Algorithmen und Datenstrukturen 2

WiSe 2021/22 · Prof. Dr. Dell
22. Februar 2022
Klausurnummer: **0000**

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift)**:** _____

Vorname (Druckschrift)**:** _____

Matrikelnummer: _____

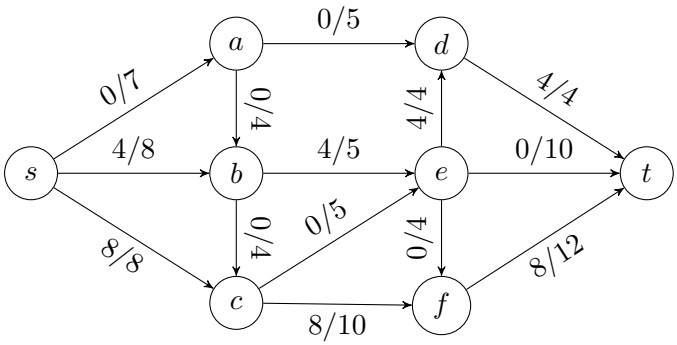
Studiengang: _____

Bitte Hinweise beachten:

- Die Klausur dauert **180 Minuten**.
- Name/Matrikelnummer **nur auf diesem Titelblatt!**
- **Kein Zusatzpapier** abgeben! Am Ende der Klausur finden Sie leere Seiten.
- **Alle elektronischen Geräte sind untersagt!** Insbesondere also Handys, Smartwatches, Taschenrechner. Ausschalten und wegpacken! Nichtbeachtung stellt einen Täuschungsversuch dar und führt zum Nichtbestehen der Klausur.
- Zugelassene Hilfsmittel:
  - Ein DIN A4 Blatt mit handschriftlichen Notizen (beidseitig).
  - Dokumentenechte Stifte in den Farben blau und schwarz. (Füller, Tipp-Ex, Tintenlöscher sind untersagt.)
- Bewertung:
  - **Immer nur eine Lösung** angeben, sonst gilt die Aufgabe als nicht gelöst.
  - Begründungen sind nur notwendig, wenn die Aufgabenformulierung dies verlangt.
  - In allen Multiple-Choice-Fragen sind **genau zwei von fünf Antworten** richtig. Wenn man x richtige Kreuze setzt und y falsche, erhält man $\max\{x-y,0\}$ Punkte, also entweder 0, 1, oder 2 Punkte. Zum Beispiel:

richtige Kreuze	x	2	2	1	1	0
falsche Kreuze	y	0	1	0	1	2
Punkte	$\max\{x-y,0\}$	2	1	1	0	0

Betrachten Sie den s-t-Fluss f in dem folgenden Flussnetzwerk (G, s, t, c):

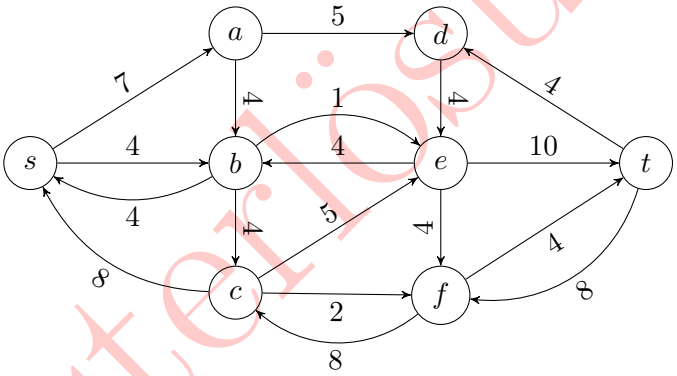


Jede Kante uv ist hierbei mit $f(uv)/c(uv)$ beschriftet, zum Beispiel trägt die Kante be den Fluss $f(be) = 4$ und hat die Kapazität $c(be) = 5$.

a) Der s-t-Fluss f hat den Wert ____12____.

(/ 1 Punkte)

b) Zeichnen Sie den zugehörigen Residualgraphen G_f in der folgenden Abbildung ein. Geben Sie dabei die Restkapazitäten durch Beschriftung an jeder Kante von G_f an.



Musterlösung:

(/ 3 Punkte)

c) Ein maximaler s-t-Fluss in (G, s, t, c) hat den Wert ____21____.

(/ 1 Punkte)

d) Ein minimaler s-t-Schnitt in (G, s, t, c) hat den Wert ____21____.

