Technische Universität Berlin, 19.02.2026

Name: ..... **Matr.-Nr.: .....**

Klausur: Berechenbarkeit und Komplexität

(Nichterlein/Herrmann/Breitkopf, Wintersemester 2025/2026)

Bearbeitungszeit:	120 Minuten
Punktzahl zum Bestehen:	50 Punkte

Aufgabe Nr.:	1	2	3	4	5	6	7	8	9	**Summe**
Punktzahl:	12	14	12	16	16	14	20	16	8	128
Davon erreicht:										

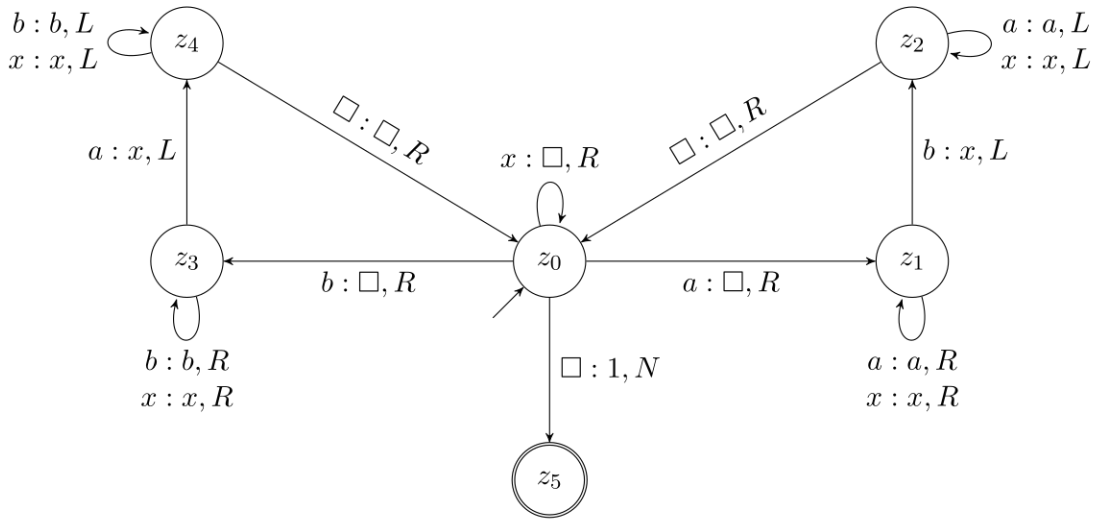
Allgemeine Hinweise:

- Es sind keinerlei Hilfsmittel erlaubt.
- Benutzen Sie einen dokumentenechten Stift in der Farbe schwarz oder blau. Insbesondere also keinen Bleistift, sondern einen Kugelschreiber.
- Beschriften Sie jedes Blatt mit ihrem Vor- und Nachnamen und ihrer Matrikelnummer.
- **Falls es in der Aufgabenstellung nicht explizit ausgeschlossen wird, so sind alle Antworten zu begründen! Antworten ohne Begründung erhalten 0 Punkte.**
- Sie dürfen alle Aussagen **und Mitteilungen** aus den Vorlesungsfolien als bekannt annehmen, es sei denn der Beweis einer solchen Aussage ist explizit gefordert.

Viel Erfolg!

Aufgabe 1: **Deterministische Turing-Maschine analysieren** (12 Punkte)

Betrachten Sie die Turing-Maschine $M = (\{z_0, z_1, z_2, z_3, z_4, z_5\}, \{a, b\}, \{a, b, x, 1, \square\}, \delta, z_0, \square, \{z_5\})$, wobei δ die folgende graphische Darstellung hat:



Beantworten Sie folgende Fragen zu der Turing-Maschine **ohne Begründung**:

(a) Geben Sie die Konfigurationsfolge (beginnend mit der Startkonfiguration z_0abba) der Turing-Maschine M bei Eingabe $abba$ an. (4 P)

(b) Geben sie die von M akzeptierte Sprache $T(M)$ an. (4 P)

