**2. Klausur zur Vorlesung**
**Theoretische Grundlagen der Informatik**
**Wintersemester 2023/2024**

# Lösung!

- Bringen Sie den Aufkleber mit Ihrer Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrer Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Am Ende der Klausur sind zusätzliche Leerseiten. Fordern Sie zusätzliches Papier bitte nur an, falls Sie den gesamten Platz aufgebraucht haben.
- Die Tackernadel darf nicht gelöst werden.
- Als Hilfsmittel ist ein beschriebenes A4-Papier erlaubt.
- Einlesezeit: 15 min
  Bearbeitungszeit: 2 h

| | Mögliche Punkte | | | | | | Erreichte Punkte | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | Σ | a | b | c | d | e | Σ |
| Aufg. 1 | 2 | 3 | 3 | 3 | – | 11 | | | | | – | |
| Aufg. 2 | 2 | 2 | 2 | 3 | – | 9 | | | | | – | |
| Aufg. 3 | 1 | 3 | 2 | 3 | – | 9 | | | | | – | |
| Aufg. 4 | 1 | 7 | 2 | – | – | 10 | | | | – | – | |
| Aufg. 5 | 2 | 1 | 4 | 2 | – | 9 | | | | | – | |
| Aufg. 6 | 3 | 1 | 1 | 5 | 2 | 12 | | | | | | |
| Σ | | | | | | 60 | | | | | | |

## **Problem 1:** Automaten 2+3+3+3= 11 Punkte

Für ein Wort $w = w_1 w_2 \dots w_n$ bezeichnen wir mit $w_i$ das $i$-te Zeichen von $w$. Die Umkehrung $w_n w_{n-1} \dots w_1$ von $w$ bezeichnen wir mit $w^R$. Sei $L$ eine Sprache. Wir definieren $L^R \coloneqq \{w^R \mid w \in L\}$ als die Sprache, die alle Umkehrungen von Wörtern in $L$ enthält.

(a) Betrachten Sie die Sprache $L_4 \coloneqq \{w \in \{0,1\}^* \colon |w| \geqslant 4, w_4 = 1\}$. Geben Sie einen NEA an, der $L_4^R$ erkennt, also die Sprache aller Umkehrungen von Wörtern aus $L_4$.

(b) Sei $L$ eine reguläre Sprache. Beschreiben Sie einen NEA, der die Sprache $L^R$ erkennt.

Wir definieren nun die Familie von Sprachen

$$L_k \coloneqq \{w \in \{0,1\}^* \colon |w| \geqslant k, w_k = 1\}$$

für $k \in \mathbb{N}^+$. Die Sprache $L_k$ besteht also aus allen Wörtern, deren $k$-tes Zeichen eine 1 ist.

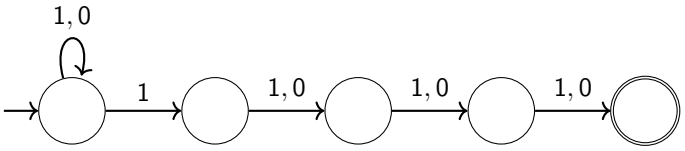
(c) Zeigen Sie für jedes $k \in \mathbb{N}^+$ die folgende Aussage. Für alle Wörter $w, w' \in \{0,1\}^k$ mit $w \neq w'$ gilt, dass sie nicht äquivalent bezüglich der Nerode-Relation von $L_k^R$ sind.

(d) Zeigen Sie, dass es kein Polynom $p$ gibt, sodass für alle regulären Sprachen die folgende Aussage gilt. Ist $L$ eine reguläre Sprache, für die ein DEA mit $n$ Zuständen existiert, so gibt es einen DEA, der $L^R$ erkennt und höchstens $p(n)$ viele Zustände hat.

*Hinweis: Betrachten Sie die Sprachen $L_k$.*

**Lösung:**

(a)




(b) Sei $A$ ein DEA, der $L$ erkennt. Wir konstruieren einen NEA $A^R$, der $L^R$ erkennt. Dazu fügen wir einen neuen Startzustand ein, den wir mit Epsilonübergängen zu allen akzeptierenden Zuständen verbinden und drehen alle Kanten des DEA $A$ um, d.h. falls $p \in \delta(q, x)$ in $A$, dann gilt für die neue Übergangsfunktion $q \in \delta^R(p, x)$ für Zustände $p, q$ und $x \in \Sigma \cup \{\varepsilon\}$. Der einzige akzeptierende Zustand des NEA $A^R$ ist der Startzustand des ursprünglichen Automaten $A$.

(c) Wir zeigen, dass alle Wörter $w, w' \in \{0,1\}^k$ mit $w \neq w'$ in verschiedenen Äquivalenzklassen liegen. Wir betrachten den größten Index $i \in \{1, \dots, k\}$ sodass $w_i \neq w'_i$. Wir dürfen aus Symmetriegründen annehmen, dass $w_i = 1$ und $w'_i = 0$. Dann gilt $w0^{i-1} \in L_k^R$, aber $w'0^{i-1} \notin L_k^R$, denn das $k$-letzte Zeichen von $w0^{i-1}$ ist $w_i = 1$, das $k$-letzte Zeichen von $w'0^{i-1}$ ist jedoch $w'_i = 0$. Somit liegen $w$ und $w'$ in verschiedenen Äquivalenzklassen.

