Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M.Sc. Stefan Karch
M.Sc. Mariia Sukhova

04.11.2022

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 4

Bearbeitungszeitraum: 21.11.2022 – 02.12.2022

Aufgabe 1 *Pythagoräische Tripel*

Ein Tripel (a, b, c), welches aus drei natürlichen Zahlen a, b und c besteht, wird ein *pythagoräisches Tripel* genannt, falls

$$a^2 + b^2 = c^2 \tag{1}$$

gilt. Schreiben Sie mit Java ein Programm mit dem Namen `PythagTripel`, welches durch reines Ausprobieren alle pythagoräischen Tripel bis zu einer oberen Grenze n findet. Gesucht sind also alle Tripel (a, b, c), mit $1 \leq a \leq b \leq c \leq n$, so dass die Gleichung (1) gilt. Gehen Sie beim Lösen dieser Aufgabe wie folgt vor:

- Erstellen Sie in der Programmklasse zunächst eine `main`–Methode. Lesen Sie in dieser Methode eine Zahl n vom Typ **`int`** von der Konsole ein und speichern Sie diese ab.
- Verwenden Sie drei ineinander geschachtelte **`for`**–Schleifen, um alle möglichen Werte für a, b und c zu durchlaufen. Prüfen Sie in der innersten Schleife mit einer **`if`**-Anweisung jeweils nach, ob die Gleichung (1) erfüllt ist. Wenn ja, so geben Sie a, b und c auf der Konsole aus.
- Testen Sie Ihr Programm für verschiedene Werte von n.

Musterlösung: Die ersten drei pythagoräischen Tripel lauten $(3, 4, 5)$, $(5, 12, 13)$, $(6, 8, 10)$

Aufgabe 2 *Der euklidische Algorithmus*

Der euklidische Algorithmus ist ein Verfahren zum bestimmen des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen z_1 und z_2. Er spielt eine große Rolle in der Zahlentheorie und wird (in erweiterter Form) bei einigen Verschlüsselungsverfahren eingesetzt.

Der euklidische Algorithmus:
Es seien $z_1, z_2, dividend, rest, temp, ggT \in \mathbb{N}, z_1 > z_2$.

$dividend$:= z_1; $rest$:= z_2;
solange $(rest \neq 0)$
    $temp$:= $rest$;
    $rest$:= $dividend \bmod rest$;
    $dividend$:= $temp$;
Schleifenende
ggT := $temp$;

Schreiben Sie ein Java-Programm, welches den größten gemeinsamen Teiler (ggT) zweier natürlicher Zahlen bestimmt. Darüber hinaus soll das Programm das kleinste gemeinsame Vielfache (kgV) der beiden Zahlen mit Hilfe des ggT bestimmen. Gehen Sie dazu folgendermaßen vor:

(a) Erstellen Sie eine öffentliche Klasse `EuklidAlg` mit einer `main`-Methode. Lassen Sie zunächst den Benutzer die Werte z_1 und z_2 für zwei entsprechende **`int`**-Variablen eingeben. Falls $z_1 < z_2$, so sollen die Werte der beiden Variablen vertauscht werden.

(b) Lassen Sie Ihr Programm den oben beschriebenen euklidischen Algorithmus durchführen. Deklarieren Sie dazu die Variablen `dividend`, `rest`, `temp` und `ggT` vom Typ **`int`**. Benutzen Sie für den Algorithmus eine `while`–Schleife:

```
while (rest != 0) {
  // Hier steht eine Folge von Anweisungen....
}
```

Diese Schleife führt die Anweisungen zwischen den geschweiften Klammern solange aus, bis die Bedingung in den runden Klammern nicht mehr erfüllt ist.

(c) Geben Sie nach erfolgter Berechnung den ggT und das kgV mit erläuterndem Text aus. Dabei ergibt sich das kgV aus $kgV = \frac{z_1 z_2}{ggT}$.

Testen Sie ihr Programm mit den Zahlenpaaren $(12, 3),\ (12, 13),\ (9, 15)$.

Hinweis: Der ggT für $(12, 3)$ ist 3 und das kgV ist 12.

Hinweis: Der Compiler achtet darauf, dass jede Variable immer einen Wert besitzt, bevor sie in einer Rechnung verwendet wird. Dabei geht er davon aus, dass **`if`**-Abfragen und Schleifen auch übersprungen werden können. Daher kann es hier nötig sein, der Variable `temp` vor der Berechnung einen Wert zuzuweisen, damit die Anweisung
ggT := $temp$;
nicht zu einem Fehler führt.

Aufgabe 3 (Pflichtaufgabe) *Das Ziegenproblem*

Das *Ziegenproblem* ist ein bekanntes Beispiel dafür, dass der spontanen Intuition bei stochastischen Fragestellungen häufig nicht zu trauen ist. Es lautet wie folgt:

> Sie sind Kandidat einer Spielshow. Vor Ihnen befinden sich drei Tore, deren Inhalt Sie gewinnen können. Hinter einem zufällig gewählten Tor befindet sich ein Auto, hinter den anderen beiden Toren jeweils eine Ziege.
>
> Sobald Sie sich für ein Tor entscheiden wählt der Moderator zuverlässig aus den beiden verbliebenen Toren ein Ziegentor aus und öffnet es, sodass nun nur noch zwei Tore zur Auswahl stehen. Er stellt Sie vor die Wahl: Möchten Sie bei ihrer ursprünglichen Wahl bleiben oder das jeweils andere Tor wählen?

