7.1 a) Durch die Aufgabenstellung ist $L \leq_e K$ gegeben.
T19 Daraus folgt, dass K mindestens so schwer ist wie L. Für K gebe es einen Algorithmus mit linearer Laufzeit, das heißt, das womöglich schwerere Problem ist in Linearzeit zu lösen. Da B nicht schwerer als A sein kann, ist es im schlimmsten Fall auch "nur" in Linearzeit zu lösen.

(Man kann auch das Problem L in Linearzeit in K überführen und das dann in Linearzeit lösen $\Rightarrow O(2n)$)

f b) Argumentation wie in a):
K sei mindestens so schwer wie L, wobei K in quadratischer Laufzeit lösbar ist.
Dann L nicht schwerer sein kann als K, ist es im schlimmsten Fall auch "nur" in Quadratzeit (etwas ist das !) lösbar. Leider nein!

c) Durch die Aufgabenstellung sei A, ein NP-hartes Problem gegeben. Wenn nun die Reduktion $A \leq_p B$ (für ein beliebiges Problem B) gelingt, muss B auch mindestens NP-hart sein. Angenommen B sei dennoch nicht NP-hart, dann kann A gar nicht NP-hart gewesen sein (Widerspruch zur Aufgabenstellung).
Die Reduktion $B \leq_p A$ bringt keine neue Einsicht, ausser dass A, das NP-harte Problem, mindestens so schwer ist, wie ein beliebiges Problem (das auch aus $P, \ldots$ stammen kann).
Angenommen, die transformierende Turingmaschine arbeite dabei
✓ nicht in polynomieller Zeit, dann hat die Reduktion keine Aussagekraft mehr über Probleme, die in P oder NP liegen.

2a) SUBSETSUM $\in$ NP. Beweis in Pseudocode.

(1) rate Lösung $S \subseteq \{1, 2, \dots n\}$, sollte $\sum_{i \in S} x_i = Z$ erfüllbar sein

(2) verifiziere Lösung S wie folgt:

```
für i = s1, s2, ... s|S|      // O(|S|) ∈ Polynomiallaufzeit
    sum += xi
falls sum == Z, return true else false // stimmt unsere Lösung?
```

PARTITION $\in$ NP Beweis in Pseudocode

(1) rate Lösung $S \subset \{1, 2, \dots n\}$, sollte $\sum_{i \in S} x_i = \sum_{\substack{j \notin S \\ 1 \leq j \leq n}} x_j$ erfüllbar sein

(2) verifiziere Lösung S wie folgt:

```
für i = x1, x2, ... x|X|      // O(|X|) ∈ Polynomialzeit
    falls i ∈ S
        sum 1 += xi
    sonst
        sum 2 += xi
falls sum1 == sum2 return true sonst false
```

RUCKSACK $\in$ NP Beweis in Pseudocode

(1) rate Lösung $S \subseteq \{1, 2, \dots n\}$, sollte $\sum_{i \in S} \text{weight}(i) \leq W$ und

$\sum_{i \in S} \text{value}(i) \geq V$ erfüllbar sind

(2) verifiziere Lösung S wie folgt:

```
für i = s1, s2, ... s|S|      // O(|S|) ∈ Polynomialzeit
    weights += weight(i)
    values += value(i)
falls weights ≤ W und values ≥ V return true sonst false
```

Schön!

7.2 b) Zu zeigen: SUBSET SUM $\leq_p$ PARTITION

Die Transformation:

Für die Eingabe X und Z fordern wir, dass Z von der Form $\frac{\sum_{i=1}^{|X|} x_i}{2}$ ist. (Also die Summe über alle Elemente in X, halbiert)

Die Transformation ist effizient berechenbar.

$\sum_{i \in S} x_i = Z \quad \in$ SUBSETSUM

$\Leftrightarrow \sum_{i \in S} x_i = \frac{\sum_{i=1}^{|X|} x_i}{2}$

$\Leftrightarrow \sum_{i \in S} x_i = \sum_{\substack{j \notin S \\ 1 \leq j \leq n}} x_j \quad \in$ PARTITION ✓

c) Zu zeigen: SUBSETSUM $\leq_p$ RUCKSACK

Die Transformation:

Wir fordern, dass weight (i) und value (i) genau auf x_i abbilden. Außerdem muss $V = W$ sein, damit wir einen exakten Wert als Lösung erhalten.

$\sum_{i \in S}$ weight $(i) \leq W$, $\sum_{i \in S}$ value $(i) \geq V \quad \in$ RUCKSACK

$\Leftrightarrow \sum_{i \in S}$ weight $(i) = W = V$

$\Leftrightarrow \sum_{i \in S} x_i = W$ ✓

$\Leftrightarrow \sum_{i \in S} x_i = Z \quad \in$ SUBSETSUM

7 3) Zu zeigen: GRAPH EMBEDDING ist NP-vollständig

Zu zeigen: GRAPHEMBEDDING ist in NP:

(1) rate injektive Abbildung $f: V_1 \to V_2$, sodass für alle

(2) verifiziere Lösung wie folgt: Kanten $\{u,v\} \in E_1$

für alle $\{u,v\} \in E_1$ //$O(|E_1|)$ gilt: $\{f(u), f(v)\} \in E_2$

falls $!(\{f(u), f(v)\} \in E_2)$

return false;

return true;

In Polynomialzeit verifizierbar $\Rightarrow$ GRAPHEMBEDDING $\in$ NP

Zu zeigen: CLIQUE $\leq_p$ GRAPH EMBEDDING

Die Transformation:

Wenn ein vollständiger Graph K_n (mit $n \in \mathbb{N}_{>0}$ als Anzahl Knoten) in einen Graphen G_2 eingebettet werden kann, enthält G_2 eine Clique der Größe n.

Die Transformation ist effizient berechenbar.

$(G, k) \in$ CLIQUE

$\Leftrightarrow (K_k, G_2) \in$ GRAPHEMBEDDING

$\Rightarrow$ GRAPHEMBEDDING $\in$ NP-Vollständig.

Diese Umformung funktioniert offensichtlich auch in die umgekehrte Richtung

✓