(d) Wir widerlegen die Aussage. Wir betrachten die Sprache $L_k$ für $k \in \mathbb{N}^+$.

Definiere den DEA $D_k = (\{s_1, \dots, s_{k+1}, f\}, \{0,1\}, \delta, s_1, \{s_{k+1}\})$ wobei

$$\begin{aligned} \forall a \in \{0,1\}, i \neq k \colon \delta(s_i, a) &= s_{i+1} \\ \delta(s_k, 1) &= s_{k+1} \\ \delta(s_k, 0) &= f \\ \forall a \in \{0,1\} \colon \delta(s_{k+1}, a) &= s_{k+1} \\ \forall a \in \{0,1\} \colon \delta(f, a) &= f. \end{aligned}$$

Der DEA $D_k$ hat $k+2$ Zustände und erkennt $L_k$. Wir setzen also $n = k+2$.

Die Sprache $L_k^R$ besteht jedoch aus allen Wörtern, deren $k$-letztes Zeichen eine 1 ist.

**Variante 1 (mit Teilaufgabe c)** Nach Aufgabe (c) liegen alle Wörter $w, w' \in \{0,1\}^k$ mit $w \neq w'$ in verschiedenen Äquivalenzklassen bezüglich der Nerode-Relation von $L_k^R$. Da es aber $2^k$ verschiedene Wörter in $\{0,1\}^k$ gibt, und all diese in verschiedenen Äquivalenzklassen liegen, hat ein kleinster DEA, der $L_k^R$ erkennt auch mindestens $2^k \in \Theta(2^n)$, also superpolynomiell viele verschiedene Zustände.

**Variante 2 (Beweis aus der Übung)** In Übung 1 haben wir gesehen, dass jeder DEA, der $L_n^R$ erkennt mindestens $2^k$ viele Zustände hat. Für hinreichend große $k$ gilt jedoch $2^k > 100k^2$ und somit ist die Aussage in Aufgabe d für Sprachen $L_k^R$ für hinreichend große $k$ nicht erfüllt.

Um zu zeigen, dass jeder DEA, der $L_k^R$ erkennt mindestens $2^k$ viele Zustände hat, sind wir wie folgt vorgegangen. Angenommen es existiert ein DEA $D_k^R = (\{0,1\}, Q, \delta_E, q_0, F)$, der $L_k^R$ erkennt und weniger als $2^k$ Zustände hat. Wir zeigen zunächst, dass es Wörter $x, y \in \{0,1\}^*$ gibt und $u, v \in \{0,1\}^{k-1}$, sodass

$$\delta(q_0, x1u) = \delta(q_0, y0v).$$

Da es weniger als $2^k$ Zustände gibt, existieren Wörter $a = a_1 \dots a_k$ und $b = b_1 \dots b_k$ mit $a \neq b$ und $\delta(q_0, a) = \delta(q_0, b)$. Es existiert eine Stelle $i$, sodass $a_i \neq b_i$. Ohne Einschränkung dürfen wir annehmen, dass $a_i = 1$ und $b_i = 0$. Wir setzen $x = a_1 \dots a_{i-1}$ und $u = a_{i+1} \dots a_k 0^{i-1}$. Analog setzen wir $y = b_1 \dots b_{i-1}$ und $v = b_{i+1} \dots b_k 0^{i-1}$. Und es folgt

$$\delta(q_0, x1u) = \delta(q_0, a0^{i-1}) = \delta(\delta(q_0, a)0^{i-1}) = \delta(\delta(q_0, b)0^{i-1}) = \delta(q_0, b0^{i-1}) = \delta(q_0, y1v).$$

Nun gilt jedoch, dass $x1u \in L_k^R$, jedoch $y0v \notin L_k^R$, aber $\delta(q_0, x1u) = \delta(q_0, y0v)$. Das ist ein Widerspruch.

## **Problem 2:** Gödelnummersprachen $2 + 2 + 2 + 3 = 9$ Punkte

Für eine gegebene Sprache $K$ sei $L_K \coloneqq \{\langle M\rangle \mid L(M) = K\}$ die Sprache der Gödelnummern aller Turingmaschinen, die genau die Wörter aus $K$ akzeptieren.

(a) Sei $K_1 = L(10^*1)$. Beschreiben Sie eine Turingmaschine $M$, sodass $\langle M\rangle \in L_{K_1}$.

(b) Sei $K_2$ regulär. Zeigen oder widerlegen Sie: Jede Turingmaschine $M$ mit $\langle M\rangle \in L_{K_2}$ macht bei Eingabe $w$ höchstens $O(|w|)$ Schritte.

