1. 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	1	3	2	–	8					–	
Aufg. 2	2	2	3	1	3	11						
Aufg. 3	2	4	1	2	–	9					–	
Aufg. 4	3	8	–	–	–	11			–	–	–	
Aufg. 5	3	3	6	–	–	12				–	–	
Aufg. 6	1	1	3	4	–	9					–	
Σ						60						

Problem 1: Automaten 2+1+3+2=8 Punkte

Sei $L \subseteq \Sigma^*$ eine Sprache. Wir haben in der Vorlesung für ein Wort $w \in \Sigma^*$ die Menge der gültigen Suffixe $S_w(L)$ bezüglich L definiert:

$$S_w(L) \coloneqq \{x \in \Sigma^* \mid wx \in L\}.$$

(a) Für ein Wort $w \in \{0,1\}^*$ und $k \in \mathbb{N}^+$ bezeichnen wir mit w_k das k-te Zeichen von w. Sei

$$L_3 \coloneqq \left\{ w \in \{0,1\}^* \mid w_{3i} = 1 \text{ für alle } i \in \mathbb{N}^+ \text{ mit } 1 \leqslant i \leqslant \frac{|w|}{3} \right\}$$

die Sprache in $\{0,1\}^*$, für die jedes dritte Zeichen eine 1 ist. Beispielsweise ist $01\mathbf{1}0 \in L_3$, aber $01\mathbf{0}0 \notin L_3$.

Geben Sie einen DEA an, der für das Wort 01 die Menge der gültigen Suffixe $S_{01}(L_3)$ erkennt.

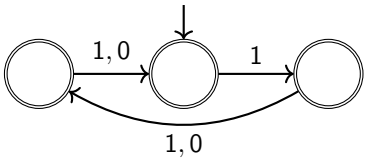
(b) Welche Wörter enthält $S_\varepsilon(L)$ für eine beliebige Sprache L?

(c) Beweisen Sie die folgende Aussage. Eine Sprache L ist genau dann regulär, wenn $S_w(L)$ für alle $w \in \Sigma^*$ regulär ist.

(d) Beweisen Sie, dass es eine nichtreguläre Sprache L gibt, bei der $S_w(L)$ für unendlich viele $w \in \Sigma^*$ regulär ist. Sie müssen nicht zeigen, dass L nicht regulär ist.

Lösung:

(a) Der folgende DEA akzeptiert $S_{01}(L_3)$:



(b) $S_\varepsilon(L) = \{x \mid \varepsilon x \in L\} = L$

(c) Ist L regulär, so existiert ein DEA $D = (\Sigma, Q, \delta, s, F)$, der L erkennt. Für jedes Wort $w \in \Sigma^*$ konstruieren wir einen DEA, der $S_w(L)$ erkennt. Dazu betrachten wir den DEA D und ändern lediglich den Startzustand zum Zustand $\delta(s, w)$. Die Wörter in $S_w(L)$ sind genau jene, die durch Hinzufügen des Präfixes w in L liegen. Also erkennt der erhaltene Automat, der die Abarbeitung des Präfixes w in L vorwegnimmt, die Sprache $S_w(L)$. Somit ist die Sprache $S_w(L)$ regulär.

Ist $S_w(L)$ regulär für jedes $w \in \Sigma^*$, so gilt dies insbesondere für $w = \varepsilon$. Da $S_\varepsilon(L) = L$ ist, folgt die Behauptung.

(d) Wähle $L = \{0^i 1^i \mid \forall i \in \mathbb{N}\} \subseteq \{0,1\}^*$. Aus der Vorlesung ist bekannt, dass L nicht regulär ist. Für alle (unendlich vielen) $w = 1^i$ (mit $i \in \mathbb{N}$) ist $S_w(L) = \emptyset$, also insbesondere regulär.

Hinweis: Für $w = 0^i$ ist $S_w(L) = \{0^j 1^k \mid j, k \in \mathbb{N}_0, j + i = k\}$ nicht regulär!

Problem 2: 1-Zustands DPDA $2 + 2 + 3 + 1 + 3 = 11$ Punkte

Ein 1-Zustands-DPDA ist ein deterministischer Kellerautomat, der nur einen Zustand hat. In dieser Aufgabe betrachten wir nur Kellerautomaten, die mit leerem Stack akzeptieren.

Hinweis: Beim Anwenden der Übergangsfunktion eines DPDA wird zuerst das oberste Stacksymbol eingelesen. Wenn der Stack leer ist und noch ein Zeichen eingelesen werden soll, kann der DPDA somit nicht weiterarbeiten und nicht mehr akzeptieren.

(a) Betrachten Sie den 1-Zustands-DPDA A mit einzigem Zustand q, Alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, Stackalphabet $\Gamma = \{Z_0, Z_1, Z_2, X\}$, Stackinitialisierung Z_0 und folgender Übergangsfunktion:

$$\delta(q, \mathsf{a}, Z_0) = (q, Z_1)$$
$$\delta(q, \mathsf{b}, Z_1) = (q, Z_0)$$
$$\delta(q, \mathsf{c}, Z_0) = (q, \varepsilon).$$

In allen bisher nicht definierten Fällen wird das gelesene Stacksymbol durch ein X überschrieben.

