Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 5

Vorlesung Theoretische Grundlagen der Informatik im WS 21/22

Ausgabe: 17. Dezember 2021
Abgabe: 14. Januar 2022 (digital im ILIAS)

Bitte bearbeiten Sie die Aufgaben **handschriftlich** und laden Sie eine eingescannte PDF-Version im Übungsmodul Ihrer ILIAS-Tutoriumsgruppe hoch! Beschriften Sie Ihren handschriftlichen Aufschrieb gut sichtbar mit Name und Matrikelnummer. Nicht handschriftliche oder unbeschriftete Abgaben werden nicht akzeptiert!

Aufgabe 1

(3 (+1) Punkte)

Die Mensa von Dr. Meta war ein solcher Erfolg, dass er sich dazu entschieden hat, zwei neue Linien L_1 und L_2 zu eröffnen. Dafür stellt er neue Minions M ein, die an den neuen Linien arbeiten sollen. Sie auf die Linien aufzuteilen ist aber gar nicht so einfach, denn es gibt einige Minions, die sich nicht leiden können. Dabei gibt die Funktion $f\colon M \to 2^M$ für jeden Minion $m \in M$ an, welche Minions $f(m)$ Minion m nicht leiden kann. Sich nicht leiden können beruht immer auf Gegenseitigkeit, also gilt für je zwei Minions $m, m' \in M$: $m \in f(m') \Leftrightarrow m' \in f(m)$. Dr. Meta möchte die Minions nun so auf die beiden Linien aufteilen, dass die Konflikte minimiert werden, also möglichst wenige Minions, die sich nicht leiden können, zusammenarbeiten müssen. Dabei müssen nicht gleich viele Minions an jeder Linie arbeiten.

Wir betrachten nun das äquivalente Maximierungsproblem. Dabei maximieren wir die Anzahl der Minions, die sich nicht leiden können und an verschiedenen Linien arbeiten.

Betrachten Sie den folgenden Greedy-Algorithmus $\mathcal{A}$: Die zwei Linien L_1, L_2 sind zunächst leer und wir gehen die Minions $m_1, \ldots, m_n$ in beliebiger Reihenfolge durch. Einen Minion $m_i \in M$ fügen wir dann zu der Linie L_k ($k \in \{1, 2\}$) hinzu, die zum Zeitpunkt der Betrachtung von m_i weniger Minions enthält, die m_i nicht leiden kann.

Zeigen Sie, dass $\mathcal{A}$ ein Approximationsalgorithmus mit relativer Güte 2 ist.

Hinweis: Verwenden Sie eine triviale obere Schranke, um den optimalen Wert abzuschätzen.

Bonusaufgabe: In einer früheren Version der Aufgabe haben wir das Minimierungsproblem betrachtet. Eine Minimierung der Konflikte innerhalb der Linien ist äquivalent dazu, die Konflikte zwischen den Linien zu maximieren. Warum eignet sich der angegebene Algorithmus trotzdem nicht für eine Approximation des Minimierungsproblems? Zeigen Sie, dass $\mathcal{A}$ keine 2-Approximation für das Minimierungsproblem ist.

Lösung:
Das Problem ist unter dem Namen MaxCut bekannt. Wir können das Problem als Graph $G = (V, E)$ modellieren, wobei jeder Knoten einen Minion repräsentiert (also $V := M$) und zwischen

zwei Knoten eine Kante existiert, wenn die zugehörigen Minions sich nicht leiden können (also $m_1 m_2 \in E \Leftrightarrow m_1 \in f(m_2)$). Gesucht ist nun eine Partionierung der Knotenmenge in zwei Mengen L_1 und L_2, die die Anzahl der Kanten zwischen L_1 und L_2 maximiert.

Der gegebene Algorithmus $\mathcal{A}$ ist polynomiell, da jeder Knoten nur einmal zu einer Menge hinzugefügt wird und für jeden Knoten alle Nachbarn betrachtet werden.

Es bleibt zu zeigen, dass $\mathcal{A}$ relative Güte 2 hat. Dafür schätzen wir zunächst den Wert einer optimalen Lösung: Die Anzahl der Kanten zwischen L_1 und L_2 kann höchstens die Anzahl aller Kanten sein, also ist $|\,\mathrm{OPT}\,| \leq |E|$.

Sei $E^i_{L_1L_2}$ die Menge der Kanten zwischen L_1 und L_2 nach dem Hinzufügen des i-ten Knotens und sei E_i die Menge der Kanten in dem von $m_1, \ldots, m_i$ induzierten Subgraphen. Wir zeigen durch Induktion, dass $|E^i_{L_1L_2}| \geq \frac{|E_i|}{2}$ für alle i gilt.