(c) Welche Funktion berechnet M? (4 P)

a) $z0 \xrightarrow{M} z1bba \xrightarrow{M} z2\square xba \xrightarrow{M} z0xba \xrightarrow{M} z0ba \xrightarrow{M} z3a \xrightarrow{M} z4\square x \xrightarrow{M} z0x \xrightarrow{M} z0 \xrightarrow{M} z51$

b) $|a| = |b|$ oder $\{((a^n b^n)^* + (b^n a^n)^*)^*\}, n \in \mathbb{N}$

=> Anzahl an a's im Wort muss gleich zur Anzahl an b's sein. Reihenfolge beliebig

c) Die TM M berechnet die halbe charakteristische Funktion $\chi'_{T(M)}(x)$ mit

$$\chi'_{T(M)}(x) := \begin{cases} 1, & \text{falls } x \in T(M) \\ \perp, & \text{falls } x \notin T(M) \end{cases}$$

Hinweise/Definitionen:

Eine (eventuell partielle) Funktion $f\colon \Sigma^* \to \Pi^*$ heißt Turing-berechenbar, wenn es eine *DTM* $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ mit $\Pi \subseteq \Gamma \setminus \{\square\}$ gibt, sodass für alle $x \in \Sigma^*, y \in \Pi^*$ gilt:

$$f(x) = y \iff \exists_{z \in E}\ z_0 x \vdash^* zy$$

Name: ..... **Matr.-Nr.:** .....

Aufgabe 2: **Primitive Rekursion** (14 Punkte)

Es soll gezeigt werden, dass $f\colon \mathbb{N}^2 \to \mathbb{N}$ primitiv rekursiv ist, wobei:

$$f(x,y) = \begin{cases} 1 & \text{falls } \sum_{i=0}^{x} i \geq y \\ 0 & \text{sonst.} \end{cases}$$

Dabei darf angenommen werden, dass folgende Funktionen primitiv rekursiv sind:

$$\begin{aligned} \mathrm{add}(x,y) &= x + y, \\ \mathrm{modsub}(x,y) &= \max\{0, x - y\}, \\ N(x) &= \begin{cases} 1 & \text{falls } x = 0 \\ 0 & \text{sonst.} \end{cases} \end{aligned}$$

Nutzen Sie zur Beantwortung der folgenden Fragen lediglich die auf dieser Seite aufgeführten Definitionen.

(a) Zeigen Sie, dass $s\colon \mathbb{N} \to \mathbb{N}, s(x) = \sum_{i=0}^{x} i$ primitiv rekursiv ist. (5 P)

(b) Zeigen Sie nun unter Verwendung von Aufgabenteil (a), dass f primitiv rekursiv ist. (5 P)

(c) Was ist $\mu(f)$? (**Ohne Begründung**) (4 P)

☐ $\mu(f)(y) = \begin{cases} 1 & \text{falls } y = 0 \\ 0 & \text{sonst} \end{cases}$

☒ $\mu(f)(y) = \begin{cases} \bot & \text{falls } y = 0 \\ 0 & \text{sonst} \end{cases}$

☐ $\mu(f)(y) = \min\{x \mid \sum_{i=0}^{x} i \geq y\}$

☐ $\mu(f)(y) = \min\{x \mid \sum_{i=0}^{y} i < x\}$

Hinweise/Definitionen:

Die Klasse der **primitiv-rekursiven Funktionen** ist die kleinste Klasse von Funktionen die

(a) folgende **Grundfunktionen** enthält:
   i) alle konstanten Funktionen $f\colon \mathbb{N}^k \to \mathbb{N}$ mit $f(x_1, \ldots, x_k) = c$
   ii) die Nachfolgerfunktion $\mathrm{suc}\colon \mathbb{N} \to \mathbb{N}$ mit $\mathrm{suc}(n) = n + 1$
   iii) die Projektionen $\pi_i^k\colon \mathbb{N}^k \to \mathbb{N}$ mit $\pi_i^k(x_1, \ldots, x_k) = x_i$

