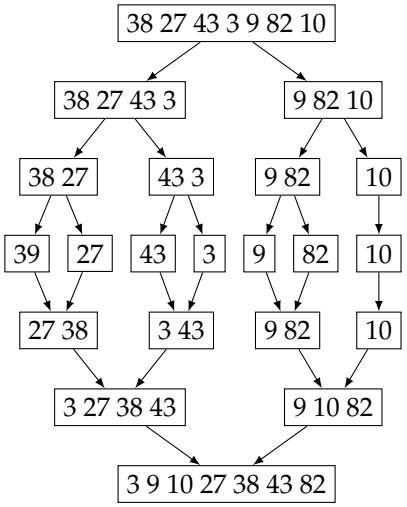
MergeSort: Sortieren durch Verschmelzen

Weiterführende Literatur:

- *Algorithmen und Datenstrukturen: Tafelübung 11, WS 2018/19*, Seite 50
- Saake und Sattler, *Algorithmen und Datenstrukturen*, Seite 131-135
- *Wikipedia-Artikel „Mergesort“*
- Schneider, *Taschenbuch der Informatik*, 6.4.2 Effiziente Sortierverfahren, Seite 192



- Funtionsweise:
  - Listen mit maximal *einem* Element sind *trivialerweise sortiert*
  - falls zu sortierende Liste mehr als ein Element beinhaltet:
    - *teile* Liste in zwei kleinere Listen auf
    - verfahre *rekursiv* mit den beiden Teillisten
    - *verschmelze* die zwei rekursiv sortierten Listen
    - dabei Sortierung in verschmolzener Liste beibehalten

  $\rightarrow$ Teile-Und-Herrsche
- Eigenschaften von MergeSort:
  - Laufzeitkomplexität: $\mathcal{O}(n \cdot log(n))$ (im Best-, Average- und Worst-Case)
  - bei geeigneter „Verschmelzung“ *stabile* Sortierung
  - durch Rekursion wachsender Aufrufstapel $\rightarrow$ *out-of-place*

Minimal zum Auswendig lernen

```
11    /**
12     * Eine Hilfsmethode für rekursives Sortieren durch Mischen des
```

```
13      * Mergesort-Algorithmus.
14      *
15      * <p><strong>Weitere Abkürzungen</strong></p>
16      *
17      * <ul>
18      * <li>i: Index links
19      * <li>j: Index rechts
20      * <li>k: Index
21      * <li>m: Index-Nummer der Mitte
22      * </ul>
23      *
24      * @param l Die linke Grenze.Die Index-Nummer, ab der das Zahlen-Feld sortiert
25      *          werden soll.
26      * @param r Die rechte Grenze. Die Index-Nummer, bis zu der das Zahlen-Feld
27      *          sortiert werden soll.
28      * @param h Eine Hilfsfeld, in dem die Zahlen temporär zwischengespeichert
29      *          werden.
30      */
31     private void mischen(int l, int r, int[] h) {
32       if (r <= l)
33         return;
34       int i, j, k;
35       int m = (r + l) / 2;
36       mischen(l, m, h);
37       mischen(m + 1, r, h);
38       for (k = l; k <= m; k++) {
39         h[k] = a[k];
40       }
41       for (k = m; k < r; k++) {
42         h[r + m - k] = a[k + 1];
43       }
44       i = l;
45       j = r;
46       for (k = l; k <= r; k++) {
47         if (h[i] < h[j]) {
48           a[k] = h[i++];
49         } else {
50           a[k] = h[j--];
51         }
52       }
53     }
54
55     public int[] sortiere() {
56       int h[] = new int[a.length];
57       mischen(0, a.length - 1, h);
58       return a;
59     }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/sortier/MergeMinimal.java`

```
9      /**
10      * Hilfsmethode für rekursives Sortieren durch Mischen
11      *
12      * @param links     Die Index-Nummer, ab der das Zahlen-Feld sortiert werden
13      *                  soll.
14      * @param rechts    Die Index-Nummer, bis zu der das Zahlen-Feld sortiert
    ↪  werden
15      *                  soll.
16      * @param hilfsFeld Eine Hilfsfeld, in dem die Zahlen temporär
17      *                  zwischengespeichert werden.
18      */
19     private void sortiereRekursiv(int links, int rechts, int[] hilfsFeld) {
20       // Wenn die rechte Grenze gleich (oder sogar kleiner) als die linke Grenze
         ↪  ist,
```

```
21      // tue nichts.
22      if (rechts <= links)
23        return;
24
25      // Zähler für diverse for-Schleifen deklarieren.
26      int indexLinks, indexRechts, index;
27      // zu sortierendes Feld teilen
28      int mitte = (rechts + links) / 2;
29      // Teilfelder sortieren
30      sortiereRekursiv(links, mitte, hilfsFeld);
31      sortiereRekursiv(mitte + 1, rechts, hilfsFeld);
32      // Hilfsfeld aufbauen
33      // Linke Hälfte übernehmen, z. B. aufsteigende Sortierung
34      for (index = links; index <= mitte; index++) {
35        // hilfsFeld[0-2]: 1 23 42
36        hilfsFeld[index] = zahlen[index];
37      }
38
39      // Rechte Hälfte umdrehen, z. B. absteigende Sortierung
40      for (index = mitte; index < rechts; index++) {
41        // hilfsFeld[3-4]: 8 7
42        hilfsFeld[rechts + mitte - index] = zahlen[index + 1];
43      }
44
45      // Ergebnisse mischen über Hilfsfeld
46      indexLinks = links;
47      indexRechts = rechts;
48      // Beispiel:
49      // indexLinks = 0, indexRechts = 4
50      // hilfsFeld: 1 23 42 7 8
51      // index = 0: if(1<8): zahlen[0] = 1 -> indexLinks++ -> indexLinks = 1
52      // index = 1: if(23<8): zahlen[1] = 8 -> indexRechts-- -> indexRechts = 3
53      // index = 2: if(23<7): zahlen[2] = 7 -> indexRechts-- -> indexRechts = 2
54      // index = 3: if(23<42): zahlen[3] = 23 -> indexLinks++ -> indexLinks = 2
55      // index = 4: if(42>42): zahlen[4] = 42 -> indexRechts-- -> indexRechts = 1
56      for (index = links; index <= rechts; index++) {
57        if (hilfsFeld[indexLinks] < hilfsFeld[indexRechts]) {
58          zahlen[index] = hilfsFeld[indexLinks++];
59        } else {
60          zahlen[index] = hilfsFeld[indexRechts--];
61        }
62      }
63    }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/sortier/Merge.java`

1

Literatur

[1] *Algorithmen und Datenstrukturen: Tafelübung 11, WS 2018/19*. `https://www.studon.fau.de/file2567217_download.html`. FAU: Lehrstuhl für Informatik 2 (Programmiersysteme).

[2] Gunter Saake und Kai-Uwe Sattler. *Algorithmen und Datenstrukturen. Eine Einführung in Java*. 2014.

[3] Uwe Schneider. *Taschenbuch der Informatik*. 7. Aufl. Hanser, 2012. ISBN: 9783446426382.

[1] Saake und Sattler, *Algorithmen und Datenstrukturen*, Seite 134 (PDF 152).

[4] *Wikipedia-Artikel „Mergesort“*. https://de.wikipedia.org/wiki/Mergesort.