

Einführung in die Programmierung  
 WS 2018/19

Übungsblatt 3: Vollständige Induktion, p-adische Zahlendarstellung

Besprechung: 12.11 - 16.11.2018

**Aufgabe 3-1** *Vollständige Induktion mit Ungleichungen*

- (a) Zeigen Sie:  $n^2 > 2n + 1$  für alle  $3 \leq n \in \mathbb{N}$ .

**Lösungsvorschlag:**

(IA)  $n = 3 : 3^2 = 9 > 7 = 2 \cdot 3 + 1$

(IV) Es gibt ein  $n \in \mathbb{N}, n \geq 3$ , sodass  $n^2 > 2n + 1$  gilt.

(IS)  $n \rightarrow n + 1$ :

$$(n+1)^2 = n^2 + 2n + 1 \stackrel{(IV)}{>} (2n+1) + 2n + 1 = 2n + 2n + 2 \stackrel{\text{da } n \geq 3}{>} 2n + 3 = 2(n+1) + 1$$

- (b) Zeigen Sie:  $2^n > n^2$  für alle  $5 \leq n \in \mathbb{N}$  (Sie dürfen 3-1-a hierfür als bewiesen annehmen).

**Lösungsvorschlag:**

(IA)  $n = 5 : 2^5 = 32 > 25 = 5^2$

(IV) Es gibt ein  $n \in \mathbb{N}, n \geq 5$ , sodass  $2^n > n^2$  gilt.

(IS)  $n \rightarrow n + 1$ :

$$2^{n+1} = 2 \cdot 2^n \stackrel{(IV)}{>} 2 \cdot n^2 = n^2 + n^2 \stackrel{(3-1/a)}{>} n^2 + 2n + 1 = (n+1)^2$$

- (c) Zeigen Sie, dass für alle  $x \in \mathbb{R}, x \geq -1$  und für alle  $n \in \mathbb{N}$  gilt:  $(1+x)^n \geq 1 + nx$

**Lösungsvorschlag:**

Wichtig: Angabe der Induktion nach  $n$ , nicht nach  $x$ .

(IA)  $n = 1 : (1+x)^1 = 1+x = 1+1x$

(IV) Es gibt ein  $n \in \mathbb{N}$ , sodass für alle  $x \in \mathbb{R}, x \geq -1$  gilt:  $(1+x)^n \geq 1 + nx$ .

(IS)  $n \rightarrow n + 1$ :

$$(1+x)^{n+1} = (1+x)^n(1+x) \stackrel{(IV), x \geq -1}{\geq} (1+nx)(1+x) = 1+x+nx+nx^2$$

$$\stackrel{nx^2 \geq 0}{\geq} 1+x+nx = 1+(n+1)x$$

*Hinweis: Da sie hier keine Gleichungen zeigen müssen, ist es manchmal notwendig, die Terme geschickt zu erweitern oder zu verringern.*

**Aufgabe 3-2** *p-adische Zahlendarstellungen*

Die Datei `p-adisch.txt` enthält folgende Tabelle:

$p = 2$	$p = 8$	$p = 10$	$p = 16$
1101	15	13	D
	76		
		26	
1111001			
			7C

Ergänzen Sie die Tabelle so, dass in jeder Zeile die verschiedenen  $p$ -adischen Zahlendarstellungen für die selbe Zahl stehen.

**Lösungsvorschlag:**

$p = 2$	$p = 8$	$p = 10$	$p = 16$
1101	15	13	D
111110	<b>76</b>	62	3E
11010	32	<b>26</b>	1A
<b>1111001</b>	171	121	79
1111100	174	124	<b>7C</b>

**Aufgabe 3-3**      *Karten umdrehen*

Gegeben ist das folgende Ein-Spieler-Spiel. Vor Ihnen liegen  $n \in \mathbb{N}$  Spielkarten verdeckt in einer Reihe nebeneinander auf dem Tisch. Ein Zug besteht immer aus zwei Schritten:

1. Eine beliebige verdeckte Karte wird ausgewählt und herumgedreht, sodass ihre Bildseite zu sehen ist.
2. Die Karte, die direkt rechts neben der gewählten Karte liegt, wird ebenfalls herumgedreht - ungeachtet dessen, ob sie noch verdeckt ist oder nicht. Gibt es dort keine Karte, dann überspringe diesen Schritt.

