

Einzelprüfung „Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)“

Einzelprüfungsnummer 46115 / 2015 / Herbst

Thema 1 / Aufgabe 3

(Unimodale Zahlenfolge)

Stichwörter: Teile-und-Herrsche (Divide-and-Conquer), Binäre Suche, Implementierung in Java

Aufgabe 3

Eine Folge von Zahlen a_1, \dots, a_n heie unimodal, wenn sie bis zu einem bestimmten Punkt echt ansteigt und dann echt fllt. 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

^ahttps://de.wikipedia.org/wiki/Unimodale_Abbildung

- (a) Entwerfen Sie einen Algorithmus, der zu (als Array) gegebener unimodaler Folge a_1, \dots, 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. Grenvergleiche, arithmetische Operationen und Arrayzugriffe knnen wie blich in konstanter Zeit ($\mathcal{O}(1)$) gettigt werden. Hinweise: binre Suche, divide-and-conquer.

Lsungsvorschlag

Wir whlen einen Wert in der Mitte der Folge aus. Ist der direkte linke und der direkte rechte Nachbar dieses Wertes kleiner, dann ist das Maximum gefunden. Ist nur linke Nachbar grer, setzen wir die Suche wie oben beschrieben in der linken Hlfte, sonst in der rechten Hlfte fort.

- (b) Begrnden Sie, dass Ihr Algorithmus tatschlich in Zeit $\mathcal{O}(\log n)$ luft.

Lsungsvorschlag

Da der beschriebene Algorithmus nach jedem Bearbeitungsschritt nur auf der Hlfte der Feld-Elemente zu arbeiten hat, muss im schlechtesten Fall nicht die gesamte Folge durchsucht werden. Nach dem ersten Teilen der Folge bleiben nur noch $\frac{n}{2}$ Elemente, nach dem zweiten Schritt $\frac{n}{4}$, nach dem dritten $\frac{n}{8}$ usw. Allgemein bedeutet dies, dass im i -ten Durchlauf maximal $\frac{n}{2^i}$ Elemente zu durchsuchen sind. Entsprechend werden $\log_2 n$ Schritte bentigt. Somit hat der Algorithmus zum Finden des Maximums in einer unimodalen Folge in der Landau-Notation ausgedrckt die Zeitkomplexitt $\mathcal{O}(\log n)$.

- (c) Schreiben Sie Ihren Algorithmus in Pseudocode oder in einer Programmiersprache Ihrer Wahl, z. B. Java, auf. Sie drfen voraussetzen, dass die Eingabe in Form eines Arrays der Gre n vorliegt.

Rekursiver Ansatz

```
public static int findeMaxRekursiv(int feld[], int links, int rechts) {
    if (links == rechts - 1) {
        return feld[links];
    }
    // bedeutet aufrunden
    // https://stackoverflow.com/a/17149572
    int mitte = (int) Math.ceil((double) (links + rechts) / 2);
    if (feld[mitte - 1] < feld[mitte]) {
        return findeMaxRekursiv(feld, mitte, rechts);
    } else {
        return findeMaxRekursiv(feld, links, mitte);
    }
}

public static int findeMaxRekursiv(int feld[]) {
    return findeMaxRekursiv(feld, 0, feld.length - 1);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java)

Iterativer Ansatz

```
public static int findeMaxIterativ(int[] feld) {
    int links = 0;
    int rechts = feld.length - 1;
    int mitte;

    while (links < rechts) {
        mitte = links + (rechts - links) / 2;
        if (feld[mitte] > feld[mitte - 1] && feld[mitte] > feld[mitte + 1]) {
            return feld[mitte];
        } else if (feld[mitte] > feld[mitte - 1]) {
            links = mitte + 1;
        } else {
            rechts = mitte - 1;
        }
    }
    return KEIN_MAX;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java)

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

Lösungsvorschlag

```
public static boolean testeUnimodalität(int[] feld) {  
    if (feld.length < 2) {
```

```

    // Die Reihe braucht mindestens 3 Einträge
    return false;
}

if (feld[0] > feld[1]) {
    // Die Reihe muss zuerst ansteigen
    return false;
}

boolean maxErreicht = false;
for (int i = 0; i < feld.length - 1; i++) {
    if (feld[i] > feld[i + 1] && !maxErreicht) {
        maxErreicht = true;
    }

    if (maxErreicht && feld[i] < feld[i + 1]) {
        // Das Maximum wurde bereits erreicht und die nächste Zahl ist größer
        return false;
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java)

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

Lösungsvorschlag

Da die Unimodalität nur durch einen Werte an einer beliebigen Stelle der Folge verletzt werden kann, müssen alle Elemente durchsucht und überprüft werden.

Lösungsvorschlag

Komplette Klasse

```

public static int findeMaxRekursiv(int feld[], int links, int rechts) {
    if (links == rechts - 1) {
        return feld[links];
    }
    // bedeutet aufrunden
    // https://stackoverflow.com/a/17149572
    int mitte = (int) Math.ceil((double) (links + rechts) / 2);
    if (feld[mitte - 1] < feld[mitte]) {
        return findeMaxRekursiv(feld, mitte, rechts);
    } else {
        return findeMaxRekursiv(feld, links, mitte);
    }
}

public static int findeMaxRekursiv(int feld[]) {
    return findeMaxRekursiv(feld, 0, feld.length - 1);
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinder.java)

Test

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class UnimodalFinderTest {

    private void testeMaxIterativ(int[] feld, int max) {
        assertEquals(max, UnimodalFinder.findeMaxIterativ(feld));
    }

    private void testeMaxRekursiv(int[] feld, int max) {
        assertEquals(max, UnimodalFinder.findeMaxRekursiv(feld));
    }

    private void testeMax(int[] feld, int max) {
        testeMaxIterativ(feld, max);
        testeMaxRekursiv(feld, max);
    }

    @Test
    public void findeMax() {
        testeMax(new int[] { 1, 2, 3, 1 }, 3);
    }

    @Test
    public void findeMaxLaengeresFeld() {
        testeMax(new int[] { 1, 3, 4, 6, 7, 8, 9, 11, 6, 5, 4, 3, 2 }, 11);
    }

    @Test
    public void keinMaxAufsteigend() {
        testeMaxIterativ(new int[] { 1, 2, 3 }, UnimodalFinder.KEIN_MAX);
    }

    @Test
    public void keinMaxAbsteigend() {
        testeMaxIterativ(new int[] { 3, 2, 1 }, UnimodalFinder.KEIN_MAX);
    }

    @Test
    public void maxNegativeZahlen() {
        testeMax(new int[] { -2, -1, 3, 1 }, 3);
    }

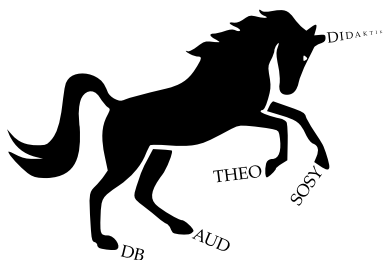
    private void testeUnimodalität(int[] feld, boolean wahr) {
        assertEquals(wahr, UnimodalFinder.testeUnimodalität(feld));
    }

    @Test
    public void unimodalität() {
        testeUnimodalität(new int[] { 1, 2, 3, 1 }, true);
    }
}
```

```
@Test
public void unimodalitätFalsch() {
    testeUnimodalität(new int[] { 1, -2, 3, 1, 2 }, false);
    testeUnimodalität(new int[] { 1, 2, 3, 1, 2 }, false);
    testeUnimodalität(new int[] { 3, 2, 1 }, false);
}

}
```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinderTest.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2015/herbst/UnimodalFinderTest.java)



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net. Der \LaTeX -Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: <https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Staatsexamen/46115/2015/09/Thema-1/Aufgabe-3.tex>