

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
17   * benötigt.
18   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem
19   * Maximum gesucht
20   * werden soll.
21   * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält
22   * das Minimum,
23   * der zweite Eintrag das Maximum.
24   */
25  public static int[] minMaxNaiv(int[] a) {
26      int max = a[0];
27      int min = a[0];
28      for (int i = 1; i < a.length; i++) {
29          if (a[i] > max) {
```

```

29         max = a[i];
30     }
31     if (a[i] < min) {
32         max = 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         if (a[n] < min) {
86             min = a[n];
87         }
88     }
89     return new int[] { min, max };
90 }
91
92 /**
93  * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert.
94  * Er
95  * funktioniert so ähnlich wie der Mergesort.
96  *
97  * nach <a href=
98  * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-
99  * maximum-value-in-an-array">enjoyalgorithms.com</a>
100  * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem
101  * Maximum
102  * gesucht werden soll.
103  * @param l Die linke Grenze.
104  * @param r Die rechts Grenze.
105  * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält
106  * das Minimum,
107  * der zweite Eintrag das Maximum.
108  */
109 int[] minMaxRekursiv(int[] a, int l, int r) {
110     int max, min;
111     if (l == r) {
112         max = a[l];
113         min = a[l];
114     } else if (l + 1 == r) {
115         if (a[l] < a[r]) {
116             max = a[r];
117             min = a[l];
118         } else {
119             max = a[l];
120             min = a[r];
121         }
122     } else {
123         int mid = l + (r - l) / 2;
124         int[] lErgebnis = minMaxRekursiv(a, l, mid);
125         int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
126         if (lErgebnis[0] > rErgebnis[0]) {
127             max = lErgebnis[0];
128         } else {
129             max = rErgebnis[0];
130         }
131         if (lErgebnis[1] < rErgebnis[1]) {
132             min = lErgebnis[1];
133         } else {
134             min = rErgebnis[1];
135         }
136     }
137     int[] ergebnis = { max, min };
138     return ergebnis;
139 }
140 }

```

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

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