(/ 1 Punkte)

e) Ein Beispiel für einen minimalen s-t-Schnitt (S, T) in (G, s, t, c) ist gegeben durch:

$S = \{s,$ _____ a, b, d _____ $\}$

$T = \{t,$ _____ c, e, f _____ $\}$

(/ 2 Punkte)

Wir betrachten eine Datenstruktur, die zwei globale Variablen s und L sowie ein Array A aufrechterhält, und die Operationen AllocateNew, Insert und Decimate bereitstellt.

```
s = 0
L = 2
A[1..L] = neues Array der Länge L

function AllocateNew()
    B[1..L] = neues Array der Länge L
    for i from 1 to s do
        B[i] = A[i]
    A = B

function Insert(x)
    if s == L then
        L = (L + 1)^2        (*)
        AllocateNew()
    s = s + 1
    A[s] = x

function Decimate()
    s = ⌊s/10⌋
    for i from 1 to s do
        A[i] = A[i · 10]
    AllocateNew()
```

Wir nehmen an, dass wir ein neues Array beliebiger Länge in Zeit $O(1)$ anlegen können. Anfangs ist die Datenstruktur leer (also $s = 0$). Im Folgenden betrachten wir eine beliebige Sequenz von n Insert und/oder Decimate Operationen und sind an den amortisierten Laufzeiten in Θ-Notation **in Abhängigkeit von n** interessiert.

a) Die amortisierte Laufzeit von Insert ist:

$\Theta($ 1 $)$.

(/ 1 Punkte)

b) Die amortisierte Laufzeit von Insert wird $\Theta(n)$, wenn wir die mit `(*)` markierte Zeile wie folgt ändern:

$L =$ $L + 1$.

(/ 1 Punkte)

c) Angenommen wir ändern die mit `(*)` markierte Zeile zu $L = L + \lceil L^{1/3} \rceil$. Die amortisierte Laufzeit von Insert ist jetzt:

$\Theta($ $n^{2/3}$ $)$.

(/ 1 Punkte)

d) Wir betrachten jetzt wieder die Datenstruktur mit der ursprünglichen Zeile `(*)`. Die amortisierte Laufzeit von DECIMATE ist $\Theta($ 1 $)$. Begründen Sie Ihre Antwort, indem Sie eine beliebige Analysemethode benutzen, die aus der Vorlesung bekannt ist. Geben Sie an, welche Methode Sie anwenden, und wenden Sie nur eine Methode an (Aggregationsmethode, Buchhaltemethode oder Potenzialmethode).

Musterlösung: Wir betrachten eine beliebige Sequenz $O_1, \ldots, O_n \in \{\text{INSERT}, \text{DECIMATE}\}$ von Operationen und müssen die Behauptung beweisen, dass die amortisierten Kosten der DECIMATE-Operationen nach oben durch $O(1)$ beschränkt sind. Da die amortisierten Kosten mindestens die best-case Laufzeit $\Omega(1)$ sind, folgt dann zusammen, dass die amortisierten Kosten $\Theta(1)$ sind.
Um die Behauptung zu beweisen, benutzen wir die Buchhaltemethode. Zunächst beobachten wir, dass die worst-case Laufzeit von DECIMATE $O(s)$ beträgt, also höchstens $C \cdot s$ Kosten entstehen, wobei C eine positive Konstante ist. Da die Datenstruktur anfangs leer ist, können wir ein Element x erst mit DECIMATE löschen, nachdem es mit INSERT eingefügt wurde; außerdem kann jedes Element nur einmal gelöscht werden. Bei jeder INSERT(x)-Operation legen wir $2 \cdot C$ Euro in das Sparbuch, um später für einen Teil der DECIMATE-Operation zu bezahlen, die das Element x löscht.
Wir müssen jetzt nur noch argumentieren, warum die Spareinlage ausreicht, um die Kosten der DECIMATE-Operationen vollständig zu decken. Wir betrachten also einen Aufruf von DECIMATE. Vor dem Aufruf sind s Elemente in der Datenstruktur, die alle mit INSERT eingefügt worden sind, weshalb der Kontostand im Sparbuch mindestens $2C \cdot s$ Euro beträgt. Wir wollen aber nur dasjenige Geld nutzen, das den jetzt zu löschenden Elementen zugeordnet ist. In DECIMATE werden mindestens $s - \lfloor s/10 \rfloor \geq s/2$ Elemente gelöscht, weshalb alleine die zu löschenden Elemente bereits Cs Euro angespart haben. Dies reicht aus, um die tatsächlichen Kosten der DECIMATE-Operation zu decken, daher sind die amortisierten Kosten $O(1)$.

