

Einführung in die Theoretische Informatik

Klausur — SoSe 2024 — 21. Oktober 2024

Nebentermin

Gruppe: Marko / Mix

Unbedingt ausfüllen

Matrikelnummer	Studiengang/Abschluss	Fachsemester
<input type="text"/>	<input type="text"/>	<input type="text"/>
Nachname	Vorname	
<input type="text"/>	<input type="text"/>	
Unterschrift	Identifikator <small>(Beliebiges Wort zur Identifikation im anonymen Notenaushang)</small>	
<input type="text"/>	<input 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	10	8	18	10	10	<b>72</b>
Punkte (erreicht)								

<b>Punkte</b>	0.. 35	36..37	38..39	40..41	42..44	45..47	48..50	51..53	54..57	58..60	61..72
<b>Note</b>	<b>5,0</b>	<b>4,0</b>	<b>3,7</b>	<b>3,3</b>	<b>3,0</b>	<b>2,7</b>	<b>2,3</b>	<b>2,0</b>	<b>1,7</b>	<b>1,3</b>	<b>1,0</b>

Note:

**Aufgabe 1: Information****(8 Punkte)****(a) Entropie****(4 Punkte)**

Sie erhalten eine codierte Nachricht  $M$ , bestehend aus  $k$  Zeichen, über dem Alphabet  $\Sigma = \{a, b, c\}$ . Die Nachricht kommt mit Wahrscheinlichkeit  $\frac{2}{3}$  aus einer Quelle  $(\Sigma, p)$  mit  $p_a = 0.5, p_b = 0.5$ , ansonsten aus einer Quelle  $(\Sigma, q)$  mit  $q_a = 0.25, q_b = 0.25$ . Berechnen Sie den erwarteten Informationsgewinn von  $M$ . Kürzen Sie dabei so weit wie möglich.

**(b) Präfix-Codes****(4 Punkte)**

Betrachten Sie die in der Tabelle angegebene Quelle  $(\Sigma, p)$ . Warum sind die gegebenen Codes  $\mathbb{C}$  und  $\mathbb{D}$  jeweils keine korrekten Präfix-Codes mit minimaler erwarteter Codewortlänge?

$\sigma \in \Sigma$	a	b	c	d	e
$p_\sigma$	$\frac{2}{10}$	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{4}{10}$	$\frac{1}{10}$
$\mathbb{C}$	10	110	011	00	111
$\mathbb{D}$	011	0111	01	0	1111

**Aufgabe 2: Umwandlung RegEx → NDEA****(8 Punkte)**

Wandeln Sie den folgenden regulären Ausdruck – gemäß dem Vorgehen aus der Vorlesung – in einen nichtdeterministischen endlichen Automaten um.

$$(c^+|d)a|(dc)^*$$

### Aufgabe 3: Rechnende Turingmaschine

(10 Punkte)

Mit  $\beta(w)$  bezeichnen wir die Anzahl der Symbole *vor* dem ersten  $b$  in einem Wort  $w$ , falls  $w$  ein  $b$  enthält. Zum Beispiel ist  $\beta(aaabaabba) = 3$ .

Erstellen Sie eine deterministische rechnende Turingmaschine mit dem Bandalphabet  $\Gamma := \{\square, a, b, 0, 1, \$\}$  und **höchstens 12 Zuständen**, die bei Eingabe  $w \in \{a, b\}^+$  die folgende Funktion berechnet und das Ergebnis **binär codiert** ausgibt:

$$f(w) := \begin{cases} \beta(w), & \text{falls } w \text{ ein } b \text{ enthält,} \\ \text{undef} & \text{sonst.} \end{cases}$$

*Hinweise:* Hinter der ausgegebenen Zahl muss ein  $\square$  stehen; was danach steht ist irrelevant. Es existieren schöne Lösungen sowohl mit als auch ohne Nutzung des  $\$$ -Symbols.

**Aufgabe 4: Turing-Vollständigkeit****(8 Punkte)**

Betrachten Sie die folgende Programmiersprache BRAINLUCK. Analog zu C++, Java, Python, etc., besteht ein BRAINLUCK-Programm aus mehreren hintereinander geschriebenen (und ausgeführten) Befehlen. Die möglichen Befehle sind:

Befehl	Bedeutung
$x_i \odot$ ;	Setzt die Variable $x_i$ auf einen zufälligen Wert aus $\{0, 1, \dots, 40\}$ .
$x_i \leftarrow x_j$ ;	Setzt die Variable $x_i$ auf den Wert der Variable $x_j$ .
$x_i \uparrow c$ ;	Erhöht die Variable $x_i$ um $c$ .
$x_i \downarrow x_j$ ;	Senkt die Variable $x_i$ um den Wert der Variable $x_j$ .
<b>repeat</b> $A$ <b>until</b> $B$ ;	Eine Schleife, die $A$ ausführt und <i>danach</i> Bedingung $B$ überprüft. Die Schleife wird beendet, wenn bei der Überprüfung $B$ gilt.
<b>unless</b> $B$ <b>do</b> $A$ <b>ok</b> ;	Nur wenn die Bedingung $B$ <i>nicht</i> gilt, wird $A$ ausgeführt.
<b>forty</b> $A$ <b>phew</b> ;	Führt $A$ vierzig mal aus.

