GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift): ______________________

Vorname (Druckschrift): ______________________

Matrikelnummer: ______________________

Studiengang: ______________________

Bitte Hinweise beachten:

- Schreiben Sie Ihren Namen nur auf dieses Titelblatt.
- Zusatzpapier darf nicht mit abgegeben werden.
- Zugelassenes Hilfsmittel: 1 Blatt DIN A4.
- Nicht zugelassene Hilfsmittel (z. B. Handys, Smartwatches, andere Geräte) stellen eine Täuschung dar und führen zum Nichtbestehen der Klausur. Schalten Sie daher alle elektronischen Geräte aus und verstauen Sie diese in Ihrer Tasche.
- 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 verlangt.
- Die Klausur dauert 180 Minuten und gilt mit 50 % der Höchstpunktzahl als bestanden.
- In allen Multiple-Choice-Fragen sind **genau zwei von fünf Antworten** richtig. Wenn man x richtige Kreuze setzt und y falsche, erhält man entweder 0, 1 oder 2 Punkte:

richtige Kreuze *(correct marks)*	x	**2**	**2**	2	2	**1**	1	1	1	0	0	0	0
falsche Kreuze *(wrong marks)*	y	**0**	**1**	2	3	**0**	1	2	3	0	1	2	3
Punkte *(points)*		**2**	**1**	0	0	**1**	0	0	0	0	0	0	0

Please note:

- *Write your name on this cover sheet only.*
- *Additional paper must not be submitted.*
- *Approved aid: 1 sheet DIN A4.*
- *Non-approved aids (e.g., mobile phones, smartwatches, other devices) constitute deception and lead to failing the exam. Turn your electronic devices off before the exam and store them in your bag.*
- *If two or more solutions are given for a problem, the problem counts as unsolved. So always choose* ***one*** *solution.*
- *Justifications are only necessary if the problem wording requires it.*
- *The exam lasts 180 minutes and is considered passed with 50 % of the maximum score.*
- *In all multiple-choice questions,* ***exactly two out of five answers*** *are correct. If you make x correct and y wrong marks, you will receive 0, 1, or 2 points according to the table above.*

Diese Seite ist für den internen Gebrauch.
Bitte leer lassen.

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

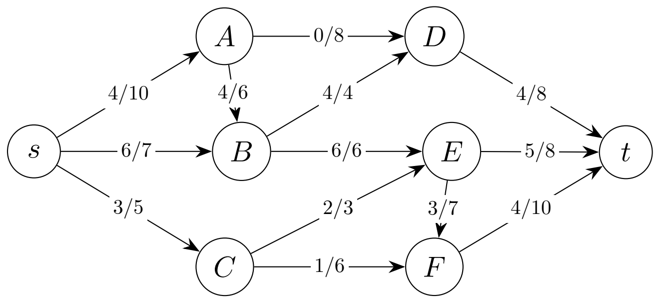
Diese Seite ist für den internen Gebrauch.
Bitte leer lassen.

Aufgabe	1	2	3	4	5
Erreichbar	20	20	20	20	20
Erreicht					

Summe	Note
/100	

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

Consider the s-t-flow f in the following flow network (G, s, t, c):



a) Welchen Wert hat ein **maximaler** Fluss in (G, s, t, c)?

*What is the value of a **maximum** flow in (G, s, t, c)?*

Antwort: 19.

(/ 3 Punkte)

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

*Provide the two sets S, T of a **minimum** cut in (G, s, t, c).*

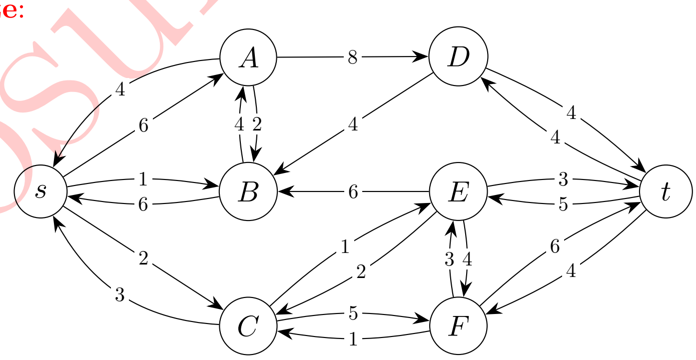
$S = \{s,$ A, B, D $\}$, $T = \{t,$ C, E, F $\}$.

(/ 3 Punkte)

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

*In the following figure, draw the residual graph G_f for the flow network (G, s, t, c) and the flow f. Provide the residual capacities by labeling each edge of G_f. Edges with residual capacity 0 **don't** need to be drawn.*

Lösungsskizze:



(/ 4 Punkte)

d) Die Hipster-Bäckerei *Raum für Brot* hat vier überteuerte Brotsorten im Angebot: Weißbrot (W), Vollkornbrot (V), Roggenbrot (R) und Sauerteigbrot (S). Basierend auf historischen Verkaufsdaten hat die Bäckerei folgende Informationen über die Produktionskapazitäten und die Nachfrage nach den Brotsorten:

- Insgesamt werden pro Woche b_W Laibe Weißbrot nachgefragt. Die Zahlen b_V, b_R und b_S sind analog definiert.
- Insgesamt werden pro Woche p_W Einheiten Weißbrot produziert. Die Zahlen p_V, p_R und p_S sind analog definiert.

Alle Zahlen sind positive, ganze Zahlen. Kann die Bäckerei die Nachfrage nach allen Brotsorten vollständig decken? Konstruieren Sie ein geeignetes Flussnetzwerk und erklären Sie, wie man damit diese Frage beantwortet.

The hipster bakery Room for Bread *offers four overpriced types of bread: white bread (W), whole grain bread (V), rye bread (R), and sourdough bread (S). Based on historical sales data, the bakery has the following information about production capacities and demand for the types of bread:*

- *A total of b_W loaves of white bread are demanded per week. The numbers b_V, b_R, and b_S are defined analogously.*
- *A total of p_W units of white bread are produced per week. The numbers p_V, p_R, and p_S are defined analogously.*

All numbers are positive integers. Can the bakery fully meet the demand for all types of bread? Construct a suitable flow network and explain how to use it to answer this question.