Kreuzen Sie an: ☐ Ich zeige die Aussage. ☐ Ich widerlege die Aussage.

(c) Sei $K_3 \notin \{\varnothing, \Sigma^*\}$ regulär. Zeigen oder widerlegen Sie: Jede Turingmaschine $M$ mit $\langle M\rangle \in L_{K_3}$ macht bei Eingabe $w$ mindestens $\Omega(|w|)$ Schritte.

Kreuzen Sie an: ☐ Ich zeige die Aussage. ☐ Ich widerlege die Aussage.

Sei $H = \{\langle M\rangle \# w \mid M \text{ hält auf Eingabe } w\}$ die Sprache des Halteproblems. Nach dem Satz von Rice ist $L_H$ unentscheidbar.

(d) Beschreiben Sie eine Turingmaschine $T$, sodass $\langle T\rangle \in L_H$. Zeigen Sie außerdem, dass die Sprache $L' = \{\langle T\rangle\}$ entscheidbar ist. Erklären Sie *kurz*, warum dies kein Widerspruch dazu ist, dass $H$ und $L_H$ unentscheidbar sind.

**Lösung:**

(a) Die Turingmaschine überprüft zuerst, ob das erste Zeichen der Eingabe eine 1 ist, geht dann ein Zeichen nach rechts und bewegt den Kopf immer weiter nach rechts, solange das gelesene Zeichen eine 0 ist. Wenn dann eine 1 gelesen wird, wandert der Kopf noch eine Position nach rechts, wenn dort die Eingabe zuende ist (das Band ist leer), akzeptiert die Turingmaschine. Wenn zu einem beliebigen Zeitpunkt der Abarbeitung etwas anderes als oben beschrieben gelesen wird, lehnt die Turingmaschine ab.

(b) Die Aussage gilt nicht. Eine Turingmaschine, die $K_2$ entscheidet kann modifiziert werden, sodass sie bei Eingabe $w$ vor dem Halten noch $|w|^3$ Schritte nach rechts läuft und erst dann hält, ohne das Akzeptanzverhalten zu modifizieren. Solange es also eine Turingmaschine gibt, die $K_2$ entscheidet, gibt es auch eine Turingmaschine, die $K_2$ entscheidet und für jede Eingabe $w$ mehr als $\mathcal{O}(|w|)$ Schritte macht.

(c) Die Aussage gilt nicht. Für $K_3 = L(\mathsf{b}(\mathsf{a} \cup \mathsf{b})^*)$ über $\Sigma = \{\mathsf{a}, \mathsf{b}\}$ kann nach dem ersten gelesenen Zeichen gehalten werden, unabhängig davon, wie lang die Eingabe ist.

(d) $T$ erhält als Eingabe ein Wort der Form $\langle M\rangle \# w$ (andernfalls wird abgelehnt). $T$ verwendet nun die universelle Turingmaschine $U$, um $M$ mit Eingabe $w$ zu simulieren. Hält $U$, so akzeptiert $T$. Andernfalls hält auch $T$ nicht und akzeptiert die Eingabe somit auch nicht.

$L'$ enthält nur ein einziges Wort, ist also endlich und somit entscheidbar. Dies ist kein Widerspruch zur Unentscheidbarkeit von $H$ bzw. $L_H$, da die Gödelnummer verglichen werden kann, ohne das Verhalten von $T$ zu betrachten.

## **Problem 3:** Grammatiken 1+3+2+3 = 9 Punkte

Für eine Sprache $L$ und $k \in \mathbb{N}^+$ definieren wir

$$L_k \coloneqq \{w^k \mid w \in L\}.$$

Betrachten Sie die Sprache $L' \coloneqq \mathsf{ab}^*\mathsf{a}$.

(a) Geben Sie für jedes $k \in \mathbb{N}^+$ die Sprache $L'_k$ an.

(b) Zeigen Sie für alle $k \in \mathbb{N}^+$ mit $k \geqslant 2$, dass $L'_k$ nicht regulär ist.

Für eine Sprache $L$ definieren wir

$$L_\infty \coloneqq \{w^k \mid w \in L, k \in \mathbb{N}^+\}$$

(c) Geben Sie eine geeignete Sprache $S$ an, sodass $L'_2 = S \cap L'_\infty$ und zeigen Sie mithilfe von $S$, dass $L'_\infty$ nicht regulär ist.

(d) Geben Sie eine Sprache $L''$ an, die unendlich viele Wörter enthält und für die $L''_k$ regulär ist für alle $k \in \mathbb{N}^+$. Bestimmen Sie für alle $k \in \mathbb{N}^+$ die Sprache $L''_k$ und beweisen Sie, dass $L''_k$ regulär ist.