(b) **und** abgeschlossen ist unter folgenden Operationen:
   i) **Komposition** von $g_1, \ldots, g_m\colon \mathbb{N}^k \to \mathbb{N}$ und $h\colon \mathbb{N}^m \to \mathbb{N}$:

$$\begin{aligned} f\colon \mathbb{N}^k \to \mathbb{N} \quad \text{mit} \quad f = h \circ (g_1, \ldots, g_m) \text{ d.h.} \\ f(x_1, \ldots, x_k) = h(g_1(x_1, \ldots, x_k), \ldots, g_m(x_1, \ldots, x_k)) \end{aligned}$$

   ii) **primitive Rekursion** mit $g\colon \mathbb{N}^k \to \mathbb{N}$ und $h\colon \mathbb{N}^{k+2} \to \mathbb{N}$:

$$\begin{aligned} f\colon \mathbb{N}^{k+1} \to \mathbb{N} \quad \text{mit} \quad f = \mathrm{pr}(h,g) \text{ d.h.} \\ f(0, x_1, \ldots, x_k) = g(x_1, \ldots, x_k) \\ f(n+1, x_1, \ldots, x_k) = h(n, f(n, x_1, \ldots, x_k), x_1, \ldots, x_k). \end{aligned}$$

Der **μ-Operator** von $g\colon \mathbb{N}^{k+1} \to \mathbb{N}$ ist definiert als:

$$\begin{aligned} f\colon \mathbb{N}^k \to \mathbb{N} \quad \text{mit} \quad f = \mu(g) \text{ d.h.} \\ f(x_1, \ldots, x_k) := \min\{n \mid g(n, x_1, \ldots, x_k) = 0 \wedge \forall_{0 \leq i < n}\, g(i, x_1, \ldots, x_k) \neq \bot\} \end{aligned}$$

a) $s(0) = c_0^0$

$s(x+1) = add \circ (suc \circ (\pi_1^2), (\pi_2^2))(s) = add(suc(x), s(x))$

b) $f(0,y) = N(y)$

$f(x+1, y) = N \circ (modsub \circ (\pi_3^3, s \circ (suc \circ \pi_1^3)))(x, f(x,y), y)$

$= N(modsub(y, s(suc(x))))$

Aufgabe 3: **Ackermannfunktion** (12 Punkte)

Zeigen Sie: Die Ackermannfunktion a ist nicht LOOP-berechenbar.

Sei $f_P(n)$ definiert als die maximale Summe aller Variablenendwerte, die das Programm P erzeugen kann, wenn die initiale Belegung höchstens Summe n hat. Sie dürfen folgendes Lemma als bewiesen voraussetzen:

Lemma: Zu jedem LOOP-Programm P existiert ein $\ell \in \mathbb{N}$ sodass für alle $n \geq \ell$ gilt: $f_P(n) < \mathrm{a}(\ell, n)$.

Vorlesungfolien Seite 46

Annahme: a LOOP-berechenbar.
-> g(n) := a(n,n) LOOP-berechenbar vermöge LOOP-Programm P.
-> es gibt ein l $\in \mathbb{N}$, sodass für alle n ≥ l gilt: fp(n) < a(l,n).
-> g(l) ≤ fp(l) < a(l,l) = g(l).

Hinweise/Definitionen:

Die Ackermannfunktion $a\colon \mathbb{N}^2 \to \mathbb{N}$ ist definiert als:

$$\begin{aligned} & \mathrm{a}(0, y) := y + 1, \\ x > 0\colon \quad & \mathrm{a}(x, 0) := \mathrm{a}(x - 1, 1) \\ x, y > 0\colon \quad & \mathrm{a}(x, y) := \mathrm{a}(x - 1, \mathrm{a}(x, y - 1)) \end{aligned}$$

Für alle $x, y \in \mathbb{N}$ gilt:

1. $y < \mathrm{a}(x, y)$
2. $\mathrm{a}(x, y) < \mathrm{a}(x, y + 1)$
3. $\mathrm{a}(x, y + 1) \leq \mathrm{a}(x + 1, y)$
4. $\mathrm{a}(x, y) < \mathrm{a}(x + 1, y)$
5. $\mathrm{a}(x + 1, 2y) < \mathrm{a}(2x + 2, y)$

Name: ..... **Matr.-Nr.:** .....

Aufgabe 4: **Entscheidbarkeit** (16 Punkte)

Begründen Sie für die folgenden Sprachen, ob sie entscheidbar sind oder nicht.

Zur Begründung von Entscheidbarkeit reicht es einen Algorithmus zur Berechnung der charakteristischen Funktion in *natürlicher Sprache* zu beschreiben.

