

# Grundlagen: Algorithmen und Datenstrukturen

## Tutorium 1

Tobias Eppacher

School of Computation, Information and Technology

5. Mai 2025

# Table of contents

Organisation

Aufgaben

E-Aufgaben

Hausaufgaben

# Organisation



## Leistungsnachweis



- **Klausur** (voraussichtlich) 11.08.2025
  - schriftliche Prüfung
  - Dauer: 90 Minuten
  - Erlaubtes Hilfsmittel: hand-beschriebenes DIN A4 Blatt
- **Wiederholungs-Klausur** (voraussichtlich) Oktober 2024
- **Warnung:** für IN0007 in München/Garching anmelden, nicht für Heilbronn (INH0008)! Auch dort gibt es im SS eine Vorlesung "Grundlagen: Algorithmen und Datenstrukturen".
- Vorbereitung durch **aktive Teilnahme** an Vorlesung und Übungsbetrieb


**Abbildung:** Aus den Vorlesungsfolien

# Zulip

## Kommunikation über Zulip (<https://zulip.in.tum.de/>)

# Kanalgruppen > Kanalgruppen   23. MÄRZ 2021


























 **TUM CS Bot**  EDITIERT 15:30

Hi! 

I have the pleasure to announce some channel groups here.

You may subscribe to a channel group in order to be automatically subscribed to all channels belonging to that group. Also, you will be kept updated when new channels are added to the group.

Just react to this message with the emoji of the channel group you like to subscribe to. Remove your emoji to unsubscribe from this group. (1, 2)

| Course | Emoji   |  | Course        | Emoji   |  | Course    | Emoji   |
|--------|---|--|---------------|---|--|-----------|---|
| ASP    |  |  | FDS           |  |  | ITSec     |  |
| BMT    |  |  | FPV           |  |  | Lambda    |  |
| CPP    |  |  | GAD           |  |  | Logic     |  |
| DS     |  |  | GBS           |  |  | Memes     |  |
| ECG    |  |  | GBS C-Vorkurs |  |  | PGdP      |  |
| EIST   |  |  | GDB           |  |  | Semantics |  |
| ERA    |  |  | GRA/S         |  |  | THEO      |  |
| ERDB   |  |  | GRA24W        |  |  |           |   |
| F:A+DS |  |  | GRNVS         |  |  |           |   |



# Zulip

## Tutoriumschannels:

1. Montag 14:00: GAD Tutorium A-14-4
2. Montag 16:00: GAD Tutorium A-16-5

Zulip DM an mich (Tobias Eppacher) für nicht öffentliche Fragen oder Feedback :)

## Aufgabe 1.1 - Division

**Input:** Ziffer[]( $x_1, x_2, \dots, x_n$ ), Ziffer  $y$

```
1: int  $i := 1$ 
2: Ziffer  $x_0 := 0$ 
3: while  $i \leq n$  do
4:   if  $x_{i-1} > 0$  then
5:     Ziffer  $z_i := g(x_{i-1} \oplus x_i, y)$ 
6:      $x_i := r(x_{i-1} \oplus x_i, y)$ 
7:      $x_{i-1} := 0$ ;
8:   else
9:     Ziffer  $z_i := g(x_i, y)$ 
10:     $x_i := r(x_i, y)$ 
11:   end if
12:    $i := i + 1$ 
13: end while
```

**Grundoperationen:**

- ▶  $g(x, y)$   
Ganzzahldivision
- ▶  $r(x, y)$  Rest
- ▶  $+$  Addition
- ▶  $:=$  Zuweisung
- ▶  $>, <, \dots$  Vergleich

*Indexberechnung keine  
Grundoperation*

## Aufgabe 1.1 - Division

Wie viele Grundoperationen hat die Schulmethode in unserem Rechenmodell im schlimmsten Fall, wenn man eine Zahl mit  $n$  Ziffern ganzzahlig durch eine Ziffer teilt? Betrachten Sie bei Ihrer Lösung jeweils jede Zeile im Pseudocode.

## Aufgabe 1.1 - Division

```
1: int  $i := 1$ 
2: Ziffer  $x_0 := 0$ 
3: while  $i \leq n$  do
4:   if  $x_{i-1} > 0$  then
5:     Ziffer  $z_i := g(x_{i-1} \oplus x_i, y)$ 
6:      $x_i := r(x_{i-1} \oplus x_i, y)$ 
7:      $x_{i-1} := 0$ ;
8:   else
9:     Ziffer  $z_i := g(x_i, y)$ 
10:     $x_i := r(x_i, y)$ 
11:   end if
12:    $i := i + 1$ 
13: end while
```



## Aufgabe 1.2 - Induktion

### Induktionsbeweis

Zu beweisende Aussage, z.B.  $\forall n \in \mathbb{N} : P(n)$

1. Induktionsanfang  
zeige Aussage für Basisfall, z.B.  $n = 1$  für  $\mathbb{N}$
2. Induktionsannahme  
z.B.  $P(n)$  gilt für ein beliebiges, frei gewähltes  $n \in \mathbb{N}$
3. Induktionsschritt  
zeige  $P(n + 1)$  mithilfe der Annahme  $P(n)$

## Aufgabe 1.2 (a)

Zeigen Sie für alle natürlichen Zahlen  $n \geq 1$  mittels Induktion die folgende Behauptung:

Die Summe der ersten  $n$  ungeraden Zahlen ist  $n^2$

(als Formel:  $\sum_{i=1}^n (2i - 1) = n^2$ )

Kennzeichnen bzw. benennen Sie in Ihrem Beweis den

Induktionsanfang, die Induktionsvoraussetzung und den Induktionsschritt.

