Algorithmen und Datenstrukturen 2

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

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift)**:** _____

Vorname (Druckschrift)**:** _____

Matrikelnummer: _____

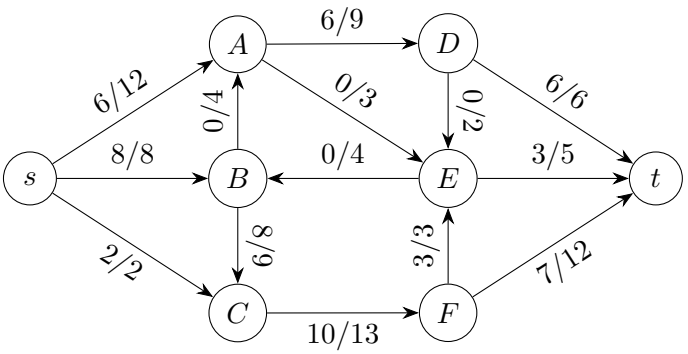
Studiengang: _____

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):

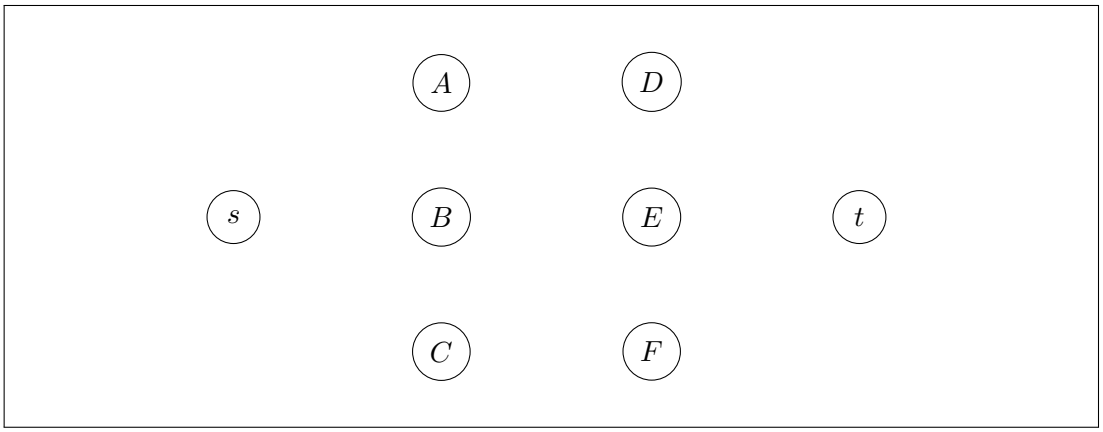


Jede Kante uv ist hierbei mit $f(uv)/c(uv)$ beschriftet, zum Beispiel trägt die Kante AD den Fluss $f(AD) = 6$ und hat die Kapazität $c(AD) = 9$.

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

(/ 1 Punkte)

b) Zeichnen Sie den zugehörigen Residualgraphen G_f für das Flussnetzwerk (G, s, t, c) und den Fluss f in der folgenden Abbildung ein. Geben Sie dabei die Restkapazitäten durch Beschriftung an jeder Kante von G_f an. Kanten mit Restkapazität 0 sollen **nicht** gezeichnet werden.



(/ 3 Punkte)

c) Der Wert eines maximalen Flusses in (G, s, t, c) ist _____.

(/ 1 Punkte)

d) Geben Sie die beiden Mengen S, T eines minimalen Schnittes in (G, s, t, c) an.

$S = \{s,$ _____$\}$,

$T = \{t,$ _____$\}$.

(/ 2 Punkte)

e) Welche zwei der folgenden fünf Aussagen über das Flussnetzwerk (G, s, t, c) und den Fluss f sind wahr?

□ Der Capacity-Scaling-Algorithmus mit Eingabe (G, s, t, c) könnte (s, B, C, F, t) als ersten augmentierenden Pfad auswählen.

□ Der Algorithmus von Ford und Fulkerson mit Eingabe (G, s, t, c) könnte (s, A, E, B, C, F, t) als ersten augmentierenden Pfad auswählen.

