Algorithmen und Datenstrukturen 2

WiSe 2022/2023 · Prof. Dr. M. Hoefer
5. April 2023 (Zweitklausur)
Klausurnummer: **0000**

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift)**:** _____

Vorname (Druckschrift)**:** _____

Matrikelnummer: _____

Studiengang: _____

Beachten Sie die folgenden Hinweise:

- Schreiben Sie Ihre persönlichen Daten **nur auf dieses Titelblatt**. **Merken oder notieren** Sie sich Ihre Klausurnummer **0000**, da nur unter dieser Nummer die Ergebnisse veröffentlicht werden.
- Legen Sie Ihre **Goethe-Card** deutlich sichtbar an Ihren Platz, damit wir während der Klausur Ihre Identität überprüfen können.
- Überprüfen Sie, ob die Klausur aus **?** durchnummerierten **Vorder- und Rückseiten** besteht. Beachten Sie, dass die Aufgabenstellungen sowohl auf den Vorder- als auch auf den Rückseiten stehen.
- Sie dürfen nur **dokumentenechte Stifte** in den Farben blau/schwarz zum Ausfüllen verwenden. Insbesondere ist die Nutzung von Tintenlöschern und Tipp-Ex untersagt.
- **Zugelassene Hilfsmittel:**
  1 Blatt DIN A4 mit handschriftlichen Notizen (beidseitig).
  Das Mitbringen nicht zugelassener Hilfsmittel stellt eine Täuschung dar und führt zum Nichtbestehen der Klausur. **Schalten Sie daher bitte alle elektronischen Geräte, insbesondere Handys und Smartwatches, vor Beginn der Klausur aus.**
- Sie müssen **alle Klausurunterlagen** (v.a. die Klausur, Zusatzpapier, DIN A4-Blatt mit handschriftlichen Notizen) nach der Bearbeitungszeit abgeben.
- Sollte der Platz unter einer Aufgabe nicht ausreichen, **nutzen Sie bitte die Zusatzblätter am Ende**. Weitere Zusatzblätter sind auf Nachfrage erhältlich. Vermerken Sie auf lose Zusatzblätter unbedingt Ihre Klausurnummer!
- Wenn sich Ihre Lösung zu einer Aufgabe teilweise oder ganz auf Zusatzblättern befindet, vermerken Sie dies entsprechend bei der Aufgabe und auf dem Zusatzblatt.
- Entscheiden Sie sich immer für **eine** Lösung. Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies explizit verlangt.
- In allen Multiple-Choice-Fragen sind genau **zwei** von **fünf** möglichen 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.
- Die Klausur ist mit Sicherheit bestanden, wenn mindestens **50%** der Höchstpunktzahl erreicht werden (ohne Berücksichtigung der Bonifikation aus den Übungen).

  Die Bearbeitungszeit beträgt **180 Minuten**.

Viel Erfolg! ♧

1 0 0 1 2 2 2 2 0 0 0 0 0

Aufgabe 1: Multiple Choice — $2 + 2 + 2 = 6$ Punkt(e)

a) Betrachten Sie eine *deterministische* Turingmaschine T mit Zustandsmenge $Q = \{q_0, q_1, q_2\}$, Eingabealphabet $\Sigma = \{0, 1\}$, Startzustand q_0, Arbeitsalphabet $\Gamma = \{0, 1, \texttt{B}\}$ und Menge $F = \{q_2\}$ aller akzeptierenden Zustände. Für $q \in Q$ ist die Zustandsüberführungsfunktion δ gegeben durch

$$\begin{aligned}
\delta(q_0, 0) &= (q_1, 0, \texttt{rechts}) & \delta(q_0, 1) &= (q_0, 1, \texttt{rechts}) \\
\delta(q_1, 0) &= (q_1, 0, \texttt{rechts}) & \delta(q_1, 1) &= (q_2, 1, \texttt{rechts}) \\
\delta(q_2, 0) &= (q_1, 0, \texttt{rechts}) & \delta(q_2, 1) &= (q_0, 1, \texttt{rechts}) \\
\delta(q\ , \texttt{B}) &= \perp
\end{aligned}$$

Kreuzen Sie die beiden zutreffenden Antwortmöglichkeiten an.