Dabei sind  $x_1, x_2, \dots$  Variablen und  $c$  eine beliebige Konstante. Wir betrachten nur natürliche Zahlen  $\mathbb{N}$ .  $A$  bezeichnet jeweils eine Sequenz von Befehlen. Bedingungen  $B$  sind beliebige logische Ausdrücke mit Variablen, Konstanten und Vergleichsoperatoren.

Zeigen Sie, dass BRAINLUCK Turing-vollständig ist.

**Aufgabe 5: NP-Vollständigkeit**

**(18 Punkte)**

**(a) Lückentext**

**(6 Punkte)**

**Aufgabe:** Vervollständigen Sie.

Eine Reduktion von  auf  ist eine  Funktion  $f$ ,  
die eine  $\mathcal{X}$ -Instanz  $f(I)$  aus einer beliebigen  $\mathcal{Y}$ -Instanz  $I$  erzeugt, sodass gilt:  
 $f(I)$  ist eine Nein-Instanz   $I$  ist eine -Instanz.

Seien  $\mathcal{A} \in \mathbf{P}$  und  $\mathcal{B}$  NP-schwer. Da  in NP liegt, existiert dann eine  
 Reduktion von  auf . Falls  
eine solche Reduktion auch in die umgekehrte Richtung existiert, ist  $\mathbf{P}$   NP.

Unabhängig vom letzten Satz: Wenn  $\mathcal{B} \notin \mathbf{NPC}$ , dann ist  $\mathcal{B}$  nicht .

**(b) Reduktion**

**(12 Punkte)**

Sie planen eine Bildungsreise, auf der Sie die Orte  $v_1, v_2, \dots, v_n$  in dieser Reihenfolge besuchen wollen; insbesondere starten Sie also an  $v_1$  und enden an  $v_n$ . Um die Etappe von einem Ort  $v_{i-1}$  nach  $v_i$  ( $1 \leq i \leq n$ ) zu reisen, müssen Sie sich immer für eine von  $r$  vielen Möglichkeiten  $M_i = \{(d_{i,1}, w_{i,1}), (d_{i,2}, w_{i,2}), \dots, (d_{i,r}, w_{i,r})\}$  entscheiden. Eine Etappenmöglichkeit  $(d_{i,j}, w_{i,j}) \in M_i$  beschreibt dabei stets ihre Dauer  $d_{i,j} \geq 1$  in Tagen und eine Zahl  $w_{i,j} \geq 1$ , die das dabei gewonnene Wissen misst.

Bis zum Ende Ihrer Ferien stehen Ihnen nur maximal  $D$  Reisetage zur Verfügung, in denen Sie aber möglichst viel Wissen  $W$  gewinnen möchten:

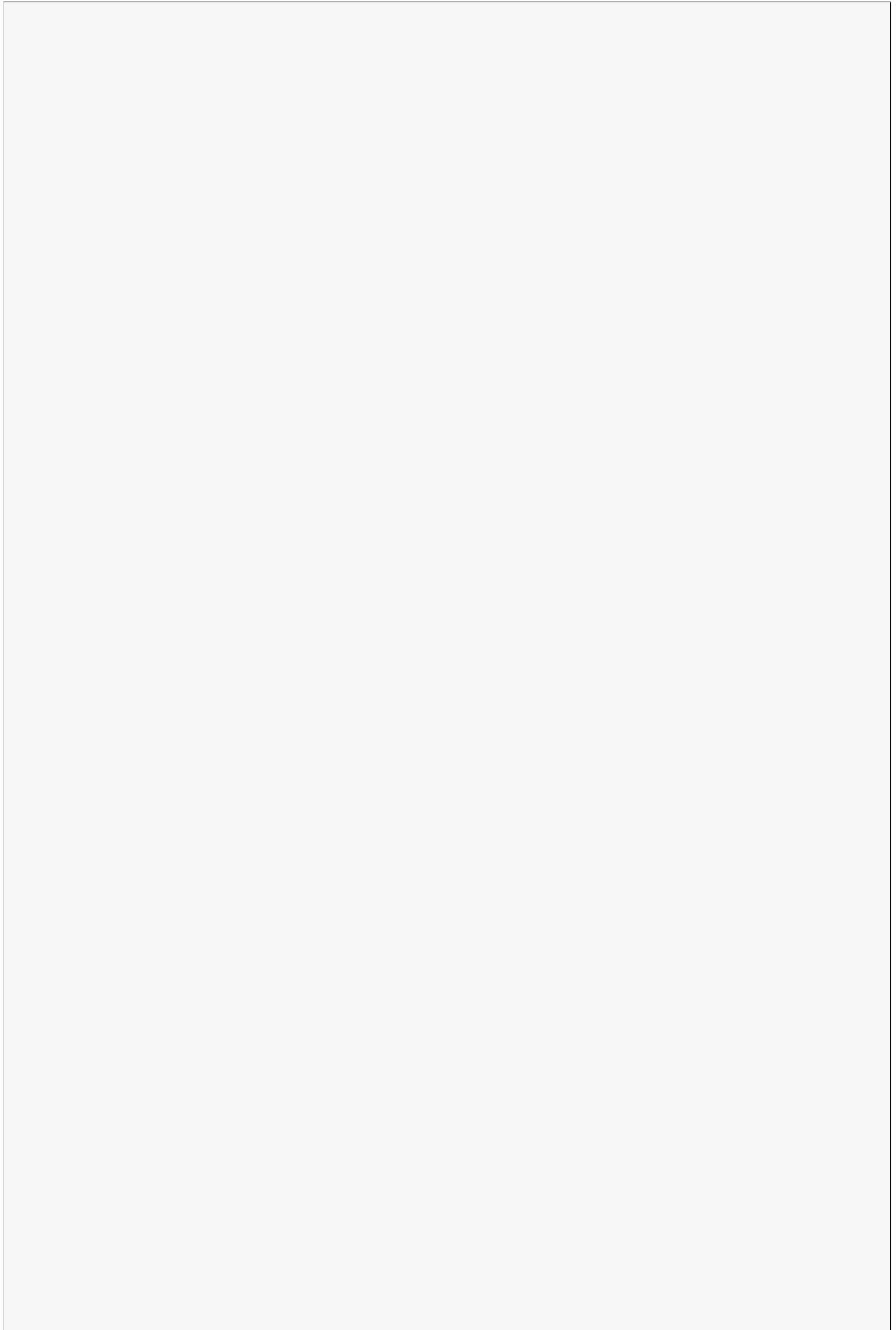
**BILDUNGSREISE**  
**Gegeben:**  $D, W, n, r \in \mathbb{N}^+$ ;  $\left\{ M_i = \{(d_{i,j}, w_{i,j}) \in (\mathbb{N}^+ \times \mathbb{N}^+)\}_{1 \leq j \leq r} \right\}_{1 \leq i \leq n}$   
**Gefragt:** Existiert  $\{(d_i^*, w_i^*) \in M_i\}_{1 \leq i \leq n}$  mit  $\sum_{i=1}^n d_i^* \leq D$  und  $\sum_{i=1}^n w_i^* \geq W$ ?