(a) $L = \{w \in \{0,1\}^* \mid \text{Die von } M_w \text{ berechnete Funktion ist LOOP-berechenbar}\}$ (5 P)

(b) $L = \{w \in \{0,1\}^* \mid \text{Die von } M_w \text{ berechnete Funktion ist WHILE-berechenbar}\}$ (5 P)

(c) $L = \left\{ w \in \{0,1\}^* \;\middle|\; \begin{array}{l}\text{Der Lese-Schreibkopf von } M_w \text{ auf Eingabe } w \\ \text{zeigt auf jede Bandzelle höchstens ein Mal}\end{array} \right\}$ (6 P)

a) Satz von Rice

R= Menge aller Turing-berechenbaren Funktionen

S = Menge aller LOOP-berechenbaren Funktionen

liegt berechnete Funktion von M_W in S -> unentscheidbar

b) Die Sprache sammelt die Kodierungen aller Turing Maschinen. L=R

$$\chi_L := \begin{cases} 1, \text{ wenn w Kodierung einer Turing Maschine} \\ 0, \text{ sonst} \end{cases}$$

c) Man baut eine Zertifikat-Maschine, wobei das Zertifikat die Bewegungsrichtung ist.

Man simuliert die Turing-Maschine M_W in folgenden 3 Schritte:

1. So lange bis in der zertifizierten Richtung vom Kopf aus nur blancs stehen. (Verwerfen, wenn M_W sich nicht in die zertifizierte Richtung bewegt)
2. Ab dem Moment merkt man sich die Zustände die M verwendet. (Verwerfen, wenn M_W sich nicht in die zertifizierte Richtung bewegt)
3. Wenn sich ein Zustand wiederholt, wird akzeptiert.

Hinweise/Definitionen:

Eine Sprache $A \subseteq \Sigma^*$ heißt **entscheidbar**, falls die charakteristische Funktion $\chi_A : \Sigma^* \to \{0,1\}$ berechenbar ist.

$$\chi_A(x) := \begin{cases} 1, & \text{falls } x \in A, \\ 0, & \text{falls } x \notin A. \end{cases}$$

Aufgabe 5: **PCP-Varianten** (16 Punkte)

Für ein endliches Alphabet Σ ist das Postsche Korrespondenzproblem die Menge

$$\mathrm{PCP} := \left\{ \left\langle \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, \ldots, \begin{pmatrix} x_\ell \\ y_\ell \end{pmatrix} \right\rangle \;\middle|\; \begin{array}{r} \ell \geq 1, x_i, y_i \in \Sigma^*, \exists_{n \geq 1}\, \exists_{i_1, i_2, \ldots, i_n \in \{1,2,\ldots,\ell\}} : \\ x_{i_1} \cdot x_{i_2} \cdot \ldots \cdot x_{i_n} = y_{i_1} \cdot y_{i_2} \cdot \ldots \cdot y_{i_n} \end{array} \right\}$$

Wir nennen n die *Länge* der Lösung. Begründen Sie für die folgenden PCP-Varianten, ob sie entscheidbar oder unentscheidbar sind.

Zur Begründung von Entscheidbarkeit reicht es einen Algorithmus zur Berechnung der charakteristischen Funktion in *natürlicher Sprache* zu beschreiben.

(a) k-$\mathrm{PCP}_a = \{P \in \Sigma^* \mid P \in \mathrm{PCP}$ und es existiert eine Lösung der Länge höchstens $k\}$

(b) k-$\mathrm{PCP}_b = \{P \in \Sigma^* \mid P \in \mathrm{PCP}$ und es existiert eine Lösung der Länge genau $k\}$

(c) k-$\mathrm{PCP}_c = \{P \in \Sigma^* \mid P \in \mathrm{PCP}$ und es existiert eine Lösung der Länge mindestens $k\}$

In Kurz:

a) man kann über alle Kombinationen mit Länge <= k iterieren -> berechenbar.
b) ähnlich, mit Länge = k -> berechenbar
c) es gibt genau dann eine Lösung für k-PCP, wenn es eine PCP Lösung gibt -> unberechenbar

In Lang:

a) die PCP-Variante ist entscheidbar, da ein Algorithmus existiert der dies berechnen kann. Man würde bei n=1 beginnen und alle möglichen (endlichen) Kombinationen bis n=k ausprobieren. Hat man nach allen möglichen Kombinationen bis einschließlich n=k kein gültiges PCP gefunden, wird 0 zurückgegeben. Wird im laufe des Durchlaufs ein gültiges PCP gefunden, stoppt der Algorithmus und gibt eine 1 zurück.

b) die PCP-Variante ist ebenfalls entscheidbar, der Algorithmus ist ähnlich zu a), mit dem Unterschied, dass nicht bei n=1 gestartet wird, sondern bei n=k (endliche Kombinationen). Wird ein gültiges PCP mit n=k gefunden, stoppt der Algorithmus und gibt eine 1 zurück. Wird kein gültiges gefunden und es wurden alle Varianten mit n=k ausprobiert, stoppt der Algorithmus und gibt eine 0 zurück.