## Aufgabe 1.2 (a)

## Aufgabe 1.2 (b)

Zeigen Sie

$$F_{n+1}F_{n-1} - F_n^2 = (-1)^n,$$

wobei  $F_n$  die  $n$ -te Fibonaccizahl nach der rekursiven Definition

$F_n = F_{n-1} + F_{n-2}$  mit den Anfangswerten  $F_0 = 0$  und  $F_1 = 1$  ist.

## Aufgabe 1.2 (b)

## Aufgabe 1.3 - Bubblesort mit Listen

```
public class List {  
  
    private static class Node {  
        private int data;  
  
        public int getData() {  
            return data;  
        }  
  
        public void setData(int data) {  
            this.data = data;  
        }  
  
        private Node next;  
  
        public Node getNext() {  
            return next;  
        }  
  
        public Node(int data, Node next) {  
            this.data = data;  
            this.next = next;  
        }  
    }  
}
```

```
    private Node head;  
    private int size;  
  
    public int getSize() {  
        return size;  
    }  
  
    public List() {}  
  
    public void prepend(int data) {  
        head = new Node ( data , head );  
        size ++;  
    }  
  
    public int get(int index) {  
        Node it = head;  
        while(index != 0) {  
            index--;  
            it = it.getNext ();  
            if(it == null)  
                throw new RuntimeException("Out of bounds");  
        }  
        return it.getData();  
    }  
}
```

## Aufgabe 1.3 - Bubblesort mit Listen

```
public void swap(int indexFirst, int indexSecond) {
    if(head == null) {
        throw new RuntimeException("Out of bounds");
    }

    if(indexFirst > indexSecond) {
        swap(indexSecond, indexFirst);
        return;
    }

    int distance = indexSecond - indexFirst;
    Node it_first = head;

    while(indexFirst != 0) {
        indexFirst--;
        it_first = it_first.getNext();
        if(it_first == null) throw new RuntimeException("Out of bounds");
    }

    Node it_second = it_first;
    while(distance != 0) {
        distance--;
        it_second = it_second.getNext();
        if(it_second == null) throw new RuntimeException("Out of bounds");
    }

    int temp = it_second.getData();
    it_second.setData(it_first.getData());
    it_first.setData(temp);
}
```

## Aufgabe 1.3 - Bubblesort mit Listen

### Implementierung

1. Welche der Methoden sind langsam, welche schnell? Wieso?
2. Die *swap*-Methode ruft sich selbst auf. Wie nennt man Methoden, die sich so verhalten? Welche Motivationen gibt es, derart zu programmieren? Welche Nachteile hat so ein Ansatz?



## Aufgabe 1.3 - Bubblesort mit Listen

```
void bubblesort ( List l) {  
    for(int i = 0; i < l.getSize(); i++)  
        for(int j = 1; j < l.getSize() - i; j++)  
            if(l.get(j - 1) > l.get(j))  
                l.swap(j - 1, j);  
}
```

1. Die Methode hat keinen Rückgabewert; funktioniert sie dennoch? Wenn ja, wieso?
2. Wie funktioniert der Algorithmus, warum liefert er stets ein sortiertes Ergebnis?

## Aufgabe 1.3 - Bubblesort mit Listen

```
void bubblesort (List l) {  
    for(int i = 0; i < l.getSize(); i++)  
        for(int j = 1; j < l.getSize() - i; j++)  
            if(l.get(j - 1) > l.get(j))  
                l.swap(j - 1, j);  
}
```

- Wie oft ungefähr wird eine Liste der Größe  $n$  durchlaufen?  
*Berücksichtigen Sie bei Ihrer Antwort nur die Implementierung des Bubblesort-Algorithmus.*
- Eignet sich unsere Implementierung einer Liste hier besonders gut oder besonders schlecht?  
*Ziehen Sie nun die Implementierung der Listenmethoden in Ihre Laufzeitüberlegung mit ein.*

## Aufgabe 1.3 - Bubblesort mit Listen

Ein bisschen PGdP :)

1. Was hat es mit der Schachtelung der Klassen auf sich?
2. Wozu wurde der Wrapper `List` implementiert, statt den Benutzer direkt auf Knoten arbeiten zu lassen? Was hat dies mit *abstrakten Datentypen* zu tun?
3. Welchen Zweck erfüllt `throw`? Wieso ist dies besser, als vordefinierte Fehlerwerte zurückzugeben?
4. Vergleichen Sie die Implementierung der Funktionen `get` und `swap`. Was fällt Ihnen dabei auf? Wann kann das ein Problem sein, und wie lässt sich dieses vermeiden?

# E-Aufgaben

- ▶ Aufgabe 1.4 - Tiefe Bäume
  - ▶ Definitionen zum Thema Bäume
  - ▶ Übung zu Induktion
- ▶ Aufgabe 1.5 - Schwere Induktion
  - ▶ Schwerer Induktionsbeweis
  - ▶ Ebenfalls gute Übung

# Hausaufgaben

- ▶ Hausaufgaben auf Artemis  
(<https://artemis.tum.de/>)
- 1. Hausaufgabe 1: Labyrinth (Deadline - 07.05.2025)
- 2. Hausaufgabe 2: Binäre Suche (Deadline - 14.05.2025)

## Fragen?

- ▶ Nach Übung gerne bei mir melden
- ▶ Tutoriumschannel oder DM an mich auf Zulip
- ▶ Vorlesungschannels von GAD auf Zulip (insbesondere bei Hausaufgaben)

## Feedback oder Verbesserungsvorschläge?

Gerne nach dem Tutorium mit mir quatschen oder DM auf Zulip

## Bis nächste Woche!