Für $i = 1$ ist $|E_i| = 0$. Der i-te Knoten m_i wird zu der Menge L_k hinzugefügt, die weniger Nachbarn von m_i enthält. Durch das Hinzufügen von m_i kommen genau $|E_i \setminus E_{i-1}|$ Kanten zu E_{i-1} hinzu. Von diesen Kanten verlaufen mindestens die Hälfte zwischen L_1 und L_2, da sonst m_i zur anderen Menge hinzugefügt worden wäre. Damit gilt

$$|E^i_{L_1L_2}| \geq |E^{i-1}_{L_1L_2}| + \frac{|E_i \setminus E_{i-1}|}{2} \overset{\text{Ind}}{\geq} \frac{|E_{i-1}|}{2} + \frac{|E_i \setminus E_{i-1}|}{2} = \frac{|E_i|}{2}$$

Damit gilt $|E^n_{L_1L_2}| \geq \frac{|E_n|}{2}$ und $\mathcal{A}$ ist ein Approximationsalgorithmus mit relativer Güte 2.

Bonusaufgabe: Der Unterschied der beiden Varianten besteht darin, was wir als Optimalwert auffassen. Falls es beispielsweise eine Lösung gibt, bei der es keine Konflikte innerhalb der Linien gibt (d.h. der Graph ist bipartit), so ist der Optimalwert des Minimierungsproblems 0, des Maximierungsproblems dagegen $|E(G)|$. Für das Maximierungsproblems genügt also eine Lösung mit $|E|/2$ Kanten zwischen den beiden Mengen, wohingegen die Lösung für das Minimierungsproblem 0, also optimal sein muss. Das Ziel der Approximation ist zwar das gleiche, die Bewertung, was „nah am Optimum“ bedeutet, unterscheidet sich aber. Um zu zeigen, dass $\mathcal{A}$ keine 2-Approximation für das Minimierungsproblem ist, betrachten wir den Pfad $P = (u, v, w)$. Da P bipartit ist, ist $\mathrm{OPT}(P) = 0$. $\mathcal{A}$ erhalte die Knoten nun in der Reihenfolge u, w, v. Für Knoten, die gleich viele Nachbarn in beiden Mengen haben, gibt uns $\mathcal{A}$ keine Garantie, also kann es passieren, dass u und w in verschiedenen Mengen landen. Dann ist die Lösung von $\mathcal{A}$ aber $1 > 2 \cdot 0$.

Aufgabe 2

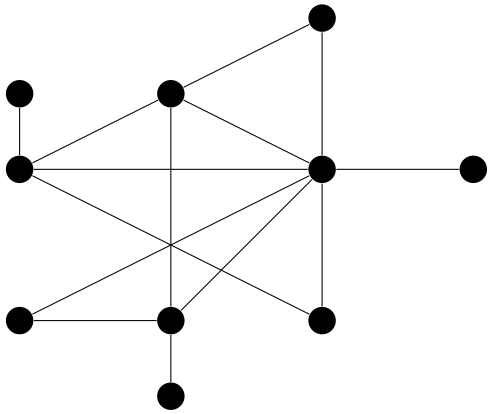
(1 (+ 3) + 3 + 1 = 5 (8) Punkte)

Für einen ungerichteten Graphen G ist ein Subgraph T ein Spannbaum von G, wenn T ein Baum ist und alle Knoten von G enthält.

Das Optimierungsproblem MIN-DEGREE-SPANNING-TREE ist wie folgt definiert: Gegeben ist ein zusammenhängender, ungerichteter Graph $G = (V, E)$. Gesucht ist ein Spannbaum T von G mit dem kleinsten Maximalgrad unter allen Spannbäumen von G. Der Maximalgrad eines Graphen G ist $\max\{\deg(v) \mid v \in V(G)\}$.

Betrachten Sie außerdem das $\mathcal{NP}$-vollständige Entscheidungsproblem HAMILTON-PFAD: Gegeben ein ungerichteter Graph G, gibt es einen Pfad P, der jeden Knoten in G genau einmal enthält?

(a) Geben Sie die Lösung des MIN-DEGREE-SPANNING-TREE für die folgende Instanz an und begründen Sie die Optimalität Ihrer Lösung:



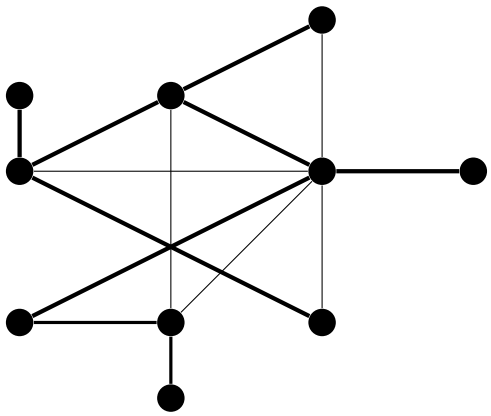
(b) *Bonusaufgabe:* Formulieren Sie das zu MIN-DEGREE-SPANNING-TREE zugehörige Entscheidungsproblem und zeigen Sie, dass es $\mathcal{NP}$-vollständig ist. Reduzieren Sie dazu von dem Problem HAMILTON-PFAD.

