Algorithmen und Datenstrukturen 2

WiSe 2022/2023 · Prof. Dr. M. Hoefer
21. Februar 2023 (Erstklausur)
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.
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung. Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies explizit verlangt.
- 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!

Lösungsskizze

Aufgabe 1: Kurzantworten — $2 + 3 + 2 = 7$ Punkt(e)

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

$$\delta(q_0, \mathtt{B}) = \bot \quad \delta(q_0, x) = (q_1, x, \mathtt{rechts})$$
$$\delta(q_2, \mathtt{B}) = \bot \quad \delta(q_1, x) = (q_1, x, \mathtt{rechts})$$
$$\delta(q_2, x) = \bot \quad \delta(q_1, \mathtt{B}) = (q_2, \mathtt{B}, \mathtt{links})$$

i) **(1 Punkt)** Welche der folgenden Eingaben wird von T *nicht* akzeptiert?

☐ 00110 ☐ 1
☐ ϵ ✓ ☐ 11111

ii) **(1 Punkt)** Die Eingabe $w = w_1 \cdots w_n$ sei in den Zellen $1, ..., n$ von T gespeichert. Über welcher Zelle befindet sich der Lesekopf von T am Ende der Berechnung?

Zelle: n

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

b) Betrachten Sie das Lastverteilungsproblem mit $m = 2$ Prozessoren und $n = 5$ Aufgaben mit Rechenzeiten

$$t_1 = 2, \quad t_2 = 4, \quad t_3 = 5, \quad t_4 = 6, \quad t_5 = 7 \, .$$

i) **(1 Punkt)** Geben Sie eine Aufteilung der Aufgaben auf die Prozessoren an, so dass der resultierende Makespan *minimal* ist.

Aufgaben für Prozessor 1: $1, 2, 4$

Aufgaben für Prozessor 2: $3, 5$

ii) **(1 Punkt)** Geben Sie den resultierenden Makespan an, wenn die Aufgaben mit der *On-line* Heuristik aus der Vorlesung zugeteilt werden.

Makespan: 14

iii) **(1 Punkt)** Geben Sie den resultierenden Makespan an, wenn die Aufgaben mit der *Off-line* Heuristik aus der Vorlesung zugeteilt werden.

Makespan: 13

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Sei A ein $\mathcal{NP}$-vollständiges Entscheidungsproblem.

Welche der folgenden Aussagen ist korrekt?

Lösungsskizze

1 004222 100004

☐ Es gilt $L \leq_p A$ für alle $\mathcal{NP}$-harten Probleme L.

☐ Für alle Sprachen $L \in \mathcal{P}$ gilt $L \leq_p A$. ✓

☐ Falls $A \leq_p B$ für ein Problem B, dann ist B ebenfalls $\mathcal{NP}$-vollständig.

☐ Wenn $A \subseteq B$ für eine Sprache $B \in \mathcal{P}$, dann gilt auch $A \in \mathcal{P}$.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Sei `A[1...n]` ein Array mit `A[`i`]` $= i \cdot (n - i)$ für $i \in \{1, \ldots, n\}$.

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

Bubble Sort: $\Theta($ n^2 $)$

Selection Sort: $\Theta($ n^2 $)$

Mergesort: $\Theta($ $n \log n$ $)$

Radix Sort mit Basis $b = n$: $\Theta($ n $)$

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Gegeben sei das Array

`A = 2 , 1 , 4 , 3 , 6 , 5 , 8 , 7`

i) **(2 Punkte)** Nehmen Sie an, `A` wird mit *Bubble Sort* sortiert. Geben Sie das Array `A` nach Ende der ersten Phase an.

Array `A`: $1, 2, 3, 4, 5, 6, 7, 8$

ii) **(2 Punkte)** Nehmen Sie an, `A` wird mit *Mergesort* sortiert. Geben Sie an, wie oft die Funktion `merge(...)` aufgerufen wird.

Anzahl Aufrufe: 7

iii) **(2 Punkte)** Nehmen Sie an, `A` wird mit *Quicksort* mit `pivot(links,rechts)=rechts` sortiert. Geben Sie an, wie oft die Funktion `partition(...)` aufgerufen wird.

Anzahl Aufrufe: 4

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

Lösungsskizze

1 0006222 100008

c) Ist die folgende Aussage korrekt?

Für jedes vergleichsorientierte Sortierverfahren gibt es Eingaben der Länge 5, so dass das Verfahren mindestens 7 Vergleiche benötigt, um die Eingabe zu sortieren.

