

66112 Frühjahr 2005

Automatentheorie / Komplexität / Algorithmen (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht

Thema Nr. 1	3
Aufgabe 14 [Klasse „DoublyLinkedList“]	3
Thema Nr. 2	8
Aufgabe 3: Hashing [Hashing mit Modulo 7]	8



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.

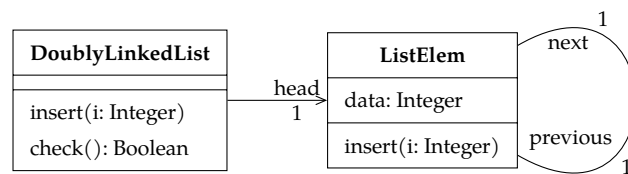


Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Thema Nr. 1

Aufgabe 14 [Klasse „DoublyLinkedList“]

Betrachten Sie folgendes Klassendiagramm, das doppelt-verkettete Listen spezifiziert. Die Assoziation `head` zeigt auf das erste Element der Liste. Die Assoziationen `previous` und `next` zeigen auf das vorherige bzw. folgende Element.



Implementieren Sie die doppelt-verketteten Listen in einer geeigneten objektorientierten Sprache (z. B. Java oder C++), das heißt:

- (a) Implementieren Sie die Klasse `ListElem`. Die Methode `insert` ordnet eine ganze Zahl `i` in eine aufsteigend geordnete doppelt-verkettete Liste `l` an die korrekte Stelle ein. Sei z. B. das Objekt `l` eine Repräsentation der Liste `[0, 2, 2, 6, 8]` dann liefert `l.insert(3)` eine Repräsentation der Liste `[0, 2, 2, 3, 6, 8]`.

```
public class ListElem {
    private int data;
    private ListElem previous;
    private ListElem next;

    public ListElem(int i) {
        data = i;
    }

    public ListElem() {
    }

    public void insert(int i) {
        ListElem newElement = new ListElem(i);
        if (i <= data) {
            if (previous != null) {
                newElement.next = this;
                newElement.previous = previous;
                previous.next = newElement;
                previous = newElement;
            } else {
                newElement.next = this;
                previous = newElement;
            }
        } else {
            if (next != null) {
```

```

        next.insert(i);
    } else {
        newElement.previous = this;
        next = newElement;
    }
}

public ListElem getPrevious() {
    return previous;
}

public ListElem getNext() {
    return next;
}

public int getData() {
    return data;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java](https://github.com/orgs/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java)

- (b) Implementieren Sie die Klasse `DoublyLinkedList`, wobei die Methode `insert` eine Zahl `i` in eine aufsteigend geordnete Liste einordnet. Die Methode `check` überprüft, ob eine Liste korrekt verkettet ist, d.h. für jedes `ListElem`-Objekt `o`, das über den `head` der Liste erreichbar ist, der Vorgänger des Nachfolgers von `o` gleich `o` ist.

```
public class DoublyLinkedList {
    private ListElem head;

    public DoublyLinkedList() {
    }

    public void insert(int i) {
        if (head != null) {
            // Immer einen neue Zahl einfügen, nicht nur wenn die Zahl kleiner ist als head.
            head.insert(i);
            // Es muss kleiner gleich heißen, sonst können mehrer gleiche Zahlen am Anfang
            // nicht eingefügt werden.
            if (i <= head.getData()) {
                head = head.getPrevious();
            }
        } else {
            head = new ListElem(i);
        }
    }

    public boolean check() {
        ListElem current = head;
        while (current.getNext() != null) {
            if (current.getNext().getPrevious() != current) {
                return false;
            }
        }
    }
}
```

```

        } else {
            current = current.getNext();
        }
    }
    return true;
}

public ListElem getHead() {
    return head;
}

public static void main(String[] args) {
    DoublyLinkedList list = new DoublyLinkedList();
    // int[] numbers = new int[] { 1 };
    // int[] numbers = new int[] { 1, 1, 1, 1, };
    // int[] numbers = new int[] { 1, 1, 1, 2, };
    // int[] numbers = new int[] { 2, 1, 1, 1, };
    // int[] numbers = new int[] { 2, 1 };
    int[] numbers = new int[] { 0, 2, 2, 6, 8, 4 };
    for (int number : numbers) {
        list.insert(number);
    }
    list.insert(3);

    ListElem current = list.getHead();
    while (current.getNext() != null) {
        System.out.println(current.getData());
        current = current.getNext();
    }
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java)

Thema Nr. 2

Aufgabe 3: Hashing [Hashing mit Modulo 7]

Gegeben seien die folgenden Zahlen: 7, 4, 3, 5, 0, 1

- (a) Zeichnen Sie eine Hash-Tabelle mit 8 Zellen und tragen Sie diese Zahlen genau in der oben gegebenen Reihenfolge in Ihre Hash-Tabelle ein. Verwenden Sie dabei die Streufunktion $f(n) = n^2 \bmod 7$ und eine Kollisionsauflösung durch lineares Sondieren.

Lösungsvorschlag

$$f(7) = 7^2 \bmod 7 = 49 \bmod 7 = 0$$

$$f(4) = 4^2 \bmod 7 = 16 \bmod 7 = 2$$

$$f(3) = 3^2 \bmod 7 = 9 \bmod 7 = 2 \text{ lineares Sondieren: } +1 = 3$$

$$f(5) = 5^2 \bmod 7 = 25 \bmod 7 = 4$$

$$f(0) = 0^2 \bmod 7 = 0 \bmod 7 = 0 \text{ lineares Sondieren: } +1 = 1$$

$$f(1) = 1^2 \bmod 7 = 1 \bmod 7 = 1 \text{ lineares Sondieren: } -1 = 0, -1 = 7$$

0	1	2	3	4	5	6	7
7	0	4	3	5			1

- (b) Welcher Belegungsfaktor ist für die Streutabelle und die Streufunktion aus Teilaufgabe a zu erwarten, wenn sehr viele Zahlen eingeordnet werden und eine Kollisionsauflösung durch Verkettung (verzeigerte Listen) verwendet wird? Begründen Sie Ihre Antwort kurz.

Lösungsvorschlag

Der Belegungsfaktor berechnet sich aus der Formel:

$$\text{Belegungsfaktor} = \frac{\text{Anzahl tatsächlich eingetragenen Schlüssel}}{\text{Anzahl Hashwerte}}$$

Der Belegungsfaktor steigt kontinuierlich, je mehr Zahlen in die Streutabelle gespeichert werden.

Die Streufunktion legt die Zahlen nur in die Buckets 0, 1, 2, 4.