Algorithmen und Datenstrukturen 2 **SoSe 2020**
Prof. Dr. Sándor Fekete **Abgabe:** 16.07.20
Arne Schmidt **Rückgabe:** ab 22.07.20

Übungsblatt 5

Abgabe der Lösungen muss bis zum 16.07.20 um 23:59 Uhr erfolgen. Lösungen müssen per Mail mit einer pdf-Datei (Name der Datei „`blatt_[nr]_[grp]_[team].pdf`“) an den jeweiligen Tutor geschickt werden. Die Email-Adressen sind unter `https://www.ibr.cs.tu-bs.de/alg/index.html` zu finden.

Hausaufgabe 1 (Graphenprobleme): **(4+1+3+3+3+3 Punkte)**
In dieser Aufgabe betrachten wir die folgenden zwei Entscheidungsprobleme:

1. Vertex Cover:

   Gegeben: Ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$

   Frage: Existiert eine Menge $VC \subseteq V$ mit $|VC| \leq k$, sodass für jede Kante $\{u, v\} \in E$ gilt: $|\{u, v\} \cap VC| \geq 1$ (Mindestens ein Endknoten ist in VC)?

2. Independent Set:

   Gegeben: Ein Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$

   Frage: Existiert eine Menge $IS \subseteq V$ mit $|IS| \geq k$, sodass für jede Kante $\{u, v\} \in E$ gilt: $|\{u, v\} \cap IS| \leq 1$ (Höchstens ein Endknoten ist in IS)?

a) Betrachte den Graphen G aus Abbildung 1.[1] Gib ein Vertex Cover der Größe $k = 4$ an. Gibt es ein kleineres Vertex Cover? Begründe deine Antwort.

b) Betrachte weiterhin den Graphen G aus Abbildung 1. Gib ein Independent Set der Größe $k = 2$ an.

c) Sei $H = (V, E)$ ein Graph.

   Zeige: VC ist genau dann ein Vertex Cover in H, wenn $IS = V \setminus VC$ ein Independent Set in H ist.

d) In der Übung #4 vom 17.06.20 haben wir eine 2-Approximation für die Optimierungsvariante von Vertex Cover betrachtet.

   (i) Zeige: Der Approximationsfaktor dieses Algorithmus ist bestmöglich.

   (ii) Warum liefert der folgende Algorithmus keine Approximation für Independent Set? Berechne ein Vertex Cover VC mit der 2-Approximation und gebe $IS = V \setminus VC$ zurück.

   (iii) Zeige: Es gibt beliebig große Instanzen, auf denen der Algorithmus aus (ii) ein größtmögliches Independent Set zurückgibt.

[1] Den Graphen gibt es als separate svg-, pdf- und ipe-Datei zum Download auf https://www.ibr.cs.tu-bs.de/courses/ss20/aud2/webpages/index.html#ueb

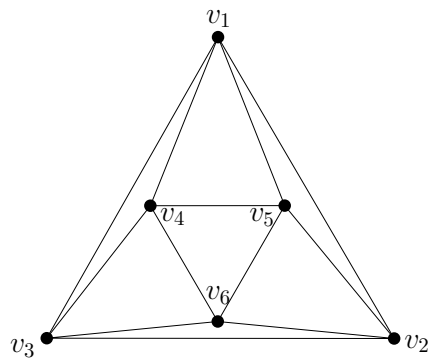

Abbildung 1: Der Oktaeder-Graph G.

Hausaufgabe 2 (Set Cover): **(4+4+2+2+1 Punkte)**
Wir betrachten das folgende Problem.

Gegeben: Eine endliche Menge U, eine Familie $\mathcal{F}$ von Teilmengen von U und eine Zahl $k \in \mathbb{N}$.

Gesucht: Ein *Set Cover* von $(U, \mathcal{F})$ der Größe höchstens k. Ein Set Cover ist eine Teilfamilie $F \subseteq \mathcal{F}$, die U überdeckt, d.h. für jedes Element $u \in U$ gibt es eine Menge $M \in F$ mit $u \in M$. Die Größe eines Set Covers ist die Anzahl an Mengen in F, d.h. $|F|$.

Wir nehmen an, dass jedes Element aus U in einer Menge aus $\mathcal{F}$ vorkommt.