- (a) Terminiert das Spiel nach endlich vielen Zügen? Begründen Sie schlüssig Ihre Antwort.
- (b) Angenommen, in jedem Zug muss die Karte links von der gewählten Karte statt der rechten Nachbarkarte umgedreht werden. Ändert dies etwas? Begründen Sie.
- (c) Nun erlauben wir, dass sich der Spieler in jedem Zug aussuchen darf, ob er im 2. Schritt die linke oder die rechte Karte umdreht. Was können Sie nun über das Terminierungsverhalten sagen?

*Hinweis: Sie können die binäre Zahlendarstellung als Repräsentation der Kartenreihe nutzen .*

### Lösungsvorschlag:

a) Ja, es terminiert.

Wir können die Reihe von Karten zwischen zwei Spielzügen als Zustand auffassen. Jeder Zustand ist repräsentierbar durch eine natürliche Zahl  $k \geq 0$ . Die Repräsentation des Startzustandes ist, mit „1“ als Zustand einer verdeckten Karte:

$$k_0 = (11 \dots 1)_2 = (2^{n+1} - 1)_{10} \in \mathbb{N}_0$$

Offensichtlich terminiert das Spiel, falls  $k_t = 0 = (00 \dots 0)_2$  erreicht wird. Es ist leicht zu sehen, dass jeder Zustand  $k_j > 0$ , mit  $0 \leq j \leq t$ .

In jedem Schritt wählen wir ein Bit  $b_i = 1$  aus ( $1 \leq i \leq n$ ), setzen dieses auf 0 und setzen, falls  $i \neq n$  gilt,  $b_{i+1}$  auf sein Komplement, also  $1 - b_{i+1}$ . Mit  $0 \leq m < t$  wird aus

$$k_m = (b_1 b_2 \dots b_i b_{i+1} \dots b_n) = (b_1 b_2 \dots 1 b_{i+1} \dots b_n)$$

durch die Spielregel

$$k_{m+1} = (b_1 b_2 \dots (1 - b_i)(1 - b_{i+1}) \dots b_n) = (b_1 b_2 \dots 0(1 - b_{i+1}) \dots b_n)$$

### Lösungsvorschlag:

Egal, ob  $b_{i+1} = 0$  oder  $b_{i+1} = 1$ , es gilt stets  $k_m > k_{m+1}$ .

Der kleinste Schritt dabei wird gemacht, wenn man die Karte  $b_m$  ganz rechts umdreht. Also ist außerdem  $k_m - k_{m+1} \geq 1$ .

Daher terminiert das Spiel mit jeder beliebigen Reihenfolge des Kartendrehens.

Falls „0“ eine verdeckte Karte darstellt, dann wird der Wert des Zustandes immer größer statt kleiner.

b) Ändert man die Strategie und nimmt die linke statt der rechten Karte, so ändert sich nichts. Drehe einfach den Tisch und das Problem ist wieder das Gleiche.

c) Falls man willkürlich wählen darf, so gibt es Konstellationen, die nicht terminieren: Betrachte zum Beispiel 01 und man wählt abwechselnd die linke und rechte Karte als zweite Karte. Nun alterniert das Spiel zwischen 01 und 10.

Offensichtlich gibt es eine Strategie, die zur Terminierung führt. Dies ist aber hier nicht gefragt, da bei Terminierung das Programm für jede Eingabe enden muss.

### Aufgabe 3-4 Java/Benutzerinteraktion

In Aufgabe 0-3 wurde ein Programm kompiliert, das eine feste Ausgabe besaß. Die meisten Programme nutzen aber unterschiedliche Eingaben, um darauf zu reagieren und spezifische Ausgaben zu erzeugen. Dazu benötigen wir eine Benutzereingabe. Es gibt einige Möglichkeiten, dies in Java zu realisieren. Vorerst nutzen wir die Scannerklasse, die einige Eingabemethoden zur Verfügung stellt.