□ Der Fluss f könnte sich als Zwischenergebnis bei der Anwendung des Algorithmus von Edmonds und Karp mit Eingabe (G, s, t, c) ergeben.

□ In (G, s, t, c) hat der Pfad (s, A, E, B, C, F, t) eine "bottleneck"-Kapazität von 13.

□ Der Capacity-Scaling-Algorithmus mit Eingabe (G, s, t, c) könnte (s, A, E, B, C, F, t) als ersten augmentierenden Pfad auswählen.

(/ 2 Punkte)

f) Gegeben sei ein Graph H mit Knotenmenge $\{1, 2, 3, 4, 5\}$. Wir führen vier Schritte von Floyd-Warshall's Algorithmus auf H aus und erhalten die folgende Matrix als Zwischenergebnis:

$$\begin{pmatrix} 0 & 7 & 5 & 3 & 5 \\ 8 & 0 & 4 & 6 & 5 \\ \infty & \infty & 0 & \infty & 1 \\ 4 & 11 & 2 & 0 & 2 \\ 6 & 13 & 4 & 2 & 0 \end{pmatrix}$$

Floyd-Warschall's Algorithmus wird noch einen letzten Schritt ausführen und ist dann fertig. Geben Sie für die folgenden Wahlen von u und v jeweils die minimale Länge eines Weges von u nach v an.

$u = 1$ und $v = 4$: _____

$u = 3$ und $v = 2$: _____

$u = 3$ und $v = 3$: _____

$u = 5$ und $v = 1$: _____

(/ 2 Punkte)

g) Das Vorlesungsteam von Professor Aglot umfasst n Personen (Tutor:innen und Senior Staff) und will jede Woche einen Lösungsspaziergang durchführen. Das Semester umfasst m Wochen. Um einen Lösungsspaziergang durchzuführen, sind für jede Woche die folgenden vier Tätigkeiten notwendig:

1. Alle Übungsaufgaben der Woche lösen
2. Folien für den Lösungsspaziergang entwerfen
3. Feedback auf Folienentwürfe geben
4. Lösungsspaziergang halten

Jede Tätigkeit benötigt in jeder Woche eine Stunde. Die i-te Person darf im ganzen Semester höchstens z_i Stunden arbeiten und nur an einer Teilmenge $T_i \subseteq \{1, \ldots, 4\}$ von Tätigkeiten arbeiten. Außerdem darf jede Person höchstens eine Tätigkeit pro Woche übernehmen.

Wir wollen eine Zuordnung ermitteln, die angibt, wer in welcher Woche welche Tätigkeiten übernehmen soll. Das heißt, wir wollen ein Array $A[1..n][1..m]$ berechnen, sodass $A[i][j] = k$ ist, wenn Person i in Woche j die Tätigkeit $k \in T_i$ übernimmt.

Beschreiben Sie einen Algorithmus, der in Polynomialzeit eine Zuordnung A bestimmt, sodass alle Lösungsspaziergänge stattfinden und niemand zu viel arbeitet; falls keine solche Zuordnung existiert, soll der Algorithmus dies erkennen.

(/ 3 Punkte)

Wir betrachten eine Funktion FUN. Ein Aufruf FUN(i) verursacht Laufzeitkosten in Höhe von T_i, wobei

$$T_i = \begin{cases} i & \text{wenn } i \text{ eine Potenz von 3 ist, also } i = 3^k \text{ für ein } k \in \mathbb{N}, \\ 1 & \text{sonst.} \end{cases}$$

Zum Beispiel verursacht der Aufruf FUN(2) die Kosten $T_2 = 1$ und der Aufruf FUN(9) die Kosten $T_9 = 9$.

a) Wir rufen die Funktion FUN wie folgt n Mal auf:

```
for i from 1 to n do
    FUN(i)
```

Welche amortisierte Laufzeit hat FUN in diesem Programmschnipsel? Geben Sie Ihre Antwort in Θ-Notation **als Funktion von n** an:

$\Theta($_____$)$

(/ 1 Punkte)

b) Beweisen Sie **mithilfe der Aggregationsmethode**, dass Ihre Behauptung in a) korrekt ist.

