Berlin, 20.07.2020

Name: ..... **Matr.-Nr.: .....**

Klausur Algorithmentheorie

(Niedermeier/Nichterlein/Bentert/Heeger, Sommersemester 2020)

Aufgabe Nr.:	1	2	3	4	5	**Summe**
Punktzahl:	10	10	8	10	12	50
Davon erreicht:						

Einlesezeit: 15 Minuten
Bearbeitungszeit: 60 Minuten
Max. Punktezahl: 50 Punkte

Allgemeine Hinweise:

- Es sind keinerlei Hilfsmittel erlaubt.
- Benutzen Sie keinen Bleistift, sondern einen Kugelschreiber in der Farbe Schwarz oder Blau.
- Beschriften Sie jedes Blatt mit Vor- und Nachnamen sowie Matrikelnummer.
- **Falls in der Aufgabenstellung nicht explizit ausgeschlossen, so sind alle Antworten zu begründen! Antworten ohne Begründung erhalten 0 Punkte.**
- Sätze aus der Vorlesung dürfen ohne Beweis verwendet werden.

Viel Erfolg!

Aufgabe 1: **Modellieren mit Min-Cost Flow** (10 Punkte)

Gegeben sei eine $n \times n$-Matrix M, wobei jeder Eintrag eine natürliche Zahl ist. Die Aufgabe besteht darin, n Einträge aus M auszuwählen, sodass in jeder Zeile und in jeder Spalte genau ein Eintrag ausgewählt wurde und die Summe der gewählten Einträge maximiert wird.

Modellieren Sie dieses Problem als ein Min-Cost Flow-Problem, sodass die Anzahl der Knoten und die Anzahl der Kanten im Flussnetzwerk polynomiell in n sind. Geben Sie hierfür die Knoten, Kanten und Kantenkapazitäten Ihres Flussnetzwerks an und beschreiben Sie, wie die Antwort bestimmt werden kann.

Eine Eingabe könnte zum Beispiel wie folgt aussehen, wobei die markierten Einträge die optimale Lösung zeigen.

$$M = \begin{pmatrix} 3 & \boxed{4} & 5 \\ \boxed{6} & 4 & 2 \\ 6 & 6 & \boxed{8} \end{pmatrix}$$

Erinnerung:

Minimum Cost Flow

Eingabe: Ein gerichteter Graph $G = (V, E)$, zwei Knoten $s, t \in V$, ein Zielwert $D \in Q^+$, Kantenkapazitäten $c\colon E \to Q^+$ und Kantenkosten $w\colon E \to Q$.

Aufgabe: Bestimme einen Fluss f vom Wert D mit minimalen Kosten, d.h. f minimiert $\sum_{e \in E} f(e) \cdot w(e)$.

Aufgabe 2: **Modellieren mit (I)LP** (10 Punkte)

Geben Sie ein (I)LP für das folgende Problem an.

Sie betreiben eine Druckerei und es ist Prüfungszeit. Sie haben m Drucker zur Verfügung (die alle gleich schnell drucken können) und bekommen für n verschiedene Lehrveranstaltungen Aufträge Klausuren zu drucken. Jede Veranstaltung i hat eine andere Klausur und benötigt eine andere Stückzahl. Die Zeit, die einer Ihrer Drucker braucht um alle Klausuren für Veranstaltung i zu drucken, beträgt t_i und ein Auftrag kann nicht auf mehrere Drucker verteilt werden. Ihr Ziel ist alle benötigten Klausuren in möglichst geringer Gesamtzeit zu drucken, das heißt, die Zeit bis der letzte Drucker fertig gedruckt hat zu minimieren.

Hinweis: Verwenden Sie eine Variable C für die Gesamtzeit.

Aufgabe 3: **Spezialfälle NP-schwerer Probleme** (8 Punkte)

Betrachten Sie das aus der Vorlesung bekannte NP-schwere Problem CLIQUE.

CLIQUE
Eingabe: Ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl k.
Frage: Gibt es k Knoten in G, die paarweise durch eine Kante verbunden sind?

Zeigen Sie, dass CLIQUE in einem der beiden folgenden Spezialfälle in Polynomzeit gelöst werden kann.

1. Die Anzahl n der Knoten und die Anzahl m der Kanten in G sind gleich, d.h., $n = m$.
2. Jede Zusammenhangskomponente des Graphen besteht aus maximal sieben Knoten.

Aufgabe 4: **Parametrisierte Algorithmen** (10 Punkte)

Betrachten Sie das folgende Graphproblem.

3-Färbbarkeit
Eingabe: Ein ungerichteter Graph $G = (V, E)$.
Frage: Lassen sich die Knoten von G so mit drei Farben färben, dass benachbarte Knoten immer unterschiedliche Farben erhalten?

Zeigen Sie, dass 3-Färbbarkeit in $\mathcal{O}(3^k \cdot (n + m))$ Zeitschritten gelöst werden kann, wobei n die Anzahl der Knoten und m die Anzahl der Kanten ist. Hierbei sei k die Größe eines kleinsten Vertex Covers von G, also die kleinste Zahl k, sodass eine Menge von k Knoten existiert, die alle Kanten von G abdeckt.

Hinweis: Sie können davon ausgehen, dass Ihnen ein Vertex Cover S der Größe k als Eingabe gegeben wurde.

Aufgabe 5: **Approximation** (12 Punkte)

Betrachten Sie das aus der Vorlesung bekannte KNAPSACK-Problem.

KNAPSACK

Eingabe: Eine Menge O von n Objekten, zusammen mit Gewichten $\text{weight}\colon O \to \mathbb{N}$, Werten $\text{value}\colon O \to \mathbb{N}$ und einer natürlichen Zahl B.

Aufgabe: Finde eine Teilmenge $O' \subseteq O$ von Objekten mit $\sum_{o \in O'} \text{weight}(o) \leq B$, die $\sum_{o \in O'} \text{value}(o)$ maximiert.

(a) Betrachten Sie die KNAPSACK-Instanz ($O = \{o_1, o_2, o_3, o_4, o_5\}, \text{weight}, \text{value}, B = 10$), wobei weight und value in Tabelle 1 beschrieben sind. Geben Sie eine optimale Lösung dieser Instanz an (ohne Beweis).

Objekt	o_1	o_2	o_3	o_4	o_5
value	2	5	3	8	4
weight	4	6	2	7	5

Tabelle 1: Die Gewichte und Werte der KNAPSACK-Instanz.

(b) Zeigen Sie, dass der folgende Algorithmus für KNAPSACK immer eine Lösung berechnet, deren Gesamtwert mindestens ein Drittel des optimalen Werts entspricht.

```
1 Lösche alle Objekte o mit weight(o) > B.
2 Sortiere O nach absteigendem Kosten-Gewicht-Verhältnis value/weight, d.h.
    O = {o_1, ..., o_n} mit value(o_1)/weight(o_1) ≥ value(o_2)/weight(o_2) ≥ ··· ≥ value(o_n)/weight(o_n).
3 Sei k maximal sodass Σ_{i=1}^{k} weight(o_i) ≤ B.
4 if Σ_{i=1}^{k} value(o_i) ≥ value(o_{k+1}) then
5 |   return {o_1, o_2, ..., o_k}
6 else
7 |   return {o_{k+1}}
8 end
```

Hinweis: Der Gesamtwert der vom Algorithmus berechnete Lösung entspricht sogar immer mindestens der Hälfte des optimalen Werts.