*Hinweis: Es genügt die Funktionsweise eines DEA oder NEA zu beschreiben, der die Sprache erkennt, um Regularität zu zeigen.*

**Lösung:**

(a) Für alle $k \in \mathbb{N}^+$ gilt

$$L'_k = \{(\mathsf{ab}^n\mathsf{a})^k \mid n \in \mathbb{N}_0\}.$$

(b) **Variante 1: (mit Nerode)** Wir verwenden den Satz von Nerode. Wir zeigen, dass für alle $i, j \in \mathbb{N}$ mit $i \neq j$ die Wörter $\mathsf{ab}^i\mathsf{a}$ und $\mathsf{ab}^j\mathsf{a}$ nicht äquivalent bezüglich der Nerode-Relation sind: Mit $w \coloneqq (\mathsf{ab}^i\mathsf{a})^{k-1}$ folgt $\mathsf{ab}^i\mathsf{a}w \in L$, aber $\mathsf{ab}^j\mathsf{a}w \notin L$. Somit gibt es unendlich viele Äquivalenzklassen bezüglich der Nerode-Relation. Die Sprache $L'_k$ ist also nicht regulär.

**Variante 2: (mit Pumping-Lemma)** Sei $k \in \mathbb{N}$ mit $k \geqslant 2$. Wir wenden das Pumping-Lemma auf das Wort $w \coloneqq (\mathsf{ab}^n\mathsf{a})^k$ an. Es gilt $w \in L_k$. Sei $uvx = w$ eine Zerlegung von $w$ mit $|uv| \leqslant n$ und $v \neq \varepsilon$.

Falls das Teilwort $v$ das Zeichen $\mathsf{a}$ enthält, so betrachten wir das Wort $w' = uv^2x$. Doch das Wort $w'$ enthält mehr als $2k$ mal das Zeichen $\mathsf{a}$ und ist deswegen nicht in $L'_k$ enthalten.

Falls das Teilwort $v$ das Zeichen $\mathsf{a}$ nicht enthält, so besteht $v$ nur aus Zeichen $\mathsf{b}$. Wir betrachten das Wort $w' = uv^2x$. Wir pumpen also nur an einer Stelle $\mathsf{b}$ auf, alle anderen Vorkommen (die mit Zeichen $\mathsf{a}$ von diesem getrennt sind) bleiben unverändert. Somit ist $w' \notin L'_k$.

Das Pumping-Lemma ist also nicht erfüllt und es folgt, dass $L'_k$ nicht regulär ist.

(c) Wir wissen nach (b), dass $L'_k$ nicht regulär ist für $k = 2$. Angenommen $L'_\infty$ ist regulär. Wir konstruieren eine reguläre Sprache $S$, sodass $L'_2 = S \cap L'_\infty$. Da reguläre Sprachen unter Schnitt abgeschlossen sind, folgt dann dass $L'_2$ regulär ist, ein Widerspruch. Wir setzen

$$S \coloneqq \{w \in \{\mathsf{a}, \mathsf{b}\}^* \mid |w|_\mathsf{a} = 2k = 4\}.$$

Es ist leicht einzusehen, dass $S$ regulär ist. Wegen $L'_2 = S \cap L'_\infty$ folgt die Behauptung.

(d) Betrachte $L'' \coloneqq \mathsf{a}^*$. Für $k \in \mathbb{N}^+$ gilt

$$L''_k = \{w \in \mathsf{a}^* \mid \exists n \in \mathbb{N}_0 \colon |w| = kn\}.$$

Die Sprache $L''_k$ besteht also aus allen Wörtern über dem Alphabet $\{\mathsf{a}\}$, deren Länge durch $k$ teilbar ist. Die Sprache $L''_k$ wird von dem DEA $D_k = (\{s_0, \ldots, s_{k-1}\}, \{\mathsf{a}\}, \delta_k, s_0, \{s_0\})$ erkannt, wobei

$$\begin{aligned} \forall i < k-1 \colon \delta_k(s_i, \mathsf{a}) &= s_{i+1} \\ \delta_k(s_{k-1}, \mathsf{a}) &= s_0. \end{aligned}$$

Somit ist $L''_k$ regulär für alle $k \in \mathbb{N}^+$.

## **Problem 4:** NP-Vollständigkeit $1 + 7 + 2 = 10$ Punkte

Wir betrachten folgendes Entscheidungsproblem:

> RECTANGLE-PACKING
> **Gegeben:** Ein Rechteck $R$ mit Höhe $h \in \mathbb{N}^+$ und Breite $b \in \mathbb{N}^+$, sowie eine Menge $M = \{R_1, \ldots, R_n\}$ weiterer Rechtecke mit Höhen $h_i \in \mathbb{N}^+$ und Breiten $b_i \in \mathbb{N}^+$.
> **Problem:** Können alle Rechtecke aus $M$ so in $R$ platziert werden, dass sie sich weder überlappen noch aus $R$ hinausragen?

Hierbei gelten die folgenden Regeln für alle RECTANGLE-PACKING Instanzen:

- $R$ sowie alle Rechtecke in $M$ sind achsenparallel.
- $R$ hat seine linke untere Ecke an Position $(0, 0)$.
- Die Rechtecke aus $M$ dürfen beliebig verschoben werden. Die Rechtecke dürfen nicht rotiert werden.
- Die Ecken von Rechtecken aus $M$ dürfen nur an ganzzahligen Koordinaten platziert werden.