(/ 2 Punkte)

c) Wir rufen die Funktion FUN wieder n Mal auf, diesmal aber wie folgt:

```
for i from 1 to n do
    FUN(3^i)
```

Welche amortisierte Laufzeit hat FUN in diesem Programmschnipsel? Geben Sie Ihre Antwort in Θ-Notation **als Funktion von n** an:

$\Theta($_____$)$

(/ 1 Punkte)

d) Wir haben eine mögliche Implementierung der Funktion FUN gefunden. Betrachten Sie folgenden Pseudocode:

```
s = 1
L = 1
function FUN(i)
    if s = L then
        L = Z · L          (*)
        for j from 1 to s do
            print "Hallo"
    else
        print "Hallo"
    s = s + 1
```

Hierbei sind s und L globale Variablen, die Anfangs 1 sind und von FUN verändert werden. Wir messen die Laufzeitkosten von FUN nur anhand der Anzahl der **print**-Anweisungen, die ausgeführt werden. Welche Konstante Z in der mit **(*)** markierten Zeile führt dazu, dass die Laufzeitkosten des i-ten Aufrufs für alle i gleich T_i sind?

$Z =$ _____.

(/ 1 Punkte)

e) Angenommen wir ändern die in d) mit **(*)** markierte Zeile zu $L = (\sqrt{L} + 1)^2$, um eine Funktion FUN2(i) zu erhalten.

Wir rufen die Funktion FUN2 n Mal wie folgt auf:

```
for i from 1 to n do
    FUN2(i)
```

Welche amortisierte Laufzeit hat FUN2 in diesem Programmschnipsel? Geben Sie Ihre Antwort in Θ-Notation **als Funktion von n** an:

$\Theta($_____$)$

(/ 1 Punkte)

Wir haben in der Vorlesung gezeigt, dass das Hamiltonkreisproblem (HC) NP-vollständig ist. Wir betrachten jetzt die folgende Variante von HC, nämlich das Entscheidungsproblem HCAE (Hamilton Cycle and Edge):

Eingabe:	Ungerichteter Graph $G = (V, E)$ mit $n = \|V\|$.
Frage:	Gibt es n verschiedene Knoten $v_1, \ldots, v_n \in V$, sodass $\{v_1, v_2\}$ eine Kante und $v_2, v_3, \ldots, v_n, v_2$ ein einfacher Kreis in G sind?

a) Zeichnen Sie einen beliebigen Graphen mit genau **fünf** Knoten, für den HCAE die Antwort **ja** fordert:

(/ 1 Punkte)

b) Zeichnen Sie einen beliebigen Graphen mit genau **fünf** Knoten, für den HCAE die Antwort **nein** fordert:

(/ 1 Punkte)

c) Das Problem HCAE ist NP-hart. Der folgende fehlerhafte Text soll angeblich beweisen, dass HCAE NP-hart ist:

> Wir geben eine Polynomialzeit-Reduktion HC $\leq$ HCAE an. Da HC aus der Vorlesung bereits als NP-hart bekannt ist, wird diese Reduktion zeigen, dass auch HCAE NP-hart ist:
>
> 1. Die Eingabe für die Reduktion ist eine Instanz $G = (V, E)$ von HC mit n Knoten.
> 2. Wir konstruieren eine Instanz G', indem wir zunächst G kopieren, also $G' = G$.
> 3. Jetzt fügen wir einen neuen Knoten z zu G' hinzu.
> 4. Außerdem fügen wir die Kanten $\{v, z\}$ für alle $v \in V$ zu G' hinzu.
> 5. Nun senden wir G' an das Orakel HCAE und behaupten, dass G eine ja-Instanz für HC ist genau dann, wenn G' eine ja-Instanz für HCAE ist.

Begründen Sie in 1-3 Sätzen möglichst genau, warum die angegebene Reduktion nicht korrekt ist:

(/ 2 Punkte)

d) Welche zwei der folgenden fünf Aussagen sind wahr?

