

Probeklausur 2 – Analyse von Algorithmen und Komplexität
SS 2026

Name: _____ Matrikel-Nr.: _____

Hinweise:

- Bearbeiten Sie *alle* Aufgaben.
- Sie haben 10 Minuten Einlesezeit. Danach haben Sie **120 Minuten** Zeit für die Klausurbearbeitung.
- Erlaubte Hilfsmittel: Ein doppelseitig handschriftlich beschriebenes Blatt Papier DIN A4, (bunte) Stifte (kein Rot, kein Grün, kein Bleistift).
- Die Klausur besteht aus **5 Aufgaben**, die alle **10 Punkte** wert sind.
- Zum Bestehen müssen mindestens **18 Punkte** erreicht werden.
- Aufgabe 5 gibt zusätzlich **10 Bonuspunkte**. Insgesamt können also **50 Punkte plus 10 Bonuspunkte** erreicht werden.

Aufgabe 1 ANWENDUNG – LPT**(5 + 2 + 3 Punkte)**

Wir betrachten das Scheduling-Problem $P \parallel C_{\max}$: n Jobs mit Bearbeitungszeiten $p_1, \dots, p_n \in \mathbb{Q}_{>0}$ sollen auf m identische Maschinen verteilt werden, sodass der Makespan $C_{\max}$ (die maximale Maschinenlast) minimal ist. Der Algorithmus LPT sortiert die Jobs zunächst *absteigend* nach Bearbeitungszeit und legt sie dann in dieser Reihenfolge nacheinander jeweils auf die aktuell am wenigsten belastete Maschine (LISTSCHEDULING); bei Gleichstand wird die Maschine mit dem kleinsten Index gewählt.

- a) Wenden Sie LPT auf die folgende Instanz mit $m = 3$ Maschinen an. Geben Sie die Sortierung, die Platzierungen mit dem jeweiligen Lastvektor (M_1, M_2, M_3) , den resultierenden Schedule (grafisch oder als Belegungsliste) sowie den Makespan an.

Job j	1	2	3	4	5	6	7
Bearbeitungszeit p_j	5	5	4	4	3	3	3

- b) Welche (multiplikative) approximative Güte hat der Algorithmus LPT in Abhängigkeit von der Maschinenzahl m ?
- c) Geben Sie eine Instanz mit $m = 2$ Maschinen an, bei der einfaches LISTSCHEDULING (ohne Vorsortierung, in der angegebenen Job-Reihenfolge) die Güte $2 - \frac{1}{m}$ erreicht. Geben Sie sowohl den von LISTSCHEDULING erzeugten Makespan als auch den optimalen Makespan an.

Aufgabe 2 VORLESUNGSBEWEIS – TRANSITIVITÄT

(10 Punkte)

Beweisen Sie den folgenden aus der Vorlesung bekannten Satz.

Seien L_1, L_2, L_3 Entscheidungsprobleme. Aus $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$ folgt $L_1 \leq_p L_3$.

Gehen Sie in Ihrem Beweis insbesondere auf die Konstruktion der Reduktion, die Korrektheit (beide Richtungen der Äquivalenz) und die polynomielle Laufzeit ein.

Aufgabe 3 NP-VOLLSTÄNDIGKEIT

(10 Punkte)

Wir betrachten die folgende Variante von SUBSETSUM.

Problem: SubsetSum-ohne-Dreierpotenzen

Eingabe: Zahlen $c_1, \dots, c_n \in \mathbb{N}_{>0}$ und ein Zielwert $K \in \mathbb{N}$, wobei keine der Größen c_i eine Dreierpotenz ist, d.h. $c_i \notin \{3^t \mid t \in \mathbb{N}_0\} = \{1, 3, 9, 27, \dots\}$ für alle $i \in [n]$.

Entscheide: Existiert eine Teilmenge $S \subseteq \{1, \dots, n\}$ mit $\sum_{i \in S} c_i = K$?

Zeigen Sie, dass SUBSETSUM-OHNE-DREIERPOTENZEN NP-vollständig ist. Sie dürfen verwenden, dass SUBSETSUM (ohne Einschränkung an die Itemgrößen) NP-vollständig ist.

Problem: k -IndependentSet

Eingabe: Ein ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}_{\geq 1}$.

Entscheide: Existiert $S \subseteq V$ mit $|S| \geq k$, sodass $\{v, u\} \notin E$ für alle $v, u \in S$ mit $v \neq u$?

