# Grundbegriffe der Informatik — Aufgabenblatt 12
# Lösungsvorschläge

| | |
|---|---|
| Matr.nr.: | |
| Nachname: | |
| Vorname: | |
| Tutorium Nr.: | Tutor*in: |

Ausgabe: 28. Januar 2021, 12:00 Uhr

Abgabe: 04. Februar 2021, 12:30 Uhr
in dem Holzkasten neben Raum -119
im UG des Info-Gebäudes (50.34)

Lösungen werden nur korrigiert, wenn sie
- handschriftlich erstellt sind (Tablet-Ausdruck erlaubt) und
- mit dieser Seite als Deckblatt
- in der oberen **linken** Ecke zusammengeheftet

**rechtzeitig** abgegeben werden.

Abgaberegeln für Teilnehmer der Online-Tutorien:
- handschriftlich erstellt (lesbare Fotos akzeptiert)
- **rechtzeitig**, mit diesem Deckblatt in **genau einer** PDF-Datei
- direkt an den entsprechenden Tutor abgeben.

---

*Von Tutor*in auszufüllen:*

erreichte Punkte

| | |
|---|---|
| Blatt 12: | / 18 [+1] |
| Blätter 7 – 12: | / 120 [+4] |

---

**Aufgabe 12.1 (0,5 + 1,5 + 2 = 4 Punkte)**

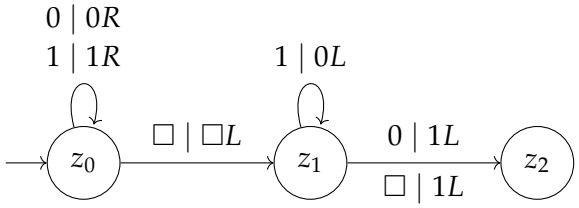
a) Geben Sie eine Turingmaschine $T_{inc}$ (als Zeichnung) an, welche die Eingabe $w \in \{0,1\}^+$ als Binärzahl auffasst und um 1 inkrementiert. Verwenden Sie höchstens 3 Zustände.

b) Geben Sie eine Turingmaschine $T_{copy}$ (als Zeichnung) an, welche die Eingabe $w \in \{0,1\}^*$ hinter ein besonderes Trennsymbol „$-$“ *kopiert*. Am Ende der Berechnung soll das Band (ausschließlich) „$w - w$“ beinhalten. Verwenden Sie höchstens 7 Zustände. Achten Sie auf eine übersichtliche Lösung!
*Anmerkung: Teilpunkte bei Verwendung von mehr als 7 Zuständen.*

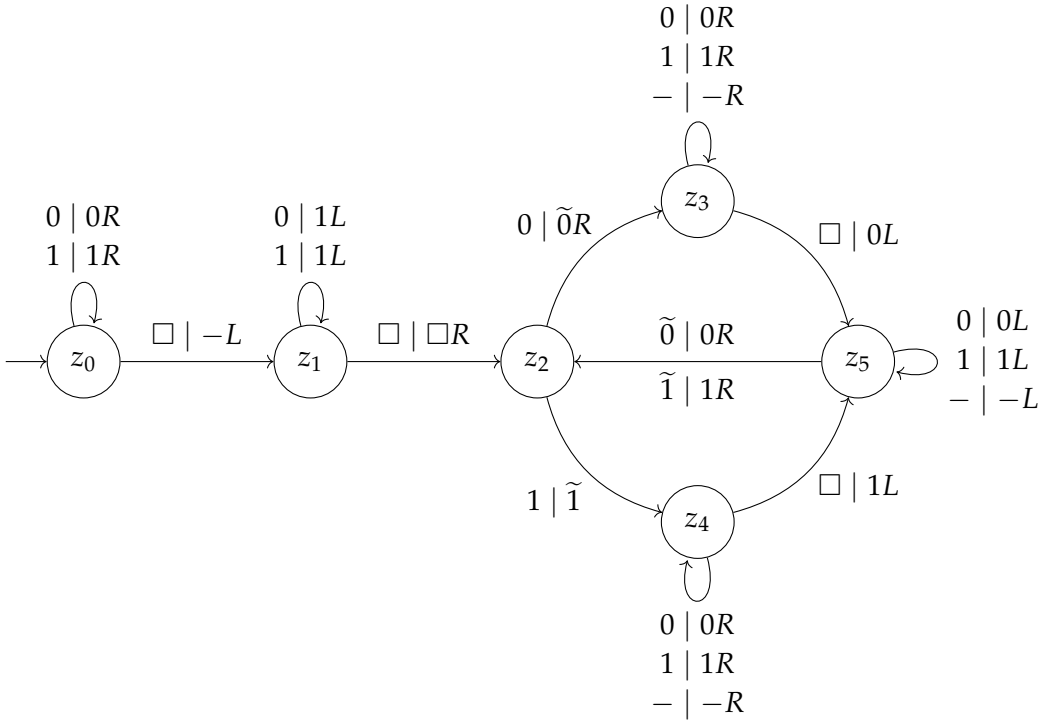
c) Beschreiben Sie textuell, wie eine Turingmaschine $T_{\mathbb{N}_0}$ vorgehen kann, um $\mathbb{N}_0$ (gegebenenfalls in irgend einer *Codierung*) *aufzuzählen*, d.h. **alle** $n \in \mathbb{N}_0$ (ggf. codiert), durch das Trennzeichen „$-$“ getrennt, auf das Band zu schreiben. Dabei soll die Bandbeschriftung nicht durch „$\square$“ unterbrochen werden. Beschreiben die Arbeitsweise der Turingmaschine präzise.