In dieser Aufgabe sollen Sie zeigen, dass RECTANGLE-PACKING NP-vollständig ist. Dazu können Sie folgendes NP-vollständiges Problem benutzen:

> BIN-PACKING
> **Gegeben:** Eine Menge $U = \{x_1, \ldots, x_n\}$ und eine Gewichtsfunktion $w \colon U \to \mathbb{N}^+$. Dazu zwei natürliche Zahlen $k, B \in \mathbb{N}^+$.
> **Problem:** Gibt es eine Aufteilung von $U$ in $k$ disjunkte Mengen $U_1, \ldots, U_k$, sodass für jedes $U_i$ das Gesamtgewicht $\sum_{x \in U_i} w(x)$ höchstens $B$ ist?

(a) Zeigen Sie, dass RECTANGLE-PACKING in NP liegt.

*Hinweis: Sie dürfen ohne Beweis benutzen, dass man in Polynomialzeit überprüfen kann, ob sich zwei achsenparallele Rechtecke überlappen.*

(b) Zeigen Sie, dass RECTANGLE-PACKING NP-schwer ist.

(c) Zeigen Sie, dass RECTANGLE-PACKING auch dann NP-schwer ist, wenn die Rechtecke aus $M$ um 90° rotiert werden dürfen.

*Hinweis: Sie dürfen Ihren Beweis aus (b) anpassen, oder annehmen, dass* RECTANGLE-PACKING *(ohne Rotation)* NP*-schwer ist.*

**Lösung:**

(a) Ein Orakel rät eine Lösung, d.h. ganzzahlige Koordinaten $(x_i, y_i)$ für die untere linke Ecke der $R_i$, $i = 1, \ldots, n$. Wir testen in Linearzeit, ob alle $R_i \in M$ in $R$ liegen, indem wir $0 \leqslant x_i \leqslant b - b_i$ und $0 \leqslant y_i \leqslant h - h_i$ prüfen, und dass sich keine zwei Rechtecke aus $M$ überlappen (was nach dem Hinweis in poly mal $n^2 =$ poly geht). Es folgt, dass RECTANGLE-PACKING in NP liegt.

(b) Wir reduzieren eine Instanz $I_{\text{BP}}$ von BIN-PACKING auf eine äquivalente Instanz $I_{\text{RP}}$ von RECTANGLE-PACKING.

*Konstruktion:* Gegeben ist eine BIN-PACKING-Instanz $I_{\text{BP}} = (U, w, k, B)$. Zuerst definieren wir das Rechteck $R$, indem wir $b \coloneqq k$ und $h \coloneqq B$ setzen. Danach fügen wir für jedes $x_i \in U$ genau ein zugehöriges $R_i$ in $M$ ein, mit Höhe $h_i = w(x_i)$ und Breite $b_i = 1$. Die Größe von $I_{\text{RP}}$ ist polynomiell in der Größe von $I_{\text{BP}}$. Alle Berechnungen benötigen ebenfalls nur Polynomialzeit.

*Äquivalenz:* Sei $P_{\text{BP}}$ eine Lösung von BIN-PACKING, also eine Aufteilung von $U$ in $k$ disjunkte Mengen $U_1, \ldots, U_k$, sodass für jedes $U_i$ das Gesamtgewicht $\sum_{x \in U_i} w(x)$ maximal $B$ ist. Sei $M_i \subseteq M$ die Menge der zu $U_i$ zugehörigen Rechtecke aus $M$. Diese haben (ohne Rotation) eine Gesamthöhe von $\sum_{M \in M_i} h(M) = \sum_{x \in U_i} w(x)$, was kleiner als $B$ ist, da $P_{\text{BP}}$ eine valide Lösung von $I_{\text{BP}}$ ist. Alle

diese Rechtecke aus $M_i$ passen also übereinander in eine Spalte (der Breite 1) von $R$. Die Spalten aller $U_i$ können nebeneinander in $R$ platziert werden (weil $b = k$), sodass wir eine Lösung $P_{\text{RP}}$ von Rectangle-Packing erhalten.

Für die andere Richtung sei nun $P_{\text{RP}}$ eine Lösung von Rectangle-Packing. Da jedes Rechteck aus $M$ Breite 1 hat, muss eine ganzzahlige Lösung $P_{\text{RP}}$ spaltenweise gefüllt sein. Entsprechend partitionieren die Spalten $M$, was wir auf $U$ übertragen. Jede Spalte in $P_{\text{RP}}$ liefert eine Teilmenge $M_i$ von Rechtecken aus $M$. Die zu $M_i$ zugehörigen Elemente $U_i$ aus $U$ haben dann Gesamtgewicht höchstens $B$, weil die die Höhe von $R$ gleich $B$ ist und die Rechtecke übereinandergestapelt nicht aus $R$ herausragen. Genauso folgt aus der Breite $k$ von $R$, dass wir höchstens $k$ Mengen $U_i$ definiert haben. Wir erhalten also eine Lösung $P_{\text{BP}}$ von Bin-Packing.