Lösungsskizze: (Entspricht Übungsaufgabe 3.3a.)
Wir konstruieren ein Flussnetzwerk mit einem Quellknoten s, einem Senkenknoten t und vier Knoten W, V, R und S für die Brotsorten. Die Kapazitäten der Kanten von s zu W, V, R und S sind p_W, p_V, p_R und p_S. Die Kapazitäten der Kanten von W, V, R und S zu t sind b_W, b_V, b_R und b_S. Ein maximaler Fluss in diesem Netzwerk hat den Wert $b_W + b_V + b_R + b_S$ genau dann, wenn die Bäckerei die Nachfrage nach allen Brotsorten vollständig decken kann.

(/ 5 Punkte)

e) Gegeben sei ein gerichteter Graph $G = (V, E)$ mit Kantengewichten, die positiv, null oder negativ sein können; negative Kreise dürfen jedoch nicht vorkommen. Außerdem seien gegeben zwei ausgezeichnete Knoten $s, t \in V$ im Graphen, sowie drei disjunkte Mengen $A, B, C \subseteq V$. Ein Weg heißt *schön*, wenn er in dieser Reihenfolge:

- bei s startet,
- dann 0 oder mehr Knoten aus V durchläuft,
- dann einen Knoten aus A besucht,
- dann 0 oder mehr Knoten aus V durchläuft,
- dann einen Knoten aus B besucht,
- dann irgendwelche Knoten aus V durchläuft,
- dann einen Knoten aus C besucht,
- dann irgendwelche Knoten aus V durchläuft,
- und schließlich bei t endet.

Geben Sie einen Algorithmus an, der in Zeit $O(|V|^3)$ die Länge eines kürzesten schönen Weges berechnet. Die Laufzeit muss begründet werden, die Korrektheit jedoch nicht. Sie dürfen Algorithmen aus der Vorlesung verwenden, ohne deren Pseudocode anzugeben.

Let a directed graph $G = (V, E)$ be given, with edge weights that can be positive, zero, or negative; however, negative cycles are not allowed. Furthermore, two distinguished nodes $s, t \in V$ are given in the graph, as well as three disjoint sets $A, B, C \subseteq V$. A path is called beautiful *if it starts at s, then visits zero or more nodes in V, then visits a node in A, then visits zero or more nodes in V, then visits a node in B, then visits zero or more nodes in V, then visits a node in C, then visits zero or more nodes in V, and finally ends at t. Describe an algorithm that computes the length of a shortest beautiful path in $O(|V|^3)$ time. The runtime must be justified, but correctness is not required. You may use algorithms from the lecture without providing their pseudocode.*

Lösungsskizze: (Entspricht Übungsaufgabe 1.11.)
Wir verwenden den Floyd-Warshall-Algorithmus, um in Zeit $O(|V|^3)$ die Längen $\operatorname{dist}(u, v)$ kürzester Wege zwischen allen Knotenpaaren (u, v) mit $u, v \in V$ zu berechnen. Wir speichern die Ergebnisse in einer Tabelle D. Der kürzeste schöne Weg hat dann die Länge $\min_{a \in A, b \in B, c \in C}(D[s, a] + D[a, b] + D[b, c] + D[c, t])$. Dieses Minimum berechnen wir mit erschöpfender Suche in Zeit $O(|A| \cdot |B| \cdot |C|) \leq O(|V|^3)$.

(/ 5 Punkte)

Aufgabe 2: Amortisiert, Randomisiert $3 + 3 + 4 + 5 + 5 = 20$ Punkte

a) Angenommen, wir führen für eine gegebene positive ganze Zahl R eine Folge von R^2 Operationen `f()` auf einer anfangs leeren Datenstruktur durch. Die Funktion `f()` läuft in amortisierter konstanter Zeit, benötigt aber im *worst-case* lineare Zeit (in Bezug auf die Anzahl bisher getätigter Aufrufe). Was ist die *worst-case* Laufzeit des folgenden Pseudocodes? Geben Sie Ihre Antwort in Θ-Notation als Funktion von R an.

Suppose we perform a sequence of R^2 operations `f()` on an initially empty data structure for some given positive integer R. The function `f()` runs in amortized constant time, but in the worst case it takes linear time (with respect to the number of calls made so far). What is the worst-case running time of the following pseudocode? Provide your answer in Θ-notation as a function of R.

Pseudocode:

```
for i from 1 to (R*R) do
    f()
```

Antwort: $\Theta\big(\underline{\quad R^2 \quad}\big)$.

(/ 3 Punkte)

b) Betrachten Sie den folgenden randomisierten Algorithmus `g(k)`:

```
1 function g(k):
2   Wähle eine Zufallszahl r gleichverteilt aus {1,2,...,k}.
3   if r == 1:
4     Erstelle ein Array von k zufälligen Zahlen.
5     Rufe MergeSort auf dem Array auf.
```

Geben Sie die erwartete Laufzeit der Funktion `g(k)` in Θ-Notation als Funktion von k an.

Consider the following randomized algorithm `g(k)`:

```
1 function g(k):
2   Choose a random number r uniformly from {1,2,...,k}.
3   if r == 1:
4     Create an array of k random numbers.
5     Call MergeSort on the array.
```

Provide the expected runtime of the function `g(k)` in Θ-notation as a function of k.

Antwort: $\Theta\big(\underline{\quad \log k \quad}\big)$.

