

Einführung in die Theoretische Informatik
Klausur — SoSe 2025 — 29.9.2025

Nebentermin

Gruppe: Weed

Unbedingt ausfüllen

Matrikelnummer	Studiengang/Abschluss	Fachsemester
<input style="width: 95%;" type="text"/>	<input style="width: 95%;" type="text"/>	<input style="width: 95%;" type="text"/>
Nachname	Vorname	
<input style="width: 95%;" type="text"/>	<input style="width: 95%;" type="text"/>	
Unterschrift	Identifikator <small>(Beliebiges Wort zur Identifikation im anonymen Notenaushang)</small>	
<input style="width: 95%;" type="text"/>	<input style="width: 95%;" type="text"/>	

Grundregeln

- Die Bearbeitungszeit der Klausur beträgt **120 Minuten**.
- Sie schreiben diese Klausur **vorbehaltlich** der Erfüllung der **Zulassungsvoraussetzung**. Das heißt: Wir werden Ihre Zulassung vor Korrektur prüfen; die Tatsache, dass Sie die Klausur mitschreiben, bedeutet keine implizite Zulassung.
- Es sind **keine Unterlagen** und auch **keine** anderen **Hilfsmittel** erlaubt.
- Benutzen Sie nur dokumentenechten (blauen oder zur Not schwarzen) **Kugelschreiber!** Bleistiftlösungen werden nicht gewertet!
- Es zählt die Antwort, die sich im dafür vorgesehenen Kästchen befindet! Soll eine andere Antwort gewertet werden, so ist diese **eindeutig** zu kennzeichnen!
- Jegliches Schummeln, und auch der Versuch desselben, führt zum Ausschluss von der Klausur und einer Bewertung mit **5,0**.

Wird vom Korrektor/Prüfer ausgefüllt

Aufgabe	1	2	3	4	5	6	7	Σ
Punkte (max)	8	8	12	8	18	8	12	74
Punkte (erreicht)								

Punkte	0.. 36	37..38	39..40	41..42	43..45	46..48	49..51	52..55	56..58	59..62	63..74
Note	5,0	4,0	3,7	3,3	3,0	2,7	2,3	2,0	1,7	1,3	1,0

Note:

Aufgabe 1: Information**(8 Punkte)****(a) Huffman-Code erstellen****(4 Punkte)**

Gegeben sei die Quelle (Σ, p) über dem Alphabet $\Sigma = \{e, t, h, i, k\}$ mit

$\sigma \in \Sigma$	e	t	h	i	k
p_σ	$2/3$	$?$	$1/8$	$1/8$	$1/16$

Konstruieren Sie einen Huffman-Code hierfür.

(b) Informationsgewinn**(4 Punkte)**

Sei \mathbb{C}_p ein *minimaler binärer* Präfix-Code für eine Quelle (Σ, p) mit $\Sigma = \{a, b, c, d\}$. Beachten Sie, dass \mathbb{C}_p für *jedes* Element aus Σ ein zulässiges Codewort definiert.

Geben Sie Quellwahrscheinlichkeiten p an, sodass der erwartete Informationsgewinn eines einzelnen *Codesymbols* $\in \{0, 1\}$ am Anfang eines Codeworts aus \mathbb{C}_p genau **0** ist.

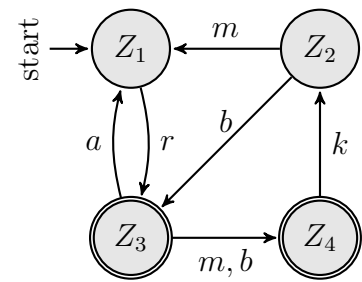
Geben Sie Quellwahrscheinlichkeiten p an, sodass der erwartete Informationsgewinn eines einzelnen *Codesymbols* $\in \{0, 1\}$ am Anfang eines Codeworts aus \mathbb{C}_p genau **1** ist.

Aufgabe 2: Umwandlung DEA \rightarrow RegEx

(8 Punkte)

Wandeln Sie den gegebenen DEA – gemäß dem Vorgehen aus der Vorlesung – in einen regulären Ausdruck um.

Hinweis: Wenn Ausdrücke zu lang werden, dürfen Sie sich gerne (eindeutige) Hilfsausdrücke definieren.