Aus der NP-Schwere von Bin-Packing folgt nun die NP-Schwere von Rectangle Packing.

(c) In der obigen Konstruktion können wir jedem Rechteck $R_i$ die Höhe $(k+1) \cdot w(x_i)$ zuweisen und dem großen Rechteck $R$ die Höhe $h \coloneqq (k+1) \cdot B$. Damit sind alle Rechtecke in $M$ so hoch, dass sie nur hochkant in $R$ platziert werden können, ohne aus $R$ herauszuragen. Der Rest des Beweises funktioniert analog.

*Alternativansatz (mit Annahme, dass rotationsfreies* Rectangle-Packing *NP-schwer ist):*
*Für eine gegebene rotationsfreie Instanz $I$ konstruiere eine Instanz $I'$ mit Rotationsoption, indem die Höhe aller Rechtecke um Faktor $(k+1)$ gestreckt wird. Eine Lösung von $I$ liefert eine Lösung von $I'$, indem alle y-Koordinaten mit Faktor $(k+1)$ multipliziert werden. In der anderen Richtung liefert eine Lösung von $I'$ eine Lösung für $I$, indem alle y-Koordinaten mit Faktor $1/(1+k)$ multipliziert werden, da eine Lösung von $I'$ keine rotierten Rechtecke enthalten kann, da jedes Rechteck dort höher ist, als $R$ breit ist.*

## **Problem 5:** Approximation $2 + 1 + 4 + 2 = 9$ Punkte

In dieser Aufgabe betrachten wir das Knotenfärbungsproblem Vertex Color.

| Vertex Color |
|---|
| **Gegeben:** Ungerichteter Graph $G = (V, E)$. |
| **Problem:** Finde das kleinste $k \in \mathbb{N}^+$, sodass es für $G$ eine $k$-Färbung $c\colon V \to \{1, \dots, k\}$ gibt. |

Erinnerung: Eine $k$-Färbung ordnet jedem Knoten eine Farbe aus $\{1, \dots, k\}$ zu, sodass je zwei benachbarte Knoten unterschiedlich gefärbt sind.

Wir betrachten nun einen Algorithmus, der schrittweise Knoten eines Graphen färbt. Dabei nennen wir eine Farbe an einem Knoten *frei*, wenn noch kein Nachbar des Knotens in dieser Farbe gefärbt wurde.

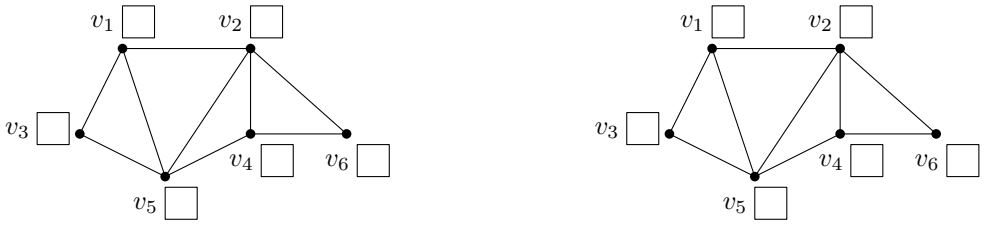
Betrachten Sie den unten beschriebenen Algorithmus $\mathcal{A}$. Er erhält als Eingabe ein Tupel $(G, \pi)$ bestehend aus einem Graphen $G = (V, E)$ und einer Knotenordnung $\pi$.

**Algorithmus $\mathcal{A}$:** Greedy Vertex Coloring

**Input:** $(G, \pi)$
1 Zu Beginn sind alle Knoten ungefärbt
2 **Für** *nächsten ungefärbten Knoten $v \in V$ in der durch $\pi$ gegebenen Reihenfolge*
3 Färbe $v$ in der kleinsten freien Farbe
4 **return** *größte verwendete Farbe*

(a) Färben Sie die unten dargestellte Instanz, indem Sie Algorithmus $\mathcal{A}$ auf $(G, \pi)$ anwenden, wobei $\pi$ durch die Indizes gegeben ist ($\pi(v_i) = i$). Schreiben Sie für jeden Knoten $v_i$ die Farbe $c(v_i) \in \mathbb{N}^+$ in die zugehörige Box. Ist diese Lösung optimal? Begründen Sie.

Falls Sie beide Kopien der Instanz bearbeiten, markieren Sie eindeutig die zu wertende Kopie.




Instanz für Teilaufgabe (a) (und eine zusätzliche Kopie)

(b) Liefert $\mathcal{A}$ für jede Eingabe $(G, \pi)$ in Polynomialzeit eine $k$-Färbung (wobei $k$ eine Zahl ist, die auch größer als die optimale Lösung von Vertex Color sein kann)? Begründen Sie.