Geben Sie einen DEA A' mit Zustandsmenge $Q = \{Z_0, Z_1, Z_2, X\}$ an, der die gleiche Sprache wie A erkennt. Hierbei ist X der explizite Müllzustand von A'.

(b) Zeigen Sie, dass jede Sprache, die von einem 1-Zustands-DPDA mit leerem Stack erkannt wird, präfixfrei ist.

Hinweis: Wir nennen eine Sprache L über einem Alphabet Σ präfixfrei, wenn es für kein $w \in L$ ein $z \in \Sigma^+$ gibt, sodass $wz \in L$.

(c) Geben Sie eine nichtreguläre Sprache L an, die von einem 1-Zustands-DPDA mit leerem Stack erkannt werden kann. Beschreiben Sie einen solchen 1-Zustands-DPDA. Sie müssen nicht zeigen, dass L nicht regulär ist.

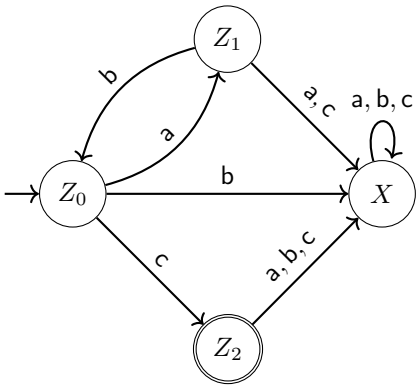
(d) Zeigen Sie, dass es reguläre Sprachen gibt, die nicht von einem 1-Zustands-DPDA mit leerem Stack erkannt werden können.

Wir betrachten nun 1-Zustands-NPDAs. Ein 1-Zustands-NPDA ist ein nichtdeterministischer Kellerautomat mit nur einem Zustand.

(e) Zeigen Sie, dass es für jeden gegebenen NEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ einen 1-Zustands-NPDA gibt, der die gleiche Sprache wie A erkennt.

Lösung:

(a) Der folgende DEA erkennt die Sprache $(\mathsf{ab})^*\mathsf{c}$, die von dem gegebenen 1-Zustands-DPDA erkannt wird:



(b) Sei L' eine Sprache, die ein Wort w und ein anderes Wort $w' = ww''$ enthält. Nehmen wir an, L' wird von einem 1-Zustands-DPDA mit leerem Stack erkannt. Um w mit leerem Stack zu erkennen, muss die Übergangsfolge $(q, w, Z_0) \overset{*}{\mapsto} (q, \varepsilon)$ existieren. Selbst wenn diese Übergangsfolge aus mehreren Übergängen (inklusive ε-Übergängen) besteht, muss diese Übergangsfolge für jedes Wort, das mit w anfängt, vollständig abgearbeitet werden (also insbesondere auch für w'), da wir deterministische Kellerautomaten betrachten.

Bei Eingabe von w' werden zunächst die gleichen Schritte durchgeführt, wie bei der Abarbeitung von w (da der Kellerautomat deterministisch ist). Der DPDA wird also in die Konfiguration (q, w'', ε) überführt. Das heißt der Stack ist leer. Somit gibt es aber keinen weiteren Übergang, die Eingabe wurde aber noch nicht vollständig eingelesen. Das Wort w' wird nicht akzeptiert. Das ist ein Widerspruch, da wir angenommen hatten, dass der 1-Zustands-DPDA, die Sprache L' erkennt.

(c) Die nichtreguläre Sprache $L = \{\mathsf{a}^n\mathsf{b}^n \mid n \in \mathbb{N}^+\}$ über $\Sigma = \{\mathsf{a}, \mathsf{b}\}$, wird vom DPDA $D = (\{q\}, \Sigma, \{Z_0, A_0, A, F\}, Z_0, \delta, \emptyset)$ mit
$\delta(q, \mathsf{a}, Z_0) = (q, A_0)$
$\delta(q, \mathsf{a}, A_0) = (q, A_0A)$
$\delta(q, \mathsf{b}, A_0) = (q, \varepsilon)$
$\delta(q, \mathsf{b}, A) = (q, \varepsilon)$
$\delta(\text{else}) = (q, F)$
erkannt.

(d) Jede reguläre Sprache, die ein Wort enthält, das ein Präfix eines anderen Worts in der Sprache ist, z.B. $L' = \{\mathsf{a}, \mathsf{aa}\}$. Nach (b) kann kein 1-Zustands-DPDA L' erkennen.

(e) Jede Sprache, die von einem NEA erkannt wird, kann auch von einem DEA erkannt werden. Deswegen genügt es die folgende Aussage zu zeigen. Sei $D = (\{q_1, \ldots, q_k\}, \Sigma, q_1, \delta, F)$ ein DEA. Dann existiert eine 1-Zustands-NPDA $P = (\{q\}, \{Z_1, \ldots, Z_k\}, \Sigma, q, Z_1, \delta', \varnothing)$ welcher mit leerem Stack die gleiche Sprache wie D akzeptiert.