Aufgabe 3: Pumping Lemma

(12 Punkte)

(a) **Definition**

(4 Punkte)

Wie lautet das Pumping Lemma für reguläre Sprachen?

Sei L eine reguläre Sprache. Dann...

... $z = uvw$ mit den Eigenschaften

(1) , (2) und (3) .

(b) **Anwendung**

(8 Punkte)

Beweisen Sie, dass $L := \{t^i c^j s^\ell \mid i = j \bmod 4, \ell = j \cdot i\}$ nicht regulär ist.

Aufgabe 4: Turingvollständigkeit

(8 Punkte)

Nehmen wir an, die Church-Turing-These würde nicht gelten und es gäbe ein Rechenmodell, das mächtiger als eine Turingmaschine wäre.

Betrachten Sie die hypothetische Programmiersprache HYPER. Sie enthält alle Programmierkonstrukte der Programmiersprache WHILE. Zusätzlich unterstützt sie noch den folgenden Anweisungstyp:

$ifHalts(A) \{B\}$

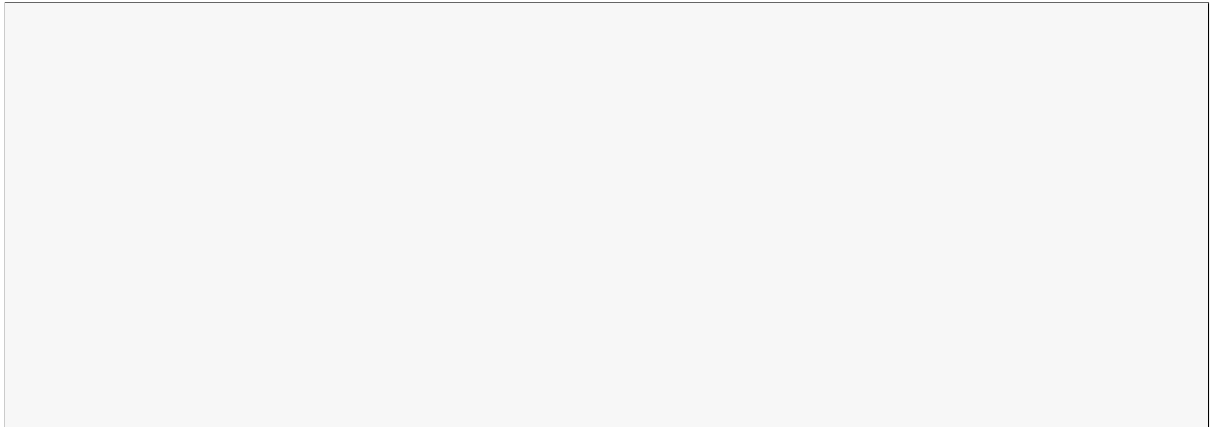
Parameter: A : Anweisungen in WHILE
 B : Anweisungen in HYPER

Verhalten: Falls das Programm A halten würde, wird B ausgeführt, sonst nicht.
(Achtung: A wird nie tatsächlich ausgeführt)

(a) Mächtigkeit

(4 Punkte)