(/ 3 Punkte)

c) Begründen Sie Ihre Antwort zu b): *Explain your answer to b):*

Lösungsskizze: (Es handelt sich um eine Analyse der erwarteten Laufzeit, ähnlich zu Übungsaufgabe 5.2.)
Zeilen 2 und 3 werden immer ausgeführt und benötigen Zeit $\Theta(\log k)$. Mit einer Wahrscheinlichkeit von $1 - 1/k$ gilt $r \neq 1$ und `g(k)` terminiert an dieser Stelle. Mit einer Wahrscheinlichkeit von $1/k$ wird ein Array von k zufälligen Zahlen erstellt und dann mit MergeSort sortiert. MergeSort ist deterministisch und hat auf jeder Eingabe der Länge k die Laufzeit $\Theta(k \log k)$. Daher ist die erwartete Laufzeit von `g(k)`:

$$\Theta\big(\log k + \tfrac{1}{k} \cdot k \log k\big) \leq \Theta(\log k)\,.$$

(/ 4 Punkte)

d) Angenommen, wir führen eine Folge von a Operationen auf einer Datenstruktur durch, wobei die b-te Operation Kosten von b verursacht, wenn b eine Quadratzahl ist, und sonst Kosten von 1. Beweisen Sie mit der Aggregationsmethode (=Summenmethode), dass die amortisierten Kosten pro Operation höchstens $O(\sqrt{a})$ sind.

Suppose we perform a sequence of a operations on a data structure in which the b-th operation costs b if b is a perfect square, and 1 otherwise. Use aggregate analysis (=summation method) to prove that the amortized cost per operation is at most $O(\sqrt{a})$.

Lösungsskizze: (Es handelt sich um eine Routine-Anwendung der Aggregationsmethode, siehe zum Beispiel Übungsaufgabe 4.1a.)
Seien c_i die Kosten der i-ten Operation. Wir haben:

$$c_i = \begin{cases} i\,, & \text{wenn } i \text{ eine Quadratzahl ist}\,, \\ 1\,, & \text{sonst}\,. \end{cases}$$

Die Gesamtkosten der a Operationen betragen:

$$\sum_{i=1}^{a} c_i \le a + \sum_{j=1}^{\lfloor\sqrt{a}\rfloor} j^2 \le a + \sum_{j=1}^{\lfloor\sqrt{a}\rfloor} a = a + \sqrt{a}\cdot a \le 2a^{3/2}\,.$$

Daher sind die Durchschnittskosten pro Operation höchstens $\frac{1}{a}\cdot 2a^{3/2} \le O(\sqrt{a})$. Nach der Aggregationsmethode sind die amortisierten Kosten pro Operation also $O(\sqrt{a})$.

(/ 5 Punkte)

e) Betrachten Sie den folgenden randomisierten Algorithmus `h(k)`, der als Eingabe eine positive ganze Zahl `k` erhält:

```
function h(k):
  Erstelle ein Array A[1..k] mit den Einträgen [1,2,...,k]
  Permutiere die Einträge von A gleichverteilt zufällig
  Setze c = 0
  Für i von 1 bis k:
    Falls A[1] das kleinste Element in A[1..i] ist:
      Erhöhe c um 1
  Liefere c zurück
```

Geben Sie den Erwartungswert für den Rückgabewert `c` von `h(k)` in Θ-Notation als Funktion von k an.

Consider the following randomized algorithm `h(k)` *that takes a positive integer* `k` *as input:*

```
function h(k):
  Create an array A[1..k] with entries [1,2,...,k]
  Permute the entries of A uniformly at random
  Set c = 0
  For i from 1 to k:
    If A[1] is the smallest element in A[1..i]:
      Increment c by 1
  Return c
```

Provide the expected value of the return value `c` *of* `h(k)` *in Θ-notation as a function of k.*

Antwort: $\mathbb{E}\big[h(k)\big] = \Theta\Big(\underline{\quad \log k \quad}\Big)$.

Begründen Sie Ihre Antwort: *Explain your answer:*

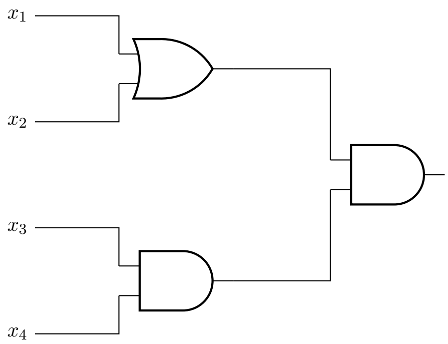
Lösungsskizze: (Variante von Übungsaufgabe 6.2.)
Sei X_i die Zufallsvariable, die angibt, ob $A[1]$ das kleinste Element in $A[1..i]$ ist. Dann ist $X_i = 1$ mit Wahrscheinlichkeit $1/i$ und $X_i = 0$ mit Wahrscheinlichkeit $1 - 1/i$. Der Erwartungswert von X_i ist daher $\frac{1}{i}$. Dann gilt für den Rückgabewert $h(k) = \sum_{i=1}^{k} X_i$. Mit Hilfe der Linearität des Erwartungswerts erhalten wir:

$$\mathbb{E}[h(k)] = \sum_{i=1}^{k} \mathbb{E}[X_i] = \sum_{i=1}^{k} \frac{1}{i} = \Theta(\log k)\,.$$

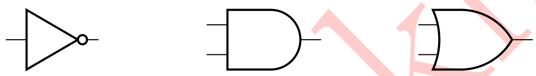
Die letzte Summe ist die k-te harmonische Zahl, die bekanntermaßen $\Theta(\log k)$ ist.