Hinweis: Sie wollten das man explizit erwähnt das es endlich viele Möglichkeiten sind oder eine genau die Anzahl an Kombinationsmöglichkeiten nennt, sonst hat man pro Teilaufgabe einen Punkt verloren.
Bei l Steinen wäre das:
a) $\sum_{n=1}^{k} l^n$ mögliche Kombinationen
b) l^k mögliche Kombinationen

c)

1. wenn es für ein PCP eine Lösung gibt, kann man diese so oft wiederholen bis die Länge k überschritten wurde.
2. existiert eine k-PCP Lösung, ist diese auch eine PCP Lösung.
3. Daraus folgt, das k-PCP mindestens so schwer wie PCP und somit unberechenbar ist.

Name: ..... **Matr.-Nr.:** .....

Aufgabe 6: **NTM konstruieren** (14 Punkte)

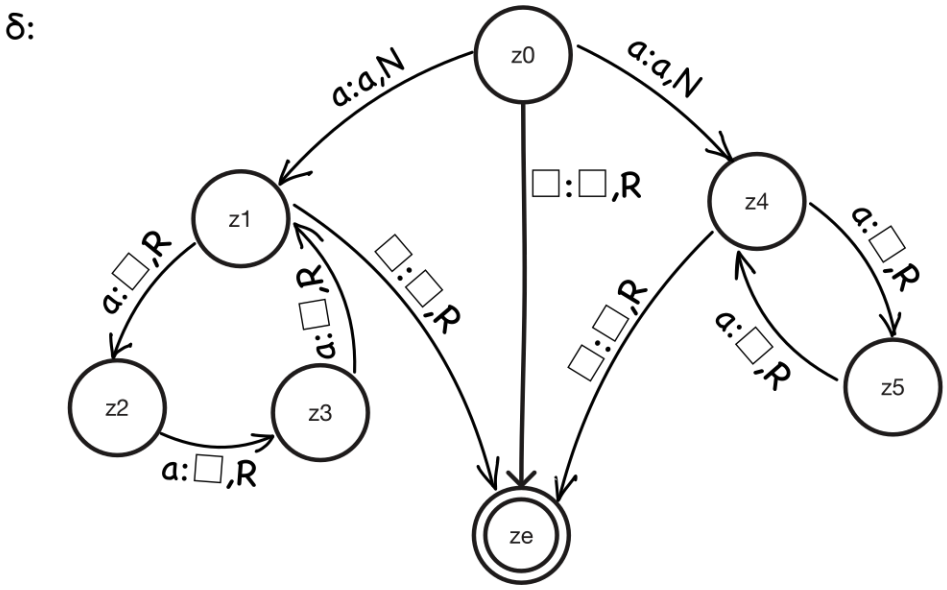
Geben Sie (**ohne Begründung**) eine nichtdeterministische Einband-Turing-Maschine M mit Eingabealphabet $\Sigma = \{a\}$ und höchstens **sieben** Zuständen an, für die gilt

$$T(M) = \{a^n \mid n \text{ ist durch zwei oder drei teilbar}\}.$$

Sie dürfen die Übergangsrelation graphisch angeben.

M = ({z0, z1, z2, z3, z4, z5, ze}, {a}, {a, □}, δ, z0, □, {ze})

δ:



Aufgabe 7: **Polynomzeitreduktion** (20 Punkte)

Betrachten Sie die folgenden beiden Probleme.

CLIQUE
Eingabe: Graph $G = (V, E)$, $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenteilmenge $X \subseteq V$ mit $|X| = k$, sodass für jedes Knotenpaar $u, v \in X$ gilt $\{u, v\} \in E$?

ALMOST CLIQUE
Eingabe: Graph $G = (V, E)$, $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenteilmenge $X \subseteq V$ mit $|X| = k$, sodass für genau ein Knotenpaar $u, v \in X$ gilt $\{u, v\} \notin E$?