- ☐ Wenn es keinen Polynomialzeitalgorithmus für HCAE gibt, dann gilt P $\neq$ NP.
- ☐ Wenn P = NP gilt, dann gibt es keinen Polynomialzeitalgorithmus für HCAE.
- ☐ Wenn P $\neq$ NP gilt, dann gibt es keinen Polynomialzeitalgorithmus für HCAE.
- ☐ Wenn P = NP gilt, dann ist jede Instanz von HCAE eine ja-Instanz.
- ☐ Wenn es einen Polynomialzeitalgorithmus für HCAE gibt, dann gilt P $\neq$ NP.

(/ 2 Punkte)

e) Beweisen Sie, dass HCAE NP-hart ist, indem Sie eine korrekte Reduktion angeben und beide Implikationsrichtungen der Korrektheit Ihrer Reduktion beweisen.

Hinweis: Für die richtige Reduktion muss nur eine Zeile in c) verändert werden! Die korrekten Zeilen brauchen Sie nicht zu wiederholen.

(/ 3 Punkte)

a) Welche zwei der folgenden fünf Sprachen sind **unentscheidbar**?

□ $\emptyset$

□ $\{ \langle M, w \rangle \mid M \text{ verlässt auf Eingabe } w \text{ den Startzustand} \}$

□ $\{ \langle M \rangle \mid M \text{ akzeptiert die Sprache } \emptyset \}$

□ $\{ \langle M, w \rangle \mid M \text{ besucht auf Eingabe } w \text{ jeden Zustand mindestens einmal} \}$

□ $\{ \langle M \rangle \mid M \text{ akzeptiert die Sprache } \textsc{Diverge} \}$

(/ 2 Punkte)

Sei $M_{\text{ystery}} = (Q, \Sigma, \Gamma, \delta, q_0, \texttt{acc}, \texttt{rej}, \square)$ eine Turingmaschine, wobei $Q := \{ q_i \mid 0 \leq i \leq 5 \}$, $\Sigma := \{0, 1, \#\}$, $\Gamma := \Sigma \cup \{\$, \square\}$ und die Überführungsfunktion δ durch die folgenden Übergänge gegeben ist:

$(q_0, 1) \to (q_1, \$, +1)$
$(q_0, \#) \to (q_5, \#, +1)$

$(q_1, 1) \to (q_1, 1, +1)$
$(q_1, \#) \to (q_2, \#, +1)$

$(q_2, x) \to (q_2, x, +1)$ für $x \in \{0, 1\}$
$(q_2, \square) \to (q_3, \square, -1)$

$(q_3, 0) \to (q_3, 1, -1)$
$(q_3, 1) \to (q_4, 0, -1)$

$(q_4, y) \to (q_4, y, -1)$ für $y \in \{0, 1, \#\}$
$(q_4, \$) \to (q_0, \$, +1)$

$(q_5, 0) \to (q_5, 0, +1)$
$(q_5, \square) \to (\texttt{acc}, \square, +1)$

b) Die Maschine M_{ystery} akzeptiert zum Beispiel das Wort 1111#00100. Geben Sie irgendein anderes Wort w_{acc} an, das von M_{ystery} **akzeptiert** wird.

$w_{\text{acc}} =$ _____

(/ 1 Punkte)

c) Geben Sie irgendein Wort $w_{\text{non-acc}}$ an, das von M_{ystery} **nicht akzeptiert** wird.

$w_{\text{non-acc}} =$ _____

(/ 1 Punkte)

d) Welche Sprache wird von der Turingmaschine M_{ystery} akzeptiert?

$$L = \left\{ x \in \{0, 1, \#\}^* \;\middle|\; \exists n, m \in \mathbb{N}\colon\ x = \underline{\qquad\qquad\qquad\qquad} \right\}$$

(/ 2 Punkte)

e) Wir betrachten die folgende Sprache:

HaltEven $= \{ \langle M, w \rangle \mid M$ hält auf Eingabe w in einer geraden Anzahl von Schritten $\}$.

Beweisen Sie, dass HaltEven unentscheidbar ist.