- ☐ T akzeptiert alle Worte, die auf '01' enden.
- ☐ T akzeptiert alle Worte, die '01' beinhalten.
- ☐ T akzeptiert alle Worte, deren vorletztes Symbol '0' ist.
- ☐ T hält nicht auf dem leeren Wort ϵ.
- ☐ T hält auf jeder Eingabe.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

b) Sei Y ein beliebiges $\mathcal{NP}$-vollständiges Problem.

Kreuzen Sie die beiden zutreffenden Antwortmöglichkeiten an.

- ☐ Um zu zeigen, dass ein Problem X $\mathcal{NP}$-vollständig ist, genügt es zu zeigen, dass $X \in \mathcal{NP}$ und $X \leq_\text{p} Y$.
- ☐ Wenn ein Problem $X \in \mathcal{P}$ mit $Y \leq_\text{p} X$, dann gilt $\mathcal{P} = \mathcal{NP}$.
- ☐ Für Y kann eine Lösung in polynomieller Zeit mittels eines nichtdeterministischen Algorithmus für `KNF-SAT` berechnet werden.
- ☐ Es gibt ein $\mathcal{NP}$-vollständiges Problem X, so dass $|X|$ endlich ist.
- ☐ Für jedes $\mathcal{NP}$-harte Problem X gilt $X \leq_\text{p}$ `KNF-SAT`.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

c) Sei M eine *nichtdeterministische* Turingmaschine und w ein Eingabewort.

Kreuzen Sie die beiden zutreffenden Antwortmöglichkeiten an.

- ☐ M akzeptiert w nicht, wenn es mindestens eine Berechnung von M auf w gibt, die nicht in einem akzeptierenden Zustand endet.
- ☐ Es gibt eine deterministische Turingmaschine M', so dass w von M akzeptiert wird genau dann wenn w von M' akzeptiert wird.
- ☐ Die Laufzeit von M auf w ist gegeben durch die Anzahl der Schritte in einer längsten akzeptierenden Berechnung von M auf w.
- ☐ Die Laufzeit von M auf w ist gegeben durch die erwartete Anzahl der Schritte in einer uniform zufälligen akzeptierenden Berechnung von M auf w.
- ☐ Die Laufzeit von M auf w ist gegeben durch die Anzahl der Schritte in einer kürzesten akzeptierenden Berechnung von M auf w.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

a) Sei n eine gerade Zahl und `A[1...n]` ein Array mit

$$\texttt{A}[i] = \begin{cases} i-1 & \text{falls } i \text{ gerade,} \\ i+1 & \text{sonst.} \end{cases}$$

Geben Sie jeweils asymptotisch exakt in Abhängigkeit von n an, welche Laufzeit der angegebene Sortieralgorithmus auf dem Array `A` hat.

Bubble Sort: $\Theta\Big($_____$\Big)$

Mergesort: $\Theta\Big($_____$\Big)$

Radix Sort mit Basis $b = 2$: $\Theta\Big($_____$\Big)$

Quicksort mit `pivot(links,rechts)=rechts`:

$\Theta\Big($_____$\Big)$

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Gegeben sei das Array `A[1...7]` mit

`A` = 2 , 3 , 9 , 7 , 6 , 5 , 1

Nehmen Sie an, `A` wird mittels `partition(p=4, links=1, rechts=7)` partitioniert. Geben Sie das resultierende Array `A` an.

Array `A`: _____

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

c) Sei n ein Vielfaches von 3 und `A[1...n]` ein Array von natürlichen Zahlen. Unter welchen Bedingungen hat Bubble-Sort immer eine Laufzeit von $\Omega(n^2)$?

Kreuzen Sie die beiden zutreffenden Antwortmöglichkeiten an.

☐ Das gröSSte Element ist $\texttt{A}[1]$.

☐ Kein Element befindet sich an der korrekten Stelle.

☐ Das kleinste Element ist unter den Elementen $\texttt{A}[\frac{2n}{3}+1], \texttt{A}[\frac{2n}{3}+2], \ldots, \texttt{A}[n]$.

☐ Das gröSSte Element ist unter den Elementen $\texttt{A}[1], \texttt{A}[2], \ldots, \texttt{A}[n/3]$.

