



Wiederholungsklausur „Komplexitätstheorie“ WS03

29. Juli 2004

NICHT MIT BLEISTIFT ODER ROTSTIFT SCHREIBEN!

Heften Sie die Blätter bei Abgabe zusammen, und tragen Sie auf jedem Blatt Ihren Namen, Vornamen, Studiennummer und Matrikel ein.

Name, Vorname:

Studiennummer und Matrikel:

Code

**abgegeben: 2 Aufgabenblätter
... eigene Blätter**

Einsichtnahme
Datum, Unterschrift

Aufgabe	1	2	3	4	5	6	7	Ges.
erreichbare Punktzahl	30	12	16	8	12	12	10	100
erreichte Punktzahl								

Es sind keine Hilfsmittel, insbesondere Taschenrechner oder Mobiltelefone, zugelassen.

Aufgabe 1 (Fragenkatalog)

[30 Punkte]

Kreuzen Sie für jede der folgenden 30 Fragen entweder „JA“ oder „NEIN“ an.

Bewertung: Ist C die Anzahl der richtigen Antworten, so errechnet sich die Anzahl P der erzielten Punkte aus $P := \frac{3}{2} \cdot \max\{0, C - 10\}$. Insbesondere: Kein Kreuz wird als falsche Antwort gewertet.

(a) Welche der folgenden Aussagen ist/sind für **alle** Probleme P, P' korrekt?

JA NEIN

- Wenn $P \leq_N P'$ und P effizient lösbar ist, dann ist auch P' effizient lösbar.
- Wenn $P \leq_N P'$ mit O -Konstante $c = 1$, dann gilt $T_{P'}(n) \geq T_P(n) - O(n)$.
- Wenn $P \leq_N P'$ mit O -Konstante $c \geq 1$ und $T_P(n) \geq d \cdot n \cdot \log n - O(n)$ für genügend große n , dann gilt $T_{P'}(n) \geq \frac{d}{c} \cdot n \cdot \log n - O(n)$ für genügend große n .

(b) Welche der nachstehenden Aussagen folgt/folgen aus dem Satz von Ben-Or?

JA NEIN

- Jedes reelle RAM-Programm für ELEMENT-UNIQUENESS benötigt für genügend große n (Länge der Eingabeliste) mindestens $0.38 \cdot n \cdot \log n + O(n^2)$ Schritte.
- Für jedes reelle RAM-Programm M zur Berechnung einer legalen Triangulierung einer Liste von Punkten in der Ebene gilt $T_M(n) = \Omega(n^2)$.
- Jedes RRAM-Programm M für das Sortieren einer Liste von reellen Zahlen benötigt für genügend große n (Länge der Liste) mindestens $0.38 \cdot n \cdot \log n - O(n)$ Schritte.

(c) Welche der folgenden Aussagen ist/sind korrekt?

JA NEIN

- Wenn $A \leq_T B$, so gibt es Polynome p, q mit $T_B(n) \leq p(n) + T_A(q(n))$.
- Wenn $A \leq_T B$ und A einen Polynomialzeitalgorithmus besitzt, so auch B .
- Wenn $P \neq NP$, dann gilt nicht $\text{CLIQUE}_{\text{strukopt}} \leq_p \text{CLIQUE}_{\text{paropt}}$.

(d) Welche der folgenden Aussagen ist/sind korrekt?

JA NEIN

Wenn $\text{SAT} \leq_p L$, dann ist L NP-schwer.

Wenn $L \leq_p L'$ und $L \in \text{P}$, dann gilt $L' \in \text{P}$.

Zu jeder $O(n^l)$ -zeitbeschränkten NTM M existiert eine $O(2^{n^k})$ -zeitbeschränkte TM M' mit $L_{M'} = L_M$.

(e) Welche der folgenden Aussagen ist/sind korrekt?

JA NEIN

Wenn $\text{P} \cap \text{NPC} = \emptyset$, dann gilt $\text{P} \neq \text{NP}$.

Wenn $\text{P} \neq \text{NP}$, dann ist SAT nicht NP-vollständig.

Wenn $\text{P} = \text{NP}$, dann ist jede nichttriviale Sprache $L \in \text{P}$ schon NP-vollständig.

(f) Welche der folgenden Aussagen ist/sind für **alle** Sprachen $L \subseteq \Sigma^*$ mit $\# \notin \Sigma$ korrekt?

JA NEIN

Wenn $L = \{x \in \Sigma^* \mid \exists y \in \{0, 1\}^{\leq p(|x|)} : x\#y \in L_0\}$ für ein Polynom p und $L_0 \in \text{P}$, dann gilt $L \in \text{P}$.

Wenn $\bar{L} = \{x \in \Sigma^* \mid \forall y \in \{0, 1\}^{\leq p(|x|)} : x\#y \notin L_0\}$ für ein Polynom p und $L_0 \in \text{P}$, dann gilt $L \in \text{NP}$.

$L \in \text{NP} \implies \exists \text{ Polynom } p \exists L_0 \in \text{P} \forall x \in \Sigma^* : x \in L \Leftrightarrow (\exists y \in \{0, 1\}^{p(|x|)} : x\#y \in L_0)$

(g) Welche der folgenden Aussagen ist/sind korrekt?