(/ 5 Punkte)

a) In der Vorlesung haben wir eine Reduktion von CircuitSat auf 3Sat gesehen. Welche 3Sat-Formel gibt diese Reduktion aus, wenn die Eingabe aus folgendem Schaltkreis besteht? Für diese Aufgabe genügt eine Umformung in eine Formel mit *höchstens* drei Literalen pro Klausel. *In the lecture, we have seen a reduction from CircuitSat to 3Sat. Which 3Sat-formula does this reduction output when given as input the following circuit? For this exercise, it is sufficient to transform to a formula with* at most *three literals per clause.*



Zur Erinnerung hier die Bedeutung der einzelnen Gatter: *Recall that the meaning of the gates in the above circuit is as follows:*



Lösungsskizze: (Entspricht Übungsaufgabe 7.1 aus der Vorlesung.)
Wir verwenden weiterhin die Variablen x_i für $1 \leq i \leq 4$ und außerdem eine Variable für den Ausgang jedes Gatters. Diese Ausgänge benennen wir von links nach rechts und oben nach unten mit g_1, g_2, g_3.
Es ergibt sich zunächst die folgende Formel:

$$(g_1 \leftrightarrow (x_1 \vee x_2)) \wedge (g_2 \leftrightarrow (x_3 \wedge x_4)) \wedge (g_3 \leftrightarrow (g_1 \wedge g_2)) \wedge g_3.$$

Daraus ergibt sich durch passende Umformung die folgende 3CNF-Formel:

$$\begin{aligned}
&(\neg g_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee g_1) \wedge (\neg x_2 \vee g_1) \wedge \\
&(\neg g_2 \vee x_3) \wedge (\neg g_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee g_2) \wedge \\
&(\neg g_3 \vee g_1) \wedge (\neg g_3 \vee g_2) \wedge (\neg g_1 \vee \neg g_2 \vee g_3) \wedge \\
&g_3
\end{aligned}$$

(/ 3 Punkte)

b) Betrachten Sie das folgende Entscheidungsproblem DOMINATINGSET:

Eingabe:	Ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$.
Frage:	Gibt es eine Teilmenge $D \subseteq V$ mit $|D| = k$, sodass es für alle $u \in V$ ein $v \in D$ gibt mit $\{u, v\} \in E$?

Zeigen Sie, dass DOMINATINGSET in NP liegt.

Consider the decision problem DOMINATINGSET:

Input:	*undirected graph $G = (V, E)$, $k \in \mathbb{N}$.*
Question:	*Is there a subset $D \subseteq V$ with $|D| = k$ such that for all $u \in V$ there is a $v \in D$ with $\{u, v\} \in E$?*

Show that DOMINATINGSET is in NP.

Lösungsskizze: (Zeigen, dass ein Problem in NP liegt, war Bestandteil von Übungsaufgabe 8.3.) Wir geben einen Verifizierungsalgorithmus für DOMINATINGSET an. Zertifikat ist ein dominating set der Größe k.
Eingabe ist also G, k, D.
Der Algorithmus prüft zunächst, ob $k < |V(G)|$. Falls nein, lehne ab. Sonst prüfe, ob $|D| = k$. Falls nein, lehne ab. Dann iteriere über alle Knoten $u \in V(G)$ und alle Knoten $v \in D$. Falls die Kante $\{u, v\}$ in G vorhanden ist, gehe weiter zum nächsten Knoten in $V(G)$. Falls diese Kante für keinen der Knoten $v \in D$ vorhanden war, lehne ab. Werden auf diese Art alle Knoten $u \in V(G)$ durchlaufen ohne abzulehnen, akzeptiere.
Es kann offenbar in Linearzeit überprüft werden, ob $k < |V(G)|$ und ob $|D| = k$. Die Knoten $u \in V(G)$ und $v \in D$ zu durchlaufen benötigt höchstens Zeit $O(|V(G)| \cdot |D|) \subseteq O(|V(G)|^2)$ und für ein Knotenpaar (u, v) zu überprüfen, ob $\{u, v\} \in E(G)$ benötigt konstante Zeit (RAM-Modell, Adjazenzmatrix angenommen). Insgesamt läuft der Algorithmus also in Polynomialzeit.

(/ 3 Punkte)

c) Sei Reach das Entscheidungsproblem, für einen gegebenen ungerichteten Graphen G und zwei gegebene Knoten $s, t \in V(G)$ festzustellen, ob es einen s-t-Pfad in G gibt. Für welche zwei der folgenden fünf Aussagen ist bereits bekannt, dass sie wahr sind? *Let Reach be the decision problem asking, for a given graph G and vertices s, t, whether there is an s-t-path in G. For which two of the following five statements is it already known that they are true?*

☐ Es gibt keine Polynomialzeitreduktion von Reach auf 3Sat. *There is no polynomial time reduction from Reach to 3Sat.*

☐ Es gibt eine Polynomialzeitreduktion von 3Sat auf Reach. *There is a polynomial time reduction from 3Sat to Reach.*

☑ Wenn $\mathrm{P} = \mathrm{NP}$ gilt, dann gibt es eine Polynomialzeitreduktion von 3Sat auf Reach. *If $\mathrm{P} = \mathrm{NP}$, then there is a polynomial time reduction from 3Sat to Reach.*

☑ Wenn $\mathrm{P} \neq \mathrm{NP}$ gilt, dann gibt es eine Polynomialzeitreduktion von Reach auf 3Sat. *If $\mathrm{P} \neq \mathrm{NP}$, then there is a polynomial time reduction from Reach to 3Sat.*

☐ Wenn $\mathrm{P} \neq \mathrm{NP}$ gilt, dann gibt es eine Polynomialzeitreduktion von 3Sat auf Reach. *If $\mathrm{P} \neq \mathrm{NP}$, then there is a polynomial time reduction from 3Sat to Reach.*

(/ 2 Punkte)

d) Seien Π und Π' zwei Entscheidungsprobleme. Welche zwei der folgenden fünf Aussagen sind wahr? *Let Π and Π' be two decision problems. Which two of the following five statements are true?*

☑ Um zu beweisen, dass Π' NP-schwer ist, genügt es zu beweisen, dass Π NP-schwer ist und es eine Polynomialzeitreduktion von Π auf Π' gibt. *In order to prove that Π' is NP-hard, it is sufficient to prove that Π is NP-hard and there is a polynomial time reduction from Π to Π'.*

☐ Um zu beweisen, dass $\mathrm{P} \neq \mathrm{NP}$ gilt, genügt es zu beweisen, dass Π in P ist, Π' in NP ist und es eine Polynomialzeitreduktion von Π auf Π' gibt. *In order to prove $\mathrm{P} \neq \mathrm{NP}$, it is sufficient to prove that Π is in P, Π' is in NP and there is a polynomial time reduction from Π to Π'.*

☐ Um zu beweisen, dass $\mathrm{P} = \mathrm{NP}$ gilt, genügt es zu beweisen, dass Π in P ist, Π' in NP ist und es eine Polynomialzeitreduktion von Π' auf Π gibt. *In order to prove $\mathrm{P} = \mathrm{NP}$, it is sufficient to prove that Π is in P, Π' is in NP and there is a polynomial time reduction from Π' to Π.*

☐ Um zu beweisen, dass Π' NP-schwer ist, genügt es zu beweisen, dass Π NP-schwer ist und es eine Polynomialzeitreduktion von Π' auf Π gibt. *In order to prove that Π' is NP-hard, it is sufficient to prove that Π is NP-hard and there is a polynomial time reduction from Π' to Π.*

☑ Um zu beweisen, dass $\mathrm{P} = \mathrm{NP}$ gilt, genügt es zu beweisen, dass Π in P ist, Π' NP-schwer ist und es eine Polynomialzeitreduktion von Π' auf Π gibt. *In order to prove that $\mathrm{P} = \mathrm{NP}$, it is sufficient to prove that Π is in P, Π' is NP-hard and there is a polynomial time reduction from Π' to Π.*

(/ 2 Punkte)

e) Wir betrachten die folgenden beiden Entscheidungsprobleme HalfClique und QuarterClique:

HalfClique
Eingabe: Ungerichteter Graph $G = (V, E)$.
Frage: Gibt es in G eine Clique der Größe mindestens $\lceil |V|/2 \rceil$?

QuarterClique
Eingabe: Ungerichteter Graph $G = (V, E)$.
Frage: Gibt es in G eine Clique der Größe mindestens $\lceil |V|/4 \rceil$?

Zeigen Sie, dass QuarterClique in Polynomialzeit auf HalfClique reduzierbar ist.

Erinnerung: Eine Clique der Größe k ist ein vollständiger Graph mit k Knoten. Die Operation $\lceil \cdot \rceil$ ist die Aufrunde-Operation. Das heißt: Ist x eine reelle Zahl, so ist $\lceil x \rceil$ die kleinste ganze Zahl n mit $n \geq x$.

Consider the following two decision problems HalfClique and QuarterClique:

HalfClique
Input: *Undirected graph $G = (V, E)$.*
Question: *Is there a clique of size at least $\lceil |V|/2 \rceil$ in G?*

QuarterClique
Input: *Undirected graph $G = (V, E)$.*
Question: *Is there a clique of size at least $\lceil |V|/4 \rceil$ in G?*

Show that QuarterClique is reducible to HalfClique in polynomial time.

Reminder: A clique of size k is a complete graph with k vertices. The operation $\lceil \cdot \rceil$ is the round-up operation. This means: If x is a real number, then $\lceil x \rceil$ is the smallest integer n such that $n \geq x$.

Lösungsskizze: (Es handelt sich um eine Variante von Übungsaufgaben 8.1 und 8.2.)
Wir geben eine Polynomialzeitreduktion von QuarterClique auf HalfClique an.
Sei G die Eingabe für unsere Reduktion.
Sei $k = \lceil |V(G)|/2 \rceil - \lceil |V(G)|/4 \rceil$. Konstruiere einen Graphen G', der sich aus G durch das Hinzufügen von $2k$ Knoten ergibt, die untereinander und zu allen ursprünglichen Knoten adjazent sind. Der Graph G' lässt sich offenbar in Polynomialzeit aus G konstruieren, da auch G' nur $O(|V(G)|)$ Knoten hat und sich für je zwei Knoten leicht feststellen lässt, ob eine Kante vorhanden ist.
Es gilt:

$$\begin{aligned}
G \in \textsc{QuarterClique} &\iff G \text{ enthält eine Clique der Größe } \lceil |V(G)|/4 \rceil \\
&\iff G \text{ enthält eine Clique der Größe } \lceil |V(G)|/2 \rceil - k \\
&\iff G' \text{ enthält eine Clique der Größe } \lceil |V(G)|/2 \rceil - k + 2k \\
&\qquad = \lceil |V(G)|/2 \rceil + k \\
&\iff G' \text{ enthält eine Clique der Größe } \lceil |V(G')|/2 \rceil \\
&\iff G' \in \textsc{HalfClique}.
\end{aligned}$$

(/ 10 Punkte)

Aufgabe 4: Berechenbarkeit $\quad 4+2+2+2+2+4+4 = 20$ Punkte

Sei $M_{\text{ystical}} := (Q, \Sigma, \Gamma, \delta, q_0, \texttt{halt}, \square)$ eine Turingmaschine, wobei $Q := \{\texttt{halt}\} \cup \{q_i \mid 0 \leq i \leq 4\}$, $\Sigma := \{a, b\}$, $\Gamma := \Sigma \cup \{\square\}$ und die Überführungsfunktion δ durch die folgenden Übergänge gegeben ist: *Let $M_{ystical} := (Q, \Sigma, \Gamma, \delta, q_0, \texttt{halt}, \square)$ be a Turing machine, where $Q := \{\texttt{halt}\} \cup \{q_i \mid 0 \leq i \leq 4\}$, $\Sigma := \{a, b\}$, $\Gamma := \Sigma \cup \{\square\}$ and the transition function δ is given by the following transitions:*

$(q_0, a) \to (q_0, a, +1)$
$(q_0, b) \to (q_1, b, +1)$
$(q_0, \square) \to (\texttt{halt}, \square, -1)$
$(q_1, a) \to (q_2, b, -1)$
$(q_1, b) \to (q_1, b, +1)$
$(q_1, \square) \to (q_4, \square, -1)$

$(q_2, a) \to (q_3, a, +1)$
$(q_2, b) \to (q_2, b, -1)$
$(q_2, \square) \to (q_3, \square, +1)$
$(q_3, b) \to (q_1, a, +1)$
$(q_4, a) \to (\texttt{halt}, a, -1)$
$(q_4, b) \to (q_4, \square, -1)$
$(q_4, \square) \to (\texttt{halt}, \square, +1)$