☐ Es gilt $\texttt{A}[i] > \texttt{A}[i+1] > \cdots > \texttt{A}[i+n/3]$ für ein $1 \leq i \leq \frac{2n}{3}$.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

d) Betrachten Sie folgenden Sortieralgorithmus für ein Array `A[1...n]`.

```
i := 1;
while i ≤ n do
    if i = 1 oder A[i] ≥ A[i − 1] then
        i := i + 1;
    else
        h := A[i];
        A[i] := A[i − 1];
        A[i − 1] := h;
        i := i − 1;
```

i) **(2 Punkte)** Geben Sie die best- sowie worst-case Laufzeit des Algorithmus asymptotisch exakt in Abhängigkeit von n an.

Best-case: $\Theta\Big($_____$\Big)$

Worst-case: $\Theta\Big($_____$\Big)$

iii) **(2 Punkte)** Geben Sie eine Instanz `A[1...n]` an, auf der der Algorithmus asymptotisch seine worst-case Laufzeit erreicht.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Aufgabe 3: GroSSe Menge mit kleiner Differenz — 7 Punkt(e)

Gegeben sei ein Array `A[1...n]` mit paarweise verschiedenen ganzen Zahlen und eine natürliche Zahl $r > 0$. Gesucht ist eine möglichst groSSe Menge M von Zahlen aus `A`, so dass die Differenz zwischen gröSSter und kleinster Zahl in M höchstens r ist.

Beschreiben Sie einen Algorithmus, welcher eine solche Menge M berechnet. Die asymptotische worst-case Laufzeit Ihres Algorithmus soll $\mathcal{O}(n \log n)$ nicht überschreiten. Sie dürfen alle aus der Vorlesung bekannten Verfahren und deren Eigenschaften ohne weitere Erläuterung verwenden.

Begründen Sie auch, warum Ihr Algorithmus korrekt arbeitet und die Laufzeitschranke einhält.

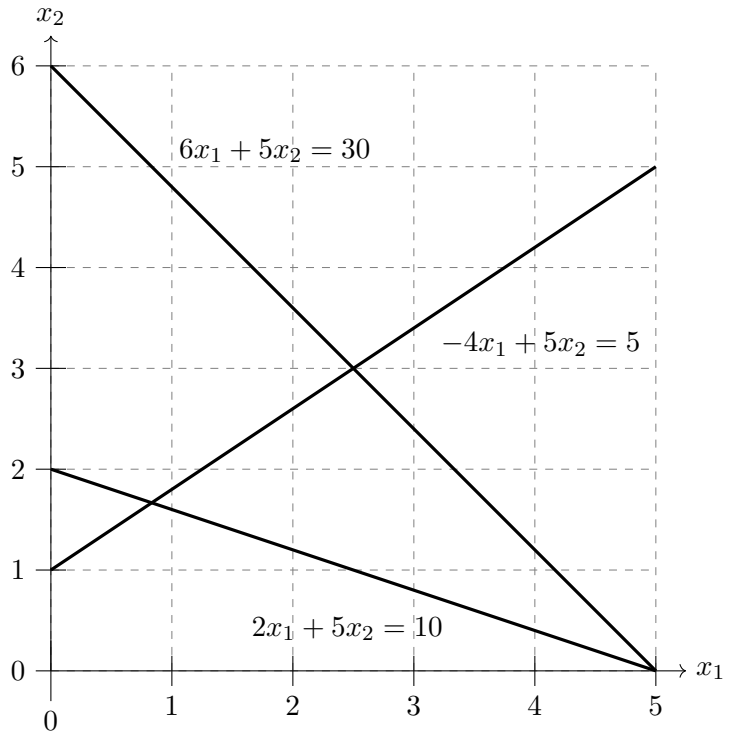
⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

a) Betrachten Sie das folgende Lineare Programm:

$$\begin{array}{lrcl} \text{Max.} & 7x_1 + 3x_2 & & \\ \text{s.d.} & 6x_1 + 5x_2 & \leq & 30 \\ & 2x_1 + 5x_2 & \geq & 10 \\ & -4x_1 + 5x_2 & \geq & 5 \\ & x_1, x_2 & \geq & 0 \end{array}$$

i) **(2 Punkte)** Schraffieren Sie in der nachfolgenden Abbildung das Lösungspolytop.

ii) **(1 Punkt)** Markieren Sie in der Abbildung eine optimale *ganzzahlige* Lösung.



⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Gegeben sei das folgende Lineare Programm:

$$\begin{array}{rrcl} \text{Max.} & 4x_1 + x_2 - 2x_3 & & \\ \text{s.d.} & 3x_1 + 2x_2 + x_3 & \leq & 12 \\ & -x_1 \quad - 4x_3 & \leq & -3 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

Geben Sie das entsprechende duale Lineare Programm an. Gehen Sie dabei nach dem Rezept aus der Vorlesung vor.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Betrachten Sie ein allgemeines Lineares Programm (P)

$$\begin{array}{rrcl} \text{Max.} & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{s.d.} & \boldsymbol{A}\boldsymbol{x} & \leq & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & \boldsymbol{0}\,, \end{array} \qquad \text{(P)}$$

wobei alle Einträge in den Vektoren $\boldsymbol{c}$ und $\boldsymbol{b}$ nicht negativ sind.

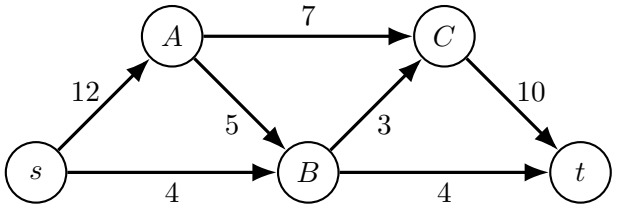
Es sei $\tilde{\boldsymbol{x}} = (\tilde{x}_1, ..., \tilde{x}_n)^T$ eine zulässige Lösung für (P) und $\tilde{\boldsymbol{y}} = (\tilde{y}_1, ..., \tilde{y}_m)$ eine zulässige Lösung für das zu (P) duale Lineare Programm.

Dabei gelte für alle $i = 1, ..., n$ entweder $\tilde{x}_i = 0$ oder $(\tilde{y}^T A)_i = c_i$. Analog gelte für alle $j = 1, ..., m$ entweder $\tilde{y}_j = 0$ oder $(A\tilde{x})_j \geq \frac{1}{\alpha} b_j$ für ein $\alpha > 1$.

Zeigen Sie, dass $\tilde{\boldsymbol{x}}$ eine α-approximative Lösung für (P) ist.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

a) Betrachten Sie das folgende Flussnetzwerk $G = (V, E, c, s, t)$ mit $V = \{A, B, C, s, t\}$:



Die Kantenbeschriftungen geben für jede Kante $e \in E$ ihre Kapazität $c(e)$ an.

Berechnen Sie mittels des Edmonds-Karp Algorithmus einen maximalen Fluss in G.

Tragen Sie für jede Runde i den gewählten augmentierenden Pfad zusammen mit dem Wert f_i des gesamten s-t-Flusses am Ende der Runde in die nachfolgende Tabelle ein (eine Runde entspricht einer Zeile).

Runde i	Augmentierender Pfad p_i	Flusswert f_i
1		

⇑⇑⇑ _____ / 9 Punkt(e) ⇑⇑⇑

b) Sei $G = (V, E, c, s, t)$ ein Flussnetzwerk mit nicht-negativen, ganzzahligen Kapazitäten.

Geben Sie für die folgenden Aussagen jeweils an, ob diese wahr oder falsch sind. Beweisen Sie Ihre Antwort.

i) **(2 Punkte)** Falls alle Kapazitäten *gerade* sind, gibt es einen maximalen Fluss f^*, so dass $f^*(e)$ für alle Kanten $e \in E$ *gerade* ist.

ii) **(2 Punkte)** Falls alle Kapazitäten *ungerade* sind, gibt es einen maximalen Fluss f^*, so dass $f^*(e)$ für alle Kanten $e \in E$ *ungerade* ist.

iii) **(2 Punkte)** Sei $\lambda > 0$ eine beliebige Konstante und c' mit $c'(e) = \lambda \cdot c(e)$ für alle $e \in E$ eine neue Kapazitätsfunktion. Jeder minimale s-t-Schnitt für Kapazitäten c ist auch minimal für Kapazitäten c'.

iv) **(2 Punkte)** Sei $\lambda > 0$ eine beliebige Konstante und c' mit $c'(e) = c(e) + \lambda$ für alle $e \in E$ eine neue Kapazitätsfunktion. Jeder minimale s-t-Schnitt für Kapazitäten c ist auch minimal für Kapazitäten c'.

