Aufgabe 7.1 - NP-Aussagen

a) **FALSCH**
Da P $\subseteq$ NP sind auch einfachere Probleme in A enthalten. Diese müssen nicht NP-Hart sein, damit ist die Aussage falsch, denn das Problem B kann auch in P liegen. Betrachtet man die Mengendarstellung, so ist die leere Menge eine Teilmenge aber bestimmt nicht NP-Vollständig.
b) **FALSCH**
Falls P = NP, dann liegt mindestens ein NP-Vollständiges Problem in P und kann in Polynomialzeit gelöst werden. Alle anderen NP-Vollständigen Probleme können in dieses überführt werden und somit auch in Polynomialzeit gelöst werden. Allerdings gilt dies nicht für NP-Harte, denn dieses können auch schwieriger als NP sein und ausserhalb dieser Menge liegen. Sie bleiben in nicht polynomieller Zeit.
c) **WAHR**
Aufgrund der Transitivität kann durch die Vollständigkeit jedes NP-Problem auch in B überführt werden. Damit können alle NP-Probleme in Polynomialzeit gelöst werden und P=NP.
d) **FALSCH**
Die Lösung von B benötigt logarithmische Zeit und die Umformung von A in B benötigt die Polynomielle Zeit $O((n+1)^k)$. Allerdings ist die so benötigte Zeit die Summe aus Umfornem + Lösen, damit benötigen wir im worst-case immer noch polynomielle Zeit.
e)**FALSCH**
Da wir nur gegeben haben, dass A NP-Hart ist, ist nicht unbedingt A $\in NP$. Ist A also $\notin NP$, dann ist A auch nicht zwingend polynomiell reduzierbar auf ein Problem $B \in NP$.

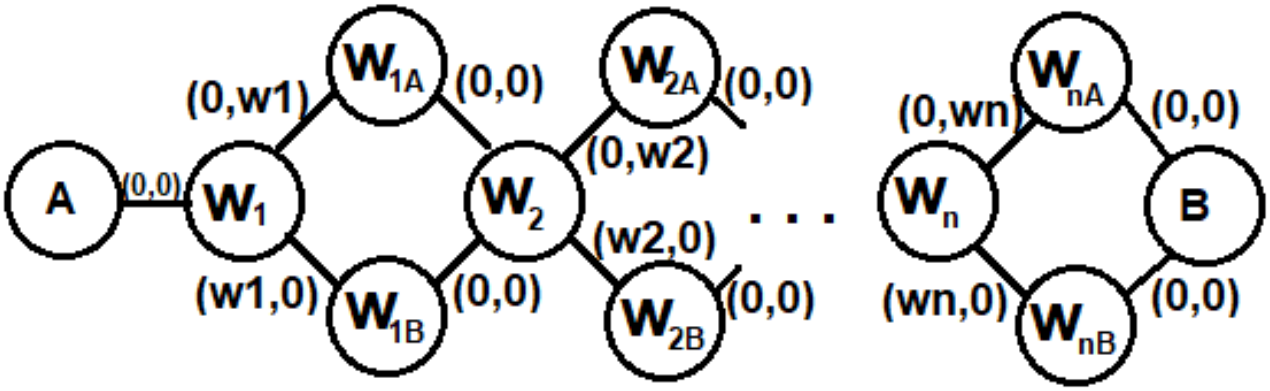
7.2 - Reduktions-Aussagen

a)
Wenn $B \in P$, dann ist B in Polynomialzeit deterministisch lösbar. Da A durch eine deterministische Umwandlung in Polynomialzeit in B umgewandelt werden kann, ist dadurch A lösbar. Die Lösung von A benötigt somit Umwandlungszeit + Lösungsaufwand für B und die Summe zweier Polynomialzeiten bleibt polynomiell.
b)
Wenn B NP-Vollständig ist, dann liegt B in NP und alle Probleme aus NP können auf B reduziert werden. Für das Problem A gibt es demnach eine umformung, sodass das Problem durch eine nicht-deterministische Turing-Maschine gelöst wird. Nämlich mit Polynomiell umformen auf B und dann lösen von B. A ist also nicht schwerer als NP, bzw. $A \in NP$.
c)
Durch die lineare Reduktion lässt sich die Eingabe in linearer Zeit umwandeln. Somit ist der gesuchte Algorithmus das umwandeln plus das Lösen von K. Die Eingabelänge für K bleibt hier in linearer Abhängigkeit von w, damit ist auch das Lösen von L in Linearzeit möglich.