JA NEIN

Für den Algorithmus FF (First Fit) gilt $\mathcal{R}_{\text{FF}} = 1.7$.

Für den Algorithmus GRS (Greedy Rucksack) gilt $\mathcal{R}_{\text{GRS}} = \infty$.

Für den Algorithmus MGRS (Maximum Greedy Rucksack) gilt $\mathcal{R}_{\text{MGRS}}^\infty \leq 2$.

(h) Welche der folgenden Aussagen ist/sind ein *offenes Problem*?

JA NEIN

MIN-TSP besitzt ein polynomielles Approximationsschema.

MIN-PARTITION besitzt ein polynomielles Approximationsschema.

MIN-BINPACKING besitzt einen Approximationsalgorithmus \mathcal{A} mit $\mathcal{R}_{\mathcal{A}} < \frac{3}{2}$.

(i) Welche der folgenden Aussagen ist/sind für **alle** NPO-Probleme \mathcal{P} korrekt?

JA NEIN

Wenn $\mathcal{P}_D \in \text{NPC}$, dann gilt $\mathcal{P}_D =_T \mathcal{P}_E =_T \mathcal{P}_C$.

Wenn $\mathcal{P}_D \in \text{NPC}$, dann gilt $\mathcal{P}'_D \leq_T \mathcal{P}_C$ für alle NPO-Probleme \mathcal{P}' .

Wenn \mathcal{P}_C einen Polynomialzeitalgorithmus besitzt, dann auch \mathcal{P}_D .

(j) Welche der folgenden Aussagen ist/sind korrekt?

JA NEIN

Für alle Sprachen L gilt: $L \in \text{co-NPC} \Leftrightarrow \bar{L} \in \text{NPC}$

Wenn NP unter Komplementbildung abgeschlossen ist, dann gilt $\text{NP} = \text{co-NP}$.

Wenn $\text{NP} \neq \text{co-NP}$, dann gilt $\text{NPC} \cap \text{co-NP} = \emptyset$.

Aufgabe 2 (Definitionen von Begriffen und Konzepten aus der Vorlesung)

[12 Punkte]

- (a) Definieren Sie die Klasse FP.
- (b) Definieren Sie den Begriff $L \leq_p L'$ (Sprache L ist polynomialzeitreduzierbar auf Sprache L').
- (c) Definieren Sie den Begriff $A \leq_T B$ (Problem A ist turingreduzierbar auf Problem B).
- (d) Definieren Sie den Begriff \mathcal{A} ist Approximationsalgorithmus für ein NPO-Problem \mathcal{P} .
- (e) Definieren Sie den Wert $m^*(x)$ einer optimalen Lösung für x sowie die Menge $SOL^*(x)$ der optimalen Lösungen für x für ein NPO-Problem \mathcal{P} .
- (f) Definieren Sie das Konstruktionsproblem \mathcal{P}_C für ein NPO-Problem \mathcal{P} .

Aufgabe 3 (Sätze bzw. Konstruktionen aus der Vorlesung)

[8+8 Punkte]

- (a) Sei $\varphi \equiv C_1 \wedge \dots \wedge C_r$ beliebige KNF-Formel, in der jede Klausel $C_i \equiv (l_{i1} \vee \dots \vee l_{is_i})$ aus mindestens 4 Literalen besteht ($s_i \geq 4$). Definieren Sie eine 3-KNF-Formel $\hat{\varphi}$ mit

$$\varphi \text{ ist erfüllbar} \iff \hat{\varphi} \text{ ist erfüllbar}$$

und beweisen Sie davon die Implikation „ \Leftarrow “.

- (b) Sei $x := (a_1, \dots, a_n, b) \in \mathbb{N}^{n+1}$, $n \geq 1$ beliebig. Definieren Sie $f(x) \in Seq(\mathbb{N})$ mit

$$x \in \text{RUCKSACK}^* \iff f(x) \in \text{PARTITION}$$

und beweisen Sie davon die Implikation „ \Leftarrow “.

Es war:

$$\text{RUCKSACK}^* := \{(a_1, \dots, a_m, b) \mid m \geq 1, \exists I \subseteq \{1, \dots, m\}: \sum_{i \in I} a_i = b\}$$

$$\text{PARTITION} := \{(a_1, \dots, a_n) \mid n \geq 2, \exists \text{Partition } I, J \text{ von } \{1, \dots, n\}: \sum_{i \in I} a_i = \sum_{j \in J} a_j\}$$

Aufgabe 4 (Sätze bzw. Konstruktionen aus der Vorlesung)

[8 Punkte]