Der Kellerautomat P hat nur einen Zustand q. Für jeden Zustand q_i des DEAs existiert ein zugehöriges Stacksymbol Z_i. Das initiale Stacksymbol entspricht dem Startzustand des DEAs. Die Übergangsfunktion δ' definieren wir für alle $i \in \{1, \ldots, k\}$ und $a \in \Sigma$ mittels

$$\begin{aligned} \delta'(q, a, Z_i) &= (q, Z_k) && \text{wobei } q_k = \delta(q_i, a) \\ \delta'(q, \varepsilon, Z_i) &= (q, \varepsilon) && \text{falls } \delta(q_i, a) \in F. \end{aligned}$$

Die Intuition ist die folgende. Wird ein Wort vom DEA abgearbeitet, so simuliert der 1-Zustands-NPDA das Verhalten, indem er den aktuellen Zustand auf dem Stack speichert. Der Stack hat dabei immer Höhe 1. Ist der aktuelle Zustand akzeptierend, so kann der 1-Zustands-NPDA ohne ein Symbol des Wortes zu lesen das oberste (und auch einzige) Stacksymbol löschen. Danach ist der Stack leer. Wurde das Wort vollständig abgearbeitet, so akzeptiert der 1-Zustands-NPDA.

Wird ein Wort w vom DEA akzeptiert, so endet die Abarbeitung des DEAs in einem akzeptierenden Zustand. Somit existiert auch eine Folge von Zustandsübergängen des NPDAs, so dass dieser ein Zeichen Z_i auf dem Stack stehen hat, der zugehörige Zustand q_i des DEAs akzeptierend ist und das Wort w vollständig abgearbeitet wurde. Somit gibt es einen Übergang, bei dem Z_i vom Stack gelöscht wird und der NPDA mit leerem Stack akzeptiert.

Wird ein Wort w vom NPDA akzeptiert, so folgt analog, dass w auch vom DEA akzeptiert wird.

Folglich akzeptiert der NPDA die gleiche Sprache wie der DEA D.

Problem 3: Grammatiken 2+4+1+2 = 9 Punkte

Wir betrachten das Alphabet $\Sigma \coloneqq \{\mathsf{a}, \mathsf{b}\}$. Für ein Wort $w \in \Sigma^*$ und ein Zeichen $x \in \Sigma$ bezeichnen wir mit $|w|_x$ die Anzahl an Vorkommen des Zeichens x im Wort w. Sei

$$L \coloneqq \{w \in \Sigma^* \mid |w|_{\mathsf{a}} = |w|_{\mathsf{b}}\},$$

die Sprache, die alle Wörter enthält, die genauso häufig das Zeichen b wie das Zeichen a enthalten.

(a) Zeigen Sie, dass L nicht regulär ist.

Betrachten Sie die kontextfreie Grammatik $G = (\Sigma, \{S, A, B\}, S, R)$ mit Ableitungsregeln

$$\begin{aligned} S &\to \varepsilon \mid \mathsf{a}B \mid \mathsf{b}A \\ A &\to S\mathsf{a}S \\ B &\to S\mathsf{b}S. \end{aligned}$$

Wir ordnen Wörtern Gewichte zu. Dazu definieren wir für $w \in \Sigma^*$ das Gewicht $c(w)$ als

$$c(w) = |w|_{\mathsf{a}} - |w|_{\mathsf{b}}.$$

Ferner definieren wir für ein Nichtterminal X der Grammatik G die Grammatik $G_X = (\Sigma, \{S, A, B\}, X, R)$. Die Sprache, die von G_X erzeugt wird, besteht also aus allen Wörtern, die von X abgeleitet werden können.

(b) Sie dürfen ohne Beweis annehmen, dass die folgenden beiden Gleichungen gelten.

$$L(G_A) = \{w \in \Sigma^* \mid c(w) = 1\} \tag{1}$$
$$L(G_B) = \{w \in \Sigma^* \mid c(w) = -1\} \tag{2}$$

Zeigen Sie die folgende Gleichung.

$$L(G_S) = \{w \in \Sigma^* \mid c(w) = 0\}.$$

Hinweis: Achten Sie darauf, beide Inklusionen zu zeigen.

(c) Ist L kontextfrei? Begründen Sie.

Wir nennen eine Turingmaschine *in-place*, wenn sie bei der Abarbeitung nur Zellen beschreibt, in denen zu Beginn die Eingabe stand.

(d) Beschreiben Sie eine nicht-deterministische in-place Turingmaschine, die die Sprache L entscheidet.

Lösung:

(a) **Variante 1: Nerode** Wir zeigen, dass L unendlich viele verschiedene Äquivalenzklassen bezüglich der Nerode-Relation hat. Dazu weisen wir nach, dass für alle $i, j \in \mathbb{N}_0$ mit $i \neq j$ die Wörter a^i und a^j nicht äquivalent bezüglich der Nerode-Relation sind. Es gilt $a^i b^i \in L$, jedoch $a^j b^i \notin L$ für $i \neq j$. Daraus folgt die Behauptung.