Wir nehmen hier an, dass das Band auch links von der Eingabe weitere Zellen hat, die zu Beginn das Blanksymbol $\square$ enthalten. *We assume here that the band also has cells left of the input, which initially contain the blank symbol $\square$.*

a) Führen Sie die Maschine M_{ystical} auf Eingabe $ababa$ aus, bis die Maschine hält. Geben Sie in geeigneter Weise die Startkonfiguration und die Konfigurationen nach jedem Berechnungsschritt an, also den Bandinhalt, die Kopfposition und den aktuellen Zustand. *Run the machine $M_{ystical}$ on input ababa until the machine halts. State the starting configuration and the configurations after each step of the computation in a suitable manner, that is, state the content of the tape, the position of the head, and the current state in each of the steps.*

Lösungsskizze: (Es handelt sich um eine Variante von Übungsaufgabe 9.1.)

Zustand	Bandinhalt (Kopfposition unterstrichen)
q_0	$\underline{a}baba$
q_0	$a\underline{b}aba$
q_1	$ab\underline{a}ba$
q_2	$a\underline{b}bba$
q_2	$\underline{a}bbba$
q_3	$a\underline{b}bba$
q_1	$aa\underline{b}ba$
q_1	$aab\underline{b}a$
q_1	$aabb\underline{a}$
q_2	$aab\underline{b}b$
q_2	$aa\underline{b}bb$
q_2	$a\underline{a}bbb$
q_3	$aa\underline{b}bb$
q_1	$aaa\underline{b}b$
q_1	$aaab\underline{b}$
q_1	$aaabb\underline{\square}$
q_4	$aaab\underline{b}$
q_4	$aaa\underline{b}$
q_4	$aa\underline{a}$
$\texttt{halt}$	$a\underline{a}a$

(/ 4 Punkte)

b) Welche Funktion f berechnet die Maschine M_{ystical}? *What function f does the machine $M_{ystical}$ compute?*

Lösungsskizze: $f(w) = a^{|w|_a}$ für alle $w \in \{a, b\}^*$, wobei $|w|_a$ die Anzahl der in w vorkommenden a ist.

(/ 2 Punkte)

c) Sei $\Sigma := \{0,1\}$. Welche zwei der folgenden fünf Sprachen sind entscheidbar?
Let $\Sigma := \{0,1\}$. Which two of the following five languages are decidable?

☐ $\{\langle M, w\rangle \mid M$ gibt auf Eingabe w das leere Wort aus$\}$
$\{\langle M\rangle \mid$ *M on input w outputs the empty string*$\}$

☑ $\{\langle M\rangle \mid$ Es gibt eine Eingabe auf der M die ersten 100 Bandzellen verlässt$\}$
$\{\langle M\rangle \mid$ *There is an input on which M leaves the first 100 band cells*$\}$

☑ $\{\langle M, w, t\rangle \mid M$ akzeptiert Eingabe w in höchstens 2^{2^t} Schritten$\}$
$\{\langle M, w, t\rangle \mid$ *M accepts input w in at most 2^{2^t} steps*$\}$

☐ $\{\langle M, M'\rangle \mid$ Es gibt eine Eingabe auf der M hält und M' nicht hält$\}$
$\{\langle M, M'\rangle \mid$ *There is an input on which M halts and M' does not halt*$\}$

☐ $\{\langle M, w, a, b\rangle \mid M$ überschreibt auf Eingabe w irgendwann Symbol a mit Symbol $b\}$
$\{\langle M, w, a, b\rangle \mid$ *M on input w at some point overwrites symbol a by symbol b*$\}$

(/ 2 Punkte)

d) Welche zwei der folgenden fünf Aussagen sind wahr?

Hinweis: Die Notation $L \leq L'$ meint hier Reduzierbarkeit mit sogenannten many-one-Reduktionen, also Reduktionen, die Instanzen des einen Problems auf Instanzen des anderen Problems abbilden, wie in der Vorlesung im Abschnitt zu Berechenbarkeit verwendet.

Which two of the following five statements are true?

Please note: The notation $L \leq L'$ here refers to reducibility with so-called many-one reductions, that is, reductions mapping instances of one problem to instances of the other, as used in the part of the lecture on computability.

☐ Es gilt $\textsc{SelfHalt} \leq \overline{\textsc{SelfHalt}}$, wobei $\overline{\textsc{SelfHalt}}$ das Komplement von $\textsc{SelfHalt}$ ist. *It holds that $\textsc{SelfHalt} \leq \overline{\textsc{SelfHalt}}$, where $\overline{\textsc{SelfHalt}}$ is the complement of $\textsc{SelfHalt}$.*

☑ Es gibt eine Sprache L, sodass sowohl $L \leq \overline{L}$ als auch $\overline{L} \leq L$ gilt, wobei $\overline{L}$ das Komplement von L ist. *There is a language L such that both $L \leq \overline{L}$ and $\overline{L} \leq L$ hold, where $\overline{L}$ is the complement of L.*

☐ Für alle Sprachen L und L' gilt: Wenn $L \leq L'$ und L' nicht semi-entscheidbar ist, dann ist auch L nicht semi-entscheidbar. *For all languages L and L' the following holds: If $L \leq L'$ and L' is not semi-decidable, then L is not semi-decidable.*

☐ Für alle Sprachen L und L' gilt: Wenn $L \subseteq L'$ und L nicht semi-entscheidbar ist, dann ist auch L' nicht semi-entscheidbar. *For all languages L and L' the following holds: If $L \subseteq L'$ and L is not semi-decidable, then L' is not semi-decidable.*

☑ Für alle Sprachen L und L' gilt: Wenn $L \leq L'$ und L' semi-entscheidbar ist, dann ist auch L semi-entscheidbar. *For all languages L and L' the following holds: If $L \leq L'$ and L' is semi-decidable, then L is semi-decidable.*

(/ 2 Punkte)

e) Sei $\mathrm{bin}\colon \mathbb{N} \to \{0,1\}^*$ die Funktion, die jede natürliche Zahl auf ihre Binärdarstellung ohne führende Nullen abbildet und sei

$$L_1 \coloneqq \left\{ \langle M \rangle \;\middle|\; \begin{array}{c} \text{Für alle } n \in \mathbb{N} \text{ erzeugt } M \text{ auf Eingabe } \mathrm{bin}(n) \text{ die Ausgabe } \mathrm{bin}(n+1) \\ \text{und verhält sich auf anderen Eingaben beliebig} \end{array} \right\}.$$

Geben Sie eine nicht-triviale Menge S_1 an, sodass $C(S_1) = L_1$ gilt und sich mit dieser Menge der Satz von Rice anwenden lässt, um zu zeigen, dass L_1 unentscheidbar ist. (Nicht-Trivialität und Korrektheit müssen nicht bewiesen werden.)

