Übungsblatt 10

Til Mohr, 405959
Andrés Montoya, 405409
Marc Ludevid Wulf, 405401

Aufgabe 4

Aufgabe 5

SetCover ist in NP. Hierzu können wir einen Verifizierer verwenden, welcher als Zertifikat eine Menge C bekommt. Der Verifizierer überprüft nun, ob $|C| \leq k$ gilt, $C \subseteq S$ gilt, und anschließend, ob $U = \bigcup_{V \in C} V$ gilt. Offensichtlich läuft dieser Verifizierer in polynomieller Zeit.

SetCover ist NP-schwer. Hierzu reduzieren wir VertexCover auf SetCover. Aus einer Instanz aus VertexCover können wir eine Instanz aus SetCover wie folgt konstruieren:
$G = (V, E)$ ist ein ungerichteter Graph mit n Knoten. Wir setzen $U = E$, $S = \mathrm{Pot}(U)$ und übernehmen das k aus der VertexCover-Instanz in das k der SetCover-Instanz. Unser C besteht nun aus Mengen C_i, $i \in [n]$, wobei jedes C_i alle Kanten aus E enthält, die einen Endknoten am Knoten i haben. Also hat C eine Größe von n. Diese Konstruktion können wir offensichtlich in polynomieller Zeit erstellen.

VertexCover $\Rightarrow$ SetCover:
G hat einen *VertexCover* U' von Größe $\leq k$. Nun führen wir unsere Konstruktion durch und bekommen die Menge C der Größe n. Anschließend können wir noch jedes C_i aus C löschen, wenn $i \notin U'$ ist. Damit hat C danach die Größe $|U'|$, insbesondere also $\leq k$. Alle verbleibenden C_i in C decken dann noch jedes Element in U ab: Alle Elemente in U sind ja die Kanten e in G. Da U' ein *VertexCover* ist, ist mind. einer der beiden Endpunkte von e in U'. Da in C auf jeden Fall die zu diesem Endpunkt zugeordnete Menge C_i enthält, die selber auch e enthält, gilt: $U = \bigcup_{C_i \in C} C_i$

SetCover $\Rightarrow$ VertexCover:
Sei C ein *SetCover* der Größe k in unserem Konstrukt. Erstelle nun aus C (bzw. den C_i) eine Menge an Knoten U'. $i \in [n]$ ist genau dann in S, falls C_i in C ist. Damit gilt $|U'| = |C| = k \leq k$. Nun betrachte jede Kante e aus U. Da C das *SetCover* von U ist, gibt es mind. eine Menge C_i in C, die e enthält. Unserer Konstruktion nach besitzen die C_i's nur Kanten, die als Endknoten den Knoten i haben. Also muss U' einen Endknoten von e besitzen. Damit gilt, dass U' ein *VertexCover* ist.

Da SetCover sowohl in NP ist, als auch NP-schwer ist, ist es NP-vollständig.

Aufgabe 6

SetPacking ist in NP. Hierzu können wir einen Verifizierer verwenden, welcher als Zertifikat eine Menge S bekommt. Der Verifizierer überprüft nun, ob $|S| = k$ gilt, $S \subseteq \text{Pot}(U)$ gilt, und anschließend, ob $C_i \cap C_j = \emptyset, i \neq j$ gilt. Offensichtlich läuft dieser Verifizierer in polynomieller Zeit.

SetPacking ist NP-schwer. Hierzu reduzieren wir IndependentSet auf SetPacking. Aus einer Instanz aus IndependentSet können wir eine Instanz aus SetPacking wie folgt konstruieren:
$G = (V, E)$ ist ein ungerichteter Graph mit n Knoten. Wir setzen $U = E$, $S = \text{Pot}(U)$ und übernehmen das k aus der IndependentSet-Instanz in das k der SetPacking-Instanz. Unser $C \subseteq S$ besteht nun aus Mengen C_i, $i \in [\,n]$, wobei jedes C_i alle Kanten aus E enthält, die einen Endknoten am Knoten i haben. Also hat C eine Größe von n. Diese Konstruktion können wir offensichtlich in polynomieller Zeit erstellen.

IndependentSet $\Rightarrow$ SetPacking:
G hat ein *IndependentSet* S' von Größe $\geq k$. Nun führen wir unsere Konstruktion durch und bekommen die Menge C der Größe n. Anschließend können wir noch jedes C_i aus C löschen, wenn $i \notin S'$ ist. Damit hat C danach die Größe $|S'vert$, insbesondere also $\geq k$. Ist $|C| > k$, so löschen wir zusätzlich noch weitere C_i's aus C, bis $|C| = k$ gilt. Alle verbleibenden C_i in C sind voneinander disjunkt: Alle Elemente in U sind ja Kanten e in G. Da S' ein *IndependentSet* ist, ist höchstens einer der beiden Endpunkte von e in S'. Hat e keinen Endpunkt in S', gibt es auch kein C_i in C, welches e besitzt. Hat e jedoch einen Endpunkt j in S', ist aber C_j nicht mehr in C, dann ist auch e nicht mehr in C vorhanden. Hat e aber einen Endpunkt j in S' und ist C_j noch in C, dann ist e auch durch C_j in C vertreten. Jedoch kann e nicht noch durch ein anderes C_i in C vertreten sein, da ja S' ein *IndependentSet* ist. Also gilt $C_i \cap C_j, i \neq j$.

SetPacking $\Rightarrow$ IndependentSet:
Sei C ein *SetPacking* der Größe k nach unserem Konstrukt mit $C_i \cap C_j, i \neq j$. Erstelle nun aus C (bzw. den C_i) eine Menge an Knoten S'. $i \in [\,n]$ ist genau dann in S', falls C_i in C ist. Damit gilt $|S'| = |C| = k \geq k$. Nun betrachte jede Kante e aus U. Da C das *SetPacking* von U ist, gibt es höchstens eine Menge C_i in C, die e enthält. Unserer Konstruktion nach besitzen die C_i's nur Kanten, die als Endknoten den Knoten i haben. Ist also e in einem C_i vertreten, muss S' i besitzen. Damit gilt, dass U' ein *IndependentSet* ist.

Da SetPacking sowohl in NP ist, als auch NP-schwer ist, ist es NP-vollständig.