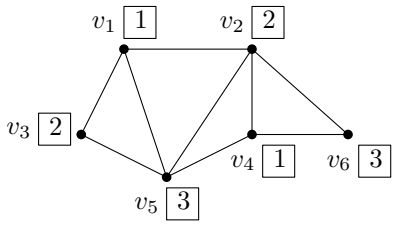
(c) Zeigen Sie, dass es kein $r \in \mathbb{R}$ gibt, sodass $\mathcal{A}$ für alle Knotenordnungen $\pi$ ein Approximationsalgorithmus mit relativer Güte $r$ ist.

*Hinweis: Sie können eine Familie von Bäumen $(T_i)_{i \in \mathbb{N}}$ mit Knotenordnungen $\pi_i$ konstruieren, für die $\mathcal{A}$ die Wurzel mit Farbe $i$ färbt. Jeder Baum kann mit 2 Farben gefärbt werden.*

(d) Zeigen Sie, dass es für jeden Graphen $G$ eine Knotenordnung $\pi$ gibt, sodass $\mathcal{A}$ auf Eingabe $(G, \pi)$ eine Lösung ausgibt, die genau so viele Farben wie eine optimale Lösung verwendet.

**Lösung:**

(a) Die Färbung $c(1) = 1, c(2) = 2, c(3) = 2, c(4) = 1, c(5) = 3, c(6) = 3$, die von $\mathcal{A}$ berechnet wurde ist optimal, da der Graph Dreiecke enthält, die nicht mit weniger als drei Farben gefärbt werden können.




(b) Ja, das Finden der kleinsten freien Farbe ist in Linearzeit möglich und $\mathcal{A}$ bearbeitet jeden Knoten genau einmal. Die Lösung ist offensichtlich gültig, da die Bedingung einer gültigen Färbung für das Färben jedes Knoten sichergestellt wird.

(c) Jeder Baum ist zweifärbbar. Aber wir können für jedes $c \in \mathbb{N}^+$ einen Baum $T_c$ angeben, dessen Wurzel $r_c$ von $\mathcal{A}$ mit Farbe $c$ gefärbt wird. Wir definieren diese $T_c$ rekursiv beginnend mit $T_1 = (\{r_1\}, \emptyset)$. Wir bauen $T_{c+1}$, indem wir $r_{c+1}$ mit der Wurzel von jedem kleineren $T_i, i \in \{1, \ldots, c\}$ verbinden. Wir wählen die Knotenordnung so, dass zuerst $T_1$ abgearbeitet wird in der für diesen Teilbaum berechneten Abarbeitungsreihenfolge, dann $T_2$ und so weiter. Die Wurzel $r_{c+1}$ wird als letztes abgearbeitet. Somit ist die kleinste freie Farbe von $r_{c+1}$ die Farbe $c + 1$. Somit gilt für die Approximationsgüte $\lim_{c\to\infty} \frac{\mathcal{A}(T_c)}{\mathrm{OPT}(T_c)} = \lim_{c\to\infty} \frac{c}{2} = \infty$.

(d) Ja, betrachte eine optimale Lösung mit Färbung $c_{\mathrm{OPT}}$. Wenn $\mathcal{A}$ alle Knoten nach aufsteigendem $c_{\mathrm{OPT}}(v)$ färbt, bekommt jeder Knoten $v$ eine Farbe kleiner gleich der in der optimalen Lösung zugewiesenen Farbe zugewiesen, da alle seine Nachbarn eine kleinere Farbe als $c_{\mathrm{OPT}}(v)$ haben.

## **Problem 6:** Entscheidbarkeit 3+1+1+5+2 = 12 Punkte

In dieser Aufgabe betrachten wir nur Gödelnummern von deterministischen Turingmaschinen mit Eingabealphabet $\Sigma := \{0, 1\}$.

Für Sprachen $L', L'' \subseteq \Sigma^*$ definieren wir die Sprache $L' \odot L''$ als

$$L' \odot L'' := \{w' \# w'' \mid w' \in L', w'' \in L''\}.$$

(a) Beweisen Sie die folgende Aussage. Sind $L'$ und $L''$ semi-entscheidbare Sprachen, so ist die Sprache $L' \odot L''$ semi-entscheidbar.

Betrachten Sie die folgenden zwei Sprachen

$$A := \{\langle M \rangle \mid M \text{ hält auf höchstens einem Wort}\}$$
$$B := \{w \in \Sigma^* \mid \text{Es existiert TM } M\colon\ M \text{ hält auf } w\}.$$

Aus der Vorlesung wissen Sie, dass die Sprache des Halteproblems

$$H := \{\langle M \rangle \# w \mid M \text{ hält auf } w\}$$

unentscheidbar ist.

(b) Zeigen Sie, dass $B$ entscheidbar ist.