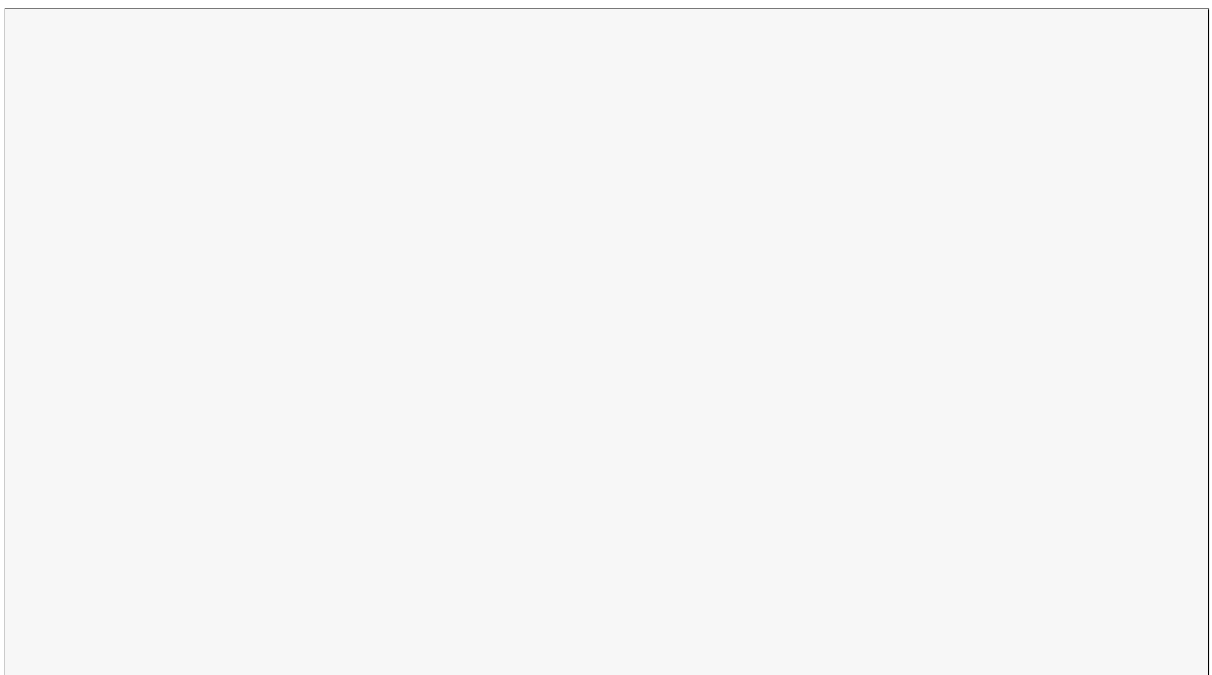
Zeigen Sie, dass HYPER echt mächtiger ist als WHILE.



(b) Halteproblem

(4 Punkte)

Ein Problem ist *HYPER-entscheidbar*, wenn es mittels eines HYPER-Programms entscheidbar ist. Skizzieren Sie, warum das Halteproblem für HYPER-Programme nicht HYPER-entscheidbar ist.



(a) Definitionen und Eigenschaften

(6 Punkte)

Eine Reduktion von \mathcal{X} auf \mathcal{Y} ist eine berechenbare Funktion f , die eine beliebige \mathcal{X} -Instanz I in eine \mathcal{Y} -Instanz $f(I)$ verwandelt, sodass $f(I)$ eine Ja-Instanz ist, genau dann wenn I eine Ja-Instanz ist.

- | | | |
|---|--------------------------|--------------------------|
| | ja | nein |
| Für alle Instanzpaare $I \neq I'$ im Definitionsbereich von f ist $f(I) \neq f(I')$. | <input type="checkbox"/> | <input type="checkbox"/> |
| Im Wertebereich von f kann es Instanzen geben, die nicht erzeugt werden. | <input type="checkbox"/> | <input type="checkbox"/> |
| Jedes Problem in NP kann auf SAT reduziert werden. | <input type="checkbox"/> | <input type="checkbox"/> |
| Falls $P \neq NP$, dann ist $NP \setminus P$ eine Teilmenge von NPC . | <input type="checkbox"/> | <input type="checkbox"/> |

Ein Problem \mathcal{A} ist **NP**-schwer, wenn gilt:

(b) Reduktion

(12 Punkte)

Wir haben Leute nach ihrer Lieblingszahl gefragt. Sei $J \subset \mathbb{N}$ die Menge der genannten Zahlen und $z_i \geq 1$ die Anzahl der Personen, die $i \in J$ als Lieblingszahl angegeben haben. Können wir J in zwei gleich große Teilmengen aufteilen, sodass in beiden Teilmengen gleich viele Leute ihre Lieblingszahl wiederfinden?

HALBHALB

Gegeben: Endliche Menge $J \subset \mathbb{N}$ und natürliche Zahlen $z_i \geq 1$ für alle $i \in J$.

Gefragt: Existiert eine Teilmenge $X \subset J$, sodass

$$|X| = |J|/2 \quad \text{und} \quad \sum_{i \in X} z_i = \sum_{i \in J \setminus X} z_i$$

Sie sollen im Folgenden zeigen, dass HALBHALB **NP**-schwer ist. Definieren Sie ein Problem aus der Vorlesung für die Reduktion.

Gegeben:

Gefragt:

Zeigen Sie nun mithilfe dieses Problems, dass HALBHALB **NP**-schwer ist.