(/ 4 Punkte)

a) Welche zwei der folgenden fünf Aussagen sind wahr?

☐ Wenn es einen Polynomialzeitalgorithmus gibt, der für einen gegebenen Graphen eine echte 4-Färbung findet, dann ist $P \neq NP$.

☑ Wenn es einen Polynomialzeitalgorithmus gibt, der für einen gegebenen Graphen eine echte 4-Färbung findet, dann ist $P = NP$.

☑ Wenn es keinen Polynomialzeitalgorithmus gibt, der für einen gegebenen Graphen eine echte 4-Färbung findet, dann ist $P \neq NP$.

☐ Wenn es keinen Polynomialzeitalgorithmus gibt, der für einen gegebenen Graphen eine echte 4-Färbung findet, dann ist $P = NP$.

☐ Wenn $P = NP$ gilt, dann gibt es keinen Polynomialzeitalgorithmus, der für einen gegebenen Graphen eine echte 5-Färbung findet.

(/ 2 Punkte)

b) Seien Π und Π' zwei Entscheidungsprobleme in NP. Welche zwei der folgenden fünf Aussagen sind wahr?

☑ Um zu beweisen, dass ein Problem Π' NP-hart ist, genügt es, zu beweisen, dass Π NP-hart ist und dass es eine Polynomialzeitreduktion von Π nach Π' gibt.

☐ Um zu beweisen, dass ein Problem Π' NP-hart ist, genügt es, zu beweisen, dass Π NP-hart ist und dass es eine Polynomialzeitreduktion von Π' nach Π gibt.

☐ Eine typische Polynomialzeitreduktion von Π nach Π' bildet Instanzen von Π' auf Instanzen von Π ab.

☐ Wenn Π' NP-hart ist und es eine Polynomialzeitreduktion von Π nach Π' gibt, dann ist Π NP-hart.

☑ Wenn $P = NP$ gilt, dann gibt es eine Polynomialzeitreduktion von Π' auf das Problem $\{ G \mid G$ ist ungerichteter Graph mit höchstens 10 Knoten $\}$.

(/ 2 Punkte)

c) Wir betrachten das folgende Entscheidungsproblem CliqueAndEdge:

Eingabe:	Ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl $k \in \mathbb{N}$.
Frage:	Gibt es $k+1$ verschiedene Knoten $s_0, s_1, \ldots, s_k \in V$, sodass $\{s_0, s_1\}$ eine Kante und $\{s_1, \ldots, s_k\}$ eine Clique in G sind?

Beweisen Sie, dass das Problem CliqueAndEdge NP-hart ist.

Musterlösung:
Wir reduzieren Instanzen (G, k) von Clique auf Instanzen (G', k) von CliqueAndEdge:
Sei (G, k) eine Eingabe für Clique (d.h., die Frage, ob es eine k-Clique in G gibt). Sei G' der Graph, wo wir mit G beginnen und für jeden Knoten v einen neuen Knoten v' sowie die Kante $\{v, v'\}$ einfügen. Diese Reduktion läuft in Polynomialzeit.
Wir beweisen die Korrektheit:
$\Rightarrow$: Falls $(G, k) \in$ Clique, dann gibt es eine Teilmenge $C \subseteq V$, sodass der von C induzierte Teilgraph von G vollständig ist. Dann ist der von C induzierte Teilgraph von G' ebenfalls vollständig. Somit ergeben die Knoten aus C zusammen mit dem neu hinzugefügten Nachbarn eines beliebigen Knotens aus C die Knoten $s_0, \ldots, s_k$ mit den gewünschten Eigenschaften und es gilt $(G', k) \in$ CliqueAndEdge.
$\Leftarrow$: Falls $(G', k) \in$ CliqueAndEdge, dann gibt es $k+1$ Knoten in G', von denen k eine Clique bilden und der $k+1$-te Knoten eine Kante zu einem der anderen Knoten hat. Ist $k > 2$, besteht die k-Clique jedoch nur aus ursprünglichen Knoten von G, da die neu hinzugefügten Knoten Grad 1 haben. Ist $k \leq 2$, kann nur einer der drei Knoten in der Lösung ein neu hinzugefügter Knoten sein, da die Distanz zwischen diesen Knoten mindestens 3 beträgt. In beiden Fällen ergibt sich also aus der Lösung direkt eine k-Clique in G und es gilt $(G, k) \in$ Clique.