Variante 2: Pumping-Lemma Betrachte $a^n b^n = uvx$ und beliebige Zerlegung $|uv| \leqslant n$, $v \neq \varepsilon$. Dann enthält v nicht das Zeichen b, und es folgt $\left|uv^0x\right|_{\mathsf{a}} < n$. Es gilt jedoch $\left|uv^0x\right|_{\mathsf{b}} = |uvx|_{\mathsf{b}} = n$. Somit folgt $uv^0x \notin L$ und das Pumping-Lemma ist nicht erfüllt.

(b)

Behauptung. $L(G_S) \subseteq \{w \in \Sigma^* \mid c(w) = 0\}$.

Beweis. Sei $w \in L(G_S)$. Ist $w = \varepsilon$, so gilt insbesondere $c(w) = 0$. Anderenfalls existiert eine Kette $S \to \mathsf{a}B \to^* w$ oder eine Kette $S \to \mathsf{b}A \to^* w$. Wir betrachten nur den ersten Fall, der andere ist analog. Es existiert also ein Wort $w_B \in L(G_B)$, sodass $w = \mathsf{a}w_B$. Mit (2) gilt $c(w_B) = -1$. Es folgt $c(w) = c(\mathsf{a}) + c(w_B) = 1 - 1 = 0$. □

Behauptung. $L(G_S) \supseteq \{w \in \Sigma^* \mid c(w) = 0\}$.

Beweis. Sei $w \in \Sigma^*$ mit $c(w) = 0$.

Fall 1. Ist $w = \varepsilon$, so folgt $w \in L(G_S)$, da es eine Ableitung $S \to \varepsilon$ gibt.

Fall 2. Ist das erste Zeichen von w ein a, so folgt $w = \mathsf{a}w'$ wobei $w' \in \Sigma^*$. Wegen $c(w) = c(\mathsf{a}) + c(w')$ ergibt sich $c(w') = -1$, also ist $w' \in L(G_B)$ nach (2). Wir erhalten $S \to \mathsf{a}B \to^* \mathsf{a}w' = w$.

Fall 3. Ist das erste Zeichen von w ein b, so folgt $w = \mathsf{b}w'$ wobei $w' \in \Sigma^*$. Wegen $c(w) = c(\mathsf{b}) + c(w')$ ergibt sich $c(w') = 1$, also ist $w' \in L(G_A)$ nach (1). Wir erhalten $S \to \mathsf{b}A \to^* \mathsf{a}w' = w$.

Somit gilt $w \in L(G_S)$. □

(c) Es gilt $L = L(G_S) = L(G)$ nach b. Somit ist L kontextfrei.

(d) Wir verwenden das Bandalphabet $\{\mathsf{a}, \mathsf{b}, X, \sqcup\}$. Die Turingmaschine geht die Eingabe von links nach rechts durch. Findet sie das Zeichen a, so überschreibt sie es mit X. Sie geht zurück zum Beginn der Eingabe und geht von links nach rechts das Wort durch. Das erste Vorkommen von b werden mit X überschrieben. Findet Sie kein b, so lehnt sie ab. Sonst geht sie zurück zum Beginn der Eingabe und sucht wieder nach dem ersten a und verfährt wie zuvor. Das passiert so lang, bis sie ablehnt, oder auf der Suche nach dem nächsten a das Ende der Eingabe erreicht. In letzterem Fall akzeptiert sie.

Problem 4: NP-Vollständigkeit 3 + 8 = 11 Punkte

GREEN-WASHING
Gegeben:

- Eine Menge U von Elementen
- eine Familie G von Teilmengen $G_1, \ldots, G_\ell \subseteq U$, welche wir als *grüne Mengen* bezeichnen
- eine Familie R von Teilmengen $R_1, \ldots, R_m \subseteq U$, welche wir als *Regenbogen-Mengen* bezeichnen
- und ein Parameter $k \in \mathbb{N}^+$, der die Anzahl der verfügbaren Farben angibt.

Problem: Können die Elemente von U mit k Farben so gefärbt werden, dass jede grüne Menge mindestens ein grünes Element enthält und jede Regenbogen-Menge keine zwei Elemente der gleichen Farbe enthält?

3SAT
Gegeben: Menge U an aussagenlogischen Variablen und eine Menge C an Klauseln mit je maximal drei Literalen.
Problem: Gibt es eine Belegung der Variablen in U, sodass in jeder Klausel mindestens ein Literal wahr ist?

(a) Geben Sie für GREEN-WASHING und 3SAT an, woraus eine Instanz bzw. Lösung des Problems im Allgemeinen besteht und geben sie jeweils ein Beispiel ohne leere Mengen an. Das Beispiel einer Lösung sollte die Lösung der zugehörigen Beispiel-Instanz sein, wählen Sie also eine Ja-Instanz. Füllen Sie dafür die Tabelle unten aus.

	3SAT	GREEN-WASHING
Instanz (allgemein)		
Instanz (Beispiel)		
Lösung (allgemein)		
Lösung (Beispiel)		

(b) Zeigen Sie, dass GREEN-WASHING NP-vollständig ist. Sie dürfen annehmen, dass 3SAT NP-schwer ist.

Hinweis: Nutzen Sie Regenbogen-Mengen der Größe 2, um zu erzwingen, dass ein Element grün und das andere nicht grün ist.

Lösung:

(a)

	3SAT	GREEN-WASHING
Instanz (allgemein)	Variablen U, Klauseln C	Elemente U, grüne Mengen G, Regenbogen-Mengen R, k
Instanz (Beispiel)	$U = \{x, y, z\}, C = \{(x \vee y \vee z)\}$	$U = \{a, b, c\}$, $G = \{\{a, b\}, \{b, c\}\}$, $R = \{\{a, c\}\}, k = 2$
Lösung (allgemein)	Variablenbelegung $f\colon U \to \{\text{true,false}\}$, sodass in jede Klausel $c \in C$ mindestens ein Literal wahr ist	Färbung $f\colon U \to \{\text{grün,blau,rot,}\dots\}$, sodass jede grüne Menge mindestens ein grünes Element enthält und keine Regenbogen-Menge zwei Elemente der gleichen Farbe enthält
Lösung (Beispiel)	$f(x) = f(y) = f(z) = \text{true}$	$f(a) = f(b) = \text{grün}, f(c) = \text{blau}$

(b) GREEN-WASHING liegt in NP, da ein Orakel jedem Element eine Farbe zuweisen kann und dann in Polynomialzeit überprüft werden kann, ob die Bedingungen an G und R erfüllt sind und ob die Anzahl verwendeter Farben höchstens k ist.

Um zu zeigen, dass GREEN-WASHING NP-schwer ist reduzieren wir von 3SAT. Sei $I = (U, C)$ eine 3SAT Instanz. Wir konstruieren daraus die GREEN-WASHING Instanz $I' = (U', G', R', k')$. Wir wählen U' als die Menge aller möglichen Literale über U (also negierte und nicht negierte Variablen der 3SAT Instanz I). Für jede Variable $v \in U$ erzeugen wir eine Regenbogen Menge $\{v, \bar{v}\}$ und für jede Klausel $C = (a \vee b \vee c) \in C$ erzeugen wir eine grüne Menge $\{a, b, c\}$ (Wenn eine Klausel mehrere Kopien des gelichen Literals beinhaltet, übernehmen wir nur eine Kopie davon in die korrespondierende grüne Menge). Wir wählen $k' = 2$. Diese Konstruktion ist polynomiell, da wir die Größe des Universums nur verdoppeln, G' und R' durch C beziehungsweise U in der Größe beschränkt sind und die Mengen in G', R' konstant groß sind (k' ist auch konstant).

Eine Lösung von I liefert uns immer direkt eine Lösung für I', indem wahre Literale grün gefärbt werden und falsche Literale blau. Da jede Regenbogen-Menge aus einem Literal und seiner Negation besteht, sind diese grün und blau gefärbt. Außerdem enthält jede Klausel ein wahres Literal, welches in der entsprechenden grünen Menge grün gefärbt wurde.

Umgekehrt liefert uns eine Lösung von I' auch eine Lösung für I. Wir setzen alle Literale, die grün sind auf wahr und alle nicht grünen Literale auf falsch. Dann sind alle Klauseln erfüllt (jede grüne Menge hat ein grünes Element $\Rightarrow$ jede Klausel hat ein wahres Literal) und jede Variable hat einen eindeutigen Wert (entweder v oder $\bar{v}$ ist wahr, aber nicht beide), da die entsprechende Regenbogen-Menge verhindert, dass beide entsprechenden Elemente die gleiche Farbe haben.

Also ist I' genau dann eine Ja-Instanz von GREEN-WASHING wenn I eine Ja-Instanz von 3SAT ist, was den Beweis der NP-Schwere von GREEN-WASHING beendet.

Problem 5: Approximation 3 + 3 + 6 = 12 Punkte

SQUARE-COVERING
Gegeben: Ein Quadrat Q mit Seitenlänge $\ell \in \mathbb{N}^+$, sowie eine Menge $M = \{Q_1, \ldots, Q_n\}$ weiterer Quadrate mit Seitenlängen $\ell_i \in \mathbb{N}^+$.
Frage: Wieviel Fläche von Q kann maximal durch Quadrate in M überdeckt werden? Hierbei dürfen sich die Q_i überlappen und auch aus Q herausragen.