(c) Zeigen Sie: Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es keinen Approximationsalgorithmus für MIN-DEGREE-SPANNING-TREE mit relativer Güte $\alpha < \frac{3}{2}$.

(d) Es gibt aber einen Approximationsalgorithmus $\mathcal{A}$ für MIN-DEGREE-SPANNING-TREE mit absoluter Güte 1. Geben Sie $\mathcal{R}_{\mathcal{A}}^{\infty}$ an. Warum ist das kein Widerspruch zu c)? Überlegen Sie sich dazu, was der Optimalwert einer Instanz sein könnte, für die $\mathcal{A}$ mindestens relative Güte $\frac{3}{2}$ hat.

Lösung:

(a) Da es drei Knoten mit Grad 1 gibt, kann der Graph keinen Hamiltonpfad haben. Der Maximalgrad eines Spannbaums ist also mindestens 3. Der folgende Spannbaum zeigt, dass Maximalgrad 3 optimal ist.



(b) Das zugehörige Entscheidungsproblem DEGREE-CONSTRAINED-SPANNING-TREE lautet: Gegeben ein zusammenhängender, ungerichteter Graph G und eine natürliche Zahl $k \in \mathbb{N}$, gibt es einen Spannbaum T von G mit Maximalgrad höchstens k?

Das Problem liegt in $\mathcal{NP}$, da eine nicht-deterministische Turingmaschine für einen gegebenen Subgraphen T in polynomieller Zeit überprüfen kann, ob T ein Spannbaum ist und ob der Maximalgrad von T höchstens k ist.

Für eine Instanz $I = (G = (V, E))$ von HAMILTON-PFAD konstruieren wir eine Instanz $I' = (G', k)$ von DEGREE-CONSTRAINED-SPANNING-TREE. Dabei setzen wir $G' \coloneqq G$ und

$k := 2$. Ein Baum mit Maximalgrad 2 ist isomorph zu einem Pfad. Gibt es also einen Spannbaum in G' mit Maximalgrad 2, ist dieser gleichzeitig auch ein Hamilton-Pfad. Umgekehrt ist ein Hamilton-Pfad auch ein Spannbaum mit Maximalgrad 2.

Da die Transformation polynomial ist, ist DEGREE-CONSTRAINED-SPANNING-TREE $\mathcal{NP}$-vollständig.

(c) Wir nehmen an, es gäbe einen Approximationsalgorithmus $\mathcal{A}$ mit relativer Güte $\alpha < \frac{3}{2}$. Wir zeigen, dass wir damit eine Instanz von HAMILTON-PFAD in polynomieller Zeit entscheiden können.

Sei Graph G eine Instanz von HAMILTON-PFAD. Wir führen den Approximationsalgorithmus $\mathcal{A}$ auf G aus. Gibt es in G einen Hamiltonpfad, dann gibt es einen Spannbaum mit Maximalgrad 2 und es gilt $\mathcal{A}(G) \leq \alpha \cdot \mathrm{OPT}(I) = \alpha \cdot 2 < 3$. Also gibt $\mathcal{A}$ einen Wert kleiner als 3 aus. Gibt es in G keinen Hamiltonpfad, ist der Maximalgrad jedes Spannbaums mindestens 3 und $\mathcal{A}$ gibt mindestens 3 aus.

Da HAMILTON-PFAD ein $\mathcal{NP}$-vollständiges Problem ist, kann unter der Annahme von $\mathcal{P} \neq \mathcal{NP}$ ein solcher Approximationsalgorithmus nicht existieren.

(d) Da $\mathcal{A}(I) \leq \mathrm{OPT}(I) + 1$ gilt für alle Instanzen I, ist $\mathcal{R}_{\mathcal{A}}^{\infty} = 1$. Ein solcher Algorithmus $\mathcal{A}$ kann aber trotzdem eine relative Güte $\alpha \geq \frac{3}{2}$ haben. Für eine Instanz I mit $\mathrm{OPT}(I) = 2$ gibt $\mathcal{A}$ den Wert $\mathcal{A}(I) = \mathrm{OPT}(I) + 1 = 3$ aus, also gilt $\mathcal{R}_{\mathcal{A}}(I) = \frac{\mathcal{A}(I)}{\mathrm{OPT}(I)} = \frac{3}{2}$. Damit ist das nicht hinreichend für einen Widerspruch zu c).

