Prof. Dr. M. Chimani
L. Enz, J. Kirstein, T. Oelschlägel, F. Stutzenstein, N. Troost
https://kreuzerl.tcs.uos.de

Universität Osnabrück
Theoretische Informatik
Sommersemester 2020

Übungsblatt 7 zur Einführung in die Theoretische Informatik

Ausgabe: 12. Juni 2020 Kreuzerl-Deadline: 21. Juni 2020

Die Aufgaben auf diesem Blatt beziehen sich auf den Vorlesungsstoff bis inklusive Kapitel 7.3

Aufgabe 7.1 Yabbadabbadoo, I want a queue!

In der Vorlesung wird behauptet, dass sich die Ackermann-Funktion mit einem GOTO-Programm schreiben lässt. Der Pseudocode der Ackermannfunktion ist jedoch rekursiv angegeben; GOTO-Programme kennen aber keine Rekursion. Diese kann man mithilfe eines sogenannten *Call-Stacks* realisieren. Eine ähnliche Datenstruktur, die nicht direkt in GOTO-Programmen gegeben ist, ist die *Queue*.

Sei W eine Konstante. Wir wollen mit GOTO-Programmen eine Queue simulieren, die natürliche Zahlen $a_1, a_2, \ldots$ enthalten kann mit $1 \leq a_i < W$ für alle i. In der Variable x_2 steht dabei die Queue und ggf. in x_3 die Anzahl der Elemente in der Queue. Überlegen Sie, wie Sie eine Queue innerhalb von x_2 darstellen können. Die zu implementierenden Operationen sind wie folgt:

- `empty()` schreibt 1 in x_1, wenn die Queue leer ist, sonst 0,
- `enqueue()` hängt die Zahl x_1 am Ende der Queue an (Sie dürfen $1 \leq x_1 < W$ annehmen),
- `dequeue()` löscht die erste Zahl der Queue (sofern die Queue nicht leer ist) und schreibt sie in x_1; ist die Queue leer, hält das Programm ohne x_1 zu verändern.

Sie können annehmen, dass Sie für Variablen eine Divison mit Rest (mit Konstanten) durchführen können. Analog kann man auch andere Anweisungen mit einem GOTO-Programm umsetzen. Daher erlauben wir die folgenden Sprachelemente ($i, j, c \in \mathbb{N}$):

- x_i := c
- x_i := $x_j \pm c$
- x_i := $x_i \pm x_j$
- x_i := $c \cdot x_i$
- x_i := $\lfloor x_i / c \rfloor$
- x_i := x_i mod c
- Verkettung ;
- if ($x_i \neq 0$) goto L_i
- halt
- goto L_i

Aufgabe 7.2 Wayne interessiert's?

Wir betrachten nun die Programmiersprache *Wayne*, die auf Variablen x_i mit $i \in \mathbb{N}$ arbeitet und folgende Anweisungen kennt:

A ; B	—	führt Anweisungen A und B nacheinander aus,
! x_i !	—	setzt Variable x_i auf den Wert 2,
{ x_i }	—	verdoppelt den Wert von Variable x_i,
[x_i]	—	halbiert den Wert von Variable x_i und rundet ab,
" w "	—	gibt die Zeichenkette $w \in \{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}, \mathrm{A}, \mathrm{B}, \ldots, \mathrm{Z}\}^*$ aus,
λ(x_i , A)	—	führt Anweisung A genau x_i-mal aus (unabhängig von späteren Änderungen an x_i).

Zum besseren Verständnis folgt ein Beispiel.
Programm: `!`x_0`!; λ(`x_0`, {`x_0`}); λ(`x_0`, "na"); "Batm"; [`x_0`]; λ(`x_0`, "a"); "n"`
Ausgabe: `nanananananananaBatmaaaan`

Beweisen Sie (z. B. durch Angabe eines Algorithmus) oder widerlegen Sie (z. B. durch Widerspruch) die Berechenbarkeit der Kolmogorov-Komplexität bzgl. Wayne.

Aufgabe 7.3 Entscheidbarkeit

Beweisen oder widerlegen Sie:

Es ist entscheidbar, ob eine Turingmaschine auf keiner möglichen Eingabe hält.