Hierbei gelten folgende Regeln für alle SQUARE-COVERING Instanzen:

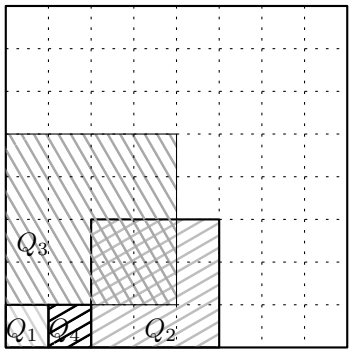
- Q sowie alle Quadrate in M sind achsenparallel.
- Q hat seine linke untere Ecke an Position $(0, 0)$.
- Die Quadrate aus M dürfen beliebig verschoben (aber nicht gedreht) werden.

Der folgende Algorithmus $\mathcal{A}$ platziert die Quadrate in M zeilenweise in aufsteigender Größe. Hierbei wird das erste Quadrat einer Zeile immer direkt oberhalb des ersten Quadrats der vorherigen Zeile platziert und alle Quadrate einer Zeile werden auf der gleichen Höhe platziert. Eine neue Zeile wird begonnen, wenn die vorherige Zeile mindestens bis zur Hälfte von Q gefüllt ist:

Algorithmus $\mathcal{A}$: SQUARE-COVERING APPROX

```
1 sortiere M aufsteigend; ab jetzt: ℓ_i ⩽ ℓ_{i+1} für alle i ∈ {1, ..., n − 1}
2 x ← ∞, y ← 0, h ← 0
3 Für i ← 1 bis n
4 |   Wenn x ⩾ ⌈ℓ/2⌉                          (⌈·⌉ bedeutet Aufrunden)
5 |   |   x ← 0, y ← y + h, h ← ℓ_i
6 |   platziere die linke untere Ecke von Q_i an Position (x, y)
7 |   x ← x + ℓ_i
8 return durch M überdeckte Fläche von Q
```

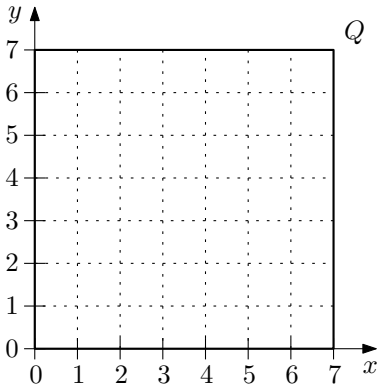
Beispiel: Auf der Eingabe $M = \{Q_1, Q_2, Q_3, Q_4\}$ mit $\ell_1 = 1$, $\ell_2 = 3$, $\ell_3 = 4$, $\ell_4 = 1$ und $\ell = 8$ platziert $\mathcal{A}$ die Quadrate aus M wie folgt und überdeckt dabei eine Fläche von 23 in Q:



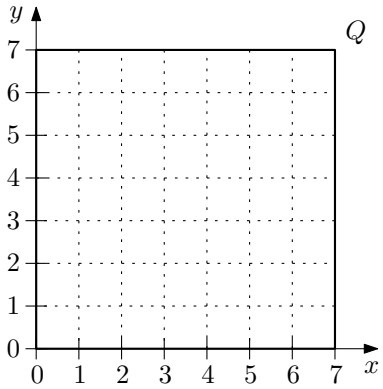
(a) Führen Sie $\mathcal{A}$ auf der Instanz I mit $M = \{Q_1, Q_2, Q_3, Q_4\}$ aus, wobei $\ell_1 = 1$, $\ell_2 = 3$, $\ell_3 = 3$, $\ell_4 = 4$ und $\ell = 7$ aus. Tragen Sie das Ergebnis in eine Kopie des auf der nächsten Seite gegebenen Quadrats Q ein.

Tragen Sie eine optimale Überdeckung von Q in eine weitere Kopie auf der nächsten Seite ein.

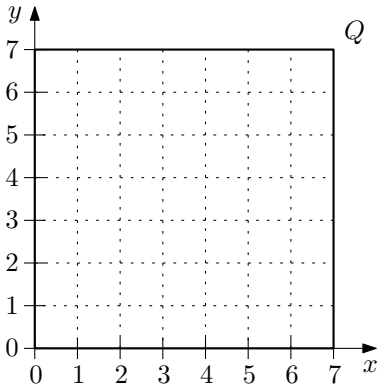
Geben Sie $\mathcal{A}(I)$ und $\mathrm{OPT}(I)$ an.



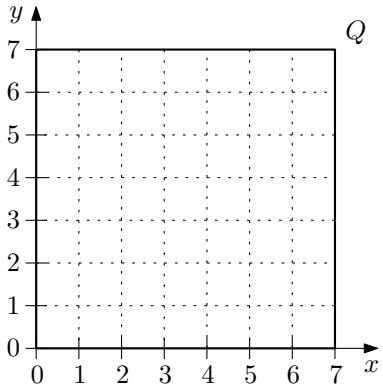
Tragen Sie hier die von $\mathcal{A}$ berechnete Lösung ein.



Kopie, falls Sie Ihre Antwort korrigieren wollen. Sollten Sie beide Kopien von Q benutzen, markieren Sie deutlich, welche bewertet werden soll!



Tragen Sie hier eine optimale Lösung ein.