Hinweis: Denken Sie vielleicht zunächst über eine Reduktion nach, in der einzelne $z_i = 0$ sind. Auch wenn Ihre Reduktion das geforderte $z_i \geq 1$ nicht erreicht, gibt es Teilpunkte.

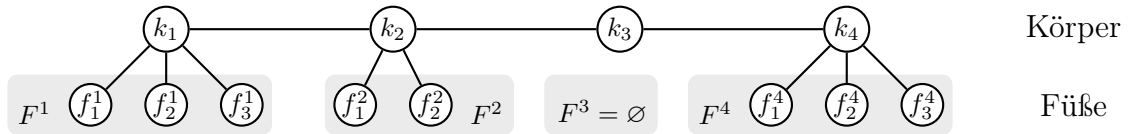
Reduktionsbeweis:

Aufgabe 6: Dynamische Programmierung

(8 Punkte)

Eine *Raupe* $R = (K, F^1, \dots, F^n)$ ist ein Graph, der aus *Körperknoten* $K = \{k_1, k_2, \dots, k_n\}$ und *Fußknoten* $\bigcup_{i=1}^n F^i$ besteht. Die Körperknoten bilden einen Pfad (in der Reihenfolge der Indizes); die Knoten aus F^i sind immer genau zu k_i adjazent. Für $1 \leq \ell \leq n$ ist $R_\ell \subseteq R$ der Teilgraph der Raupe mit den Knoten $\bigcup_{i=1}^{\ell} (\{k_i\} \cup F^i)$.

Beispiel:



RAUPENÜBERDECKUNG

Gegeben: Raupe R , Gewichte $w_v \in \mathbb{N}$ für jeden Knoten v in R , Maximalgewicht $W \in \mathbb{N}$.

Gefragt: Existiert ein Vertex Cover C für R mit Gewicht $\sum_{c \in C} w_c \leq W$?

Geben Sie einen deterministischen Linearzeit-Algorithmus an, der mittels dynamischer Programmierung das Problem RAUPENÜBERDECKUNG exakt löst. Berechnen Sie dafür ein Array M mit Einträgen $M[\ell, b] \in \mathbb{N}$ für $\ell \in \{1, 2, \dots, n\}$ und $b \in \{\mathbf{true}, \mathbf{false}\}$. Dabei sei $M[\ell, \mathbf{true}]$ das kleinste Gewicht eines Vertex Covers von R_ℓ in dem k_ℓ enthalten ist und $M[\ell, \mathbf{false}]$ das kleinste Gewicht eines Vertex Covers von R_ℓ in dem k_ℓ **nicht** enthalten ist.

```
// Initialisierung
```

```
// Dynamische Programmierungs-Schleife:
```

```
// Ergebnis:
```

Aufgabe 7: Randomisierte Algorithmen

(12 Punkte)

(a) Definitionen

(6 Punkte)

Ordnen Sie Komplexitätsklassen P , RP , BPP , $ZPP(MC)$ und $ZPP(LV)$ den gegebenen Eigenschaften gemäß ihrer kanonischen Definition zu:

		polynomielle Laufzeit	
		immer	erwartet
Korrektheit	immer		
	wahrscheinlich		

Was ist die kleinste Wahrscheinlichkeit mit der ein Algorithmus korrekt antworten muss, um zu zeigen, dass das damit gelöste Problem in BPP liegt?

(b) Algorithmen

(6 Punkte)

Sei \mathcal{X} ein Problem in NP . Sei \mathcal{A} ein Algorithmus mit Laufzeit $\mathcal{O}(n^3)$, der als Eingabe eine \mathcal{X} -Instanz der Größe n erhält. Bei Ja-Instanzen liefert \mathcal{A} mit Wahrscheinlichkeit $1/2$ einen Ja-Zeugen und terminiert sonst ohne Zeugen. Bei Nein-Instanzen liefert \mathcal{A} mit Wahrscheinlichkeit $1/4$ einen Nein-Zeugen und terminiert sonst ebenfalls ohne Zeugen.

Nutzen Sie \mathcal{A} , um einen Algorithmus anzugeben, der immer korrekt antwortet. Was ist seine erwartete Laufzeit in \mathcal{O} -Notation?

Notizen: