

Einzelprüfung „Theoretische Informatik / Algorithmen (vertieft)“

Einzelprüfungsnummer 66115 / 2020 / Herbst

Thema 1 / Teilaufgabe 2 / Aufgabe 2

(Bäume)

Stichwörter: Binärbaum, AVL-Baum

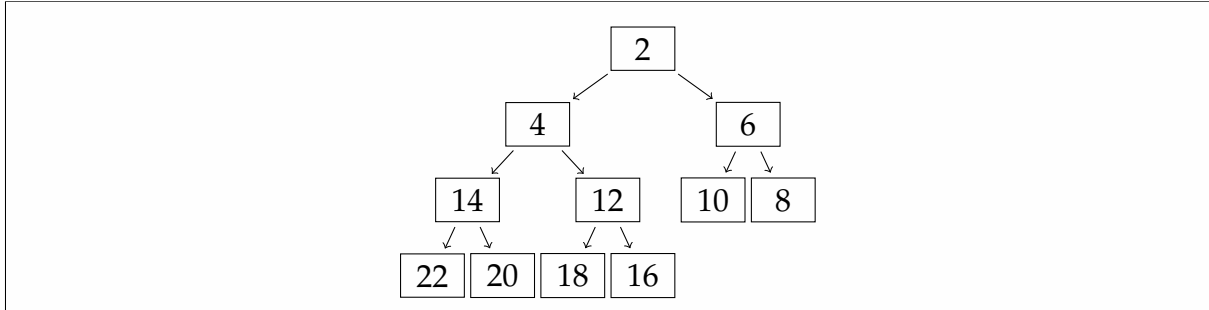
Wir betrachten ein Feld A von ganzen Zahlen mit n Elementen, die über die Indizes $A[0]$ bis $A[n-1]$ angesprochen werden können. In dieses Feld ist ein binärer Baum nach den folgenden Regeln eingebettet: Für das Feldelement mit Index i befindet sich

- der Elternknoten im Feldelement mit Index $\lfloor \frac{i-1}{2} \rfloor$,
- der linke Kindknoten im Feldelement mit Index $2 \cdot i + 1$, und
- der rechte Kindknoten im Feldelement mit Index $2 \cdot i + 2$.

(a) Zeichnen Sie den durch das folgende Feld repräsentierten binären Baum.

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|------|---|---|---|----|----|----|---|----|----|----|----|
| A[i] | 2 | 4 | 6 | 14 | 12 | 10 | 8 | 22 | 20 | 18 | 16 |

Lösungsvorschlag



(b) Der folgende rekursive Algorithmus sei gegeben:

Pseudo-Code / Pascal

```

procedure magic(i, n : integer) : boolean
begin
  if (i > (n - 2) / 2) then
    return true;
  endif
  if (A[i] <= A[2 * i + 1] and A[i] <= A[2 * i + 2] and
    magic(2 * i + 1, n) and magic(2 * i + 2, n)) then
    return true;
  endif
  return false;
end
  
```

Java-Implementation

```
public static boolean magic(int i, int n) {
    System.out.println(String.format("i: %s n: %s", i, n));
    if (i > (n - 2) / 2) {
        return true;
    }
    if (A[i] <= A[2 * i + 1] && A[i] <= A[2 * i + 2] && magic(2 * i + 1, n) && magic(2
    ↪ * i + 2, n)) {
        return true;
    }
    return false;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

Gegeben sei folgendes Feld:

| | | | | |
|------|---|---|---|----|
| i | 0 | 1 | 2 | 3 |
| A[i] | 2 | 4 | 6 | 14 |

Führen Sie `magic(0,3)` auf dem Feld aus. Welches Resultat liefert der Algorithmus zurück?

Lösungsvorschlag

true

- (c) Wie nennt man die Eigenschaft, die der Algorithmus `magic` auf dem Feld A prüft? Wie lässt sich diese Eigenschaft formal beschreiben?

Lösungsvorschlag

Die sogenannte „Haldeneigenschaft“ bzw. „Heap-Eigenschaft“ einer Min-Halde. Der Schlüssel eines jeden Knotens ist kleiner (oder gleich) als die Schlüssel seiner Kinder.

Ein Baum erfüllt die Heap-Eigenschaft bezüglich einer Vergleichsrelation „ $>$ “ auf den Schlüsselwerten genau dann, wenn für jeden Knoten u des Baums gilt, dass $u_{\text{wert}} > v_{\text{wert}}$ für alle Knoten v aus den Unterbäumen von u .

- (d) Welche Ausgaben sind durch den Algorithmus `magic` möglich, wenn das Eingabefeld aufsteigend sortiert ist? Begründen Sie Ihre Antwort.

Lösungsvorschlag

true. Eine sortierte aufsteigende Zahlenfolge entspricht den Haldeneigenschaften einer Min-Heap.

- (e) Geben Sie zwei dreielementige Zahlenfolgen (bzw. Felder) an, eine für die `magic(0,2)` den Wert **true** liefert und eine, für die `magic(0,2)` den Wert **false** liefert.

Lösungsvorschlag