Kopie, falls Sie Ihre Antwort korrigieren wollen. Sollten Sie beide Kopien von Q benutzen, markieren Sie deutlich, welche bewertet werden soll!

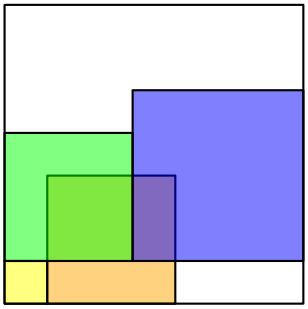
(b) Zeigen Sie, dass in einer von $\mathcal{A}$ berechneten Lösung jeder Punkt in Q von höchstens zwei Quadraten überdeckt wird.

(c) Zeigen Sie, dass $\mathcal{A}$ ein polynomieller Approximationsalgorithmus mit relativer Gütegarantie 4 für Square-Covering ist. (Sie dürfen ohne Beweis annehmen, dass die Return-Anweisung in Zeile 8 nur polynomielle Laufzeit benötigt.)

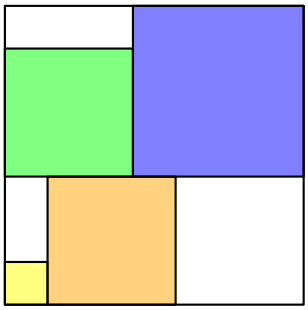
Hinweis: Unterscheiden Sie, ob $\mathcal{A}$ Quadrate so platziert, dass sie aus Q herausragen oder nicht und zeigen Sie für jeden Fall die Approximationsgüte.

Lösung:

(a) Q_1 gelb, Q_2 orange, Q_3 grün, Q_4 blau:



Lösung von $\mathcal{A}$.
$\mathcal{A}(I) = 29$



Eine optimale Lösung.
$\mathrm{OPT}(I) = 35$

(b) Der Approximationsalgorithmus $\mathcal{A}$ ordnet die Quadrate „zeilenweise“ an, Quadrate der gleichen Zeile überlappen sich nicht. Dabei entspricht die Höhe einer Zeile (also der Abstand zur Zeile darüber) gerade der Höhe des kleinsten Quadrats in der Zeile. Die Quadrate aus M sind nach aufsteigender Größe sortiert, sodass jedes Quadrat (wenn überhaupt) nur in die darüber liegende Zeile hineinragt. Insbesondere wird jeder Punkt $p \in Q$ von höchstens zwei Quadraten aus M überdeckt.

(c) $\mathcal{A}$ plaziert alle Quadrate legal und gibt danach die Größe der überdeckten Fläche zurück, also liefert der Algorithmus eine korrekte Lösung für Square-Covering.

Laufzeit: Das Sortieren von M benötigt Zeit in $O(n \log n)$. Die Schleife macht n Iterationen. Jede weitere Zeile benötigt konstante Laufzeit (außer Zeile 8, die laut Aufgabenstellung Polynomialzeit benötigt). Insgesamt hat $\mathcal{A}$ also polynomielle Laufzeit.

Approximationsgüte:

Wie in Teilaufgabe (b) gezeigt, wird jeder Punkt in Q von maximal zwei Quadraten überdeckt. Sollten alle $Q_i \in M$ vollständig innerhalb von Q liegen, hätten wir demnach eine relative Gütegarantie von 2 (was besser ist als 4).

Ein Quadrat kann entweder oben oder rechts aus Q herausragen. Wenn ein Quadrat Q_i oben aus Q herausragt ist die linke Hälfte von Q unterhalb von Q_i komplett überdeckt. Wenn Q_i also oberhalb von $\frac{\ell}{2}$ platziert wurde, ist das untere linke Viertel schon komplett überdeckt, also kann keine Lösung mehr als vier mal so viel Fläche von Q überdecken. Sonst muss ℓ_i größer als $\frac{\ell}{2}$ sein um aus Q herausragen zu können. In diesem Fall ist aber die von Q_i in Q überdeckte Fläche schon mindestens ein viertel von Q, da die linke untere Ecke von Q_i im linken unteren Viertel von Q platziert wurde, seine rechte obere Ecke aber oberhalb von Q liegt.

Wenn Q_i rechts aus Q herausragt muss ℓ_i größer als $\frac{\ell}{2}$ sein. Wenn Q_i nicht schon oben aus Q herausragt, ist also eine mindestens $\frac{\ell}{2}$ hohe Zeile in der rechten Hälfte von Q durch Q_i überdeckt, was uns wieder die gewünschte Approximationsgüte liefert.

Problem 6: Entscheidbarkeit 1+1+3+4 = 9 Punkte

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

Sei $L \subseteq \Sigma^*$ und $w \in \Sigma^*$ ein Wort. Wir sagen L ist *w-präfixfrei*, wenn w von keinem Wort aus L ein Präfix ist. Das heißt, hängen wir ein beliebiges $z \in \Sigma^*$ an w an, so gilt $wz \notin L$.

(a) Geben Sie eine Sprache L' an, die 0-präfixfrei ist.