Betrachten Sie folgende Reduktion von k -CLIQUE auf k -INDEPENDENTSET. Für eine Eingabe $(G = (V, E), k)$ bilde die Instanz $(G' = (V, \bar{E}), k')$ mit

$$\bar{E} = \{\{v, u\} \mid v, u \in V, v \neq u, \{v, u\} \notin E\} \quad (\text{Komplementgraph}), \quad k' = k.$$

- a) Geben Sie die Anzahl der Knoten $|V'|$ und die Anzahl der Kanten $|E'|$ für die resultierende k -INDEPENDENTSET-Instanz in Abhängigkeit von der originalen k -CLIQUE-Instanz an.
- b) Beweisen Sie Laufzeit-Lower-Bounds für k -INDEPENDENTSET bezüglich der Knotenzahl $|V'|$ und der Kantenanzahl $|E'|$ basierend auf der ETH und der oben beschriebenen Reduktion.

Hinweis: Unter der ETH ist k -CLIQUE weder in $2^{o(|V|)} \cdot |I|^{O(1)}$ noch in $2^{o(\sqrt{|E|})} \cdot |I|^{O(1)}$ lösbar.

Problem: SubsetSum

Eingabe: Zahlen $c_1, \dots, c_n$ und ein Zielwert K .

Entscheide: Existiert $S \subseteq \{1, \dots, n\}$ mit $\sum_{j \in S} c_j = K$?

Betrachten Sie folgende Reduktion von 3-EXACTCOVER (3-EC) auf SUBSETSUM. Bei 3-EXACTCOVER sind eine Grundmenge U und eine Familie $F = \{S_1, \dots, S_{|F|}\}$ von Dreiermengen $S_j \subseteq U$ gegeben; gesucht ist eine exakte Überdeckung von U . Die Reduktion erzeugt *pro Menge* $S_j \in F$ ein Item c_j , kodiert als Zahl zur Basis $(n+1)$, wobei $n = \text{Anzahl der Items} = |F|$ ist (Ziffer 1 an den drei Stellen der Elemente von S_j). Der Zielwert K ist die Zahl mit Ziffer 1 an allen $|U|$ Stellen.

- c) Geben Sie die Itemzahl n der resultierenden SUBSETSUM-Instanz in Abhängigkeit von der originalen 3-EXACTCOVER-Instanz an.
- d) Beweisen Sie einen Laufzeit-Lower-Bound für SUBSETSUM bezüglich der Itemzahl n basierend auf der ETH und der oben beschriebenen Reduktion.

Hinweis: Unter der ETH ist 3-EXACTCOVER weder in $2^{o(\sqrt{|U|})} \cdot |I|^{O(1)}$ noch in $2^{o(\sqrt[4]{|F|})} \cdot |I|^{O(1)}$ lösbar.

Aufgabe 5 APPROXIMATIVE ALGORITHMEN – 2ApproxVC (2 + 5 + 3 + (10) Punkte)

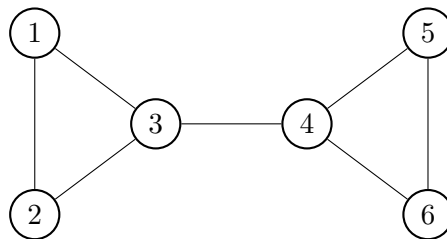
Beim Problem VERTEXCOVER ist ein ungerichteter Graph $G = (V, E)$ gegeben; gesucht ist eine möglichst kleine Knotenmenge $C \subseteq V$, die jede Kante abdeckt, d.h. $u \in C$ oder $v \in C$ für jede Kante $\{u, v\} \in E$. Wir betrachten den folgenden Approximationsalgorithmus, der die Kanten in einer gegebenen Reihenfolge durchläuft.

```
Algorithmus 2ApproxVC( $G = (V, E)$ )
1   $C \leftarrow \emptyset$ 
2  foreach Kante  $\{u, v\} \in E$  (in gegebener Reihenfolge) do
3    if  $u \notin C$  and  $v \notin C$  then
4       $C \leftarrow C \cup \{u, v\}$ 
5    fi
6  od
7  return  $C$ 
```

- a) Wenden Sie 2APPROXVC auf den folgenden Graphen G an. Die Kanten werden in dieser Reihenfolge durchlaufen:

$\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 6\}, \{5, 6\}$.

Geben Sie den Ablauf (welche Kanten führen zu einer Aufnahme?) sowie die Größe $|C|$ der berechneten Überdeckung an.



- b) Zeigen Sie, dass 2APPROXVC die (multiplikative) Güte 2 hat, also $|C| \leq 2 \cdot |C^*|$ für ein minimales Vertex Cover C^* gilt.
- c) Betrachten Sie den Kreis C_n auf n Knoten für *gerades* n . Wie groß ist ein minimales Vertex Cover von C_n , und welche Kantenreihenfolge führt zur schlechtesten Approximationsrate von 2APPROXVC? Geben Sie diese Rate an.
- d) **BONUS:** Geben Sie eine Instanzfamilie an, deren Approximationsrate von 2APPROXVC (bei ungünstigster Kantenreihenfolge) gegen 2 konvergiert, und weisen Sie dies nach.