- (a) Erstellen Sie wie in Aufgabe 0-3 eine .java-Datei mit dem Namen **Eingabe.java**. Geben Sie dieser Datei vorerst das Hello-World-Programm als Inhalt. Kompilieren Sie das Programm. Folgende Fehlermeldung wird erscheinen:

```
Eingabe.java:3: error: class HelloWorld is public,
should be declared in a file named HelloWorld.java
public class HelloWorld {
    ^
1 error
```

Java-Dateien enthalten immer genau eine öffentliche Klasse (public class). Ändern Sie daher den

Namen der Klasse von `HelloWorld` zu `Eingabe`. Ändern Sie außerdem die Ausgabe von `"Hello World"` zu `"Wie alt bist du: "`.

- (b) Nach der Ausgabe soll eine ganze Zahl eingelesen werden. Dazu importieren Sie mittels `import java.util.Scanner;` am Dateianfang eine Klasse, die Ihnen dabei hilft, Texteingaben zu verarbeiten. Danach nutzen Sie `Scanner sc = new Scanner(System.in);`, um dem Programm zu erklären, dass es einen Scanner vorbereiten soll, der Eingaben von der Eingabekonzole (`System.in`) erwartet. Dieser heißt `sc`. Anschließend soll der Scanner eine ganze Zahl des Benutzers einlesen: `int alter = sc.nextInt();`. Geben Sie auf geeignete Weise die Eingabe in der Konsole aus, z.B.: `"Dein Alter ist 13."`. Das Grundgerüst für diese Aufgabe liegt als `Eingabe.java` zum Download bereit.
- (c) Nun reagieren Sie altersgerecht auf die Eingabe. Personen mit einem Alter über oder gleich 16 haben Zugriff auf Bier. Allen anderen wird dies verweigert. Dazu brauchen wir ein Konstrukt zur Abfrage von Bedingungen. Dabei gibt es im Programm eine Verzweigung, deren erster Block nur durchlaufen wird, wenn die Bedingung erfüllt ist. Ansonsten wird entweder das Konstrukt ganz übersprungen, oder der zweite Block durchgearbeitet. Im folgenden sehen Sie die Umsetzung in Java:

```
if(Bedingung) {  
    //Wenn Bedingung erfuehlt ist, dann tue alles in diesem Block.  
}  
else {  
    //Wenn Bedingung nicht erfuehlt ist, dann tue dies hier.  
}
```

Formulieren Sie eine geeignete Bedingung zur Altersabfrage und geben Sie in beiden Anweisungsblöcken eine entsprechende Ausgabe. Beispielausgaben können wie folgt aussehen:

```
Wie alt bist du: 13  
Kein Bier fuer dich.
```

```
Wie alt bist du: 28  
Hier ist dein Bier.
```

- (d) Für die Motivierten: Fragen Sie bei positiver Altersabfrage nach, wieviel Biere  $n$  gewünscht sind, und geben sie dann  $n$ -mal das Wort Bier aus. Falls  $n$  negativ ist, verweigern Sie die Bierausgabe. Dazu benötigen Sie zusätzlich zum `if`-Konstrukt noch eine `while`-Schleife.

### Lösungsvorschlag:

```
import java.util.Scanner;
public class Eingabe {
    public static void main(String[] args) {
        System.out.print("Wie alt bist du: ");    //a)

        Scanner sc = new Scanner(System.in);
        int alter = sc.nextInt();
        System.out.print("Dein Alter ist " + alter);    //b)

        if (alter >= 16) {                            //c)
            System.out.println("Hier ist dein Bier.");
        }
        else {
            System.out.println("Kein Bier fuer dich.");
        }

        System.out.println("Wie viele Bier moechtest du?");    //d)
        int anzahl = sc.nextInt();
        while(anzahl > 0){
            System.out.print("Bier");
            anzahl = anzahl - 1;
        }
    }
}
```