**Lösung 12.1**

a)




b)




c) Da das Bandalphabet von Turingmaschinen endlich sein muss, wählen wir die Binärcodierung und zählen $Repr_2(n) \forall n \in \mathbb{N}_0$ auf. Die in Teilaufgabe a) und b) angegebenen Turingmaschinen $T_{inc}$ und $T_{copy}$ lassen sich mit kleinen Anpassungen und Ergänzungen als *Programmteile kombinieren* um die geforderte Funktionalität zu „implementieren“. Zunächst schreibt $T_{\mathbb{N}_0}$ „0“ und links davon „$-$“ auf das Band.

Danach spult Sie wieder zur „0“. Ab hier wechseln sich die folgenden *Programmteile* ab:

- Zuerst wird der aktuelle Wortteil, *links* und *rechts* jeweils durch „□“ oder „−“ begrenzt, vor die aktuelle Bandbeschriftung kopiert. Analog wie in $T_{copy}$ (nur eben nach links). Anschließend wird zum *linkesten* Zeichen der Bandbeschriftung gespult.
- Daraufhin startet ein *Programmteil* analog zu $T_{inc}$, der das aktuell *linkeste* Teilwort (vor dem ersten „"“) inkrementiert. Anschließend wird wieder zum *linkesten* Zeichen der Bandbeschriftung gespult.

*Anmerkung:*
*In der Aufgabe wurde keine genaue Reihenfolge oder Richtung der Aufzählung gefordert. Intuitiver wäre eine Aufzählung nach rechts* $0 - 1 - 10 - 11 - 100 - ....$ *Dies ist genauso gut möglich, allerdings muss dabei darauf geachtet werden, dass Binärzahlen beim inkrementieren ggf. eine Ziffer länger werden. In diesem Fall müsste man das aktuelle Teilwort um ein Feld nach rechts verschieben.*
*Anmerkung2: Work smart and hard!*

**Aufgabe 12.2 (1 + 1,5 + 1,5 = 4 Punkte)**
Gegeben folgender Algorithmus $S$, bei dem $z \in \mathbb{N}_+$ eine beliebige *Eingabe* darstellt.

```
k ← z
i ← 0
while k > 1 do
    if k mod 3 = 0 then
        k ← k div 3
        i ← i + 1
    fi
    if k mod 2 = 0 then
        k ← k div 2
        i ← i + 1
    fi
od
```

a) Welche Werte kann $z$ haben, damit nach Ausführung des Algorithmus $i = 4$ gilt?
b) Welche Bedingung muss $z$ (abhängig von $n$) erfüllen, damit für ein beliebiges aber festes $n \in \mathbb{N}_0$ nach Ausführung des Algorithmus gilt: $i = n$?
c) Gibt ein gültiges Hoare Tripel $\{P\}S\{Q\}$, mit dem sich zeigen lässt, dass nach der Ausführung von $S$ mit Eingabe $z = 325$, $i = 6$ gilt?
   - Falls ja: Geben Sie $P$ und $Q$ an. (Beweis **nicht** gefordert!)
   - Falls nein: Begründen Sie wieso nicht.