Let $\mathrm{bin}\colon \mathbb{N} \to \{0,1\}^*$ be the function that maps any natural number to its binary representation without leading zeroes and let

$$L_1 \coloneqq \left\{ \langle M \rangle \;\middle|\; \begin{array}{c} \textit{For all } n \in \mathbb{N} \textit{ the machine } M \textit{ on input } \mathrm{bin}(n) \textit{ outputs } \mathrm{bin}(n+1) \\ \textit{and it behaves arbitrarily on other inputs} \end{array} \right\}.$$

Provide a non-trivial set S_1 such that $C(S_1) = L_1$ and, consequently, Rice's Theorem can be applied using S_1 to show that L_2 is undecidable. (Non-triviality and correctness do not need to be proven.)

Lösungsskizze: (Es handelt sich um eine Variante von Übungsaufgabe 10.3b.)

$$S_1 \coloneqq \left\{ f\colon \{0,1\}^* \to \{0,1\}^* \;\middle|\; \begin{array}{c} f \text{ ist berechenbare Funktion und} \\ f(\mathrm{bin}(n)) = \mathrm{bin}(n+1) \text{ für alle } n \in \mathbb{N} \end{array} \right\}$$

(/ 2 Punkte)

f) Sei die Menge S_2 wie folgt definiert:

$$S_2 \coloneqq \left\{ f\colon \{0,1\}^* \to \{0,1\}^* \;\middle|\; \begin{array}{c} f \text{ ist berechenbare Funktion und es gibt ein } n \in \mathbb{N}, \\ \text{sodass } f \text{ auf } \textit{allen} \text{ Eingaben der Länge } n \text{ und auf} \\ \textit{keiner} \text{ Eingabe einer anderen Länge definiert ist} \end{array} \right\}.$$

Zeigen Sie, dass S_2 nicht-trivial ist (wie im Satz von Rice definiert).

Let the set S_2 be defined as follows:

$$S_2 \coloneqq \left\{ f\colon \{0,1\}^* \to \{0,1\}^* \;\middle|\; \begin{array}{c} f \textit{ is a computable function and there is an } n \in \mathbb{N} \\ \textit{such that } f \textit{ is defined on all inputs of length } n \textit{ and} \\ \textit{on no input of any other length} \end{array} \right\}.$$

Show that S_2 is non-trivial (as defined in Rice's Theorem).

Lösungsskizze: (Es handelt sich um eine Variante von Übungsaufgabe 10.3b.)
Die überall undefinierte Funktion Ω ist nicht in S_2, also ist S_2 nicht die Menge aller berechenbaren Funktionen.
Die Funktion $f_1\colon \{0,1\}^* \to \{0,1\}^*$ mit $f(0) = f(1) = 1$ und f undefiniert für alle anderen Eingaben ist in S_2, also ist $S_2 \neq \emptyset$.

(/ 4 Punkte)

g) Sei die Sprache L_2 wie folgt definiert:

$$L_2 \coloneqq \{\langle M_1, M_2\rangle \mid M_1 \text{ hält auf Eingabe } \langle M_1\rangle \text{ oder } M_2 \text{ hält auf Eingabe } \langle M_2\rangle\}$$

Sei die Funktion r wie folgt definiert: Jede gegebene Eingabe $\langle M\rangle$ wird von r auf $\langle M, \Omega\rangle$ abgebildet, wobei Ω eine Maschine ist, die auf keiner Eingabe hält. Zeigen Sie, dass die Funktion r eine Reduktion von SelfHalt auf L_2 ist.

Let the language L_2 be defined as follows:

$$L_2 \coloneqq \{\langle M_1, M_2\rangle \mid M_1 \text{ halts on input } \langle M_1\rangle \text{ or } M_2 \text{ halts on input } \langle M_2\rangle\}$$

Let the function r be defined as follows: Any input $\langle M\rangle$ is mapped by r to $\langle M, \Omega\rangle$, where Ω is a machine that halts on no input. Show that r is a reduction from SelfHalt to L_2.

Lösungsskizze: (Es handelt sich um eine Variante von Übungsaufgabe 10.3a.)
Die Funktion r ist offenbar berechenbar, da nur die Pairing-Funktion für das Paar (M, Ω) berechnet werden muss, wobei Ω eine feste Maschine ist, die nicht von der Eingabe abhängt.
Weiterhin gilt:

$$\begin{aligned}
\langle M\rangle \in \textsc{SelfHalt} &\iff \langle M\rangle \text{ hält auf Eingabe } \langle M\rangle \\
&\iff \langle M\rangle \text{ hält auf Eingabe } \langle M\rangle \text{ oder } \Omega \text{ hält auf Eingabe } \langle \Omega\rangle \\
&\iff \langle M, \Omega\rangle \in L_2
\end{aligned}$$

(/ 4 Punkte)

Gegeben sei folgendes, in kanonischer Form gegebene lineare Programm (LP_1):
Consider the following linear program (LP_1) given in canonical form:

$$\begin{array}{rrl} \text{maximize} & 2x_1 + 4x_2 + 3x_3 & \\ \text{subject to} & x_1 + 2x_2 + x_3 & \leq 10 \\ & 2x_1 + x_2 - 2x_3 & \leq 8 \\ & x_1, x_2, x_3 & \geq 0 \end{array}$$

a) Welchen Wert hat die zulässige Lösung $(x_1, x_2, x_3) = (0, 0, 10)$ von (LP_1)?
What is the value of the feasible solution $(x_1, x_2, x_3) = (0, 0, 10)$ of (LP_1)?

