Aufgabe 3

Eine Folge von Zahlen $a_1, \ldots, a_n$ heiße unimodal, wenn sie bis zu einem bestimmten Punkt echt ansteigt und dann echt fällt. Zum Beispiel ist die Folge $1, 3, 5, 6, 5, 2, 1$ unimodal, die Folgen $1, 3, 5, 4, 7, 2, 1$ und $1, 2, 3, 3, 4, 3, 2, 1$ aber nicht.

> **Exkurs: Unimodale Abbildung**
> Eine unimodale Abbildung oder unimodale Funktion ist in der Mathematik eine Funktion mit einem eindeutigen (lokalen und globalen) Maximum wie zum Beispiel $f(x) = -x^2$. [a]
>
> [a] https://de.wikipedia.org/wiki/Unimodale_Abbildung

(a) Entwerfen Sie einen Algorithmus, der zu (als Array) gegebener unimodaler Folge $a_1, \ldots, a_n$ in Zeit $\mathcal{O}(\log n)$ das Maximum $\max a_i$ berechnet. Ist die Folge nicht unimodal, so kann Ihr Algorithmus ein beliebiges Ergebnis liefern. Größenvergleiche, arithmetische Operationen und Arrayzugriffe können wie üblich in konstanter Zeit ($\mathcal{O}(1)$) getätigt werden. Hinweise: binäre Suche, divide-and-conquer.

```java
public class UnimodalFinder {

  /**
   * https://gist.github.com/viniru/6f134fecc98a15465bae2149ef89a3f7
   *
   * @param a
   * @param l
   * @param h
   */
  public static int findeMaxRekursiv(int a[], int l, int h) {
    int mid = (l + h) / 2;
    if (a[mid] < a[mid + 1]) {
      if (a[mid + 1] > a[mid + 2]) {
        return a[mid + 1];
      } else {
        return findeMaxRekursiv(a, mid + 1, h);
      }
    }

    return findeMaxRekursiv(a, l, mid);

  }

  /**
   * https://github.com/yosriady/Other-Java-
↪  code/blob/master/Unimodal.java
   *
   * @param A
   * @param size
   * @return
   */
  public static int findeMaxIterativ(int[] A, int size) {
    int begin = 0;
    int end = size - 1;
    int mid;
```

```
37
38      while (begin < end) {
39        mid = begin + (end - begin) / 2;
40        if (A[mid] > A[mid - 1] && A[mid] > A[mid + 1]) {
41          return A[mid];
42        } else if (A[mid] > A[mid - 1]) {
43          begin = mid + 1;
44        } else {
45          // if the element on the left of mid is bigger
46          end = mid - 1;
47        }
48      }
49
50      return -1;
51    }
52
53    public static void main(String[] args) {
54      int[] test = { 1, 3, 4, 6, 7, 8, 9, 11, 6, 5, 4, 3, 2 };
55
56      System.out.println(findeMaxIterativ(test, test.length));
57
58      int a[] = { 1, 2, 3, 1 };
59      System.out.println(findeMaxRekursiv(a, 0, a.length - 1));
60
61    }
62
63  }
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java

(b) Begründen Sie, dass Ihr Algorithmus tatsächlich in Zeit $\mathcal{O}(\log n)$ läuft.

(c) Schreiben Sie Ihren Algorithmus in Pseudocode oder in einer Programmiersprache Ihrer Wahl, z. B. Java, auf. Sie dürfen voraussetzen, dass die Eingabe in Form eines Arrays der Größe n vorliegt.

(d) Beschreiben Sie in Worten ein Verfahren, welches in Zeit $\mathcal{O}(n)$ feststellt, ob eine vorgelegte Folge unimodal ist oder nicht.

(e) Begründen Sie, dass es kein solches Verfahren (Test auf Unimodalität) geben kann, welches in Zeit $\mathcal{O}(\log n)$ läuft.