(c) Zeigen Sie, dass für alle Gödelnummern $\langle M \rangle$ folgende Äquivalenz gilt:

$$\langle M \rangle \in A \iff \langle M \rangle \# 0 \in A \odot B$$

(d) Zeigen Sie, dass $A \odot B$ nicht entscheidbar ist, indem Sie von der Sprache des Halteproblems reduzieren. Sie dürfen in dieser Aufgabe nicht den Satz von Rice verwenden.

(e) Zeigen Sie unter der Annahme, dass $A \odot B$ nicht semi-entscheidbar ist, dass $A$ nicht semi-entscheidbar ist.

**Lösung:**

(a) Da $L'$ und $L''$ semi-entscheidbar sind, existieren TM $T'$ und $T''$, die $L'$ bzw. $L''$ semi-entscheiden. Wir konstruieren eine TM $T$, die das $L' \odot L''$ semi-entscheidet. Zuerst überprüft $T$, ob die Eingabe $w$ die richtige Form hat, also genau ein # beinhaltet, und wir bezeichnen den Teil von $w$ links des Trennzeichens # mit $w'$ und den Teil rechts davon mit $w''$, alle anderen Eingaben werden sofort abgelehnt.

**Variante 1: (parallel ausführen)** Die TM $T$ arbeitet auf zwei Bändern. Bei Eingabe $w' \# w''$ führt sie abwechselnd je einen Schritt der Bearbeitung von $T'$ auf $w'$ auf dem ersten Band aus, und einen Schritt der Bearbeitung von $T''$ auf $w''$ auf dem zweiten Band. Die TM $T$ akzeptiert, wenn sowohl $T'$ als auch $T''$ akzeptieren.

Ist $w' \# w'' \in L' \odot L''$, so akzeptieren $T'$ und $T''$ nach endlich vielen Schritten, somit akzeptiert auch $T$. Die TM $T$ semi-entscheidet die Sprache $L' \odot L''$.

**Variante 2: (nacheinander ausführen)** Bei Eingabe $w' \# w''$ führt die TM $T$ erst $T'$ auf $w'$ aus. Wenn $T'$ akzeptiert, so führt $T$ anschließend $T''$ auf $w''$ aus und übernimmt das Akzeptanzverhalten. Lehnt $T'$ ab, so lehnt auch $T$ ab.

Für jedes Wort $w' \# w'' \in L' \odot L''$ gilt, dass sowohl $T'$ als auch $T''$ die Eingaben $w'$ bzw. $w''$ akzeptieren, also insbesondere nach endlich vielen Schritten halten. Die TM $T$ semi-entscheidet nach Konstruktion also die Sprache $L' \odot L''$.

(b) Es gilt $B = \Sigma^*$, da jedes Wort $w$ in einer Sprache enthalten ist, die von einer Turingmaschine entschieden wird. Die Sprache $\Sigma^*$ ist regulär, somit insbesondere entscheidbar.

(c) Da $B = \Sigma^*$ gilt und somit insbesondere $0 \in B$ folgt die Behauptung.

(d) Wir reduzieren vom Halteproblem. Angenommen $A \odot B$ ist entscheidbar, so existiert eine TM $T$, welche $A \odot B$ entscheidet. Wir konstruieren eine TM $T_H$, welche das Halteproblem entscheidet. Die TM $T_H$ soll ein Tupel $\langle M \rangle \# w$ genau dann akzeptieren, wenn $M$ auf $w$ hält. Wir konstruieren eine TM $M_w$, welche die Eingabe ignoriert und $M$ auf $w$ ausführt. Falls $M$ auf $w$ hält, so hält auch $M_w$. Da $M_w$ die Abarbeitung von $w$ durch $M$ simuliert, hält $M_w$ nicht, falls $M$ auf $w$ nicht hält. Es folgt:

- $M_w$ hält auf jeder Eingabe, falls $M$ auf $w$ hält.
- $M_w$ hält auf keiner Eingabe, falls $M$ nicht auf $w$ hält.

Somit gilt

$$M \text{ hält auf } w \iff \langle M_w \rangle \notin A \iff \langle M_w \rangle \# 0 \notin A \odot B. \tag{1}$$

wobei wir in der zweiten Äquivalenz Aufgabe c verwenden.

Die TM $T_H$ simuliert nun $T$ auf der Eingabe $\langle M_w \rangle \# 0$ und akzeptiert, wenn $T$ die Eingabe ablehnt, und lehnt ab, wenn $T$ die Eingabe akzeptiert. Da $T$ auf jeder Eingabe hält, hält auch $T_H$ auf jeder Eingabe und entscheidet nach (1) das Halteproblem. Das ist ein Widerspruch. Somit ist $A \odot B$ unentscheidbar.

(e) Nach b ist $B$ entscheidbar, somit insbesondere semi-entscheidbar. Angenommen $A$ ist semi-entscheidbar, so folgt mit a, dass $A \odot B$ semi-entscheidbar ist. Nach Voraussetzung ist aber $A \odot B$ nicht semi-entscheidbar. Das ist ein Widerspruch. Somit ist $A$ nicht semi-entscheidbar.