Antwort: 30.

(/ 2 Punkte)

b) Geben Sie das duale lineare Programm für (LP_1) an:
Formulate the dual linear program for (LP_1):

Lösungsskizze: (Es handelt sich um eine Variante von Übungsaufgabe 13.1.)

$$\begin{array}{lrcrcl} \min & 10y_1 & + & 8y_2 & & \\ \text{s.t.} & y_1 & + & 2y_2 & \geq & 2 \\ & 2y_1 & + & y_2 & \geq & 4 \\ & y_1 & - & 2y_2 & \geq & 3 \\ & y_1, & & y_2 & \geq & 0 \end{array}$$

(/ 3 Punkte)

c) Geben Sie eine optimale zulässige Lösung des primalen linearen Programms (LP_1) an. (Hinweis: Untersuchen Sie die zulässige Lösung $(3, 0)$ im dualen Programm.)
Provide an optimal feasible solution of the primal linear program (LP_1). (Hint: Examine the feasible solution $(3, 0)$ in the dual program.)

Antwort: $(0, 0, 10)$.

(/ 2 Punkte)

d) Begründen Sie Ihre Antwort zu c). *Justify your answer to c).*

Lösungsskizze: Wir wenden die schwache Dualität an: Die zulässige Lösung $(y_1, y_2) = (3, 0)$ im dualen Programm hat den Wert 30. Da der Wert der zulässigen Lösung $(x_1, x_2, x_3) = (0, 0, 10)$ im primalen Programm ebenfalls 30 ist, folgt aus der schwachen Dualität, dass beide Lösungen optimal sind.

(/ 3 Punkte)

e) In einer Stadt gibt es B Bäckereien $b_1, \dots, b_B$ und S Supermärkte $s_1, \dots, s_S$. Jede Bäckerei b_i kann täglich bis zu K_i Brotlaibe backen. Jeder Supermarkt s_j hat einen täglichen Bedarf von B_j Brotlaiben. Die Kosten für den Transport eines Brotlaibs von einer Bäckerei b_i zu einem Supermarkt s_j werden als C_{ij} in Euro angegeben.

Formulieren Sie ein ganzzahliges lineares Programm (ILP), das die gesamten Transportkosten minimiert. Dabei muss der tägliche Bedarf aller Supermärkte gedeckt werden, ohne die Backkapazitäten der Bäckereien zu überschreiten.

Hinweise:

- Führen Sie Variablen T_{ij} ein, die angeben, wie viele Brotlaibe von der Bäckerei b_i zum Supermarkt s_j transportiert werden sollen.
- Die Nebenbedingungen sollten sicherstellen, dass der Bedarf jedes Supermarkts gedeckt wird und die Produktionskapazität jeder Bäckerei nicht überschritten wird.
- Alle Variablen und Konstanten sollen ganzzahlig und nicht negativ sein.

In a city there are B bakeries $b_1, \dots, b_B$ and S supermarkets $s_1, \dots, s_S$. Each bakery b_i can bake up to K_i loaves of bread per day. Each supermarket s_j has a daily demand of B_j loaves of bread. The cost for transporting a loaf of bread from a bakery b_i to a supermarket s_j is given as C_{ij} in Euro.

Formulate an integer linear program (ILP) that minimizes the total transport costs. The daily demand of all supermarkets must be met without exceeding the baking capacities of the bakeries.

Hints:

- *Introduce variables T_{ij} that indicate how many loaves of bread should be transported from bakery b_i to supermarket s_j.*
- *The constraints should ensure that the demand of each supermarket is met and the production capacity of each bakery is not exceeded.*
- *All variables and constants should be non-negative integers.*

Lösungsskizze: (Es handelt sich um eine Umformulierung von Übungsaufgabe 13.3.) Das ganzzahlige lineare Programm (ILP) zur Minimierung der Transportkosten für die Brotverteilung von Bäckereien zu Supermärkten kann wie folgt formuliert werden:

Zielfunktion Minimiere die Gesamtkosten der Brotlieferungen:

$$\min \sum_{i=1}^{B} \sum_{j=1}^{S} C_{ij} T_{ij}$$

Nebenbedingungen

(a) Jeder Supermarkt s_j erhält genügend Brotlaibe, um seinen täglichen Bedarf zu decken:

$$\forall j \in \{1, \ldots, S\} : \sum_{i=1}^{B} T_{ij} \geq B_j$$

(b) Die Anzahl der von jeder Bäckerei b_i gelieferten Brotlaibe überschreitet nicht ihre tägliche Backkapazität:

$$\forall i \in \{1, \ldots, B\} : \sum_{j=1}^{S} T_{ij} \leq K_i$$

(c) Die Anzahl der transportierten Brotlaibe muss ganzzahlig und nicht negativ sein:

$$\forall i \in \{1, \ldots, B\} \, \forall j \in \{1, \ldots, S\} : T_{ij} \in \mathbb{Z}, T_{ij} \geq 0$$

Lösungsansatz Um das ILP zu lösen, kann ein geeigneter Solver für ganzzahlige lineare Programme verwendet werden. Die tatsächlichen Werte für T_{bs} hängen von den spezifischen Kapazitäten der Bäckereien, den Bedarfen der Supermärkte und den Transportkosten ab. Der Solver findet eine Kombination von Werten für T_{bs}, die die Zielfunktion minimiert und dabei alle Nebenbedingungen erfüllt.

(/ 10 Punkte)

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 22 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 22 was placed at the corresponding task.*

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 23 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 23 was placed at the corresponding task.*

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 24 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 24 was placed at the corresponding task.*