

Aufgabe 2 [Minimum und Maximum]

- (a) Argumentieren Sie, warum man das Maximum von n Zahlen nicht mit weniger als $n - 1$ Vergleichen bestimmen kann.

Wenn die n Zahlen in einem unsortierten Zustand vorliegen, müssen wir alle Zahlen betrachten, um das Maximum zu finden. Wir brauchen dazu $n - 1$ und nicht n Vergleiche, da wir die erste Zahl zu Beginn des Algorithmus als Maximum definieren und anschließend die verbleibenden Zahlen $n - 1$ mit dem aktuellen Maximum vergleichen.

- (b) Geben Sie einen Algorithmus im Pseudocode an, der das Maximum eines Feldes der Länge n mit genau $n - 1$ Vergleichen bestimmt.

```

5  public static int bestimmeMaximum(int[] a) {
6      int max = a[0];
7      for (int i = 1; i < a.length; i++) {
8          if (a[i] > max) {
9              max = a[i];
10         }
11     }
12     return max;
13 }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java

- (c) Wenn man das Minimum und das Maximum von n Zahlen bestimmen will, dann kann das natürlich mit $2n - 2$ Vergleichen erfolgen. Zeigen Sie, dass man bei jedem beliebigen Feld mit deutlich weniger Vergleichen auskommt, wenn man die beiden Werte statt in zwei separaten Durchläufen in einem Durchlauf geschickt bestimmt.

```

15  /**
16   * Diese Methode ist nicht optimiert. Es werden  $2n - 2$  Vergleiche benötigt.
17   *
18   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
19   *         werden soll.
20   *
21   * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
22   *         der zweite Eintrag das Maximum.
23   */
24  public static int[] minMaxNaiv(int[] a) {
25      int max = a[0];
26      int min = a[0];
27      for (int i = 1; i < a.length; i++) {
28          if (a[i] > max) {
29              max = a[i];
30          }
31          if (a[i] < min) {
32              min = a[i];
33          }
34      }
35      return new int[] { min, max };
36  }
37
38  /**
39   * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
40   * betrachtet. Die Anzahl der Vergleiche reduziert sich auf  $3n/2 + 2$  bzw.
41   *  $3(n-1)/2 + 4$  bei einer ungeraden Anzahl an Zahlen im Feld.
42   *
43   * nach <a href=
```

```

44  * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-minimum-
→ comparisons/">techiedelight.com</a>
45  *
46  * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
47  *         werden soll.
48  *
49  * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
50  *         der zweite Eintrag das Maximum.
51  */
52  public static int[] minMaxIterativPaarweise(int[] a) {
53      int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
54      int n = a.length;
55
56      boolean istUngerade = (n & 1) == 1;
57      if (istUngerade) {
58          n--;
59      }
60
61      for (int i = 0; i < n; i = i + 2) {
62          int maximum, minimum;
63
64          if (a[i] > a[i + 1]) {
65              minimum = a[i + 1];
66              maximum = a[i];
67          } else {
68              minimum = a[i];
69              maximum = a[i + 1];
70          }
71
72          if (maximum > max) {
73              max = maximum;
74          }
75
76          if (minimum < min) {
77              min = minimum;
78          }
79      }
80
81      if (istUngerade) {
82          if (a[n] > max) {
83              max = a[n];
84          }
85
86          if (a[n] < min) {
87              min = a[n];
88          }
89      }
90      return new int[] { min, max };
91  }
92
93  /**
94   * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
95   * funktioniert so ähnlich wie der Mergesort.
96   *
97   * nach <a href=
98   * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-value-in-an-
→ array">enjoyalgorithms.com</a>
99   *
100  * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
101  *         gesucht werden soll.
102  * @param l Die linke Grenze.
103  * @param r Die rechts Grenze.
104  *
105  * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
106  *         der zweite Eintrag das Maximum.
107  */
108  int[] minMaxRekursiv(int[] a, int l, int r) {
109      int max, min;
110      if (l == r) {

```

```
111     max = a[l];
112     min = a[l];
113 } else if (l + 1 == r) {
114     if (a[l] < a[r]) {
115         max = a[r];
116         min = a[l];
117     } else {
118         max = a[l];
119         min = a[r];
120     }
121 } else {
122     int mid = l + (r - l) / 2;
123     int[] lErgebnis = minMaxRekursiv(a, l, mid);
124     int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
125     if (lErgebnis[0] > rErgebnis[0]) {
126         max = lErgebnis[0];
127     } else {
128         max = rErgebnis[0];
129     }
130     if (lErgebnis[1] < rErgebnis[1]) {
131         min = lErgebnis[1];
132     } else {
133         min = rErgebnis[1];
134     }
135 }
136 int[] ergebnis = { max, min };
137 return ergebnis;
138 }
139
140 }
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

Github: Staatsexamen/46115/2021/03/Thema-2/Teilaufgabe-1/Aufgabe-2.tex