**Lösung 12.2**
a) $z \in \{16, 24, 36, 54, 81\}$
b) $z = 3^a \cdot 2^b$, mit: $a + b = n,\ a, b \in \mathbb{N}_0\}$
c) Nein, da die Ausführung von $S$ bei *Eingabe* $z = 325$ nicht terminiert, ist nach Definition zwar jedes Hoare-Tripel $\{z = 325\}S\{Q\}$ gültig, aber eben auch nicht nicht aussagekräftig genug um die Behauptung zu beweisen.
*Anmerkung: Dies ist plausibel, da die zu zeigende Aussage schlicht nicht gilt.*

**Aufgabe 12.3 (1 + 2 + 3 = 6 Punkte)**
Entwerfen Sie für jede Teilaufgabe einen Algorithmus $A_i$ der die erforderliche Ausgabe berechnet und geben Sie diesen in der Syntax der *einfachen Programmiersprache* aus der Vorlesung (Kapitel 17, Folie 16) an.
Betrachten Sie hierbei stets $n \in N_+$ als *Eingabe* und weisen sie die gesuchten Werte der Variable $c$ zu.

a) $A_1$ soll $\frac{n \cdot (n+1)}{2}$ berechnen und dabei **keine** anderen arithmetischen Operationen als Addition verwenden.
b) $A_2$ soll berechnen, wie oft $n$ maximal durch beliebige $t \in \mathbb{N}_+, t \geq 2$ ohne Rest geteilt werden kann.
c) Beweisen Sie die Korrektheit von $A_1$ mittels einem geeigneten Hoare-Tripel.

**Lösung 12.3**

a)
$z \leftarrow 0$
$i \leftarrow 0$
**while** $i < n$ **do**
  $z \leftarrow z + (i + 1)$
  $i \leftarrow i + 1$
**od**

*Anmerkung: Diese Variante des Algorithmus lässt sich leicht beweisen.*

b)
$c \leftarrow 0$
$t \leftarrow 2$
**while** $n > 1$ **do**
  **while** $n \bmod t = 0$ **do**
    $n \leftarrow n \operatorname{div} t$
    $c \leftarrow c + 1$
  **od**
  $t \leftarrow t + 1$
**od**

c)

$\{0 = 0\}$
$\{0 = \frac{0(0+1)}{2} \wedge 0 < n\}$
$z \leftarrow 0$
$\{z = \frac{0(0+1)}{2} \wedge 0 < n\}$
$i \leftarrow 0$
$\{z = \frac{i(i+1)}{2} \wedge i < n\}$
**while** $i < n$ **do**
$\{z = \frac{i(i+1)}{2} \wedge i \leq n\}$
$\{z = \frac{(i^2+i)}{2}\}$
$\{z = \frac{i^2+2i+i+2-2i-2}{2}\}$
$\{z = \frac{(i+1)(i+2)-2(i+1)}{2}\}$
$\{z = \frac{(i+1)(i+2)}{2} - (i+1)\}$
$\{z + (i+1) = \frac{(i+1)(i+2)}{2}\}$
$z \leftarrow z + (i+1)$
$\{z = \frac{(i+1)(i+2)}{2}\}$
$i \leftarrow i + 1$
$\{z = \frac{i(i+1)}{2}\}$
**od**
$\{z = \frac{i(i+1)}{2} \wedge i = n\}$
$\{z = \frac{n(n+1)}{2}\}$

**Aufgabe 12.4 (0,5 + 3 + 0,5 [+1] = 4 [+1] Punkte)**

Dr. Meta mag das Hoare-Kalkül nicht. Seit er den Übungsbetrieb nutzt, um güstig „Freiwillige“ für seine Organisation zu rekrutieren plagt es ihn jährlich. Sobald er die Weltherrschaft übernimmt, plant er es sofort abzuschaffen und seinem Erfinder aus Protest den Adelstitel abzuerkennen.

Viel besser gefällt ihm seine eigene, „revolutionäre“ Idee zum Beweis der Korrektheit von *iterativen* Algorithmen:

Hinreichend starke Schleifeninvarianten $I$ aufstellen und deren Gültigkeit induktiv beweisen. Dabei wird die Anzahl bisheriger Schleifendurchläufe $n$ als *Induktionsvariable* verwendet. Als Induktionsanfang wird stets (mindestens) $n = 0$ gewählt, um die Gültigkeit vor der Schleife zu beweisen. Im Induktionsschritt wird gezeigt, dass sofern $I$ nach dem $n$-ten Durchlauf gilt, die Gültigkeit für künftige Durchläufe erhalten bleibt. Hierzu anhand des Programmcodes stichhaltig argumentiert, wieso die Veränderungen innerhalb von Schleifendurchläufen die Invariante unberührt lassen.