⇑⇑⇑ _____ / 8 Punkt(e) ⇑⇑⇑

Aufgabe 6: P und NP — $2 + 2 + 2 + 4 = 10$ Punkt(e)

Seien $\mathcal{NPC}$ und $\mathcal{NPH}$ die Klassen aller $\mathcal{NP}$-vollständigen bzw. aller $\mathcal{NP}$-harten Probleme.

a) Es seien L_1 und L_2 zwei Entscheidungsprobleme.

Kreuzen Sie die beiden zutreffenden Antwortmöglichkeiten an.

- ☐ Falls $\mathcal{P} \neq \mathcal{NP}$ und $L_1, L_2 \in \mathcal{P}$, dann gilt $L_1 \cap L_2 \in \mathcal{P}$.
- ☐ Falls $\mathcal{P} \neq \mathcal{NP}$ und $L_1, L_2 \in \mathcal{NP}$, dann gilt *nicht* $L_1 \cap L_2 \in \mathcal{NP}$.
- ☐ Falls $\mathcal{P} = \mathcal{NP}$ und $L_1 \in \mathcal{NPH}$, dann gilt $L_1 \in \mathcal{P}$.
- ☐ Falls $\mathcal{P} \neq \mathcal{NP}$ und $L_1 \in \mathcal{NPH}$, dann gilt $L_1 \in \mathcal{NPC}$.
- ☐ Falls $\mathcal{P} = \mathcal{NP}$ kann gleichzeitig $L_1 \in \mathcal{NPH}$ und $L_1 \in \mathcal{P}$ gelten.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

b) Betrachten Sie die Sprache $L = \{a^n b^n \mid n \geq 0\}$. Es wird der folgende Versuch unternommen, die Aussage $\mathcal{P} = \mathcal{NP}$ zu zeigen:

Die Reduktion `Clique` $\leq_{\mathrm{p}}$ L *ist gegeben durch eine transformierende Turingmaschine* M *mit*

$$M\left((G,k)\right) = \begin{cases} aabb & \textit{falls } G \textit{ eine Clique der GröSSe } k \textit{ besitzt,} \\ aab & \textit{sonst.} \end{cases}$$

Weil `Clique` $\mathcal{NP}$*-vollständig ist und* `Clique` $\leq_p L$ *sowie* $L \in \mathcal{P}$ *gilt, folgt* $\mathcal{P} = \mathcal{NP}$. □

Begründen Sie, warum die Aussage dadurch nicht bewiesen ist.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

c) Erläutern Sie, warum $\mathcal{P} \subseteq \mathcal{NP}$ gilt.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

d) Zeigen Sie, dass $\mathcal{P} \cap \mathcal{NPC} = \emptyset$, falls $\mathcal{P} \neq \mathcal{NP}$.

Hinweis: Sie dürfen die Aussage in c) hier auch ohne Beweis verwenden.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Aufgabe 7: NP-Vollständigkeit — $1 + 3 + 7 = 11$ Punkt(e)

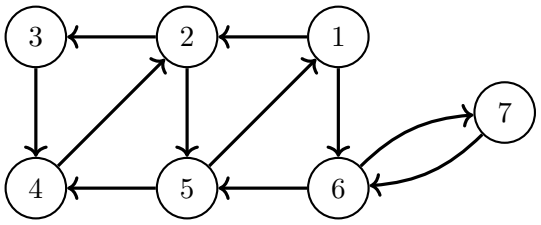
In einem *gerichteten* Graph $G = (V, E)$ heiSSt eine Knotenmenge $W \subseteq V$ *kreiseliminierend*, wenn das Entfernen der Knoten in W mit allen inzidenten Kanten den Graph kreisfrei macht.

Im Problem `Kreiseliminierung` ist ein *gerichteter* Graph $G = (V, E)$ und ein Parameter k gegeben. Es soll entschieden werden, ob es eine kreiseliminierende Menge $W \subseteq V$ der GröSSe $|W| = k$ gibt.

Formal ist die Sprache `Kreiseliminierung` wie folgt definiert:

$$\texttt{Kreiseliminierung} := \big\{(G, k) \mid G \text{ ist gerichteter Graph und es gibt kreiseliminierende Knotenmenge der GröSSe } k\big\}$$

a) Betrachten Sie folgenden Graphen:



Geben Sie eine möglichst kleine kreiseliminierende Knotenmenge W an.