(Für $\mathrm{OPT}(I) > 2$ gilt $\mathcal{R}_{\mathcal{A}}(I) < \frac{3}{2}$. Für die relative Güte α des Algorithmus insgesamt ist aber nur der schlechteste Fall entscheidend.)

Aufgabe 3

(3 + 1 = 4 Punkte)

Sei Π ein Minimierungsproblem mit ganzzahliger Optimierungsfunktion. Bezeichne das Entscheidungsproblem, ob $\mathrm{OPT}_{\Pi}(I) \leq K$ für eine gegebene Instanz I, als Π_K. Sei Π_K $\mathcal{NP}$-schwer für ein $K \in \mathbb{N}$.

(a) Zeigen Sie, dass unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ kein Approximationsalgorithmus $\mathcal{A}$ für Π mit relativer Güte $\mathcal{R}_{\mathcal{A}} < 1 + \frac{1}{K}$ existiert.

(b) Folgern Sie, dass es unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ kein PTAS für Π geben kann.

Lösung:

(a) Wir nehmen an, dass ein Approximationsalgorithmus $\mathcal{A}$ mit relativer Güte $\mathcal{R}_{\mathcal{A}} < 1 + \frac{1}{K}$ für Π existiert. Wir verwenden $\mathcal{A}$, um Π_K zu entscheiden. Sei also I eine Instanz von Π_K.

Ist $\mathrm{OPT}_{\Pi}(I) \leq K$, gilt:

$$\mathcal{A}(I) < \left(1 + \frac{1}{K}\right) \mathrm{OPT}_{\Pi}(I) \leq \left(1 + \frac{1}{K}\right) \cdot K = K + 1$$

Da die Optimierungsfunktion von Π ganzzahlig ist, gilt somit $\mathcal{A}(I) \leq K$.

Ist $\mathrm{OPT}_{\Pi}(I) > K$, gilt auch $\mathcal{A}(I) > K$, da Π ein Minimierungsproblem ist und damit $\mathcal{A}(I) \geq K + 1$.

Damit können wir also entscheiden, ob $\mathrm{OPT}_{\Pi} \leq K$ gilt. Da $\mathcal{A}$ ein polynomieller Algorithmus ist und Π_K $\mathcal{NP}$-schwer, ist $\mathcal{P} = \mathcal{NP}$.

(b) Gäbe es für Π ein PTAS, gäbe es per Definition für jedes $\epsilon > 0$ einen Approximationsalgorithmus A_ϵ mit relativer Güte $\leq 1 + \epsilon$, insbesondere also auch für $\epsilon := \frac{1}{K}$. Dieser kann aber nach Teilaufgabe a) nicht existieren, wenn $\mathcal{P} \neq \mathcal{NP}$ gilt.

Aufgabe 4 (1 + 1 + 2 + 1 + 1 = 6 Punkte)

Das Maximierungsproblem MaxCover ist wie folgt definiert: Gegeben seien ein $k \in \mathbb{N}$, ein Universum U mit $|U| = n$ Elementen und eine Menge von $m \geq k$ Teilmengen $\mathcal{S} = \{S_1, \dots S_m\}$ mit $S_i \subseteq U$. Gesucht ist eine Teilmenge $\mathcal{X} \subseteq \mathcal{S}$ mit $|\mathcal{X}| = k$, sodass $\bigcup_{S_i \in \mathcal{X}} S_i$ maximale Kardinalität hat. D.h. wir wollen k Mengen aus $\mathcal{S}$ so wählen, dass diese Mengen die maximale Anzahl an Elementen in U abdecken.

(a) Betrachten Sie die folgende MaxCover-Instanz mit $k = 3$:

$$U = \{0, 1, 2, 3, 4, 5, 6, 7, 8\},\ \mathcal{S} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$$

mit $S_1 = \{2, 3, 6\}$, $S_2 = \{1, 2, 3, 4\}$, $S_3 = \{1, 4, 8\}$, $S_4 = \{0, 4, 5\}$, $S_5 = \{0, 1, 2\}$ und $S_6 = \{5, 6\}$. Geben Sie k Mengen in $\mathcal{S}$ an, die ausgewählt werden, um eine optimale Lösung zu erreichen.

Betrachten Sie den folgenden Greedy-Algorithmus $\mathcal{A}$, der immer die Menge auswählt, die maximal viele der noch nicht abgedeckten Elemente abdeckt.

Algorithmus 1: MaxCover-Approximation $\mathcal{A}$

```
X ← ∅                                          // Lösung
Y ← ∅                                  // Bereits abgedeckte Elemente
for i ← 1, ..., k do
    Wähle X_i ∈ S, dass |X_i ∩ (U \ Y)| maximiert wird
    Y ← Y ∪ X_i
    X ← X ∪ {X_i}
```

(b) Wenden Sie $\mathcal{A}$ auf der Instanz aus a) an. Wählen Sie bei mehreren Möglichkeiten die Menge mit dem kleinsten Index. Geben Sie die Reihenfolge an, in der die Mengen gewählt werden.