Eine (unvollständige) Polynomzeitreduktion CLIQUE $\leq_m^p$ ALMOST CLIQUE ist gegeben durch die Funktion $\langle (G = (V = \{v_1, v_2, \ldots, v_n\}, E), k) \rangle \mapsto \langle (G' = (V', E'), k') \rangle$, wobei

$$V' = V \cup \{x, y\}$$
$$E' = E \cup \{\{x, v_i\} \mid v_i \in V\} \cup \{\{y, v_i\} \mid v_i \in V\}$$
$$k' = \boxed{\text{k + 2}}$$

(a) Setzen Sie einen Wert für k' ein, sodass die Reduktion korrekt ist. (**ohne Begründung**) (3 P)

Zeigen Sie die Korrektheit der Reduktion, das heißt zeigen Sie dass für alle (G, k) gilt:

(b) $\langle (G, k) \rangle \in$ CLIQUE $\implies \langle (G', k') \rangle \in$ ALMOST CLIQUE (8 P)

(c) $\langle (G', k') \rangle \in$ ALMOST CLIQUE $\implies \langle (G, k) \rangle \in$ CLIQUE (9 P)

b) Sei ⟨(G,k)⟩ ∈ CLIQUE, dann gilt dass es eine Knotenteilmenge X ⊆ V mit |X| = k gibt, sodass alle Knoten untereinander verbunden sind. Wir beweisen nun, dass es auch eine ALMOST CLIQUE ist.
Nach Konstruktion werden 2 Knoten x und y hinzugefügt, welche mit allen anderen Knoten verbunden sind, dadurch entshet eine Clique der Größe k+1.
Da x und y nicht untereinander verbunden sind, entsteht eine ALMOST CLIQUE der Größe k+2, in welcher genau 1 Knotenpaar nicht verbunden ist.
Somit gilt ⟨(G`,k`)⟩ ∈ ALMOST CLIQUE.

c) Angenommen wir haben eine ALMOST CLIQUE $A \subseteq V'$ in G' mit $|A| = k+2$.
Wir bauen die ALMOST CLIQUE zunächst wie folgt um:
Für das Knotenpaar $\{u,v\}$ mit $u,v \in A$, für welches gilt das $\{u,v\} \notin E'$, tauschen wir diese zwei Knoten jeweils für x und y ein. x und y sind jeweils zu allen anderen Knoten verbunden, weswegen sie vereinzelt nicht die Eigenschaft der CLIQUE kaputt machen, haben aber keine Kante zueinander wodurch die Eigenschaften einer ALMOST CLIQUE weiterhin erfüllt werden.
Wir wählen unsere CLIQUE $C \subseteq V$ in G nun wie folgt:
$C = \{ v \mid v \in A \land v \neq x \land v \neq y\}$.
Wir schmeißen für unsere Clique also genau das Paar x,y raus.
Da das dass einzige Paar ohne Kante zueinander war, haben wir nun eine CLIQUE.
Da wir 2 Knoten rausschmeißen, ist $|C| = |A|-2 = k+2 - 2 = k$.
Wir haben unser Budget also genau eingehalten.

Name: ..... **Matr.-Nr.:** .....

Aufgabe 8: **Vermischtes zu P, NP, coNP, PSPACE** (16 Punkte)

(a) Begründen Sie kurz folgende Teilmengenbeziehungen: (10 P)
LOGSPACE $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE

(b) Ist eine der in Aufgabenteil (a) genannten Teilmengenbeziehungen strikt ($\subsetneq$)? (3 P)

(c) Welche der folgenden Aussagen sind bewiesenermaßen korrekt? (**ohne Begründung**) (3 P)

- [x] $\mathrm{DTIME}(n) \subseteq \mathrm{NTIME}(n^2)$
- [ ] $\mathrm{NTIME}(n^2) \subseteq \mathrm{NTIME}(n)$
- [ ] $\mathrm{NTIME}(n) \subseteq \mathrm{DTIME}(n^2)$
- [ ] $\mathrm{NTIME}(n^2) \subseteq \mathrm{DTIME}(n)$

a)

LOGSPACE ⊆ P:

Die DTM, die f in Logarithmischem Raum berechnet, kann maximal durch $|Q| \cdot \log(n) \cdot 2^{\log(n)}$ Konfigurationen laufen, das sind $|Q| \cdot \log(n) \cdot n$, was in $O(n^2)$ liegt.