Aufgabe 7.4 Game of Another Life

Game of Another Life ist eine Maschine, die ein gegebenes unendliches Spielfeld mit Koordinaten in $\mathbb{Z}^2$ nach bestimmten Regeln modifiziert. Das Spielfeld ist ein unendliches Gitter, in dem jedes Quadrat für eine Zelle steht, die entweder den Wert 0 (tot) oder 1 (lebendig) hat. Eine Zuweisung $f\colon \mathbb{Z}^2 \to \{0,1\}$ nennen wir *Konfiguration* (des Spielfeldes). Wir sagen eine Konfiguration ist *endlich*, genau dann wenn $|f^{-1}(1)| \in \mathbb{N}$, d. h. es nur endlich viele lebendige Zellen gibt. Mit

$$N(x,y) := \sum_{\Delta_1=-1}^{1} \sum_{\Delta_2=-1}^{1} f(x+\Delta_1, y+\Delta_2)$$

bezeichnen wir die Anzahl der lebendigen Zellen in der Nachbarschaft von Zelle (x,y), zuzüglich der Zelle selbst. Die Maschine führt, startend auf einer initialen Konfiguration, immer wieder den folgenden Simulationsschritt aus. Sei f die aktuelle Konfiguration. Wir bestimmen die nächste Konfiguration f' durch

$$f'(x,y) := \begin{cases} 1 & \text{falls } N(x,y) = 3 \\ f(x,y) & \text{falls } N(x,y) = 4 \\ 0 & \text{sonst} \end{cases}$$

Betrachten Sie das Entscheidungsproblem GameOfAnotherLife:

> **Gegeben:** Zwei endliche Konfigurationen $f_1, f_2\colon \mathbb{Z}^2 \to \{0,1\}$.
> **Frage:** Kann aus f_1 nach einer endlichen Anzahl von Iterationen f_2 entstehen?

Sie dürfen im Folgenden voraussetzen, dass GameOfAnotherLife unentscheidbar ist.

(a) Formulieren Sie das zugehörige Co-Problem.

(b) Zeigen oder widerlegen Sie, dass GameOfAnotherLife *semi-entscheidbar* ist.

(c) Zeigen oder widerlegen Sie, dass GameOfAnotherLife *co-semi-entscheidbar* ist.

Aufgabe 7.5 EdgeCover ist *NP*-schwer?

Betrachten Sie das wie folgt definierte Entscheidungsproblem EdgeCover (Obacht: eine ungerichtete Kante ist eine 2-elementige Menge von Knoten),

> **Gegeben**: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$.
> **Gefragt**: Gibt es eine Kantenteilmenge $F \subseteq E$, sodass $|F| \leq k$ und $\forall v \in V\ \exists e \in F\colon v \in e$?

sowie das Entscheidungsproblem SetCover:

> **Gegeben:** Ein Universum $U := \{1, 2, \ldots, n\}$, eine Menge $S \subseteq \mathcal{P}(U)$ und eine Zahl $\ell \in \mathbb{N}$.
> **Gefragt:** Gibt es eine Teilmenge $C \subseteq S$ sodass $|C| \leq \ell$ und $\bigcup_{c \in C} c = U$?

Sie dürfen im Folgenden vorraussetzen, dass SetCover *NP*-schwer ist. Wo liegt der Fehler in folgendem Beweis der *NP*-Vollständigkeit von EdgeCover?

Beweis. Wir zeigen zuerst, dass EdgeCover $\in$ *NP*. Sei (V, E, k) eine Instanz, dann raten wir eine Kantenteilmenge $F \subseteq E$ und können $|F| \leq k$ in Linearzeit prüfen. Ebenso können wir trivial in $\mathcal{O}(|V|\,|F|)$ prüfen ob jeder Knoten überdeckt ist.

Es bleibt zu zeigen, dass EdgeCover auch *NP*-schwer ist. Wir beweisen die Aussage, indem wir EdgeCover deterministisch in Polynomialzeit auf SetCover reduzieren. Sei also $J := (V, E, k)$ eine beliebige EdgeCover-Instanz. Dann konstruieren wir die SetCover-Instanz $J' := (U, S, \ell)$ durch $U := V$, $S := E$ und $\ell := k$. Es ist leicht zu sehen, dass eine Lösung $L \subseteq E$ für J genau dann existiert, wenn auch J' die Lösung $L \subseteq S$ hat; denn jeder Knoten aus V muss ja von L überdeckt werden, ebenso muss das gesamte Universum U der Vereinigung der 2-elementigen Teilmengen von L entsprechen. □

Aufgabe 7.6 Game of Life

Sei G ein Gitter in Conway's Game of Life. Das Gitter, das durch i-faches Anwenden der in der Vorlesung vorgestellten Regeln entsteht, notieren wir als G_i. Sei $|G|$ die Anzahl der lebenden Zellen in G und $L(G) := \max\{i \in \mathbb{N} \mid |G_i| > 0\}$, wobei wir auch den Wert ∞ zulassen, falls $\{i \in \mathbb{N} \mid |G_i| > 0\}$ unbeschränkt ist.

Geben Sie alle Gitter G mit $|G| \leq 3$ und $L(G) > 0$ an.

Hinweis 1: Welche Auswirkungen haben *Verschiebung*, *Spiegelung* und *Rotation* auf ein Gitter?
Hinweis 2: Welchen „Aktionsumfang" haben die Regeln?

Go n-éirí an t-ádh leat!