(/ 4 Punkte)

Aufgabe 5: Randomisierte Algorithmen — $2 + 2 + 2 = 6$ Punkte

Professor Regloh möchte die schönste Lösung für eine Sternaufgabe ermitteln, um die Autoren mit einem Preis auszuzeichnen. Für die ausgewählte Sternaufgabe wurden n Lösungen abgegeben.

a) Angenommen Professor Regloh geht wie folgt bei seiner Auswahl vor:

> **while** noch nicht alle Lösungen betrachtet wurden **do**
> Wähle uniform zufällig eine von allen n Lösungen aus und betrachte diese.

Wie groß ist die Wahrscheinlichkeit, dass Professor Regloh nach drei Iterationen der while-Schleife *genau drei verschiedene* Lösungen betrachtet hat? Geben Sie das Ergebnis **als Funktion von n** an.

Antwort: _____

Wie viele Lösungen muss Professor Regloh in Erwartung betrachten, bis er alle Lösungen mindestens einmal betrachtet hat? Geben Sie das Ergebnis in Θ-Notation als Funktion von n an:

$\Theta($_____$)$

(/ 2 Punkte)

b) Angenommen, die fantastische Lernplattform El Doom sortiert die n abgegebenen Lösungen gleichverteilt zufällig, dann geht Professor Regloh wie folgt vor:

Er geht der zufälligen Reihenfolge $L_1, \ldots, L_n$ nach alle abgegebenen Lösungen durch. Wann immer er dabei eine noch schönere als die schönste bisher gesehene Lösung findet, schreibt er eine begeisterte Discord-Nachricht an seine Doktorandin Oel. Dabei ist für ihn im paarweisen Vergleich immer eine Lösung eindeutig schöner als die andere.

Sei X_i eine Zufallsvariable, die den Wert 1 annimmt, wenn Professor Regloh für die i-te Lösung L_i eine Discordnachricht an Oel schickt, und 0 wenn nicht. Dann gilt:

$P(X_i = 1) =$ _____

Wie viele Discordnachrichten empfängt Oel von Professor Regloh im Erwartungswert? Geben Sie das Ergebnis in Θ-Notation **als Funktion von n** an:

$\Theta($_____$)$

(/ 2 Punkte)

c) Welche zwei der folgenden fünf Aussagen sind wahr?

□ Um den Median in einem sortierten Array zu bestimmen, ist der randomisierte *Select*–Algorithmus die asymptotisch schnellste bekannte Methode.

□ Es ist vorteilhaft für den *Quicksort*–Algorithmus, wenn das gewählte Pivotelement immer das maximale Element ist.

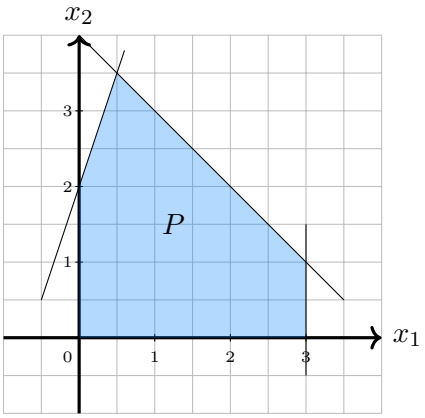
□ Es ist vorteilhaft für den *Quicksort*–Algorithmus, wenn das gewählte Pivotelement immer das Median-Element ist.

□ Wir wissen, dass randomisierte Algorithmen in erwarteter Polynomialzeit NP-harte Probleme lösen können.

□ Um den Median in einem unsortierten Array zu bestimmen, ist der randomisierte *Select*–Algorithmus die asymptotisch schnellste bekannte Methode.

(/ 2 Punkte)

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



Sei das lineare Programm (LP_1) gegeben durch $\max x_1$ unter der Nebenbedingung $(x_1, x_2) \in P$. Welche zwei der folgenden fünf Aussagen bezüglich (LP_1) sind wahr?

☐ Die Lösung $(3, 0)$ mit Zielfunktionswert 2 ist die eindeutige optimale Lösung für (LP_1).