Begründen Sie Ihre Antwort.

Lösungsskizze: Die Aussage ist richtig. Jedes vergleichsbasierte Sortierverfahren hat für jede Eingabelänge einen Sortierbaum, der mindestens so viele Blätter haben muss, wie es Ordnungstypen für diese Eingabelänge gibt. Für Eingaben der Länge 5 gibt es $5! = 120$ Ordnungstypen. Haben im Baum alle Blätter maximal Tiefe t, gibt es maximal 2^t Blätter. Somit muss es ein Blatt der Tiefe mindestens 7 geben, denn andernfalls gäbe es maximal $2^6 = 64 < 120$ Blätter. Da jede Kante nach unten im Baum einem Vergleich entspricht, muss es mindestens 7 Vergleiche geben.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Lösungsskizze

Aufgabe 3: Sortieren II — 7 Punkt(e)

Gegeben seien zwei Arrays `A[1...n]` und `B[1...n]` von natürlichen Zahlen. Die Zahlen in `A` sind aus der Menge $\{1, 2, \ldots, n^3\}$, die Zahlen in `B` sind aus der Menge $\{2^i \mid i \in \{0, 1, \ldots, n\}\}$.

Beschreiben Sie ein Verfahren, welches ein aufsteigend sortiertes Array `C[1...2n]` der Zahlen aus `A` und `B` berechnet. Die asymptotische worst-case Laufzeit soll $\mathcal{O}(n)$ nicht überschreiten. Sie dürfen alle aus der Vorlesung bekannten Verfahren und deren Eigenschaften ohne weitere Erläuterung verwenden.

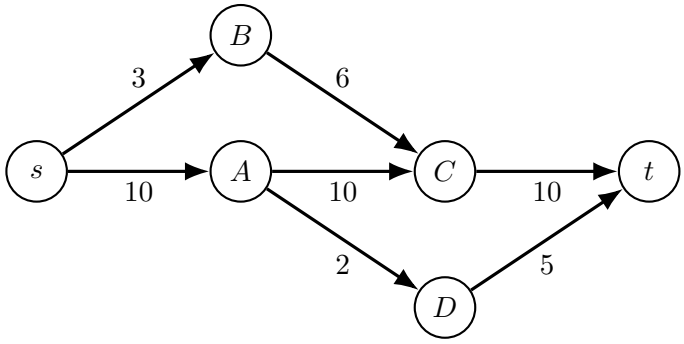
Begründen Sie auch, warum Ihr Verfahren die Laufzeitschranke einhält.

Lösungsskizze: Zuerst wird das Array `A` mittels Radixsort in Zeit $\mathcal{O}(n)$ sortiert. Anschließend wird $\log_2(\cdot)$ auf jedes Element von `B` angewendet, dies hat ebenfalls Laufzeit $\mathcal{O}(n)$. Die Elemente in `B` sind nun aus $\{0, \ldots, n\}$ und können mittels Distribution Counting in Zeit $\mathcal{O}(n)$ sortiert werden. Nach dem Sortieren, wird Array `B` mittels $2^{(\cdot)}$ rücktransformiert. Nachdem beide Arrays `A` und `B` sortiert sind, können sie mittels Merge zu einem sortierten Array `C` zusammengeführt werden. Dies hat ebenfalls Laufzeit $\mathcal{O}(n)$.

⇑⇑⇑ _____ / 7 Punkt(e) ⇑⇑⇑

Lösungsskizze

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



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

a) Berechnen Sie mittels des Ford-Fulkerson Algorithmus einen maximalen Fluss in G.

Wählen Sie dafür in jeder Runde i des Algorithmus einen augmentierenden Pfad p_i und erhöhen Sie den Fluss entlang dieses Pfades so viel wie möglich. Tragen Sie die von Ihnen gewählten augmentierenden Pfade zusammen mit dem Wert f_i des s-t-Flusses am Ende jeder Runde in die nachfolgende Tabelle ein (eine Runde entspricht einer Zeile).

Runde i	Augmentierender Pfad p_i	Flusswert f_i
1	$p_1 = (s, A, C, t)$	10
2	$p_2 = (s, B, C, A, D, t)$	12

Lösungsskizze: Es gibt zwei weitere mögliche Lösungen:

- $p_1 = (s, B, C, t)$, $f_1 = 3$
  $p_2 = (s, A, C, t)$, $f_2 = 10$
  $p_3 = (s, A, D, t)$, $f_3 = 12$
- $p_1 = (s, B, C, t)$, $f_1 = 3$
  $p_2 = (s, A, D, t)$, $f_2 = 5$
  $p_3 = (s, A, C, t)$, $f_3 = 12$

⇑⇑⇑ _____ / 9 Punkt(e) ⇑⇑⇑

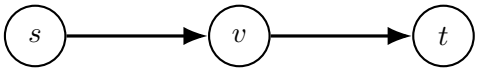
Lösungsskizze

1 0009222 100009

b) Zeichnen Sie ein Flussnetzwerk H, für das der minimale s-t-Schnitt nicht eindeutig ist.

Geben Sie auSSerdem zwei verschiedene minimale s-t-Schnitte in H an.

Lösungsskizze: Mögliche Antwort:



mit minimalen s-t-Schnitten (S_1, T_1) und (S_2, T_2), wobei

$$S_1 = \{s, v\}, \quad T_1 = \{t\}$$

$$S_2 = \{s\}, \quad T_1 = \{v, t\}$$

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Sei nun $G = (V, E, c, s, t)$ ein beliebiges Flussnetzwerk.

Beweisen Sie die folgende Aussage:

Wenn $(S_1, V \setminus S_1)$ und $(S_2, V \setminus S_2)$ zwei minimale s-t-Schnitte sind, dann ist $(S, V \setminus S)$ mit $S = S_1 \cap S_2$ ebenfalls ein minimaler s-t-Schnitt.

Hinweis: Max-Flow-Min-Cut Theorem.

Lösungsskizze: Sei f^* ein maximaler s-t-Fluss. Nach dem Max-Flow-Min-Cut Theorem gilt $c(S_1, V \setminus S_1) = c(S_2, V \setminus S_2) = |f^*|$ und jede Schnittkante $(u, v) \in S_i \times V \setminus S_i$ ist maximal ausgelastet, also $f^*((u, v)) = c((u, v))$. Da jede Schnittkante $(u, v) \in S \times V \setminus S$ auch Schnittkante für mindestens einen der beiden Schnitte S_i ist, ist jede solche Kante ebenfalls voll ausgelastet. Für den Wert des Schnittes S folgt somit $c(S, V \setminus S) = c(S_i, V \setminus S_i)$.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Lösungsskizze

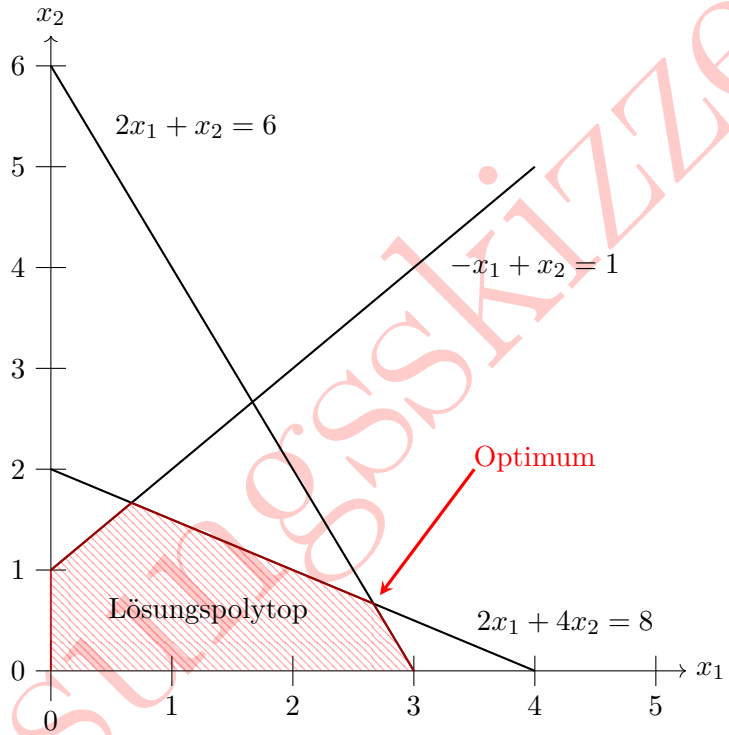
1 0 1 0 2 2 2 1 0 0 0 0 5

a) Betrachten Sie das folgende lineare Programm:

$$\begin{array}{lrcl} \text{Max.} & 12x_1 + 13x_2 & & \\ \text{s.d.} & 2x_1 + 4x_2 & \leq & 8 \\ & 2x_1 + x_2 & \leq & 6 \\ & -x_1 + x_2 & \leq & 1 \\ & x_1, x_2 & \geq & 0 \end{array}$$

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

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



Lösungsskizze

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Gegeben sei das folgende lineare Programm (P):

$$\begin{array}{lrcl} \text{Max.} & 7x_1 - 14x_2 & & \\ \text{s.d.} & x_1 - 4x_2 - 2x_3 & \leq & 1 \\ & 2x_1 + 2x_2 + 3x_3 & \leq & 9 \\ & x_1, x_2, x_3 & \geq & 0 \end{array} \tag{P}$$

i) **(3 Punkte)** Bestimmen Sie das zu (P) duale lineare Programm (D). Gehen Sie dabei nach dem Rezept aus der Vorlesung vor.

Lösungsskizze: Bestimme die Vektoren $\boldsymbol{c}$ und $\boldsymbol{b}$ sowie die Matrix $\boldsymbol{A}$ aus (P):

$$\boldsymbol{c}^T = (7, -14, 0), \qquad \boldsymbol{b}^T = (1, 9), \qquad \boldsymbol{A} = \begin{pmatrix} 1 & -4 & -2 \\ 2 & 2 & 3 \end{pmatrix}.$$

Da $\boldsymbol{b}$ zwei Einträge hat, folgt $\boldsymbol{y}^T = (y_1, y_2)$ mit $y_1, y_2 \geq 0$. Das duale LP (D) sieht wie folgt aus:

$$\begin{array}{lrcl} \text{Min.} & y_1 + 9y_2 & & \\ \text{s.d.} & y_1 + 2y_2 & \geq & 7 \\ & -4y_1 + 2y_2 & \geq & -14 \\ & -2y_1 + 3y_2 & \geq & 0 \\ & y_1, y_2 & \geq & 0 \end{array}$$

ii) **(3 Punkte)** Seien $(\boldsymbol{x}^*)^T = (3, 0, 1)$ und $(\boldsymbol{y}^*)^T = (3, 2)$.

Zeigen Sie, dass $\boldsymbol{x}^*$ eine optimale Lösung für (P) und $\boldsymbol{y}^*$ eine optimale Lösung für (D) ist.

Lösungsskizze: Durch Einsetzen von $\boldsymbol{x}^*$ in (P) sowie $\boldsymbol{y}^*$ in (D), erkennt man, das beide Vektoren jeweils *zulässige* Lösungen sind - alle Nebenbedingungen sind tight, bis auf $-4 \cdot 3 + 2 \cdot 2 = -8 \geq -14$ in (D). Aufgrund *Schwacher Dualität* gilt somit $\boldsymbol{c}^T\boldsymbol{x} \leq \boldsymbol{y}^T\boldsymbol{b}$. Wegen

$$\boldsymbol{c}^T\boldsymbol{x} = 7 \cdot 3 - 14 \cdot 0 = 21 \qquad \text{sowie} \qquad \boldsymbol{y}^T\boldsymbol{b} = 1 \cdot 3 + 9 \cdot 2 = 21$$

folgt, dass $\boldsymbol{x}^*$ und $\boldsymbol{y}^*$ optimale Lösungen für (P) bzw. (D) sind.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

Lösungsskizze

Aufgabe 6: P und NP

$4 + 4 + 4 = 12$ Punkt(e)

Die Sprache `Trio` sei wie folgt definiert:

$$\texttt{Trio} := \{G \mid G \text{ ist ein ungerichteter Graph und enthält eine Clique der GröSSe 3}\}$$

a) Zeigen Sie, dass $\texttt{Trio} \in \mathcal{P}$.

Lösungsskizze: Folgender Algorithmus entscheidet `Trio` in Zeit $\mathcal{O}(n^3)$, wenn der Graph als $n \times n$ Adjazenzmatrix A gegeben ist:

```
for(i=1...n){
    for(j=1...n){
        for(k=1...n){
            if(1 = A[i][j] = A[j][k] = A[k][i])
                return JA;
        }
    }
}
return NEIN;
```

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Ist die folgende Aussage korrekt?

Wenn $L \leq_p$ `Trio` für eine Sprache L gilt, dann ist $L \in \mathcal{P}$.

Begründen Sie Ihre Antwort.

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

Lösungsskizze: Die Aussage ist richtig. Es gelte $L \leq_p$ `Trio`. Folgender Algorithmus entscheidet L in polynomieller Zeit: Transformiere die Eingabe w für L in polynomieller Zeit in eine Eingabe $M(w)$ für `Trio`. Löse `Trio` auf Eingabe $M(w)$ in polynomieller Zeit mit dem Algorithmus aus a) und übernehme die Antwort.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

c) Ist die folgende Aussage korrekt?

Wenn $\mathcal{P} \neq \mathcal{NP}$, dann gilt nicht `KNF-SAT` $\leq_p$ `Trio`.

Begründen Sie Ihre Antwort.

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

Lösungsskizze: Die Aussage ist richtig. Angenommen $\mathcal{P} \neq \mathcal{NP}$. Wenn `KNF-SAT` $\leq_p$ `Trio`, dann gibt es einen Algorithmus, der jede Instanz von `KNF-SAT` in polynomieller Zeit entscheidet: Transformiere die KNF-Formel in eine Instanz von `Trio` und nutze den Algo aus a) um das Problem zu lösen. `KNF-SAT` ist jedoch $\mathcal{NP}$-hart, und somit würde $\mathcal{P} = \mathcal{NP}$ folgen (denn $K \leq_p$ `KNF-SAT` für alle $K \in \mathcal{NP}$, da `KNF-SAT` $\mathcal{NP}$-hart ist.)

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Lösungsskizze

Ein Flugzeughersteller hat eine Menge T neuer Flugzeugtypen konstruiert, die im Rahmen eines Testprogramms getestet werden sollen. Der Hersteller verfügt über n Piloten, wobei Pilot $i \in \{1, \ldots, n\}$ die Flugzeugtypen in der Menge $F_i \subseteq T$ fliegen kann. Zur Durchführung des Testprogramms soll ein möglichst kleines Team von Piloten zusammengestellt werden, so dass für jeden Flugzeugtyp mindestens *zwei* Piloten im Team sind, die diesen fliegen können.

Formal wird die Sprache `Flugzeugtest` wie folgt definiert:

$$\texttt{Flugzeugtest} := \big\{(T, F_1, \ldots, F_n, k) \mid \text{es gibt } I \subseteq \{1, \ldots, n\} \text{ mit } |I| = k \text{ und für alle } t \in T \text{ gibt es mindestens zwei } i \in I \text{ mit } t \in F_i \big\}$$

a) Betrachten Sie folgende Instanz des Problems. Die Flugzeugtypen sind:

$$T = \big\{\text{AeroTuring},\ \text{JetFord},\ \text{SkyMonds},\ \text{AirKarp}\big\}$$

Es stehen vier Piloten zur Wahl, die folgende Flugzeugtypen fliegen können:

$$F_1 = \{\text{AeroTuring},\ \text{SkyMonds},\ \text{AirKarp}\},$$
$$F_2 = \{\text{JetFord},\ \text{SkyMonds}\},$$
$$F_3 = \{\text{JetFord},\ \text{AirKarp}\},$$
$$F_4 = \{\text{AeroTuring},\ \text{JetFord},\ \text{AirKarp}\}$$

Geben Sie eine minimale Auswahl $I \subseteq \{1, 2, 3, 4\}$ von Piloten an, so dass für jeden Flugzeugtyp mindestens *zwei* Piloten vorhanden sind.

$I = \{$ 1, 2, 4 $\}$

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

b) Zeigen Sie, dass `Flugzeugtest` $\in \mathcal{NP}$.

Lösungsskizze: Für eine gegebene Pilotenmenge $I \subseteq \{1, \ldots, n\}$ kann wie folgt in polynomieller Zeit überprüft werden, ob jeder Flugzeugtyp mindestens zwei Piloten hat: Durchlaufe die Flugzeugtypen und zähle für jeden Typ $t \in T$, wie viele Piloten $i \in I$ dieses fliegen können. Sind am Ende für jedes Flugzeug mindestens zwei Piloten vorhanden, gebe JA aus, ansonsten NEIN.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

Lösungsskizze

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

Lösungsskizze: Wir zeigen die Reduktion `SetCover` $\leq_p$ `Flugzeugtest`. Da `SC` $\mathcal{NP}$-hart ist, folgt $\mathcal{NP}$-härte von `FT`.

Transformation:
Sei $(A_1, \ldots, A_m, k)$ mit $U = \cup_{i=1}^{m} A_i$ Eingabe für `SC`. Führe die Pilotenmenge $\{1, \ldots, m+1\}$ ein. Pilot $i \in \{1, \ldots, m\}$ kann die Flugzeuge der Menge A_i fliegen. Der „universelle Pilot“ $m+1$ kann die Flugzeuge der Menge U fliegen. Schreibe die transformierte Eingabe $(A_1, \ldots, A_m, U, k+1)$.

$(A_1, \ldots, A_m, k) \in$ `SC` $\Rightarrow (A_1, \ldots, A_m, U, k+1) \in$ `FT`:
Seien $i_1, \ldots, i_k$ Indizes, so dass $U = \cup_{j=1}^{k} A_{i_j}$. Wähle die Pilotenmenge $\{i_1, \ldots, i_k, m+1\}$. Dann ist für jedes Flugzeug mindestens ein Pilot in $\{i_1, \ldots, i_k\}$. Zusammen mit dem „universellen Piloten“ sind also für jedes Flugzeug zwei Piloten vorhanden.

$(A_1, \ldots, A_m, U, k+1) \in$ `FT` $\Rightarrow (A_1, \ldots, A_m, k) \in$ `SC`:
Sei $I \subseteq \{1, \ldots, m+1\}$ eine Menge von $|I| = k+1$ Piloten, so dass jedes Flugzeug mindestens zwei Piloten hat. Wird ein beliebiger Pilot i^* aus I entfernt, verbleibt für jedes Flugzeug mindestens einer. Damit gilt $\cup_{i \in I \setminus \{i^*\}} A_i = U$ und somit $(A_1, \ldots, A_m, k) \in$ `SC`.

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑

Lösungsskizze

a) Die Sprache L_1 sei wie folgt definiert:

$$L_1 := \{\langle M \rangle \mid M \text{ ist eine Turingmaschine und } M \text{ hält } \textit{nur} \text{ auf dem leeren Wort}\}$$

Zeigen Sie, dass die Sprache L_1 unentscheidbar ist.

Lösungsskizze: Wir zeigen die Reduktion $H_\epsilon \leq L_1$.

Transformation:
Sei o.B.d.A. $\langle M \rangle$ für eine TM M Eingabe für H_ϵ. Schreibe die transformierte Eingabe $\langle M' \rangle$, wobei M' eine TM ist, die zunächst überprüft, ob das Band leer ist. Falls Nein, geht M' in eine Endlosschleife. Falls Ja, arbeitet M' wie M.

$\langle M \rangle \in H_\epsilon \Rightarrow \langle M' \rangle \in L_1$
Wenn M auf ϵ hält, dann hält M' auch auf ϵ, denn M' arbeitet auf dem leeren Wort genau wie M.

$\langle M \rangle \in H_\epsilon \Leftarrow \langle M' \rangle \in L_1$

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

b) Betrachten Sie die folgende Sprache:

$$L_2 := \{\langle M \rangle \mid M \text{ ist eine Turingmaschine und der Lesekopf von } M \text{ liest auf Eingabe } \epsilon \text{ keine Zelle auf dem Band mehrfach}\}$$

Ist die Sprache L_2 entscheidbar? Beweisen Sie Ihre Antwort.

Lösungsskizze: Ja. Die Maschine M muss auf dem leeren Wort beim Lesen des ersten `B` den Kopf Richtung $r \in \{\texttt{rechts}, \texttt{links}\}$ bewegen, sonst liest sie die erste Zelle mehrfach. Danach muss sie in jedem Schritt stets Richtung r gehen, bis sie entweder einen Zustand erreicht, in dem sie schon einmal war (Endlosschleife) oder anhält. Dies kann anhand der Überführungsfunktion überprüft werden: Setze $(q, \cdot, r) = \delta(q, \texttt{B})$ beginnend mit $q = q_0$, bis sich ein Zustand wiederholt, oder ein undefinierter Übergang erreicht ist, oder die Kopfrichtung r sich ändert. Da es nur endlich viele Zustände q gibt, terminiert das Verfahren.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

c) Ist die Sprache $\overline{L_2}$ rekursiv aufzählbar? Beweisen Sie Ihre Antwort.

Lösungsskizze: Ja. Da L_2 nach dem Beweis in b) entscheidbar ist, ist $\overline{L_2}$ auch entscheidbar und somit insbesondere auch rekursiv aufzählbar, denn jede TM die eine Sprache entscheidet, akzeptiert insbesondere auch alle JA-Instanzen.

⇑⇑⇑ _____ / 2 Punkt(e) ⇑⇑⇑

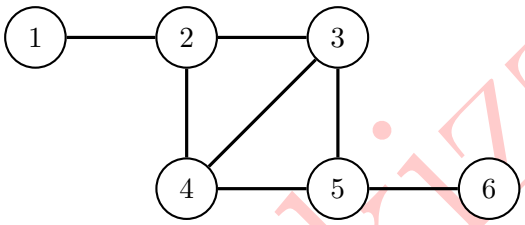
Lösungsskizze

Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit $V = \{1, \ldots, n\}$ und eine natürliche Zahl $k < n$. Eine Kante $e \in E$ wird durch eine Menge $S \subseteq V$ von Knoten überdeckt, wenn $S \cap e \neq \emptyset$. In G soll nun eine Menge $S \subseteq V$ von $|S| = k$ Knoten gefunden werden, so dass möglichst viele Kanten durch S überdeckt werden.

Betrachten Sie den folgenden Approximationsalgorithmus für das Problem:

$S := \emptyset$;
for $i = 1 \ldots k$ **do**
  Wähle einen Knoten $v \in V \setminus S$ mit maximalem Grad. Wenn es mehrere solche gibt, wähle den mit kleinstem Index; `// grad`$(v) = |\{e \in E : v \in e\}|$
  $S := S \cup \{v\}$; `// füge Knoten` v `zu` S `hinzu`
Gebe S aus;

a) Gegeben sei folgende Instanz I des Problems mit $k = 2$:



i) **(1 Punkt)** Geben Sie die vom Algorithmus berechnete Knotenmenge S_{alg} an.

$S_{\text{alg}} = \{$ 2, 3 $\}$

ii) **(1 Punkt)** Geben Sie eine optimale Knotenmenge S_{opt} an.

$S_{\text{opt}} = \{$ 2, 5 $\}$

iii) **(1 Punkt)** Geben Sie den Approximationsfaktor des Algorithmus auf I an.

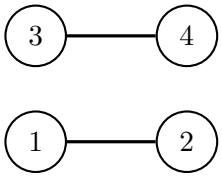
Approximationsfaktor: 6/5

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Lösungsskizze

b) Zeigen Sie, dass der Algorithmus *nicht* δ-approximativ ist für $\delta < 3/2$.

Lösungsskizze:
Betrachte folgenden Graphen mit $k = 2$:



Der Algorithmus wählt die Knoten 1 und 2 und überdeckt somit eine Kante. Optimal ist aber, die Knoten 1 und 3 zu wählen und damit alle beiden Kanten zu überdecken.

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

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

Lösungsskizze: Sei $d_1 \geq \cdots \geq d_n$ die absteigend sortierte Folge von Knotengraden. Die k Knoten in einer optimalen Lösung überdecken maximal $\sum_{i=1}^{k} d_i$ Kanten. Es folgt

$$\text{Opt} \leq \sum_{i=1}^{k} d_i.$$

Der Algorithmus überdeckt mindestens $\sum_{i=1}^{k} d_i/2$ Kanten, denn jede Kante ist maximal doppelt überdeckt. Es folgt

$$\text{Alg} \geq \sum_{i=1}^{k} d_i/2 \geq \text{Opt}/2.$$

⇑⇑⇑ _____ / 3 Punkt(e) ⇑⇑⇑

Lösungsskizze

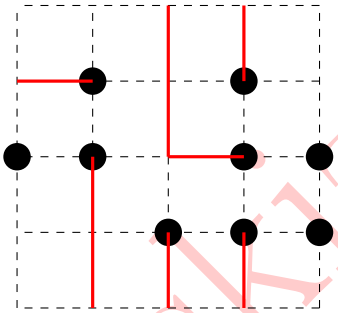
Aufgabe 10: Exklusiver Urlaub — $4 + 6 = 10$ Punkt(e)

Das Gelände der Ferienanlage Schöneruh kann durch einen quadratischen Gittergraphen G_d mit d Knoten entlang jeder Seite modelliert werden. Dabei entsprechen alle Kanten Wegen und alle Knoten Kreuzungen von Wegen. An bestimmten Kreuzungen (nicht unbedingt an allen) stehen insgesamt $k > 0$ Hütten, in denen Gäste untergebracht sind.

Da die Gäste richtig entspannen wollen, möchten sie einander auf keinen Fall begegnen. Im Problem `FreiePfade` sollen daher Pfade von jeder Hütte zum Rand der Anlage bestimmt werden, so dass keine zwei Pfade eine gemeinsame Kreuzung nutzen. Die Pfade dürfen nur entlang der Kanten im Gitter verlaufen.

a) Betrachten Sie die beiden folgenden Eingaben für `FreiePfade` mit $d = 5$ und $k = 9$ Hütten. Die schwarzen Punkte markieren die Positionen $h_1, ..., h_9$ der Hütten. Geben Sie jeweils an, ob es eine Lösung für die Eingabe gibt oder nicht. Falls ja, zeichnen Sie eine entsprechende Lösung ein; falls nicht, begründen Sie dies kurz anhand der Abbildung.

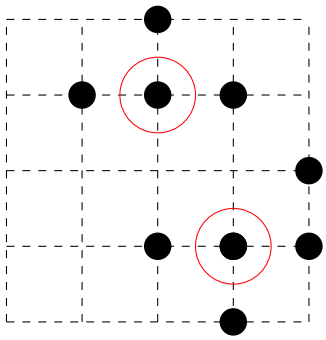
i) **(2 Punkte)**



☐ Ja, es gibt eine Lösung (eingezeichnet). ✓

☐ Nein, es gibt keine Lösung. Begründung:

ii) **(2 Punkte)**



☐ Ja, es gibt eine Lösung (eingezeichnet).

☐ Nein, es gibt keine Lösung. Begründung: ✓

Lösungsskizze: Alle Pfade von den beiden umzirkelten Hütten aus schneiden sich stets.

⇑⇑⇑ _____ / 4 Punkt(e) ⇑⇑⇑

Lösungsskizze

b) Gegeben sei ein Gittergraph G_d zusammen mit den Positionen von k Hütten $h_1, ..., h_k$.

Beschreiben Sie, wie mittels eines Flussnetzwerks in polynomieller Laufzeit bestimmt werden kann, ob eine Lösung für `FreiePfade` existiert oder nicht. Beschreiben Sie dabei das Flussnetzwerk explizit. Begründen Sie auch die Korrektheit Ihrer Idee.

Lösungsskizze: Das Problem `FreiePfade` kann auf das Problem `Max-Flow` aus der Vorlesung *mit knotendisjunkten Pfaden* reduziert werden (offensichtlich gilt hier, dass knotendisjunkte Pfade auch kantendisjunkte Pfade sind). Als solches kann es dann mittels eines der aus der Vorlesung bekannten Verfahren zur Berechnung maximaler Flüsse (Algorithmus von Ford-Fulkerson oder Edmonds-Karp) in polynomieller Laufzeit gelöst werden.

Sei $H = \{h_1, ..., h_k\}$ die Menge aller mit Hütten besetzten Knoten und $R = \{r_1, ..., r_l\}$ die Menge aller Knoten am Rand von G_d.

Das Flussnetzwerk wird wie folgt konstruiert:

1. Wandle G_d in einen gerichteten Graph um, indem jede (ungerichtete) Kante $e = \{u, v\} \in G_d$ durch die zwei direkte Kanten (v, u) und (u, v) ersetzt wird.
2. Füge eine Quelle s hinzu. Verbinde s mit allen Hütten, d.h., füge Kanten (s, h_i) für alle $i = 1, ..., k$ ein.
3. Füge eine Senke t hinzu. Verbinde alle Knoten am Rand von G_d mit t, d.h., füge Kanten (r_j, t) für alle $j = 1, ..., l$ ein.
4. Weise jeder Kante im Netzwerk die Kapazität $c(e) = 1$ zu.

An dieser Stelle würde ein maximaler Fluss mit Wert k (sofern er denn existiert) eine Lösung ausgeben, die *noch nicht* berücksichtigt, dass sich keine zwei Pfade in einem Knoten berühren dürfen. Es muss also sichergestellt werden, dass der Fluss durch jeden Knoten höchstens 1 ist. Dafür wird jeder Knoten v im Netzwerk durch zwei Knoten v_{in} und v_{out} ersetzt und die Kanten von v werden wie folgt angepasst:

- Alle *eingehenden* Kanten von v werden zu eingehenden Kanten von v_{in}.
- Alle *ausgehenden* Kanten von v werden zu ausgehenden Kanten von v_{out}.
- Darüber hinaus wird eine Kante $(v_{\text{in}}, v_{\text{out}})$ eingefügt, ebenfalls mit Kapazität 1.

Da die Gesamtkapazität, die die Senke s verlässt, k beträgt, ist jeder maximale Fluss f^* durch das konstruierte Flussnetzwerk höchstens k. Gilt $|f^*| = k$, so existieren knotendisjunkte (und damit kantendisjunkte) Pfade von den Hütten zu den Knoten am Rand. Ansonsten gibt es von mindestens einer Hütte aus keinen Pfad zum Rand, da die Kapazität der dafür benötigten Kanten bereits ausgeschöpft sind.

Lösungsskizze

⇑⇑⇑ _____ / 6 Punkt(e) ⇑⇑⇑