Im Folgenden zeigen wir, dass $\mathcal{A}$ ein relativer Approximationsalgorithmus mit relativer Güte $(1 - \frac{1}{e})^{-1}$ ist. Sei Y_{OPT} die Menge der Elemente, die durch eine optimale Lösung abgedeckt wird. Sei $\Delta_i = |Y_{\text{OPT}}| - |Y_i|$, wobei Y_i die Menge der überdeckten Elemente ist nach Auswahl der i-ten Menge von $\mathcal{A}$.

(c) Zeigen Sie, dass in jedem Schritt für die Anzahl der neu abgedeckten Elemente

$$|Y_i \setminus Y_{i-1}| \geq \frac{\Delta_{i-1}}{k}$$

gilt.

(d) Zeigen Sie mit Hilfe von vollständiger Induktion, dass

$$\Delta_i \leq \left(1 - \frac{1}{k}\right)^i \cdot |Y_{\text{OPT}}|$$

gilt.

(e) Zeigen Sie, dass $\mathcal{A}$ ein Approximationsalgorithmus mit relativer Güte $(1 - \frac{1}{e})^{-1}$ ist.
Hinweis: Es gilt $\left(1 - \frac{1}{n}\right)^n < \frac{1}{e}$ *für alle* $n \in \mathbb{N}$.

Lösung:

(a) Die Mengen S_1, S_3 und S_4 decken acht Elemente $\{0, 1, 2, 3, 4, 5, 6, 8\}$ ab. Da 7 in keiner Menge in $\mathcal{S}$ vorkommt, ist das eine optimale Lösung.

(b) $\mathcal{A}$ wählt die Mengen in der folgenden Reihenfolge: S_2, S_4, S_1. Diese decken sieben Elemente $\{0, 1, 2, 3, 4, 5, 6\}$ ab.

(c) Vor der Auswahl der i-ten Menge gibt es noch mindestens Δ_{i-1} Elemente in $Y_{\text{OPT}} \setminus Y_{i-1}$. Das sind die Elemente, die durch optimale Lösung abgedeckt werden, aber nach dem $(i-1)$-ten Schritt noch nicht durch $\mathcal{A}$. Da eine optimale Lösung die Elemente in Y_{OPT} mit k Mengen abdecken kann, kann sie insbesondere auch die Elemente in $Y_{\text{OPT}} \setminus Y_{i-1}$ mit k Mengen abdecken. Dann muss es eine Menge geben, die mindestens einen Anteil von $\frac{1}{k}$ der Elemente in $Y_{\text{OPT}} \setminus Y_{i-1}$ abdeckt. Da $\mathcal{A}$ die Menge auswählt, die die meisten verbleibenden Elemente abdeckt, werden durch die Auswahl der i-ten Menge auch mindestens $\frac{\Delta_{i-1}}{k}$ bisher noch nicht abgedeckte Elemente abgedeckt.

(d) Für $i = 0$ wird noch keine Menge von $\mathcal{A}$ abgedeckt, also ist $|Y_i| = 0$ und $\Delta_i = |Y_{\text{OPT}}|$.

Für $i > 0$ nehmen wir an, dass die Behauptung für $i - 1$ gilt, also $\Delta_{i-1} \leq \left(1 - \frac{1}{k}\right)^{i-1} \cdot |Y_{\text{OPT}}|$. Dann gilt:

$$
\begin{aligned}
\Delta_i &= \Delta_{i-1} - |Y_i \setminus Y_{i-1}| && \text{neu abgedeckte Elemente verringern Differenz } |Y_{\text{OPT}}| - |Y_i| \\
&\leq \Delta_{i-1} - \frac{\Delta_{i-1}}{k} && \text{wegen c)} \\
&= \Delta_{i-1} \left(1 - \frac{1}{k}\right) \\
&\leq \left(1 - \frac{1}{k}\right)^i \cdot |Y_{\text{OPT}}| && \text{IV}
\end{aligned}
$$

Durch Induktion folgt die Behauptung.

(e) Nach Auswahl der k-ten Menge gilt

$$|Y_{\text{OPT}}| - |Y_k| = \Delta_k \leq \left(1 - \frac{1}{k}\right)^k \cdot |Y_{\text{OPT}}| < \frac{1}{e} \cdot |Y_{\text{OPT}}|$$

, also

$$|Y_k| \geq \left(1 - \frac{1}{e}\right) |Y_{\text{OPT}}|$$

Dabei ist $|Y_k|$ die Anzahl der Elemente, die durch $\mathcal{A}$ abgedeckt werden. Da der Algorithmus polynomielle Laufzeit hat, ist er ein Approximationsalgorithmus mit relativer Güte $(1 - \frac{1}{e})^{-1}$.