Um das Beweisverfahren zu demonstrieren, haben sie folgenden Algorithmus:

```
z ← 0
i ← 0
while i < n do
    // Invariante: z ≥ i − 5
    if i mod 2 = 0 then
        z ← z − 1
    else
        z ← z + 3
    fi
    i ← i + 1
od
```

Dr. Meta will wissen, welche Bedingungen für *Eingaben* $n \in \mathbb{N}_0$ gelten müssen, damit $z \geq 8$ gilt. B. Scheuert hat hierfür direkt die starke Invariante $z \geq i - 5$ aufgestellt. Leider ist es ihm nicht gelungen, diese mit dem Hoare-Kalkül zu beweisen.

a) Beim Versuch, die Gültigkeit der Schleifeninvariante mittels Hoare-Kalkül zu beweisen, würden Sie auf ein Problem stoßen. Welche Information benötigt ein Korrektheitsbeweis, die bei Betrachtung eines einzelnen Schleifendurchlaufs nicht berücksichtigt wird?
b) Beweisen Sie die Schleifeninvariante induktiv.
c) Für welche $n \in \mathbb{N}_0$ gilt: $z \geq 8$?
d) Ist Dr. Metas Beweisprinzip für den Beweis der Korrektheit eines *iterativen* Algorithmus $A$ ausreichend? Begründen Sie. Gehen Sie der Einfachheit halber davon aus, dass $A$ nach der Schleife keine weiteren Programmzeilen hat.
*Diese Aufgabe gibt 1 Bonuspunkt!*

**Lösung 12.4**

a) Man muss für den Beweis berücksichtigen, dass nach einem Schleifendurchlauf mit $i \bmod 2 = 0$ stets auch ein Durchlauf mit $i \bmod 2 = 1$ folgt.
b) Wir schreiben $z_n$, bzw. $i_n$ für den Wert von $z$, bzw. $i$ nach dem $n$-ten Schleifendurchlauf.

**IA:** Wir wählen hier zwei Induktionsanfänge:
$n = 0$: Vor dem Durchlauf gilt: $z_0 = 0 = i_0 \geq i_0 - 5 = -5$.
$n = 1$:
Nach dem ersten Durchlauf gilt: $z_1 = z_0 - 1 = -1 \geq i_1 - 5 = i_0 - 4 = -4$.
Die jeweiligen Werte ergeben sich durch Ausführung der Programmzeilen.

**IV:** Nach dem $n$-ten Schleifendurchlauf gelte $z_n \geq i_n + 5$, für ein beliebiges, aber festes $n \in \mathbb{N}_0$.

**IS:** Schritt von $n \to n + 2$: // deshalb zwei Induktionsanfänge
Fallunterscheidung:
Fall I: $i_{n+1} \bmod 2 = 0 \Rightarrow i_{n+2} \bmod 2 = 1$:
$z_{n+2} = z_{n+1} + 3 = z_n - 1 + 3 = z_n + 2$.
Fall II: $i_{n+1} \bmod 2 = 1 \Rightarrow i_{n+2} \bmod 2 = 0$:
$z_{n+2} = z_{n+1} - 1 = z_n + 3 - 1 = z_n + 2$.
Es gilt also immer: $z_{n+2} = z_n + 2$. Ebenso stets $i_{n+2} = i_{n+1} + 1 = i_n + 2$.
Somit folgt nach IV: $z_{n+2} = z_n + 2 \geq i_n + 2 = i_{n+2}$.

*Anmerkung: Durch den Induktionsschritt $n \to n + 2$ wird der zweite Induktionsanfang notwendig, da sonst die Invariante nur für $n = 0, 2, 4, \ldots$ gezeigt wäre. Durch explizites Zeigen für $n = 1$ ist sie auch für $n = 1, 3, 5, \ldots$ und somit $\forall n \in \mathbb{N}_0$ gezeigt.*

c) für $n \in \{i \in \mathbb{N} \mid i = 8 \vee i \geq 10\}$. Die *schwächere* Invariante $z \geq i - 1$ lässt sich analog beweisen und die Korrektheit für $n = 8$ explizit prüfen.
d) Nicht ganz. Es muss zur Vollständigkeit noch gezeigt werden, dass die Schleife sofern sie überhaupt betreten- auch wieder verlassen wird. Nur unter dieser Bedingung lässt sich aus einer *hinreichend starken* Invariante $I$ auch die Korrektheit des Algorithmus folgern.

Mit dieser absolut revolutionären Idee für Algorithmenkorrektheitsbeweise gewappnet macht sich Dr. Meta auf um seine Welteroberungsalgorithmen gründlich zu prüfen. Da Welteroberungsalgorithmen naturgemäß unglaublich kompliziert sind, lässt er Sie vorerst hier zur Weiterbildung zurück.
Wen der Abschied schmerzt, sei versichert:
Der Bösewicht den Sie nicht brauchen, aber verdienen, ist gewiss irgendwann wieder zur Stelle!