7.3

Erhalten wir von einer nicht deterministischen Turing-Maschine eine Menge von erratenen Paaren, lässt sich sehr einfach prüfen, ob diese Paare die gegebenen Grenzen einhalten. Damit ist TTiT auf jeden Fall in NP. Um nun noch die NP-Härte zu zeigen, kann eine polynomielle Reduktion von Subset-Sum auf TTiT gefunden werden, denn da Subset-Sum NP-vollständig ist (nach Skript) ist durch die Transitivität dann auch TTiT NP-vollständig.
Diese Reduktion lässt sich folgendermaßen realisieren:

Die Manipulation M(w) einer Eingabe von einer Menge $w_i,\ i \in \{1, n\}$ an Zahlen und einer Summe S für Subset-Sum wird angepasst auf Paare und Begrenzungen k und t sowie einen Graphen für das TTiT-Problem.
Für jede Zahl aus Subset-Sum werden zwei Wege über je einen weiteren Knoten hinzugefügt, dabei werden die Kosten so gewählt, dass ein Weg Benzinkosten 0 und der andere Zeitkosten 0 hat. Die jeweils anderen Kosten sind die Werte der Zahlen aus Subset-Sum. Damit gibt es immer ein Wegteilstück mit entweder Benzin- oder Zeitkosten w_i Nun können die Begrenzungen für TTiT mit Benzinkosten genau S aus Subset-Sum und Zeitkosten $(\sum_{i=1}^{n} w_i) - S$ gewählt werden. Somit wird über die Weg-Wahl für TTiT jede Zahl aus Subset-Sum entweder für die Grenze S als Benzinkosten oder für den Rest als Zeitkosten gewählt. Gibt es also eine Lösung für Subset-Sum mit Summe S, so können an diesen Zahlen die Wege mit Benzinkosten gewählt werden und an den anderen die Zeitkosten. Damit erhalten wir S Benzinkosten und den Rest als Zeitkosten, wie gefordert.
Gibt es ein Lösung für TTiT, so müssen beide Grenzen eingehalten sein, das geht durch die Wahl üder die Gesamtsumme - S aber nur mit den exakten Werten. Also gibt es einen Weg mit Benzinkosten S und die entsprechende Wahl der Teilmenge hat für Subset-Sum die Summe S.
Skizze zur Veranschaulichung:



Wir können Subset-Sum somit polynomiell auf TTiT reduzieren und haben oben bereits gezeigt, dass TTiT ∈ NP, Damit ist TTiT NP-vollständig.

7.4 - HALF-CLIQUE

Wir zeigen zunächst, dass dieses Problem in NP liegt. Wir können entscheiden, ob ein Graph G mit V Knoten eine Clique der Größe V/2 besitzt, indem wir polynomiell die Anzahl an Knoten bestimmen und halbieren (das sei k). Das ist durch "Hin-Und-Herlaufen" in quadratischer Zeit möglich. Nun können wir die nicht-deterministische Turingmaschinge für das CLIQUE-Problem nehmen und nach der Lösung für den Graphen G mit einer CLIQUE-Größe von k fragen. Diese Entscheidungsfrage ist nach Vorlesung in NP. Nun ist noch NP-Härte zu zeigen :
Wir manipulieren eine Eingabe G=(V,E) mit einem k für CLIQUE, sodass Half-Clique die Lösung findet. Also eine deterministische in Polynomialzeit arbeitende Turing-Maschine, die die Eingabe umwandelt, und die Eingabe ist genau dann in CLIQUE, wenn die manipulierte in Half-Clique ist. Wir müssen zunächst eine Unterscheidung treffen bezüglich k aus CLIQUE (G,k) mit G = (V,E). Ist k = $\frac{|V|}{2}$, dann ist die Äquivalenz gegeben. Ist k > $\frac{|V|}{2}$, dann müssen wir in der Manipulation Knoten ohne Kanten hinzufügen.
Fall k> $\frac{|V|}{2}$:
Der bisherige Graph hat |V| Kknoten, durch hinzufügen von 2k - |V| unverbundenen Knoten erhalten wir |V|+2k-|V| = 2k Knoten. Die Größe von vorhandenen Cliquen wurde nicht verändert. Hatte der Ursprüngliche Graph eine Clique der Größe k, so hat der manipulierte sie auch und ist ∈ Half-Clique. Hat der manipulierte eine Half-Clique, dann ist diese auch im ursprünglichen vorhanden. Das hinzufügen von unverbundenen Knoten kann durch eine deterministische

Turing-Maschine in Polynomialzeit erfolgen.
Fall k$< \frac{|V|}{2}$:
Wir können nun neue Knoten Hinzufügen, und diese mit allen bisher vorhandenen und auch bereits eingefügten verbinden. Damit vergrößern wir die ursprünglich vorhandene(n) Clique(n). Wenn wir genau |V| -2k Knoten hinzufügen, erhält der manipulierte Graph genau 2(|V| -k) Knoten. Er liegt also in HALF-CLIQUE, wenn er nach dem ergänzen eine Clique der Größe (|V|-k) hat! Da wir aber |V| -2k ergänzt haben, war die vorherige Größe der Clique |V| -k -(|V| -2k) = k. Also liegt er auch in Clique(G,k). Waren im Ursprungsgraphen k Knoten in einer Clique, so sind alle diese mit den neuen verbunden und somit hat der manipulierte nun eine Clique der Größe |V| -k = $\frac{2(|V|-k)}{2}$, also Half-Clique. Auch dieses hinzufügen von Kanten und Knoten kann durch eine deterministische Turing-Maschine in Polynomialzeit erfolgen.
Damit haben wir eine polynomielle Reduktion von der NP-vollständigen CLIQUE-Entscheidung auf HALF-CLIQUE und aufgrund der Transitivität ist HALF-CLIQUE damit NP-Hart. Da es wie oben beschrieben in NP liegt. Somit ist HALF-CLIQUE auch NP-vollständig.