W08P02 - Rekursives Sortieren

PGdP-TUT Fr-08 Shiqi Li

mergeSort(List<Integer>)

Basisfall

- Nur ein Element => sortiert List
  => return dieses List
- sortiert List=> return List

```
if (list.size() <= 1) {
    return list;
}
boolean sorted = true;
for (int i = 1; i < list.size(); i++) {

    if (list.get(i) < list.get(i - 1)) {
        sorted = false;
        break;
    }
}
if (sorted) {
    return list;
}
```

Letzte rekursive Aufruf: [3, 6]

- In 2 Teile unterteilen. List.subList(int fromIndex, int toIndex)

```
List<Integer> list1 = list.subList(0, (list.size()+1) / 2);
List<Integer> list2 = list.subList((list.size()+1) / 2, list.size());
```

- Sortieren 2 Teile separate

```
list1 = mergeSort(list1);
list2 = mergeSort(list2);
```

- Merge 2 Ergebnisse von 2 Teilen

stoogeSort(int[ ])

- Achtung:
- Objekt Array ändert nicht, nur die Elemente vertauscht.

- Länger des Arrays <= 2
  - Ist ein sortiertes Array

Basisfall

```
int length = to - from;
if (length == 2 && array[from] > array[from + 1]) {
    int tmp = array[from];
    array[from] = array[from + 1];
    array[from + 1] = tmp;
}
if (length <= 2) {
    return;
}
```

Letzte rekursive Aufruf: Länge von Array ist 3

- Array in 3 Teilarrays unterteilen

```
int third = from + length / 3;
int twoThirds = to - length / 3;
```

- Sortieren 3 Teilarrays separat durch rekursive Aufruf nach Aufgabenstellung

```
stoogeSortHelper(array, from, twoThirds);
stoogeSortHelper(array, third, to);
stoogeSortHelper(array, from, twoThirds);
```