Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 2 / Aufgabe Nr. 2

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.

```java
public static int bestimmeMaximum(int[] a) {
  int max = a[0];
  for (int i = 1; i < a.length; i++) {
    if (a[i] > max) {
      max = a[i];
    }
  }
  return max;
}
```

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.

```java
/**
 * Diese Methode ist nicht optimiert. Es werden 2n - 2 Vergleiche benötigt.
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
 *          werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxNaiv(int[] a) {
  int max = a[0];
  int min = a[0];
  for (int i = 1; i < a.length; i++) {
    if (a[i] > max) {
      max = a[i];
    }
    if (a[i] < min) {
      max = a[i];
    }
  }
  return new int[] { min, max };
}

/**
 * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
 * betrachtet. Die Anzahl der Vergleiche reduziert sich auf 3n/2 + 2 bzw.
 * 3(n-1)/2 + 4 bei einer ungeraden Anzahl an Zahlen im Feld.
 *
 * nach <a href=
```

```java
   * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-minimum-
↪  comparisons/">techiedelight.com</a>
   *
   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
   *          werden soll.
   *
   * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das Minimum,
   *         der zweite Eintrag das Maximum.
   */
  public static int[] minMaxIterativPaarweise(int[] a) {
    int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
    int n = a.length;

    boolean istUngerade = (n & 1) == 1;
    if (istUngerade) {
      n--;
    }

    for (int i = 0; i < n; i = i + 2) {
      int maximum, minimum;

      if (a[i] > a[i + 1]) {
        minimum = a[i + 1];
        maximum = a[i];
      } else {
        minimum = a[i];
        maximum = a[i + 1];
      }

      if (maximum > max) {
        max = maximum;
      }

      if (minimum < min) {
        min = minimum;
      }
    }

    if (istUngerade) {
      if (a[n] > max) {
        max = a[n];
      }

      if (a[n] < min) {
        min = a[n];
      }
    }
    return new int[] { min, max };
  }

  /**
   * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
   * funktioniert so ähnlich wie der Mergesort.
   *
   * nach <a href=
   * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-value-in-an-
↪  array">enjoyalgorithms.com</a>
   *
   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
   *          gesucht werden soll.
   * @param l Die linke Grenze.
   * @param r Die rechts Grenze.
   *
   * @return Ein Feld mit zwei Einträgen. Der erste Einträg enthält das Minimum,
   *         der zweite Eintrag das Maximum.
   */
  int[] minMaxRekursiv(int[] a, int l, int r) {
    int max, min;
    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`

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