$W = \Big\{$ _____ $\Big\}$

⇑⇑⇑ _____ / 1 Punkt(e) ⇑⇑⇑

b) Beschreiben Sie ein konkretes Verfahren, um zu zeigen, dass `Kreiseliminierung` $\in \mathcal{NP}$.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Zeigen Sie, dass `Kreiseliminierung` $\mathcal{NP}$-hart ist.

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

a) Die Sprache L_1 sei wie folgt definiert:

$$L_1 := \{\langle M\rangle \mid M \text{ ist eine Turingmaschine und } |\mathrm{L(M)}|{=}2\}$$

Zeigen Sie, dass die Sprache L_1 unentscheidbar ist.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Sei $|w|$ die Anzahl der Zeichen eines Wortes w. Eine Turingmaschine M heiSSt *Kürzer*, wenn sie für jedes Eingabewort w, auf dem sie hält, als Ausgabe eine Bitfolge w' mit $|w'| < |w|$ schreibt.

Zeigen Sie mit dem Satz von Rice, dass die folgende Sprache L_2 unentscheidbar ist:

$$L_2 := \{\langle M\rangle \mid M \text{ ist ein Kürzer}\}$$

Begründen Sie insbesondere die nicht-Trivialität der Menge S.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

c) Die Sprache L_3 sei wie folgt definiert:

$$L_3 := \{\langle M_1 \rangle \# \langle M_2 \rangle \mid M_1 \text{ und } M_2 \text{ sind Turingmaschinen und } L(M_1) \cap L(M_2) \neq \emptyset\}$$

Ist die Sprache L_3 rekursiv aufzählbar? Beweisen Sie Ihre Antwort.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Aufgabe 9: Approximationsalgorithmen — $3 + 5 = 8$ Punkt(e)

Gegeben sei ein ungerichteter Graph $G = (V, E)$. Ein *2-Labeling* von G ist eine Funktion $L : V \mapsto \{Blau, Rot\}$, die jedem Knoten eines der Label *Blau* oder *Rot* zuordnet. Eine Kante $\{u, v\} \in E$ heiSSt *gemischt* unter L, falls $L(u) \neq L(v)$, die Label der beiden Endknoten also verschieden sind. Im Problem `MaxMixedEdges` soll ein 2-Labeling gefunden werden, welches die Anzahl der gemischten Kanten maximiert.

Betrachten Sie folgenden Algorithmus für das Problem:

Setze das Label aller Knoten auf *Blau*;
repeat
  $changed := false$;
  if *es gibt einen Knoten $v \in V$, so dass die Änderung des Labels von v die Anzahl gemischter Kanten strikt erhöht* **then**
    Ändere das Label von v;
    $changed := true$;
until $changed = false$;

a) Begründen Sie, warum der Algorithmus polynomielle Laufzeit hat.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Zeigen Sie, dass der Algorithmus stets eine 2-approximative Lösung ausgibt.

⇑⇑⇑ _____ / 5 Punkt(e) ⇑⇑⇑

Aufgabe 10: Umsortieren — $2 + 7 = 9$ Punkt(e)

Sei $\boldsymbol{P}$ eine $n \times n$-Matrix, mit Einträgen P_{ij}, wobei i die Zeile und j die Spalte bezeichnet. Jeder Eintrag P_{ij} ist entweder 0 oder 1. Die Einträge in den Positionen P_{ii} heiSSen *Diagonaleinträge*.

Je zwei Zeilen i und i' in $\boldsymbol{P}$ können *vertauscht* werden, d.h., für alle $k = 1, ..., n$ werden die Einträge P_{ik} und $P_{i'k}$ vertauscht. Die Matrix $\boldsymbol{P}$ ist *kommutativ*, wenn die Zeilen in $\boldsymbol{P}$ so vertauscht werden können (in einer beliebigen Reihenfolge), dass $P_{ii} = 1$ für alle Diagonaleinträge in $\boldsymbol{P}$ gilt.

a) Welche der folgenden 4×4-Matrizen sind *nicht* kommutativ?

Kreuzen Sie die beiden zutreffenden Antwortmöglichkeiten an.

☐ $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$ ☐ $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$ ☐ $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

☐ $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$ ☐ $\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

b) Beschreiben Sie, wie mittels eines Flussnetzwerks in polynomieller Zeit bestimmt werden kann, ob eine gegebene Matrix $\boldsymbol{P}$ kommutativ ist oder nicht. Beweisen Sie auSSerdem die Korrektheit Ihres Verfahrens.

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