Heapsort

Weiterführende Literatur:

- *Algorithmen und Datenstrukturen: Tafelübung 11, WS 2018/19*, Seite 45
- *Wikipedia-Artikel „Heapsort“*
- Saake und Sattler, *Algorithmen und Datenstrukturen*, Seite 407-413 14.6.1 Sortieren mit Bäumen: HeapSort

HeapSort • HeapSort ; Haldensortierung • füge alle Elemente in eine Max-Halde (bzw. Min-Halde) ein • das Maximum (bzw. Minimum) eines jeden Teilbaums steht in der Wurzel • solange noch Elemente in der Halde sind: • entnimm die Wurzel der Halde • füge sie vorne an die sortierte Liste an (bzw. am hintersten freien Platz ins sortierte Array ein) ; alle Elemente werden dabei aufsteigend (bzw. absteigend) sortiert • Eigenschaften von HeapSort: • Laufzeitkomplexität: $O(n \cdot \log n)$ • sortiert auf Grund der Eigenschaften der Halde instabil • in-place-Implementierung möglich (s.u.)

```
3   /**
4    * Sortiere ein Zahlen-Feld mit Hilfe des Heapsort-Algorithmus. (Nach Saake
5    * Seite 412)
6    */
7   public class Heap extends Sortieralgorithmus {
8     private void versickere(int index, int letzterIndex) {
9       int i = index + 1, j;
10      // zahlen[i] hat linken Sohn
11      while (2 * i <= letzterIndex) {
12        // zahlen[j] ist linker Sohn von zahlen[i]
13        j = 2 * i;
14        // zahlen[i] hat auch rechten Sohn
15        if (j < letzterIndex)
16          if (zahlen[j - 1] < zahlen[j])
17            // zahlen[j] ist jetzt kleiner
18            j++;
19        if (zahlen[i - 1] < zahlen[j - 1]) {
20          vertausche(i - 1, j - 1);
21          // versickere weiter
22          i = j;
23        } else {
24          // halte an, heap-Bedingung erfüllt
25          break;
26        }
27      }
28    }
29
30    /**
31     * Sortiere ein Zahlen-Feld mit Hilfe des Heapsort-Algorithmus. *
32     *
33     * @return Das sortierte Zahlenfeld.
34     */
35    public int[] sortiere() {
36      int i;
37      for (i = zahlen.length / 2; i >= 0; i--)
38        versickere(i, zahlen.length);
39      for (i = zahlen.length - 1; i > 0; i--) {
40        // tauscht jeweils letztes Element des Heaps mit dem ersten
41        vertausche(0, i);
```

```
42          // heap wird von Position 0 bis i hergestellt
43          versickere(0, i);
44        }
45        return zahlen;
46      }
47
48      public static void main(String[] args) {
49        new Heap().teste();
50      }
51    }
```

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

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] *Wikipedia-Artikel „Heapsort“*. `https://de.wikipedia.org/wiki/Heapsort`.