☐ Der Knoten $(3, 1)$ ist *locally optimal* bezüglich der Zielfunktion von (LP_1).

☐ Das duale Programm von (LP_1) hat 5 Variablen und 2 Constraints.

☐ Die Lösung $(3, 0.5)$ ist eine optimale Lösung für (LP_1).

☐ Eine mögliche Reihenfolge von Knoten, die der Simplex Algorithmus auf der Suche nach einer optimalen Lösung für (LP_1) betrachten kann, ist durch $(0, 0), (0, 2), (0, 0), (3, 0)$ gegeben.

(/ 2 Punkte)

b) Welche zwei der folgenden fünf Aussagen sind wahr?

☐ Durch eine LP Relaxierung kann immer eine optimale Lösung für ein MILP in polynomieller Zeit gefunden werden.

☐ Für ein LP mit n Variablen und m Constraints gibt es höchstens $\frac{n!}{m!(n-m)!}$ *basic feasible solutions*.

☐ Die Simplex Methode benutzt ein geometrisches Divide-and-Conquer Verfahren, um eine optimale Lösung für ein LP zu finden.

☐ Wenn das primale Programm eine zulässige Lösung hat, dann ist das zugehörige duale Programm beschränkt.

☐ Von der Wahl des Zielfunktionsvektors ist abhängig, ob ein Knoten zulässig ist.

(/ 2 Punkte)

c) Seien das primale Program (P) und das duale Program (D) gegeben durch:

$$
\begin{array}{llrl}
 & \max & c^T x & \\
(P) & & Ax & \leq b \\
 & & x & \geq 0
\end{array}
\qquad\qquad
\begin{array}{llrll}
 & \min & b^T y & & \\
 & & A^T y & \geq c & (D) \\
 & & y & \geq 0 &
\end{array}
$$

Weak Duality. Wenn x eine zulässige Lösung für (P) und y eine zulässige Lösung für (D) ist, dann gilt $c^T x \leq b^T y$.

Beweisen Sie, dass Weak Duality gilt.

(/ 1 Punkte)

d) Gegeben sei folgendes lineares Programm (LP_2):

$$
\begin{array}{llrcrcrl}
\max & & x_1 & + & 5x_2 & + & 34x_3 & \\
\text{s.t.} & & 2x_1 & + & 8x_2 & + & 55x_3 & \leq 144 \\
 & & 2x_1 & + & 13x_2 & & & \leq 233 \\
 & & 3x_1 & + & 21x_2 & + & 89x_3 & \leq 377 \\
 & & x_1 & , & x_2 & , & x_3 & \geq 0
\end{array}
$$

Geben Sie das duale lineare Programm für (LP_2) an:

(/ 3 Punkte)

Wir erinnern uns, dass ein c-Approximationsalgorithmus für ein Minimierungsproblem immer eine Lösung von Wert $\widetilde{\mathrm{OPT}}$ ausgibt, wobei gilt:

$$\mathrm{OPT} \leq \widetilde{\mathrm{OPT}} \leq c \cdot \mathrm{OPT}\,.$$

Hier ist ein heuristischer Algorithmus für das Minimum Vertex-Cover Problem:

function MyVertexCover(G)
  $S = \emptyset$
  while es gibt einen Knoten v in G, der mindestens einen Nachbarn hat **do**
    $S = S \cup \{v\}$
    Lösche v sowie alle an v inzidenten Kanten aus G.
  return S

a) Beweisen Sie, dass MyVertexCover(G) immer ein Vertex-Cover von G zurückliefert:

(/ 2 Punkte)

b) Beweisen Sie, dass MyVertexCover kein 7-Approximationsalgorithmus für das Minimum Vertex-Cover Problem ist:

(/ 2 Punkte)

c) Wir betrachten jetzt nur Eingaben G mit Maximalgrad höchstens 13, das heißt, jeder Knoten von G hat mindestens 0 und höchstens 13 Nachbarn.

Beweisen Sie, dass MyVertexCover(G) ein 14-Approximationsalgorithmus für das Minimum Vertex-Cover Problem auf solchen Eingaben ist.

(/ 4 Punkte)