Das NPO-Problem MIN-BINPACKING = (D, O, SOL, m, \min) ist wie folgt gegeben:

$$D = \{(a_1, \dots, a_n, c) \in \mathbb{N}^{n+1} \mid n \geq 1, 0 < a_i \leq c \text{ für } i = 1, \dots, n\}$$

$$O = \{(r_1, \dots, r_n) \in \mathbb{N}^n \mid 1 \leq r_i \leq n \text{ für } i = 1, \dots, n\}$$

$$SOL(x) = \{(r_1, \dots, r_n) \in O \mid \forall j = 1, \dots, n: \sum_{\substack{1 \leq i \leq n \\ r_i = j}} a_i \leq c\} \text{ für } x = (a_1, \dots, a_n, c) \in D$$

$$m(x, y) = \max\{r_1, \dots, r_n\} \text{ für } x = (a_1, \dots, a_n, c) \in D \text{ und } y = (r_1, \dots, r_n) \in SOL(x)$$

Sei \mathcal{A} ein Approximationsalgorithmus für MIN-BINPACKING, so daß für alle $x \in D$ gilt:

$$(*) \quad \frac{m_{\mathcal{A}}(x)}{m^*(x)} \leq r \text{ für eine Konstante } 1 \leq r < \frac{3}{2}$$

Konstruieren daraus einen Polynomialzeitalgorithmus \mathcal{B} , so daß für alle $x = (a_1, \dots, a_m) \in \mathbb{N}^m$ mit $m \geq 2$ und $0 < a_i$ für $i = 1, \dots, m$ gilt:

$$(**) \quad x \in \text{PARTITION} \iff \mathcal{B} \text{ akzeptiert } x$$

und beweisen Sie die Aussage (**).

Aufgabe 5 (Reduktionsmethode für NPC-Sprachen)

[12 Punkte]

Das Problem BASE besteht in folgender Aufgabe: Gegeben $m \geq 1$ Städte mit *Präferenzen* c_1, \dots, c_m , eine Menge von *Städteverbindungen* $E \subseteq \{(i, j) \mid 1 \leq i, j \leq m, i \neq j\}$ sowie eine *Präferenzschranke* c , entscheide, ob eine *Basis* I für E existiert, d.h. eine Städteauswahl $I \subseteq \{1, \dots, m\}$ mit

$$\forall (i, j) \in E: i \in I \text{ oder } j \in I,$$

mit *Gesamtpräferenz* $m(I) := \sum_{i \in I} c_i \leq c$. Formal erhalten wir die Sprache

$$\text{BASE} := \{\text{bin}(c_1) \# \dots \# \text{bin}(c_m) \# \langle E \rangle \# \text{bin}(c) \mid \exists \text{ Basis } I \text{ für } E \text{ mit } m(I) \leq c\}$$

wobei $\langle E \rangle := a_{11} \dots a_{1m} \dots a_{m1} \dots a_{mm}$ die Adjazenzmatrix von E ist.

Zeigen Sie: BASE ist NP-vollständig.

Aufgabe 6 (Approximationsalgorithmen)

[12 Punkte]

Betrachten Sie den folgenden Approximationsalgorithmus BEST FIT (BF) für das BINPACKING-Problem.

Eingabe: $(a_1, \dots, a_n, b) \in \mathbb{Q}^{n+1}$ mit $n \geq 1$ und $0 < a_i \leq b$ für $i = 1, \dots, n$
Ausgabe: Aufteilung $r_1, \dots, r_n \in \{1, \dots, n\}$ der n Objekte in Bins mit Kapazität b
for $i := 1, 2, \dots, n$ **do**
 $r_i :=$ Nummer eines maximal beladenen Bins, in das „Objekt“ a_i noch paßt, falls ein solches Bin existiert, andernfalls $r_i :=$ kleinste Nummer eines leeren Bins.

Sei $m_{\text{BF}}(x)$ die Anzahl der Bins, die der Algorithmus BF auf Eingabe x benutzt, und $m^*(x)$ die Anzahl der Bins einer optimalen Verteilung für x .

Zeigen Sie: Für alle Eingaben x gilt: $\frac{m_{\text{BF}}(x)}{m^*(x)} < 2$

Aufgabe 7 (Turingreduzierbarkeit)

[10 Punkte]

Das NPO-Problem RUCKSACK = (D, O, SOL, m, \max) ist wie folgt gegeben:

$$\begin{aligned} D &= \{(a_1, \dots, a_n, c_1, \dots, c_n, b) \in \mathbb{N}^{2n+1} \mid n \geq 1, 0 < a_i \leq b \text{ für } i = 1, \dots, n\} \\ O &= \{I \mid I \subseteq \{1, \dots, n\}, n \geq 1\} \\ SOL(x) &= \{I \mid I \subseteq \{1, \dots, n\}, \sum_{i \in I} a_i \leq b\} \text{ für } x = (a_1, \dots, a_n, c_1, \dots, c_n, b) \in D \\ m(x, I) &= \sum_{i \in I} c_i \text{ für } x = (a_1, \dots, a_n, c_1, \dots, c_n, b) \in D \text{ und } I \in SOL(x) \end{aligned}$$

- RUCKSACK_C sei die Aufgabe, auf Eingabe $x \in D$ eine optimale Lösung $I^* \in SOL(x)$ zu berechnen.
- RUCKSACK_E sei die Aufgabe, auf Eingabe $x \in D$ den Wert $m^*(x)$ einer optimalen Lösung für x zu berechnen.

Zeigen Sie: RUCKSACK_C \leq_T RUCKSACK_E

Viel Erfolg!