Betrachten Sie die Sprache

$$S := \{\langle M\rangle \# w \mid L(M) \text{ ist } w\text{-präfixfrei}\}.$$

(b) Sei $M_\emptyset$ die Turingmaschine, die jedes Wort ablehnt. Zeigen Sie, dass $\langle M_\emptyset\rangle \# w \in S$ für alle Wörter $w \in \Sigma^*$.

Aus der Vorlesung wissen Sie, dass die universelle Sprache

$$L_u := \{\langle M\rangle \# w \mid w \in L(M)\}$$

unentscheidbar ist.

(c) Betrachten Sie ein Tupel $\langle M\rangle \# w$ bestehend aus der Gödelnummer einer Turingmaschine M und einem Wort $w \in \Sigma^*$. Konstruieren Sie eine deterministische Turingmaschine $T_{M,w}$, sodass

$$w \in L(M) \iff L(T_{M,w}) \text{ ist nicht } w\text{-präfixfrei.}$$

Begründen Sie, warum die von Ihnen angegebene Turingmaschine $T_{M,w}$ die obige Eigenschaft hat. Ihre Konstruktion muss aus w und $\langle M\rangle$ berechenbar sein.

(d) Zeigen Sie, dass S unentscheidbar ist, indem Sie von der universellen Sprache reduzieren. Sie dürfen in dieser Aufgabe nicht den Satz von Rice verwenden.

Hinweis: Verwenden Sie Aufgabe (c).

Lösung:

(a) Jede Sprache, die kein Wort enthält welches mit 0 beginnt, ist 0-präfixfrei. Insbesondere gilt das für $L' = 1^*$.

(b) Es gilt $L(M_\varnothing) = \varnothing$. Also gilt insbesondere für jedes $w \in \Sigma^*$, dass für alle Suffixe $z \in \Sigma^*$

$$wz \notin \varnothing = L(M_\varnothing).$$

Somit ist $L(M_\varnothing)$ w-präfixfrei für jedes $w \in \Sigma^*$. Es folgt $\langle M_\varnothing\rangle \# w \in S$ für alle $w \in \Sigma^*$.

(c) Wir konstruieren eine TM $T_{M,w}$, für die gilt

$$L(T_{M,w}) = \begin{cases} \Sigma^*, & w \in L(M) \\ \varnothing, & w \notin L(M). \end{cases}$$

Die TM $T_{M,w}$ arbeitet wie folgt. Sie ignoriert die Eingabe und führt M auf der Eingabe w aus. Somit hat sie die obige Eigenschaft.

Wir müssen nun noch zeigen, dass

$$w \in L(M) \iff \exists z \in \Sigma^* \colon wz \in L(T_{M,w}).$$

Ist $w \in L(M)$, so akzeptiert die TM $T_{M,w}$ jedes Wort in Σ^*. Insbesondere akzeptiert die TM $T_{M,w}$ das Wort wz mit $z = \varepsilon$ (an dieser Stelle könnte man jeden Suffix $z \in \Sigma^*$ wählen).

Existiert ein $z \in \Sigma^*$, sodass $wz \in L(T_{M,w})$, so gilt insbesondere, dass $L(T_{M,w}) \neq \varnothing$. Also gilt $L(T_{M,w}) = \Sigma^*$ und $w \in L(M)$ nach Konstruktion von $T_{M,w}$.

(d) Wir reduzieren von der universellen Sprache. Angenommen S ist entscheidbar, dann existiert eine TM T, die S entscheidet. Wir konstruieren eine TM T', die die universelle Sprache entscheidet.

Die TM T' erhält ein Tupel $\langle M \rangle \# w$ und soll das Tupel genau dann akzeptieren, wenn $w \in L(M)$ liegt. Die TM T' konstruiert wie in Aufgabe c die TM $T_{M,w}$.

Ist $w \in L(M)$, so gilt nach Teilaufgabe c, dass $L(T_{M,w})$ *nicht* w-präfixfrei. Das heißt, das Tupel $\langle T_{M,w} \rangle \# w$ liegt nicht in der Sprache S und die TM T akzeptiert die Eingabe $\langle T_{M,w} \rangle \# w$ nicht.

Wird das Tupel $\langle T_{M,w} \rangle \# w$ von der TM T nicht akzeptiert, so ist $L(T_{M,w})$ w-präfixfrei. Mit Teilaufgabe c folgt, dass $w \notin L(M)$.

Wir haben also gezeigt, dass

$$w \in L(M) \iff \langle T_{M,w} \rangle \# w \notin S. \tag{3}$$

Die TM T' führt die TM T auf der Eingabe $\langle T_{M,w} \rangle \# w$ aus und invertiert ihr Akzeptanzverhalten. Da T immer hält, hält auch T' immer. Mit (3) folgt, dass T' die universelle Sprache entscheidet, denn

$$w \in L(M) \iff \langle T_{M,w} \rangle \# w \notin S \iff \langle M \rangle \# w \in L(T')$$

Das ist ein Widerspruch, da die universelle Sprache unentscheidbar ist. Somit ist S unentscheidbar.