Definieren Sie zunächst das Problem SUBSETSUM; achten Sie darauf ggf. andere Bezeichner als in BILDUNGSREISE zu benutzen:

**Name:** SUBSETSUM  
**Gegeben:**  
  
**Gefragt:**

Zeigen Sie nun mithilfe von SUBSETSUM, dass BILDUNGSREISE NP-schwer ist.

*Hinweis:* Beachten Sie, dass  $d_{i,j}, w_{i,j} \geq 1$ . Falls Ihnen dazu keine gute Reduktion einfällt: Sie können, für etwas weniger Punkte, auch eine Reduktion angeben, bei denen diese Werte teilweise 0 sind.



## Aufgabe 6: Pseudopolynomiell

(10 Punkte)

### (a) Definition

(4 Punkte)

Wie lautet die Definition von **stark** NP-vollständig? Sie dürfen in Ihrer Definition davon ausgehen, dass NP-Vollständigkeit schon definiert ist.

### (b) Bildungsreise

(6 Punkte)

Betrachten Sie abermals das Problem BILDUNGSREISE aus Aufgabe 5(b), allerdings als Optimierungsproblem: Wir möchten während der maximal  $D$ -tägigen Bildungsreise möglichst viel Wissen gewinnen. Im Gegensatz zum Entscheidungsproblem haben wir also kein  $W$  gegeben, sondern möchten das größtmögliche zulässige  $W$  berechnen.

Geben Sie dafür einen polynomiellen Algorithmus unter der Bedingung an, dass der Wert von  $D$  polynomiell in der Eingabelänge beschränkt ist. Nutzen Sie dabei die Idee der dynamischen Programmierung und berechnen Sie ein geeignetes zwei-dimensionales Array  $R[i, t]$ , wobei  $v_i$  einen Ort und  $t$  eine Anzahl an Tagen repräsentiert.

*Fortsetzung Aufgabe 6:*

## Aufgabe 7: Randomisierte Algorithmen

(10 Punkte)

(a) **Definition**

(4 Punkte)

Definieren Sie  $ZPP(MC)$ .

(b) **Algorithmus**

(6 Punkte)

Gegeben sei ein **BPP**-Algorithmus  $\mathcal{A}$ , der Ja-Instanzen mit Wahrscheinlichkeit 99% korrekt erkennt, und Nein-Instanzen mit Wahrscheinlichkeit 51% korrekt erkennt.

Erstellen Sie darauf basierend einen **BPP**-Algorithmus  $\mathcal{B}$  mit Fehlerwahrscheinlichkeit kleiner 33%, der dazu die minimal notwendige Anzahl an  $\mathcal{A}$ -Aufrufen benötigt. Begründen Sie die Anzahl und die Wahrscheinlichkeiten.

*Algorithmus und Analyse:*

*Fortsetzung Aufgabe 7:*

*Notizen:*

*Viel Erfolg!*