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

Aufgabe 4

QAP ist in NP. Hierzu können wir einen Verfiizierer verwenden, welcher als Zertifikat eben ein solches σ bekommt und dann überprüft, ob die Aussage gilt. Offensichtlich geht das in polinomieller Zeit.

QAP ist NP-schwer. Hierzu reduzieren wir HAMILTONKREIS auf QAP. Für das Hamiltonproblem über Graphen $G = (V, E)$ sein $m = n = |V|$ und sei $c_{i,j} = 1$, falls es eine Kante zwischen V_i und V_j gibt und $c_{i,j} = n + 1$ sonst. Außerdem sei $d_{i,j} = 1$, falls $(i + 1) \bmod n = j$, andernfalls $d_{i,j} = 0$. Schließlich sei $b = n$. Diese Berechnung ist offensichtlich polynomiell.

HAMILTONKREIS $\Rightarrow$ QAP.
Wenn es einen Hamiltonkreis $H = (v_{\pi(1)}, ..., v_{\pi(n)})$ gibt, dann gibt es auch für das QuadraticAssignment eine Lösung mit $\sigma = \pi$. Dies liegt daran, dass für diese Permutation gilt, dass alle $d_{\sigma(i),\sigma(j)} = 1$ gilt, dass dann auch $c_{i,j} = 1$ ist, und somit die Summe der Multiplikationen von c und d gleich n ist, da alle anderen Werte von d gleich 0 sind.

QAP $\Rightarrow$ HAMILTONKREIS
Wenn es eine Lösung σ für das QuadraticAssignment gibt, gilt für diese Lösung folgendes: σ ist eine Permutation über Knoten. Außerdem wissen wir, dass zwischen jeder dieser Knoten eine Kante existieren muss, denn angenommen zwischen zwei Knoten v_i und v_j keine Kante existiert und somit $c_{i,j} = n + 1$ ist und $(\pi(i) + 1) \bmod n = \pi(j)$ sei (also j kommt eins nach i in der Permutation), dann wäre also $d_{\pi(i),\pi(j)} = 1$ und die Multiplikation beider würde $n + 1$ ergeben. Somit könnte die Schranke b nicht mehr eingehalten werden. Eine Permutation der Knoten, die in der Reihenfolge, wie sie in der Permutation stehen, über Kanten verbunden sind, ist ein Hamiltonkreis.

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

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

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 Hamiltonkreis auf SetCover. Für ein Hamiltonproblem über Graphen $G = (V, E)$ und $V = \{v_1, ..., v_n\}$ sei $U = \{v_{1_{in}}, v_1, v_{1_{out}}, ..., v_{n_{in}}, v_n, v_{n_{out}}\}$. Außerdem sei für jedes $e = (v_i, v_j)$ in E eine Menge $\{v_{i_{out}}, v_{j_{in}}\}$ in S. Außerdem sei $k = \frac{|U|}{2} = |V|n$. Diese Berechnung ist offensichtlich in polynomieller Zeit möglich.

Hamiltonkreis ⇒ SetCover:
Wenn es einen Hamiltonkreis $H = (v_{\pi(1)}, ..., v_{\pi(n)})$ gibt, dann gibt es Kanten $E_H = (e_1, ..., e_n)$, sodass jede Kante e_i von $v_{\pi(i)}$ nach $v_{\pi(i)+1}$ verläuft. Spezialfall ist e_n, dass von $v_{\pi(n)}$ nach $v_{\pi(1)}$ verläuft. Also gibt es in S eine Untermenge C, wobei $C_i = \{v_{j_{out}}, v_{k_{in}}\}$ der Kante e_i entspricht, die von v_j nach v_k verläuft. Da diese Untermenge genau die Kanten des Hamiltonkreis entsprechen, ist jedes Element $v_{i_{out}}$ und $v_{i_{in}}$ genau ein Mal vertreten und die Kardinalität von C ist genau k.

SetCover ⇒ Hamiltonkreis:
Für eine gegebene Lösung C für gegebenes S und k gilt folgendes: Da $k = \frac{|U|}{2}$, jede Menge in C genau 2 Elemente hat und die Vereinigung der Mengen in C alle Knoten von U abdeckt, ist jedes Element von U genau ein Mal in C vertreten.
Nun lässt sich ein Hamiltonkreis folgendermaßen konstruieren: Starte bei v_1 und suche das C_i, dass $v_{1_{out}}$ enthält. In derselben Menge wird ein $v_{i_{in}}$ sein. Dieses v_i ist der nächste Knoten. Wiederhole, bis v_1 wieder erreicht wird. Dass das Durchlaufen der Knoten in der so produzierten Reihenfolge auch im Problemgraphen möglich ist, liegt an der Konstruktion von S. Zu zeigen ist aber noch, dass zum einen kein Knoten doppelt abgelaufen wird, und dass keiner nicht durchlaufen wird. Das erste stimmt, da wie oben gezeigt wurde, jedes Element von U genau einmal in C vorhanden sein wird, und dementsprechend jeweils nur ein $v_{i_{in}}$ und ein $v_{i_{out}}$ pro Knoten v_i existieren wird. Dass jeder Knoten erreicht wird liegt daran, dass der Kreis k Kanten hat und kein Knoten 2 Mal durchlaufen wird, also k Knoten durchlaufen werden, und das ist genau die Kardinalität der Knoten in G.

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

Übungsblatt 10

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

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 $|S'|$, 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.