Aufgabe 4

Gegeben ist ein Array a von ganzen Zahlen der Länge n, z.B.:

i	0	1	2	3	4	5	6	7	8	9
a	5	-6	4	2	-5	7	-2	-7	3	5

Im Beispiel ist also $n = 10$. Es soll die maximale Teilsumme berechnet werden, also der Wert des

Ausdrucks Ok LIS max =

Im Beispiel ist dieser Wert 8 und wird für $i = 8$,$j = 10$ erreicht. Entwerfen Sie ein Divide-And-Conquer Verfahren, welches diese Aufgabenstellung in Zeit $\mathcal{O}(n \log n)$ löst. Skizzieren Sie Ihre Lösung hinreichend detailliert.

Tipp: Sie sollten ein geringfügig allgemeineres Problem lösen, welches neben der maximalen Teilsumme auch noch die beiden „maximalen Randsummen" berechnet. Die werden dann bei der Endausgabe verworfen.

```
3  /**
4   * Teilsumme.java Klasse mit Algorithmen für die Berechnung des größten
5   * gemeinsamen Teilers zweier Ganzzahlen Algorithmen und Datenstrukturen,
6   * Auflage 4, Kapitel 2.1
7   *
8   * nach Prof. Grude, Prof. Solymosi, (c) 2000-2008: APSIS GmbH 22. April 2008
9   * http://public.beuth-hochschule.de/oo-plug/A&D/prog/kap21/Teilsumme.java
10  */
11 public class Teilsumme {
12
13   private static int rechtesRandMax(final int[] folge, int links, int rechts) {
      ↪ //
14     // requires 0 <= links <= rechts < folge.length
15     // berechnet rechtes Randmaximum in folge zwischen links und rechts
16     int bisherMax = 0, bisherSum = 0;
17     for (int i = rechts; i >= links; i--) {
18       bisherSum += folge[i];
19       bisherMax = Math.max(bisherMax, bisherSum);
20     }
21
22     return bisherMax;
23   }
24
25   private static int linkesRandMax(final int[] folge, int links, int rechts) {
26     // requires 0 <= links <= rechts < folge.length
27     // berechnet linkes Randmaximum in folge zwischen links und rechts
28     int bisherMax = 0, bisherSum = 0;
29     for (int i = links; i <= rechts; i++) {
30       bisherSum += folge[i];
31       bisherMax = Math.max(bisherMax, bisherSum);
32     }
33
34     return bisherMax;
35   }
36
37   private static int maxTeilsummeRekursiv(final int[] folge, int links, int
      ↪ rechts) {
38     // requires 0 <= links <= rechts < folge.length
39     // berechnet maximale Teilsumme in folge zwischen links und rechts
40     if (links == rechts) // nur ein Element
41       return Math.max(0, folge[links]);
42     else {
```

```
43          final int mitte = (rechts + links) / 2;
44          final int maxLinks = maxTeilsummeRekursiv(folge, links, mitte);
45          final int maxRechts = maxTeilsummeRekursiv(folge, mitte + 1, rechts);
46          final int rechtesMax = rechtesRandMax(folge, links, mitte);
47          // linke Hälfte
48          final int linkesMax = linkesRandMax(folge, mitte + 1, rechts);
49          // rechte Hälfte
50          return Math.max(maxRechts, Math.max(maxLinks, rechtesMax + linkesMax));
51        }
52      }
53
54      /**
55       * Berechnet die maximale Teilsumme von folge rekursiv mit logarithmischer
56       * Zeitkomplexität
57       *
58       * @param folge
59       * @return maximale Teilsumme in folge
60       */
61      public static int maxTeilsummeRekursiv(final int[] folge) {
62        // berechnet maximale Teilsumme von folge
63        return maxTeilsummeRekursiv(folge, 0, folge.length - 1);
64      }
65
66      public static void main(String[] args) {
67        final int[] testfolge = { +5, -8, +3, +3, -5, +7, -2, -7, +3, +5 };
68        // final int[] testfolge = {+5, -8, +3, +3, -5, +2, +7, -2, -7, +3, -1, +5};
69
70        int[] folge;
71        if (args.length > 0) {
72          folge = new int[args.length];
73          for (int i = 0; i < args.length; i++)
74            folge[i] = Integer.parseInt(args[0]);
75        } else
76          folge = testfolge;
77
78        int erg = maxTeilsummeRekursiv(folge);
79        System.out.println("maxTeilsummRekursiv = " + erg);
80      }
81    }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2012/herbst/Teilsumme.java`