P ⊆ NP:

Jede DTM ist auch eine NTM.

NP ⊆ PSPACE:

NP = Sprachen, die in P-Zeit von einer Zertifikat-Maschine berechnet werden können. Da das Zertifikat polynomiell groß ist und die Maschine in P-Zeit nur P-Raum beschreiben kann, ist die Maschine durchs Iterieren über die Zertifikate in P-Raum simulierbar.

b) Nach dem Platzhierarchiesatz ist LOGSPACE eine strikte Teilmenge von PSPACE

Aufgabe 9: **Komplexität von Problemen** (8 Punkte)

Kreuzen Sie alle vollständig korrekten Aussagen über folgendes Problem an (**ohne Begründung**):

Exact Vertex Cover (EVC)
Eingabe: Graph $G = (V, E)$, $k \in \mathbb{N}$.
Frage: Gibt es eine Knotenmenge $X \subseteq V$ mit $|X| = k$, sodass $e \cap X \neq \emptyset$ für alle $e \in E$ und gibt es keine Knotenmenge $X' \subseteq V$ mit $|X'| = k - 1$, sodass $e \cap X' \neq \emptyset$ für alle $e \in E$?

(a) ☐ EVC $\in$ NP. Als Zertifikat für Ja-Instanzen dient eine Knotenmenge X. Eine deterministische Turing-Maschine kann in Polynomzeit überprüfen, ob X ein Vertex Cover der Größe k ist. (1 P)

(b) ☐ EVC $\in$ coNP. Als Zertifikat für Nein-Instanzen dient eine Knotenmenge X der Größe $k-1$. Eine deterministische Turing-Maschine kann in Polynomialzeit überprüfen, ob X ein Vertex Cover der Größe $k - 1$ ist. (1 P)

(c) ☒ EVC $\in$ PSPACE. Mit polynomiellem Platzverbrauch kann eine deterministische Turing-Maschine alle Teilmengen von Knoten betrachten und überprüfen, ob ein Vertex Cover der Größe k existiert und keines der Größe $k - 1$. (1 P)

(d) ☐ EVC $\in$ P. Es reicht alle Teilmengen von Knoten der Größe k und $k - 1$ zu betrachten. Für jede Teilmenge kann eine deterministische Turing-Maschine jeweils in Polynomialzeit überprüfen, ob sie ein Vertex Cover darstellt. Da es lediglich $O(|V|^k)$ Teilmengen der Größe k und $k - 1$ gibt, ergibt sich daraus ein Algorithmus, dessen Laufzeit polynomiell in der Eingabegröße ist. (1 P)

Kreuzen Sie alle vollständig korrekten Aussagen über folgendes Problem an (**ohne Begründung**):

2-Numerierbarkeit (2-NUM)
Eingabe: Zusammenhängender Graph $G = (V, E)$.
Frage: Gibt es eine Funktion $f\colon V \to \{1, 2\}$, sodass für jede Kante $\{u, w\} \in E$ gilt: $f(u) > f(w)$ oder $f(u) < f(w)$?

(e) ☒ 2-NUM $\in$ NP. Als Zertifikat für Ja-Instanzen kann eine Funktion f angegeben werden. Eine deterministische Turing-Maschine kann in Polynomialzeit überprüfen, dass für alle Kanten $\{u, w\} \in E$ gilt: $f(u) > f(w)$ oder $f(u) < f(w)$. (1 P)

(f) ☒ 2-NUM $\in$ coNP. Als Zertifikat für Nein-Instanzen kann ein Teilgraph $H \subseteq G$ angegeben werden. Eine deterministische Turing-Maschine kann in Polynomialzeit überprüfen, ob H einen Kreis ungerader Länge in G bildet. Ist dies der Fall kann keine Funktion f mit den gewünschten Eigenschaften existieren und die Maschine kann ablehnen. (1 P)

(g) ☒ 2-NUM $\in$ PSPACE. Mit polynomiellem Platzverbrauch kann eine deterministische Turing-Maschine alle Funktionen f ausprobieren. Ist eine davon eine valide 2-Numerierung, wird akzeptiert, sonst abgelehnt. (1 P)

(h) ☒ 2-NUM $\in$ P. Folgender Algorithmus löst 2-NUM in Polynomialzeit. Beginne bei einem beliebigen Knoten v und weise diesem die Zahl 1 zu. Führe den nachfolgenden Schritt so lange wie möglich aus: Solange es noch einen Knoten w ohne Nummerierung gibt, dessen bereits nummerierten Nachbarn alle die gleiche Nummer haben, gib w die dazu komplementäre Nummer. Wenn alle Knoten nummeriert sind, gib Ja aus, ansonsten Nein. (1 P)

Name: ..... **Matr.-Nr.:** .....

