip bearbeitet haben.

Quicksort ist ein sehr schnellet Sortieralgorithmus. Er kommt in der Praxis häufig zum Einsatz. Zahlreiche Standardbibliotheken verschiedener Programmiersprachen enthalten Methoden um zum Beispiel Arrays zu sortieren, die in als Quicksort implementiert sind. Zum Beispiel hat die Standardbibliothek der Programmiersprache C eine Funktion namens `qsort`. Quicksort verwendet ein Teile-und-herrsche-Prinzip.

## Modellvorstellung

Stell dir vor die Schüler der 7a wollen sich wie die Orgelpfeifen der Größe nach geordnet aufstellen:

Zunächst wählt man die erste Person als „Vergleichsgröße“ aus, der Fachbegriff für das Element, das als Vergleichselement verwendet wird ist **Pivotelement**.

Jetzt teilt man das Problem in zwei Unterprobleme auf: Alle Schülerinnen die kleiner als das Pivotelement sind stellen sich links davon auf, alle die größer oder gleich sind rechts:

Das Pivotelement scheidet jetzt aus dem Verfahren aus, es bleibt an dem Platz, an dem es sich jetzt befindet. Jetzt haben wir zwei „kleinere“ Sortierprobleme geschaffen: Die Schülerinnen, die links setehen sind ungeordnet, die größeren auf der rechten Seite ebenfalls. Auf jeden Fall – auch im ungünstigsten[1] – sind die neuen Teilproblem(e) kleiner als das ursprüngliche Problem.

In den beiden Teilmengen verfährt man jetzt wie gerade in der Ausgangsmenge:

- Pivotelement wählen (die erste Schülerin ganz links)
- Menge in zwei Teile teilen: Kleiner und größer/gleich Pivotelement

Dieses Vorgehen wird jetzt wiederholt bis der Basisfall eintritt.

**(A1)**

Führe das Verfahren mit Stift und Papier zu Ende, bis du die Schüler nach Körpergröße sortiert hast.

**Frage:** Was ist der Basisfall beim sortieren der Schülergruppen? Wann kannst du also direkt ohne weitere Überlegung eine sortierte Schülergruppe zurückgeben?

Antwort:

Leere Arrays und Arrays mit nur einem Element stellen den Basisfall dar. Du kannst solche Arrays unverändert zurückgeben – es gibt nichts zu sortieren

# Quicksort

## Leere Arrays und solche mit nur einem Element

Wir legen den Basisfall zugrunde: Wenn unser Array leer ist oder nur ein Element hat, ist es sortiert und kann direkt als sortiertes Array zurückgegeben werden:

```
public ArrayList<Integer> quicksort(ArrayList<Integer> listToSort) {
   if(listToSort.size() < 2) {
     return listToSort;
   }
   [...]
}
```

## Arrays mit zwei oder mehr Elementen

Arrays mit **zwei Elementen** sind ebenfalls einfach zu bearbeiten: Man muss lediglich die beiden Elemente vergleichen und wenn nötig vertauschen, bevor man das dann sortierte Array zurückgibt.

Spannend wird es, wenn das Array drei Elemente hat:

Wir gehen vor, wie oben angedacht:

- Pivotelement wählen (erstes Element des Arrays)
- Partitionieren in *Elemente kleiner als Privot*, *Pivot* und *Elemente größer/gleich Pivot*:

Bislang haben wir als Pivotelemet stets einfach das erste Element des Arrays gewählt - tatsächlich ist es zunächst unerheblich, welches der Elemente man dazu heranzieht.

Untersuche, ob die Auswahl des Pivotelements einen Einfluss auf das Ergebnis des sortiervorgangs hat.

Das sortierte Array erhält man anschließend zuverlässig als:

[1)]

welches ist der ungünstigste Fall?

From:
https://wiki.qg-moessingen.de/ - **QG Wiki**

Permanent link:
**https://wiki.qg-moessingen.de/faecher:informatik:oberstufe:algorithmen:sortieren:quicksort:start?rev=1643641948**

Last update: **31.01.2022 16:12**