Aufgabe 5 (2 + 4 = 6 Punkte)

Das Maximierungsproblem d-WeightedIndSet ist wie folgt definiert: Gegeben sei ein Graph $G = (V, E)$ mit Maximalgrad d und eine Knotengewichtsfunktion $w \colon V \to \mathbb{N}$. Gesucht ist eine unabhängige Menge $S \subseteq V$ mit maximalem Gewicht $w(S) \coloneqq \sum_{v \in S} w(v)$.

Betrachten Sie den folgenden Greedy-Algorithmus $\mathcal{A}$. Dieser beginnt mit $S = \emptyset$ und fügt immer den Knoten mit maximalem Gewicht zu S hinzu. Nachdem ein Knoten v zu S hinzugefügt wird, werden v, alle Nachbarn $N(v)$ von v und alle inzidenten Kanten aus dem Graphen gelöscht. Das wird so lange wiederholt, bis der Graph leer ist.

(a) Zeigen Sie, dass $\mathcal{A}$ tatsächlich eine unabhängige Menge ausgibt und in polynomieller Zeit läuft.

(b) Zeigen Sie, dass $\mathcal{A}$ ein Approximationsalgorithmus ist mit relativer Güte d ist.

Lösung:

(a) Angenommen, die ausgegebene Menge S ist keine unabhängige Menge, dann gibt es zwei Knoten $u, v \in S$ mit $uv \in E$. Entweder u oder v wurde von $\mathcal{A}$ zuerst ausgewählt, o.B.d.A. sei u zuerst gewählt worden. Da aber $\mathcal{A}$ danach alle Nachbarn von u (insbesondere auch v) aus dem Graphen löscht, kann v nicht mehr von $\mathcal{A}$ gewählt werden. Die Knoten u und v können also nicht gleichzeitig in S sein.

In jedem Schritt einen Knoten mit maximalem Gewicht finden und die Nachbarn und inzidente Kanten löschen geht in polynomieller Zeit. Da höchstens $|V|$ Knoten zu S hinzugefügt werden können, hat $\mathcal{A}$ auch insgesamt polynomielle Laufzeit.

(b) Sei $I = (G = (V, E), w)$ eine Instanz von d-WeightedIndSet und sei S^* eine optimale Lösung von I. Wir betrachten die unabhängige Menge $S_\mathcal{A}$, die von $\mathcal{A}$ ausgegeben wird. Der Wert der Lösung ist dann $\sum_{v \in S_\mathcal{A}} w(v)$.

Wir schätzen $w(S^*)$ nach oben ab. Dazu definieren wir für jedes $v \in S$ die folgende Menge:

$$\tilde{N}(v) = \{u \in S^* \mid u \text{ wurde durch Wahl von } v \text{ gelöscht}\}$$

Da jeder Knoten in G irgendwann gelöscht wird, damit der Algorithmus terminiert, wird auch jeder Knoten in S^* irgendwann gelöscht. Wir weisen also jedem Knoten $u \in S^*$ genau einen Knoten $v \in S_\mathcal{A}$ zu. Es gilt also $\bigcup_{v \in S_\mathcal{A}} \tilde{N}(v) = S^*$.

Weniger formal ist die Wahl von $v \in S_\mathcal{A}$ also Schuld daran, dass wir die Knoten in $\tilde{N}(v)$ (die in der optimalen Lösung S^* enthalten sind) danach nicht mehr wählen können. Es bleibt zu zeigen, dass es „nicht so schlimm“ ist, wenn wir v statt $\tilde{N}(v)$ wählen.

Im Folgenden wollen wir zeigen, dass für alle $v \in S_\mathcal{A}$ gilt: $w(\tilde{N}(v)) \leq d \cdot w(v)$.

Es gilt entweder $\tilde{N}(v) = \{v\}$ oder $\tilde{N}(v) \subseteq N(v)$, da nur Nachbarn von v und v selbst bei der Wahl von v gelöscht werden. Außerdem können nicht gleichzeitig v und ein Nachbar von v in S^* sein, da S^* sonst keine unabhängige Menge bilden würde.

Wir unterscheiden also die beiden Fälle:

(1) $\tilde{N}(v) = \{v\}$
In diesem Fall ist offensichtlich $w(\tilde{N}(v)) = w(v) \leq d \cdot w(v)$

(2) $\tilde{N}(v) \subseteq N(v)$
Da die Knoten in $N(v)$ vor der Wahl von v noch nicht gelöscht und insbesondere nicht gewählt worden sind und $\mathcal{A}$ immer den Knoten mit maximalem Gewicht wählt, gilt für alle $u \in \tilde{N}(v)$: $w(u) \leq w(v)$. Außerdem ist $|\tilde{N}(v)| \leq d$, da G Maximalgrad d hat.