Theoreme aus der Vorlesung

Theorem 1. Zu jeder k-Band-TM M gibt es eine Einband-TM Q mit $T(M) = T(Q)$ (bzw. $f_M = f_Q$).

Theorem 2. Sei L eine Liste aller 1-stelligen, LOOP-berechenbaren Funktionen. Dann ist $g\colon \mathbb{N} \to \mathbb{N}$ mit

$$g(n) := L_n(n) + 1$$

total und *nicht LOOP-berechenbar.*

Theorem 3. Eine Sprache A ist genau dann rekursiv aufzählbar, wenn sie semi-entscheidbar ist.

Theorem 4. Es gibt eine universelle Turing-Maschine M^U, die bei Eingabe $w, x \in \{0,1\}^*$ die Turing-Maschine M_w auf x simuliert. Wenn M_w auf x in $t \in \mathbb{N}$ Schritten hält, dann hält M^U auf w, x in höchstens $c \cdot t \cdot \log t$ Schritten, wobei $c \in \mathbb{N}$ von M_w aber nicht von x abhängt.

Theorem 5. Das spezielle Halteproblem $K := \{w \in \{0,1\}^* \mid M_w \text{ hält auf Eingabe } w\}$ ist unentscheidbar.

Theorem 6 (Satz von Rice)**.** Sei $\mathcal{R}$ die Menge aller Turing-berechenbaren Funktionen. Sei $\mathcal{S} \subseteq \mathcal{R}$ eine nicht-triviale Teilmenge von $\mathcal{R}$ (d.h. $\mathcal{S} \neq \emptyset$ und $\mathcal{S} \neq \mathcal{R}$). Dann ist $\mathcal{C}(\mathcal{S}) := \{w \mid$ die von M_w berechnete Funktion liegt in $\mathcal{S}\}$ unentscheidbar.

Theorem 7 (Satz von Rice für Sprachen)**.** Sei $\mathcal{A}$ die Menge aller Typ-0 Sprachen. Sei $\mathcal{S} \subseteq \mathcal{A}$ eine nicht-triviale Teilmenge von $\mathcal{A}$ (d.h. $\mathcal{S} \neq \emptyset$ und $\mathcal{S} \neq \mathcal{A}$). Dann ist $\mathcal{C}(\mathcal{S}) := \{w \mid T(M_w) \in \mathcal{S}\}$ unentscheidbar.

Theorem 8 (deterministischer Zeithierarchiesatz)**.** Seien $f, g\colon \mathbb{N} \to \mathbb{N}$ monoton wachsend und zeitkonstruierbar mit $f \log(f) \in o(g)$. Dann gilt:

$$\mathrm{DTIME}\,(f(n)) \subsetneq \mathrm{DTIME}\,(g(n))$$

Theorem 9 (nichtdeterministischer Zeithierarchiesatz)**.** Seien $f, g\colon \mathbb{N} \to \mathbb{N}$ monoton wachsend und zeitkonstruierbar mit $f(n+1) \in o(g(n))$. Dann gilt:

$$\mathrm{NTIME}\,(f(n)) \subsetneq \mathrm{NTIME}\,(g(n))$$

Theorem 10 (Satz von Cook und Levin)**.** SAT ist NP-vollständig.

Theorem 11. Vertex Cover$\leq_m^p$Hitting Set.

Theorem 12. Tautologie ist coNP-vollständig.

Theorem 13 (deterministische Platzhierarchiesatz)**.** Seien $f, g\colon \mathbb{N} \to \mathbb{N}$ monoton wachsend und platzkonstruierbar mit $f \in o(g)$. Dann gilt:

$$\mathrm{DSPACE}\,(f(n)) \subsetneq \mathrm{DSPACE}\,(g(n))$$