# W04H02 - Mega Merge Sort

Hausaufgabe Medium

Points: 3 of 6 Assessment: automatic

Submission due: 9 months ago

Complaint due: 9 months ago

Resume practice in exercise

50% (9 months ago) GRADED

## Recent results:

0% (9 months ago) GRADED

0% (9 months ago) GRADED

50% (9 months ago) GRADED

Show all results

Tasks:

## Mega Merge Sort

Nachdem du den Gartuinen mit ihrem Gnomroboter geholfen hast, sind immer mehr naturliebende Schuloine der Gruppe beigetreten. Darüber freut sich die Gruppe natürlich. Jetzt steht der alljährliche Planzenwettbewert vor der Tür und die Verantwortlichen mussten feststellen, dass ein Gnomrobote nicht mehr mit den Sortieren hinterher kommt. Glücklicherweise konnten der Roboter mehrfach nachgebaut werden. Alles was jetzt noch fehlt ist ein Sortieralgorithmus, der auch mehrere Roboter effektiv einsetzen kann. Und natürlich brauchen die Gartuine dafür wieder deine Hilfe!

## Aufgabe

In dieser Aufgabe geht es darum, MegaMergeSort zu implementieren. Dabei handelt es sich um eine Variante des bekannten Divide-and-Conquer Sortieralgorithmus MergeSort. Bei MergeSort werden die zu sortierenden Elemente zuerst in möglichst gleichgroße Bereiche geteilt, indem jeder Bereic immer wieder halbiert wird. Anschließend werden diese Bereiche wieder zusammengeführt (gemergt), wobei sie sortiert werden. MegaMergeSort funktioniert sehr ähnlich, allerdings werden Bereiche nicht halbiert, sondern in eine festgelegte Anzahl an Teilbereiche zerlegt.

### Mergen zweier Arrays No results

Zuerst sollst du die Methode `int[] merge(int[], int[])` implementieren, die die beiden übergebenen Arrays zu einem neuen zusammenfügt (merget) und dieses zurückgibt. Das Mergen funktioniert dabei wie bei normalem MergeSort: Man beginnt bei beiden Arrays beim ersten Element, vergleicht diese und fügt das kleinere im Ausgabe-Array ein. Das eingefügte Element überspringt man nun im Eingabe-Array und wiederholt die Prozedur, bis alle Elemente in der Ausgabe eingefügt wurden. Die Länge des zurückgegebenen Arrays entspricht natürlich der Summe der Längen der beiden Eingabearrays.

▼ Beispiel für das Mergen zweier Arrays

```
merge([1, 3], [2, 4])

Vergleiche 1 und 2 -> füge 1 in die Ausgabe ein: [1]
Vergleiche 3 und 2 -> füge 2 in die Ausgabe ein: [1, 2]
Vergleiche 3 und 4 -> füge 3 in die Ausgabe ein: [1, 2, 3]
Nur 4 ist noch übrig -> füge 4 in die Ausgabe ein: [1, 2, 3, 4]
```

### Mergen mehrerer Arrays No results

Jetzt sollst du `int[] merge(int[][], int, int)` implementieren. Diese Methode soll rekursiv die inneren Arrays aus `arrays` im Indexbereich `from` (inklusiv) bis `to` (exklusiv) mergen (also `arrays[from]`, `arrays[from + 1]`, ... und `arrays[to - 1]` sollen gemerget werden) und das daraus resultierende Array zurückgeben. Falls der Bereich `[from, to[` leer ist, soll ein Array der Länge `0` zurückgegeben werden. Die genaue Mergereihenfolge siehst du in folgendem Beispiel:

▼ Beispiel für das Mergen mehrerer Arrays

```
merge([[1], [2], [3], [4]], 0, 4)

merge von [3] und [4] -> [3, 4]
merge von [2] und [3, 4] -> [2, 3, 4]
merge von [1] und [2, 3, 4] -> [1, 2, 3, 4]
```

### MegaMergeSort No results

Als Letztes fehlt noch die Methode `int[] megaMergeSort(int[], int, int, int)` (die zugehörige Methode `int[] megaMergeSort(int[], int)`, mit der MegaMergeSort ausgeführt wird, ist schon gegeben). Diese Methode sortiert das `array` im Bereich `[from, to[` mit MegaMergeSort, wobei das Array in jeder Ebene in `div`-Teile aufgeteilt werden soll. Auch hier soll, falls `[from, to[` leer ist, ein Array der Länge `0` zurückgegeben werden. Beim Aufteilen gibt es Folgendes zu beachten: Es wird rekursiv aufgeteilt, bis ein Bereich maximal `1` Element enthält. Falls die Länge des zu sortierende Bereichs nicht restlos durch `div` teilbar ist, so sollen die ersten Teilbereiche entsprechend um 1 größer sein. Im Beispiel sieht das wie folgt aus:

▼ Beispiel für das Splitten eines Arrays

```
megaMergeSort([1, 2, 3, 4, 5, 6, 7, 8], 3, 0, 8)

8/3 = 2 Rest 2
Split der 1. Ebene: [1, 2, 3], [4, 5, 6], [7, 8]
Split der 2. Ebene:
  für [1, 2, 3]: 3/3 = 1 Rest 0 => [1], [2], [3]
  für [4, 5, 6]: 3/3 = 1 Rest 0 => [4], [5], [6]
  für [7, 8]: 2/3 = 0 Rest 2 => [7], [8], []
```

## Hinweis

- In der gesamten Aufgabe sind Methoden aus `java.util.Arrays` verboten.
- Wie du nach dem Klonen der Aufgabe sehen wirst, tragen die Methoden in dieser Aufgabe nicht die uns bekannten Köpfe `public static int[] merge(..)` o.Ä., sondern stattdessen Köpfe wie `protected int[] merge(..)`. Dies ist eine Notwendigkeit dafür, wie unsere Tests aufgebaut sind. Du kannst von einer der drei Methoden aus die anderen wie gewohnt mit einem Call wie `merge(.. parameter ..)` aufrufen. Von der `main()`-Methode aus kannst du das allerdings nicht mehr genauso. Daher haben wir dir die `main()`-Methode bereits implementiert. Um dein Verfahren zu testen, musst du nur noch `arr` und den zweiten Parameter im Call darunter entsprechend abändern. Was dieser neue Methodenkopf genau bedeutet, weißt du dann in Woche 07.

Lösungsvorschlag

Exercise details

| | |
|---|---|
| Release date: | Nov 10, 2022 18:30 |
| Submission due: | Nov 20, 2022 18:00 |
| Complaint due: | Nov 27, 2022 18:00 |

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left.