```
A = new int[] { 1, 2, 3 };
System.out.println(magic(0, 2)); // true

A = new int[] { 2, 1, 3 };
System.out.println(magic(0, 2)); // false
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

- (f) Betrachten Sie folgende Variante `almostmagic` der oben bereits erwähnten Prozedur `magic`, bei der die Anweisungen in Zeilen 3 bis 5 entfernt wurden:

Pseudo-Code / Pascal

```
procedure almostmagic(i, n : integer) : boolean
begin
  // leer
  // leer
  // leer
  if (A[i] <= A[2 * i + 1] and A[i] <= A[2 * i + 2] and
      magic(2 * i + 1, n) and magic(2 * i + 2, n)) then
    return true;
  endif
  return false;
end
```

Java-Implementation

```
public static boolean almostmagic(int i, int n) {
  System.out.println(String.format("i: %s n: %s", i, n));
  if (A[i] <= A[2 * i + 1] && A[i] <= A[2 * i + 2] && magic(2 * i + 1, n) && magic(2
    ↪ * i + 2, n)) {
    return true;
  }
  return false;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

Beschreiben Sie die Umstände, die auftreten können, wenn `almostmagic` auf einem Feld der Größe `n` aufgerufen wird. Welchen Zweck erfüllt die entfernte bedingte Anweisung?

Lösungsvorschlag

Wird die Prozedur zum Beispiel mit `almostmagic(0, n + 1)` aufgerufen, kommt es zu einem sogenannten „Array-Index-Out-of-Bounds“ Fehler, d. h. die Prozedur will auf Index des Feldes zugreifen, der im Feld gar nicht existiert. Die drei zusätzlichen Zeilen in der Methode `magic` bieten dafür einen Schutz, indem sie vor den Index-Zugriffen auf das Feld `true` zurückgeben.

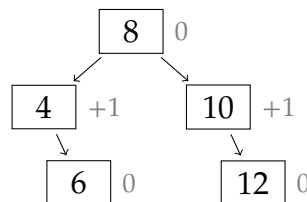
```
// A = new int[] { 1, 2, 3 };
```

```
// System.out.println(almostmagic(0, 4)); // Exception in thread "main"
↪ java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length
↪ 3
```

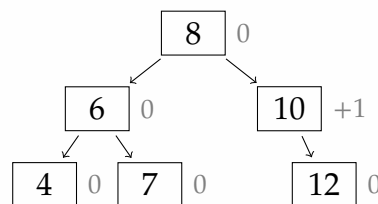
Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2020/herbst/Baum.java)

- (g) Fügen Sie jeweils den angegebenen Wert in den jeweils angegebenen AVL-Baum mit aufsteigender Sortierung ein und zeichnen Sie den resultierenden Baum vor möglicherweise erforderlichen Rotationen. Führen Sie danach bei Bedarf die erforderliche(n) Rotation(en) aus und zeichnen Sie dann den resultierenden Baum. Sollten keine Rotationen erforderlich sein, so geben Sie dies durch einen Text wie „keine Rotationen nötig“ an.

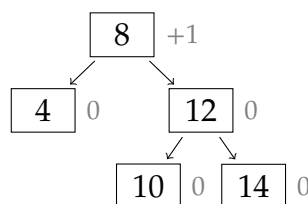
- (i) Wert 7 einfügen



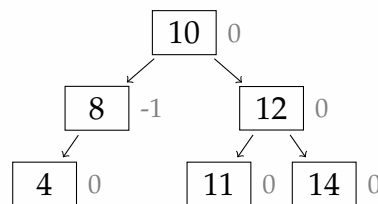
Lösungsvorschlag



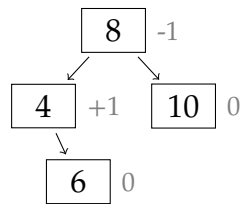
- (ii) Wert 11 einfügen



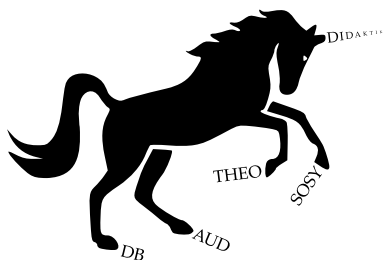
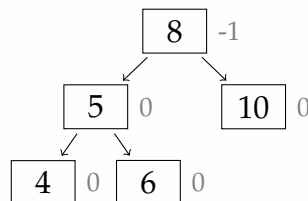
Lösungsvorschlag



- (iii) Wert 5 einfügen



Lösungsvorschlag



Die Bschlangaul-Sammlung

Hermine Bschlangauland 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 TeX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: <https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Staatsexamen/66115/2020/09/Thema-1/Teilaufgabe-2/Aufgabe-2.tex>