(/ 5 Punkte)

a) Sei $\Sigma = \{0, 1\}$. Welche zwei der folgenden fünf Sprachen L sind **unentscheidbar**?

☐ $L = \Sigma^*$

☐ $L = \{\, a_1 \dots a_n \in \Sigma^* \mid n \in \mathbb{N} \text{ und } a_1 + \dots + a_n = 9 \,\}$

☐ $L = \{\, \langle M, w \rangle \mid M \text{ hält auf Eingabe } w \text{ nach höchstens } |w|^2 \text{ Schritten} \,\}$

☑ $L = \{\, \langle M, w \rangle \mid M \text{ hält auf Eingabe } w \text{ nach mindestens } |w|^2 \text{ Schritten} \,\}$

☑ $L = \{\, \langle M \rangle \mid M(w) = 1 \text{ für alle } w, \text{ die gerade Zahlen kodieren} \,\}$

(/ 2 Punkte)

b) Welche zwei der folgenden fünf Aussagen sind wahr?

☑ Für alle Sprachen L gilt: Wenn L entscheidbar ist, dann ist $\overline{L}$ semi-entscheidbar.

☐ Jede semi-entscheidbare Menge ist entscheidbar.

☑ Für alle Sprachen L und L' gilt: Wenn es eine Reduktion von L nach L' gibt und L' entscheidbar ist, dann ist L entscheidbar.

☐ Für alle Sprachen L und L' gilt: Wenn es eine Reduktion von L nach L' gibt, dann ist L unentscheidbar oder L' entscheidbar.

☐ Für alle Sprachen L und L' gilt: Wenn $L \subseteq L'$ gilt und L' entscheidbar ist, so ist auch L entscheidbar.

(/ 2 Punkte)

c) Wir betrachten die folgende Sprache:

HaltDiverge $= \{ \langle M, N, w \rangle \mid M$ hält auf Eingabe w und N hält auf Eingabe w nicht $\}$.

Beweisen Sie, dass L unentscheidbar ist.

Musterlösung:
Aus der Vorlesung ist bekannt, dass die Sprache Halt mit

$$\text{Halt} = \{ \langle M, w \rangle \mid M \text{ hält auf Eingabe } w \}$$

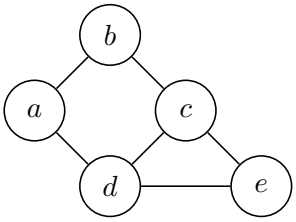
unentscheidbar ist. Wir geben eine Reduktion von Halt auf HaltDiverge an, um zu zeigen, dass HaltDiverge ebenfalls unentscheidbar ist.
Sei $\langle M, w \rangle$ eine beliebige Instanz von Halt. Sei N eine triviale Turingmaschine, die auf keiner Eingabe hält. Bei Eingabe $\langle M, w \rangle$ stellt unsere Reduktion die Orakelanfrage $\langle M, N, w \rangle$ an das Orakel für HaltDiverge. Wenn die Antwort des Orakels ja ist, geben wir ja aus, ansonsten nein.
Die Reduktion ist offensichtlich berechenbar. Es verbleibt, zu zeigen, dass sie korrekt ist: Es gilt

$$\begin{aligned} \langle M, w \rangle \in \text{Halt} &\Leftrightarrow M \text{ hält auf Eingabe } w \\ &\Leftrightarrow M \text{ hält auf Eingabe } w \text{ und } N \text{ hält auf Eingabe } w \text{ nicht} \quad (\text{da } N \text{ nie hält}) \\ &\Leftrightarrow \langle M, N, w \rangle \in \text{HaltDiverge} \end{aligned}$$

Das beweist die Korrektheit der Reduktion.

(/ 4 Punkte)

a) Gegeben sei folgender Graph G:



Geben Sie eine Sequenz von Kanten an, sodass die schrittweise Kontraktion dieser Kanten zu einem minimalen Schnitt von G führt:

$\{c,d\},\{\{c,d\},e\},\{\{c,d,e\},a\}$.