In beiden Fällen gilt also $w(\tilde{N}(v)) \leq d \cdot w(v)$.

Dann ist

$$w(S^*) = w(\bigcup_{v \in S_\mathcal{A}} \tilde{N}(v)) = \sum_{v \in S_\mathcal{A}} w(\tilde{N}(v)) \leq \sum_{v \in S_\mathcal{A}} d \cdot w(v) = d \cdot \sum_{v \in S_\mathcal{A}} w(v) = d \cdot w(S_\mathcal{A})$$

Damit ist $\mathcal{A}$ also ein Approximationsalgorithmus mit relativer Güte d.

Aufgabe 6 (1 + 1 + 2 + 3 = 7 Punkte)

Das Minimierungsproblem DreiecksfreierGraph ist wie folgt definiert: Gegeben ist ein Graph $G = (V, E)$ mit $|V| = n$ und $|E| = m$. Gesucht ist eine kardinalitätsminimale Kantenmenge $E' \subseteq E$, sodass durch Löschen von E' aus Graph G ein Graph $G' = (V, E \setminus E')$ entsteht, der dreiecksfrei ist. Ein Graph ist dreiecksfrei, wenn es keine drei Knoten $u, v, w \in V$ mit $u \neq v$, $v \neq w$, $u \neq w$ gibt, die paarweise zueinander adjazent sind, also ein Dreieck bilden.

Es kann gezeigt werden, dass die Entscheidungsvariante von DreiecksfreierGraph $\mathcal{NP}$-vollständig ist.

In der Vorlesung haben wir gesehen, dass sich jedes $\mathcal{NP}$-vollständige Problem als *ganzzahliges lineares Programm* (engl. *Integer Linear Program*, ILP) darstellen lässt. Um DreiecksfreierGraph als ILP darzustellen, führen wir für jede Kante $e \in E$ eine Variable $x_e \in \{0, 1\}$ ein. Dabei soll $x_e = 1$ bedeuten, dass wir e aus G löschen (also $e \in E'$), und $x_e = 0$, dass wir e nicht löschen.

Im Folgenden ist ein noch unvollständiges ILP für DreiecksfreierGraph gegeben, das wir mit DG-$\mathbb{N}$ bezeichnen:

Gegeben:
DreiecksfreierGraph-Instanz $I = (G = (V, E))$
Variablen:
Für jede Kante $e \in E$ eine Variable x_e
Nebenbedingungen:
(I) Für jede Kante $e \in E$: $x_e \in \{0, 1\}$
(II) [ ]
Zielfunktion:
Minimiere [ ]

(a) Vervollständigen Sie die zweite Nebenbedingung, die garantiert, dass $G' = (V, E \setminus E')$ dreiecksfrei ist, und die Minimierungsfunktion.

Eine gängige Methode, ILPs zu approximieren, ist die *LP-Relaxierung*. Dabei wird die Beschränkung aufgehoben, dass die Variablen ganzzahlige Werte haben müssen. Das dadurch entstehende *lineare Programm* (LP) lässt sich in Polynomialzeit lösen. Im Fall von SetCover erhalten wir das Problem DG-$\mathbb{R}$. Der einzige Unterschied gegenüber DG-$\mathbb{N}$ ist die Nebenbedingung (I). Die Variablen x_e dürfen nun beliebige reelle Zahlen aus dem Intervall $[0, 1]$ sein. Dementsprechend ist auch die Zielfunktion reellwertig.

Sei im Folgenden $\mathrm{OPT}_{\mathbb{R}}(I)$ der Wert von L für eine optimale Lösung von DG-$\mathbb{R}$ und $\mathrm{OPT}_{\mathbb{N}}(I)$ der Wert von L einer optimalen Lösung von DG-$\mathbb{N}$.

(b) Zeigen Sie, dass für jede DreiecksfreierGraph-Instanz $I = (G = (V, E))$ gilt: $\mathrm{OPT}_{\mathbb{R}}(I) \leq \mathrm{OPT}_{\mathbb{N}}(I)$.

(c) Geben Sie eine DreiecksfreierGraph-Instanz $I = (G = (V, E))$ an, für die $\mathrm{OPT}_{\mathbb{R}}(I) < \mathrm{OPT}_{\mathbb{N}}(I)$ gilt. Geben Sie jeweils eine optimale Lösung an.