Das Ziegenproblem besteht nun aus folgender Frage:

> Stellt es für den Gewinn des Autos einen Vorteil oder Nachteil dar, das Angebot des Moderators anzunehmen?

Das Problem lässt sich durch elementare Methoden der Stochastik lösen. Wir wollen stattdessen eine Simulation duchführen um einen Hinweis auf das richtige Ergebnis zu erhalten oder unsere Vermutung zu überprüfen. Dazu führen wir das Experiment jeweils mit beiden Strategien mehrmals durch. Die *relative Trefferhäufigkeit*

$$\hat{p} = \frac{\text{Anzahl der Treffer}}{\text{Anzahl der Versuche}}$$

einer Strategie wird uns näherungsweise die Gewinnchance eines Autos liefern, insofern wir den Gewinn eines Autos als "Treffer" werten.

Schreiben Sie ein Java-Programm welches die Situation des Ziegenproblems jeweils mit den Strategien "Tor behalten" und "Tor wechseln" simuliert und in beiden Fällen die relative Trefferhäufigkeit berechnet. Gehen Sie dazu folgendermaßen vor:

- Erstellen Sie eine öffentliche Klasse `Ziegenproblem.java` sowie deren `main`-Methode. Erzeugen Sie zunächst eine ganzzahlige Variable `versuche`, welche die Anzahl der durchzuführenden Versuche angibt. Initialisieren Sie diese Variable mit dem Wert `10000`.
- Simulieren Sie das Ziegenproblem mit der Strategie "Tor behalten". Erstellen Sie dazu eine **`int`**-Variable `treffer` die Sie mit 0 initialisieren. Schreiben Sie daraufhin eine **`for`**-Schleife, deren Anzahl an Schleifendurchläufen dem Inhalt von `versuche` entspricht. In dieser Schleife werden folgende Schritte durchgeführt:
  - Erzeugen Sie zwei ganzzahlige Variablen `tor_auto` und `tor_wahl`. Diese Variablen sollen den Wert des Tores (Tor 1, Tor 2 oder Tor 3) abspeichern. Erzeugen Sie dazu zufällige ganze Zahlen $w, x \in \{1, 2, 3\}$ und weisen Sie deren Werte jeweils den Variablen `tor_auto` und `tor_wahl` zu.

    Hinweis: Eine zufällige natürliche Zahl im Bereich $\{a, ..., b-1\}$ erhalten Sie mit dem Ausdruck

    ```
    (int)((a+(b-a)*Math.random())
    ```
  - Ein Treffer entspricht dem Ereignis, dass das Tor `tor_auto` und das von `tor_wahl` übereinstimmen. Erhöhen Sie in diesem Fall den Wert von `treffer`.

  Berechnen Sie nun die relative Trefferhäufigkeit dieser Strategie und geben Sie diese auf der Konsole aus.

- Simulieren Sie nun das Ziegenproblem mit der Strategie "Tor wechseln". Initialisieren Sie wie im vorherigen Aufgabenteil die Variable `treffer` mit 0 und erzeugen Sie eine entsprechende **`for`**-Schleife. In dieser werden folgende Schritte durchgeführt:
  - Belegen Sie die Variablen `tor_auto` und `tor_wahl` zufällig wie zuvor.
  - Erzeugen Sie eine ganzzahlige Variable `tor_offen` mit einem zufälligen Wert $y \in \{1,2,3\} \setminus \{\texttt{tor\_auto}, \texttt{tor\_wahl}\}$. Das bedeutet, dass das offene Tor weder dem gewählten noch dem Tor mit dem Hauptgewinn entsprechen darf.

    Hinweis: Eine einfache Möglichkeit einen zufälligen Wert aus $A \setminus B$ zu ziehen ist:

    Ziehe $y \in A$ wiederholt bis $y \notin B$
  - Erzeugen Sie eine ganzzahlige Variable `tor_alternative` mit dem Wert $z \in \{1,2,3\} \setminus \{\texttt{tor\_offen}, \texttt{tor\_wahl}\}$. Dieser Schritt entspricht dem Vorgang, wenn die Wahl des Tors nochmal geändert wird. Das alternative Tor darf also nicht das schon Geöffnete und nicht die ursprüngliche Wahl sein.
  - Erhöhen Sie den Wert von `treffer`, falls das alternative Tor und das Tor des Autos übereinstimmen.

  Berechnen Sie auch hier wieder die relative Trefferhäufigkeit und geben Sie diese auf der Konsole aus.

- Unterscheiden sich die Strategien? Welche ist die bessere?

Hinweis: Achten Sie bei den Berechnungen der relativen Häufigkeiten auf die Regeln der Integerdivision.

Musterlösung: Trefferwahrscheinlichkeit ohne Wechsel: $\frac{1}{3}$
Trefferwahrscheinlichkeit mit Wechsel: $\frac{2}{3}$

Fragen 3 *Das Ziegenproblem*

- Was ist der Unterschied zwischen `System.out.println()` und `System.out.print()`?
- Aus was für Teilen besteht eine **`for`**-Schleife und wie könnten diese ersetzt werden?
- Was ist der Unterschied zwischen einer **`while`**- und einer **`do`**-Schleife?