Musterlösung:
Minimale Schnitte von G sind:

A	B	Eine mögliche Sequenz von Kanten um A, B zu erhalten
$\{b\}$	$\{a,c,d,e\}$	$\{c,d\},\{\{c,d\},e\},\{\{c,d,e\},a\}$
$\{a\}$	$\{b,c,d,e\}$	$\{c,d\},\{\{c,d\},e\},\{\{c,d,e\},b\}$
$\{e\}$	$\{a,b,c,d\}$	$\{a,b\},\{\{a,b\},c\},\{\{a,b,c\},d\}$
$\{a,b\}$	$\{c,d,e\}$	$\{c,d\},\{a,b\},\{\{c,d\},e\}$

Geben Sie die genaue Wahrscheinlichkeit an, dass der Kontraktionsalgorithmus einen minimalen Schnitt von G ausgibt, wenn als erste Kante $\{c,e\}$ kontrahiert wird:

$\frac{7}{10}$.

Musterlösung:
Der Algorithmus kann maximal 3 Kanten auswählen und die Kante $\{c,d\}$ muss kontrahiert werden, um einen minimalen Schnitt zu finden. Wenn als erste Kante $\{c,e\}$ kontrahiert wird, dann ergibt sich als Gegenwahrscheinlichkeit, dass $\{c,d\}$ nicht als die nächsten beiden Kanten ausgewählt wird:

$$P(!\{c,d\}) = \frac{3}{5}\cdot\frac{2}{4} = \frac{3}{10}.$$

Somit erhalten wir den minimalen Schnitt mit einer Wahrscheinlichkeit von $\frac{7}{10}$.

(/ 2 Punkte)

b) Wir erinnern uns an den *Select*-Algorithmus: Für ein gegebenes Array $S=[a_1,\ldots,a_n]$ und eine gegebene Zahl $k\in\{1,\ldots,n\}$ liefert $Select(S,k)$ die k-kleinste Zahl im Array S zurück. Beispielsweise liefert $Select(S,1)$ die kleinste Zahl in S.

Intern muss der Algorithmus vor jedem Rekursionsschritt ein Pivotelement auswählen. Wir implementieren den Algorithmus nun mit der folgenden Pivotregel:

(R): Wähle zufällig das minimale oder maximale Element als nächstes Pivotelement.

Wir rufen $Select(S,k)$ mit Pivotregel (R) auf. Geben Sie die Laufzeit in Θ-Notation abhängig von n an:

$\Theta($ n^2 $)$.

Sei $B = [4, 5, 1, 6, 3, 2]$ ein Array von ganzen Zahlen. Geben Sie eine mögliche Reihenfolge von Pivotelementen an, die für den Aufruf $Select(B, 5)$ mit Pivotregel (R) ausgewählt werden können:

Ein Beispiel wären die Pivotelemente in der Reihenfolge: $6, 1, 2, 3, 4$.

(/ 2 Punkte)

c) Wir betrachten den *Quicksort*-Algorithmus, um das Array $C = [4, 5, 1, 6, 3, 2]$ zu sortieren. Geben Sie eine Reihenfolge von Pivotelementen an, damit der Aufruf $Quicksort(C)$ die kleinstmögliche Anzahl von Vergleichen benötigt:

Man muss immer den Median als Pivotelement wählen, also: $3, 1, 5$.

(/ 1 Punkte)

d) Professor Regloh möchte auf seiner Website gerne Porträtfotos von sich und seinen wissenschaftlichen Mitarbeiter:innen veröffentlichen. Dazu beauftragt er seine Sekretärin Aidualc die Namen von m Fotograf:innen aus Frankfurt und ein Beispiel ihrer Arbeit zusammenzustellen. Es ist allerdings allgemein bekannt, dass es nur k gute Fotograf:innen in Frankfurt gibt. Um eine:n gute:n Fotograf:in zu beauftragen, geht er wie folgt vor:

> FindeFotograf:in
> Ziehe einen uniform zufälligen Namen n_i eines:r Fotograf:in. Prüfe dann anhand von n_i's Beispielfotos, ob n_i gut ist. Wenn ja, dann beauftrage n_i damit, Porträtfotos für Professor Regloh zu machen. Wenn das nicht der Fall ist, suche weiter.