Wir betrachten nun folgenden Algorithmus $\mathcal{A}$, der eine (nicht notwendigerweise optimale) Lösung für DG-$\mathbb{N}$ berechnet:

1. Berechne eine optimale Lösung $X = (x_{e_1}, \ldots, x_{e_m})$ mit $x_{e_i} \in [0, 1]$ für DG-$\mathbb{R}$.
2. Generiere eine Lösung $X' = (x'_{e_1}, \ldots, x'_{e_m})$ mit $x'_{e_i} \in \{0, 1\}$ für DG-$\mathbb{N}$:
   Wähle jedes x'_e wie folgt:
$$x'_S = \begin{cases} 1 & \text{falls } x_S \geq \frac{1}{3} \\ 0 & \text{sonst.} \end{cases}$$

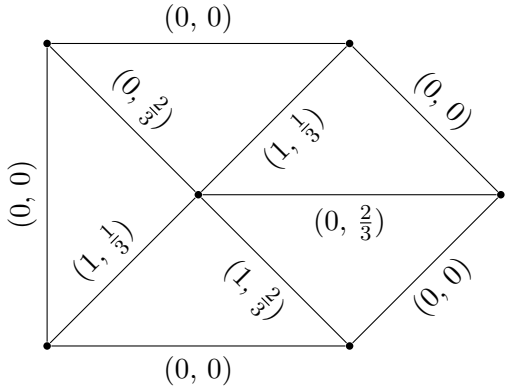
(d) Zeigen Sie, dass $\mathcal{A}$ ein 3-Approximationsalgorithmus für DG-$\mathbb{N}$ ist.

Lösung:

(a) Für jedes Dreieck $\{u, v, w\} \subseteq V^3$ in G muss mindestens eine Kante gelöscht werden, also muss uv, vw oder wu in E' sein, also muss für alle Dreiecke in G mit Knoten u, v, w gelten: $x_{uv} + x_{vw} + x_{wu} \geq 1$ Die Anzahl der gewählten Kanten soll minimal sein, also soll $\sum_{e \in E} x_e$ minimiert werden.

(b) Jede Lösung von DG-$\mathbb{N}$ erfüllt die Nebenbedingungen von SC-$\mathbb{R}$ und ist somit auch eine Lösung von DG-$\mathbb{R}$. Es kann also nicht $\mathrm{OPT}_{\mathbb{R}}(I) > \mathrm{OPT}_{\mathbb{N}}(I)$ gelten, weil $\mathrm{OPT}_{\mathbb{R}}(I)$ dann nicht optimal wäre.

(c) Betrachten Sie den folgenden Graphen:



Die erste Komponente gibt eine optimale Lösung X für DG-$\mathbb{N}$ und die zweite Komponente eine optimale Lösung X' für DG-$\mathbb{R}$ an. Dabei ist $\sum_{e \in E} x_e = 3 > \frac{8}{3} = \sum_{e \in E} x'_e$.

(d) Zunächst muss gezeigt werden, dass $\mathcal{A}$ tatsächlich eine gültige Lösung für DG-$\mathbb{N}$ berechnet. Dazu nehmen wir an, dass $\mathcal{A}$ keine gültige Lösung ausgibt. Dann existiert ein Dreieck $\{u, v, w\}$ in G mit $x'_{uv} + x'_{vw} + x'_{wu} = 0$, d.h. keine der drei Kanten wird gelöscht. Also gilt für alle drei Kanten in der Lösung von DG-$\mathbb{R}$: $x_{uv}, x_{vw}, x_{wu} < \frac{1}{3}$. Dann ist aber $x_{uv} + x_{vw} + x_{wu} < \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$. Das heißt, X ist keine gültige Lösung von DG-$\mathbb{R}$, was ein Widerspruch ist.

Außerdem gilt

$$|\mathcal{A}(I)| = \sum_{e \in E} x_e = \sum_{\substack{x_e = 1 \\ e \in E}} 1 \overset{x'_e \geq \frac{1}{3}}{\leq} \sum_{\substack{x_e = 1 \\ e \in E}} 3 \cdot x'_e \leq 3 \cdot \sum_{e \in E} x'_e = 3 \cdot \mathrm{OPT}_{\mathbb{R}}(I) \overset{a)}{\leq} 3 \cdot \mathrm{OPT}_{\mathbb{N}}(I) \,,$$

weshalb $\mathcal{A}$ relative Güte 3 hat.

Da die Anzahl der Dreiecke in einem Graphen durch n^3 beschränkt ist, ist auch die Anzahl der Nebenbedingung und damit auch die LP-Instanz polynomiell. Diese kann in polynomieller Zeit optimal gelöst werden und $\mathcal{A}$ kann die zugehörige ganzzahlige Lösung offensichtlich in Linearzeit berechnen. Insgesamt ist $\mathcal{A}$ also ein Approximationsalgorithmus mit relativer Güte 3.