Übungsblatt 10

Adrian Schollmeyer

Aufgabe 1

(a)

Listing 1: MinMaxMain.java

```
package me.adrian.aup10;

public class MinMaxMain
{
    public static int[] minMax(int[] data)
    {
        return minMax(data, 0, data.length);
    }

    // endIterator points to one element past the last element of the selected range
    public static int[] minMax(int[] data, int beginIterator, int endIterator)
    {
        if (endIterator <= beginIterator)
            throw new IllegalArgumentException("Invalid iterator positions");

        if (endIterator - beginIterator == 1)
            return new int[] { data[beginIterator], data[beginIterator] };

        if (endIterator - beginIterator == 2)
            return (data[beginIterator] <= data[endIterator - 1]) ?
                   (new int[] { data[beginIterator], data[endIterator - 1] }) :
                   (new int[] { data[endIterator - 1], data[beginIterator] }) ;

        int[] lastMinMax = minMax(data, beginIterator, endIterator - 2);
        int curMin, curMax;
        if (data[endIterator - 1] <= data[endIterator - 2]) {
            curMin = data[endIterator - 1];
            curMax = data[endIterator - 2];
        } else {
            curMin = data[endIterator - 2];
            curMax = data[endIterator - 1];
        }

        int actualMin = (curMin <= lastMinMax[0]) ? curMin : lastMinMax[0];
```

```
        int actualMax = (curMax > lastMinMax[1]) ? curMax : lastMinMax[1];

        return new int[] { actualMin, actualMax };
    }

    public static void main(String[] args)
    {
        // Example input
        int[] res = minMax(new int[] { 2, 3, 4, 5, 3, 2, 9, 1 });
        System.out.println(res[0] + "\t" + res[1]);
    }
}
```

(b)

Die Rekurrenzgleichung für die Anzal der Vergleiche des angegebenen Algorithmus' ist

$$T(n) = T(n-2) + 3 \tag{1}$$
$$T(1) = 0 \tag{2}$$
$$T(2) = 1 \tag{3}$$

Zudem gilt, sofern n eine Zweierpotenz ist:

$$T(n) \leq \frac{3}{2}n \tag{4}$$

Beweis.

Induktionsanfang:

$$T(1) = 0 \leq \frac{3}{2} \cdot 1 \quad \checkmark \tag{5}$$
$$T(2) = 1 \leq \frac{3}{2} \cdot 2 = 3 \tag{6}$$

Induktionsvorraussetzung:

$$T(n-2) \leq \frac{3}{2}(n-2) \tag{7}$$

Induktionsbehauptung:

$$T(n) \leq \frac{3}{2}n \tag{8}$$

Induktionsschritt:

$$T(n) = T\left(\frac{n}{2}\right) + 3 \tag{9}$$
$$\overset{\text{IV}}{\leq} \frac{3}{2}(n-2) + 3 \tag{10}$$
$$= \frac{3}{2}n - 3 + 3 \tag{11}$$
$$= \frac{3}{2}n \tag{12}$$

□

Aufgabe 2

Listing 2: AbsoluteMajorityMain.java

```
package me.adrian.aup10;

public class AbsoluteMajorityMain
{
    public static int absoluteMajority(int[] data)
    {
        return absoluteMajority(data, 0, data.length);
    }

    /**
     * @return The value of the absolute majority, or -1, if there is no absolute majority
     */
    public static int absoluteMajority(int[] data, int beginIterator, int endIterator)
    {
        if (endIterator <= beginIterator)
            throw new IllegalArgumentException("Invalid iterator positions");

        if (endIterator - beginIterator == 1)
            return data[beginIterator];

        if (endIterator - beginIterator == 2)
            return isDuplicate(data[beginIterator], data[endIterator - 1]) ? data[beginIterator] : -1;

        int middleIterator = beginIterator + (endIterator - beginIterator) / 2;

        int lowerMaj = absoluteMajority(data, beginIterator, middleIterator);
        int upperMaj = absoluteMajority(data, middleIterator, endIterator);

        int lowerMajDuplicates = getDuplicateAmount(data, lowerMaj, beginIterator, endIterator);
        int upperMajDuplicates = getDuplicateAmount(data, upperMaj, beginIterator, endIterator);

        if (lowerMaj != -1 && lowerMajDuplicates > (endIterator - beginIterator) / 2)
            return lowerMaj;

        if (upperMaj != -1 && upperMajDuplicates > (endIterator - beginIterator) / 2)
```

```
            return upperMaj;

        return -1;
    }

    private static int getDuplicateAmount(int[] data, int baseValue, int
        beginIterator, int endIterator)
    {
        if (endIterator <= beginIterator)
            throw new IllegalArgumentException("Invalid iterator positions");

        int n = 0;
        for (int i = beginIterator; i < endIterator; i++)
            n += isDuplicate(baseValue, data[i]) ? 1 : 0;

        return n;
    }

    private static boolean isDuplicate(int x, int y)
    {
        return x == y;
    }

    public static void main(String[] args)
    {
        System.out.println(absoluteMajority(new int[] { 1, 4, 4, 4, 3, 2, 4,
            3, 4, 3, 3, 4, 4}));
    }
}
```