Sei n_j ein:e gute:r Fotograf:in. Sei X_j eine Zufallsvariable, die den Wert 1 annimmt, wenn Professor Regloh n_j beauftragt und 0 wenn nicht. Dann gilt:

$P(X_j = 1) =$ $\frac{1}{k}$.

Was ist die erwartete Laufzeit von FindeFotograf:in? Oder in anderen Worten, was ist die erwartete Anzahl von Porträtfotos, die Professor Regloh anschauen muss?

$\frac{m}{k}$

Musterlösung: In jeder Runde ist die Wahrscheinlichkeit eine:n gute:n Fotograf:in anzuschauen $\frac{k}{m}$. Diese „Erfolgswahrscheinlichkeit“ ist in jeder Runde gleich und die einzelnen Runden sind unabhängig. Die Anzahl Runden bis zum ersten „Erfolg“ ist geometrisch verteilt, der Erwartungswert ist daher $\frac{1}{\frac{k}{m}} = \frac{m}{k}$.

(/ 2 Punkte)

a) Welche zwei der folgenden fünf Aussagen sind wahr?

☐ Jede *basic feasible solution* eines linearen Programms ist eine optimale Lösung.

☑ Jedes lineare Programm, dessen zugehöriges Polyhedron beschränkt und nicht leer ist, hat eine optimale Lösung.

☐ Jede optimale Lösung in einem linearen Programm ist auch eine *basic feasible solution*.

☐ In jedem linearen Programm, das eine optimale Lösung hat, ist die optimale Lösung eindeutig.

☑ Die Menge aller optimalen Lösungen eines linearen Programms ist konvex.

(/ 2 Punkte)

b) Welche zwei der folgenden fünf Aussagen sind wahr?

☐ Jede optimale Lösung des dualen Programms ist eine zulässige Lösung für das primale Programm.

☐ Jede zulässige Lösung x des primalen Programms hat einen Wert $c^T x$, der echt kleiner ist als der Wert der optimalen Lösung des dualen Programms.

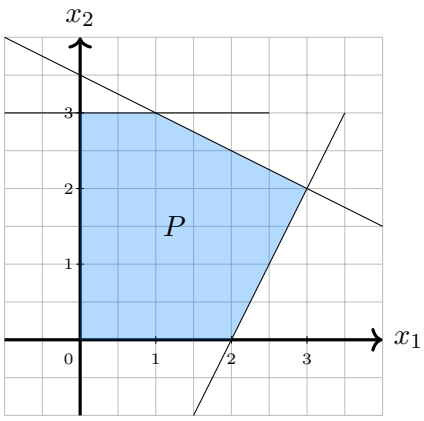
☐ Es gibt immer eine zulässige Lösung x des primalen Programms und eine zulässige Lösung y des dualen Programms, sodass $c^T x = b^T y$ gilt.

☑ Wenn das primale Programm unbeschränkt ist, dann ist das duale Programm *infeasible*.

☑ *Complementary slackness* kann benutzt werden, um von einer optimalen Lösung des dualen Programms eine optimale Lösung des primalen Programms zu berechnen.

(/ 2 Punkte)

c) Gegeben sei folgendes Polytop P in $\mathbb{R}^2$:



Geben Sie alle zulässigen Basen (*basic feasible solutions*) von P der Form (x_1, x_2) an:

$(0,0), (0,3), (1,3), (3,2), (2,0)$.

Betrachten Sie nun das folgende lineare Programm (LP_1):

$\max \begin{pmatrix} 1 & 3 \end{pmatrix}^T \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ unter der Nebenbedingung $(x_1, x_2) \in P$

Geben Sie eine optimale Lösung für (LP_1) mit dem zugehörigen Zielfunktionswert an:

$(1,3)$ mit Zielfunktionswert 10 .

(/ 2 Punkte)

d) Gegeben sei folgendes lineare Programm (LP_2):

$$\begin{array}{lrcrcl} \min & x_1 & + & 16x_2 & & \\ \text{s.t.} & 2x_1 & & & \leq & 64 \\ & 4x_1 & + & 32x_2 & \geq & 128 \\ & 8x_1 & & & = & 256 \\ & x_1 & & & \geq & 0 \\ & & & x_2 & \in & \mathbb{R} \end{array}$$

Geben Sie (LP_2) in Standardform an:

Musterlösung:

$$\begin{array}{lrcrcrcrcrcl} \max & -x_1 & - & 16x_2^+ & + & 16x_2^- & & & & & \\ \text{s.t.} & 2x_1 & & & & & + & s_1 & & & = 64 \\ & 4x_1 & + & 32x_2^+ & - & 32x_2^- & & & - & s_2 & = 128 \\ & 8x_1 & & & & & & & & & = 256 \\ & x_1 & , & x_2^+ & , & x_2^- & , & s_1 & , & s_2 & \geq 0 \end{array}$$

(/ 3 Punkte)

Sei $G = (V, E)$ ein ungerichteter Graph. Eine Knotenmenge $I \subseteq V$ heißt *unabhängig*, falls $(v, w) \notin E$ für alle $v, w \in I$ gilt. Wir definieren das Entscheidungsproblem IndependentSet wie folgt:

Problem:	IndependentSet
Eingabe:	Ein Graph G mit n Knoten und eine Zahl $k \in \mathbb{N}$.
Frage:	Enthält G eine unabhängige Knotenmenge mit k Knoten?

a) Sei $N(v)$ die Menge aller Nachbarn von Knoten v. Beweisen Sie: Für alle maximalen unabhängigen Mengen $I \subseteq V$ und alle Knoten $v \in V$ gilt $(N(v) \cup \{v\}) \cap I \neq \emptyset$.

Musterlösung:

Beweis. Sei I eine maximale unabhängige Menge von G und sei $v \in V$ ein Knoten von G. Wir beweisen die Behauptung durch Widerspruch: Angenommen $N(v) \cup \{v\}$ und I sind disjunkt. Dann ist die Menge $I \cup \{v\}$ eine unabhängige Menge, denn kein Nachbar von v ist in I enthalten. Da $I \cup \{v\}$ größer ist als I, erhalten wir einen Widerspruch zu der Maximalität von I. Daher muss die Annahme falsch sein, also können $N(v) \cup \{v\}$ und I nicht disjunkt sein. □

(/ 3 Punkte)

b) Sei Δ der Maximalgrad von G, also die kleinste Zahl Δ, sodass jeder Knoten von G höchstens Δ Nachbarn hat. Überlegen Sie sich einen **rekursiven** *bounded search tree* Algorithmus, der das IndependentSet-Problem in einer Laufzeit von

$$f(k, \Delta) \cdot \text{poly}(n)$$

löst, wobei $\text{poly}(n)$ ein Polynom in n und $f(k, \Delta)$ eine beliebige Funktion ist, die nur von k und Δ abhängt. Beschreiben Sie Ihren Algorithmus **in Pseudocode**.

Hinweis: Aufgabenteil a) könnte nützlich sein.

Musterlösung:

```
function FindIS(G = (V, E), k)
    return FindISAux(G, k, ∅, V)
function FindISAux(G = (V, E), k, Sol, Candidates)
    if |Sol| = k then
        return true
    Wähle beliebigen Knoten v ∈ Candidates
    for all w ∈ N_G[v] do
        if w ∉ Candidates then
            continue
        if FindISAux(G, k, Sol ∪ {w}, Candidates ∖ N_G[w]) then
            return true
    return false
```

(/ 5 Punkte)

c) Betrachten Sie folgenden rekursiven Algorithmus:

```
function Rec(n, k)
    if k ≤ 0 then return 0
    else
        for i from 1 to n do
            print "Irgendeine Ausgabe, die nur wegen der Laufzeit wichtig ist."
        return Rec(n − 99, k − 1) + Rec(n − 7, k − 1) + Rec(n − 5, k − 1)
```

Geben Sie jetzt die Laufzeit von Rec in Θ-Notation in Abhängigkeit von n und k an:

$\Theta\Big($ $3^k n$ $\Big)$.

Beweisen Sie, dass Rec tatsächlich die von Ihnen angegebene Laufzeit hat.

Musterlösung:
Wir betrachten den Rekursionsbaum von Rec. Jeder Knoten ist mit der aktuellen Eingabe (n, k) markiert. Jeder Knoten (n, k) mit $k > 0$ hat drei Kinder, die mit $(n - 99, k - 1)$, $(n - 7, k - 1)$, $(n - 5, k - 1)$ markiert sind. Es handelt sich also um einen ternären Baum der Tiefe k, und der hat $O(3^k)$ Knoten. An jedem Knoten durchläuft Rec eine for-Schleife der Länge n. Insgesamt ist die Laufzeit also $\Theta(3^k n)$.

(/ 2 Punkte)