Als Beispiel betrachte $U := \{1, 2, 3, 4, 5, 6\}$ und $\mathcal{F} := \{\{1, 2\}, \{1, 4\}, \{3, 6\}, \{2, 3, 4\}, \{1, 2, 5\}, \{2, 3\}\}$, sowie $k = 3$. $F := \{\{1, 4\}, \{3, 6\}, \{1, 2, 5\}\}$ ist ein Set Cover von $(U, \mathcal{F})$. Es kann schnell überprüft werden, dass es für $k = 2$ kein Set Cover gibt. Eine graphische Darstellung dieser Instanz ist in Abbildung 2 abgebildet.

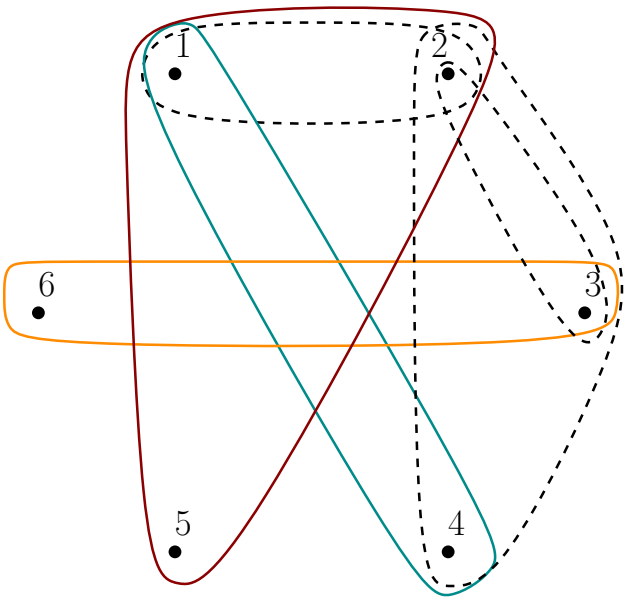

Abbildung 2: Beispiel einer Instanz von Set Cover. Punkte entsprechen den Elementen in U, Kreise entsprechen den Mengen in $\mathcal{F}$. Die farbige Auswahl entspricht einem Set Cover.

a) Zeige, dass Set Cover NP-schwer ist. (Hinweis: Nutze Vertex Cover.)

Da Set Cover also NP-schwer ist, bietet es sich an, Approximationsalgorithmen zu betrachten, um das kleinste k zu finden. Der folgende Algorithmus (GreedySC) versucht ein möglichst kleines Set Cover zu bestimmen.

```
function GreedySC(U, F)
    C := ∅                                  ▷ Menge der bereits überdeckten Elemente
    C̄ := U                                  ▷ Menge der noch zu überdeckenen Elemente
    SC := ∅                                 ▷ Set Cover

    while C ≠ U do
        S := argmax_{M∈F} |M ∩ C̄|           ▷ Menge mit den meisten nicht überdeckten Elementen

        α := 1/|S ∩ C̄|
        For each s ∈ S ∩ C̄ do kosten(s) := α

        C := C ∪ S
        C̄ := C̄ \ S
        SC := SC ∪ {S}

    return SC
```

Algorithmus 1: Algorithmus GreedySC zum Finden eines Set Covers. In jeder Iteration wird diejenige Menge aufgenommen, die die meisten nicht überdeckten Elemente besitzt.

b) Wende GreedySC auf folgende Instanz an: $U := \{1, \ldots, 10\}$, $\mathcal{F} := \{F_1, \ldots, F_5\}$ mit $F_1 = \{1, 2, 3, 7, 9\}$, $F_2 = \{4, 5, 6, 8, 10\}$, $F_3 = \{1, 2, 3, 4, 5, 6\}$, $F_4 = \{7, 8\}$ und $F_5 = \{9, 10\}$.
Gib dabei nach jeder Iteration der while-Schleife S, α, C sowie $\overline{C}$ an. Führe nach Ablauf des Algorithmus für alle $s \in U$ die kosten(s) auf.

c) Zeige: In jeder Iteration von GreedySC gilt $\alpha \leq \mathrm{OPT}/|\overline{C}|$.
(Hinweis: In jeder Iteration kann $\overline{C}$ mit $\leq \mathrm{OPT}$ Elementen überdeckt werden.)

d) Sei $U = \{e_1, \ldots, e_n\}$, wobei die Elemente in der Reihenfolge aufgelistet seien, in der sie GreedySC in C einfügt. Folgere mit Hilfe von c): Für alle $k \leq n$ gilt $\mathrm{kosten}(e_k) \leq \frac{\mathrm{OPT}}{n-k+1}$.

e) Folgere: GreedySC liefert eine H_n-Approximation von Set Cover, wobei $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$.
(Hinweis: Verteile die Kosten des gefundenen Set Covers mit Hilfe von kosten(e_i) auf U.)