Grundkurs Theoretische Informatik - Beweise von NP-Vollständigkeit

Stefan Neubert

März 2021

Die folgenden Beweise sind Lösungen zur Aufgabe 21.6.3 (Beweise von NP-Vollständigkeit) aus dem Buch *Grundkurs Theoretische Informatik*, erschienen 2021 beim Rheinwerk Verlag.

1 Subset-Sum ist in NP

Zur Erinnerung hier nochmals die Problemdefinition von Subset-Sum:

> **Subset-Sum**
>
> **Gegeben:** Eine Menge von n Objekten mit den Größen $a_1, a_2, \ldots, a_n \in \mathbb{N}$ sowie eine Zahl $b \in \mathbb{N}$.
>
> **Gefragt:** Gibt es eine Teilmenge $A' \subseteq \{1, 2, \ldots, n\}$ der Objekte mit der Gesamtgröße $\sum_{i \in A'} a_i = b$?

Wir beweisen, dass Subset-Sum in NP ist, indem wir einen Polynomialzeitverifizierer angeben: Gegeben eine Instanz $(a_1, a_2, \ldots, a_n, b)$ sowie als Zertifikat eine Indexmenge $A' \subseteq \{1, 2, \ldots, n\}$ der Länge $|A'| \in O(n)$, überprüft unser Verifizierer, ob $\sum_{i \in A'} a_i = b$ gilt. Ist dies der Fall, akzeptiert der Algorithmus die Eingabe, ansonsten verwirft er sie.

Damit prüft der Verifizierer exakt die Definition von Subset-Sum und ist somit korrekt, da es exakt dann ein entsprechendes Zertifikat gibt, falls es eine Menge A' gibt, die die Bedingung erfüllt. Sind die Objekte als Array gegeben und ist die Teilmenge als Bitstring der Länge n codiert (das Objekt a_i ist genau dann in A', falls an Stelle i im Bitstring eine 1 steht), benötigt die Berechnung der Summe lediglich $O(n)$ Schritte. Somit hat der Verifizierer polynomielle Laufzeit und Subset-Sum ist in NP.

2 Clique ist NP-vollständig

Zur Erinnerung hier nochmals die Problemdefinitionen von Clique und Independent Set:

> **Clique**
>
> **Gegeben:** Ein Graph $G = (V, E)$, eine natürliche Zahl $k \leq |V|$.
>
> **Gefragt:** Gibt es eine Teilmenge $C \subseteq V$ mit $|C| = k$, sodass jedes Paar von Knoten in C durch eine Kante verbunden ist?

> **Independent Set (IS)**
>
> **Gegeben:** Ein Graph $G = (V, E)$, eine natürliche Zahl $k \leq |V|$.
>
> **Gefragt:** Gibt es eine Teilmenge $I \subseteq V$ mit $|I| = k$, sodass kein Paar von Knoten aus I durch eine Kante verbunden ist?

Bekannt ist, dass IS NP-vollständig ist. Wir zeigen daher mit IS $\leq_p$ Clique, dass Clique NP-schwer ist, und mit Clique $\leq_p$ IS, dass Clique in NP ist.

Als erstes zeigen wir IS $\leq_p$ Clique und geben dafür eine Reduktionsfunktion f an: Gegeben ist also ein Graph $G = (V, E)$ und eine natürliche Zahl $k \leq |V|$. Die Reduktionsfunktion konstruiert daraus den Komplementgraphen $\overline{G} = (V, \overline{E})$ mit den Kanten $\overline{E} = \{\, \{u, v\} \mid u, v \in V \wedge u \neq v \wedge \{u, v\} \notin E \,\}$, in dem also genau diejenigen Knoten benachbart sind, die es in G nicht sind. Zusätzlich gibt die Reduktionsfunktion die Zahl k unverändert zurück.

Da der Komplementgraph in $O(|V|^2)$ Schritten konstruiert werden kann und die Zahl k unverändert durchgereicht wird, kann f deterministisch in Polynomialzeit berechnet werden.

Es verbleibt zu zeigen, dass für alle Instanzen (G, k) gilt:

$$(G, k) \in \text{IS} \Leftrightarrow f(G, k) = (\overline{G}, k) \in \text{Clique}.$$

Sei dafür zunächst $(G, k) \in$ IS. Es gibt also nach Definition von IS eine Menge $I \subseteq V$ mit $|I| = k$, sodass kein Paar von Knoten aus I durch eine Kante verbunden ist. Per Konstruktion sind daher alle Knoten in I im Komplementgraphen $\overline{G}$ paarweise durch eine Kante verbunden und I ist eine Clique der Größe k in $\overline{G}$. Also gilt $(\overline{G}, k) \in$ Clique.

Sei umgekehrt $(\overline{G}, k) \in$ Clique. Dann gibt es nach Definition von Clique eine Menge $C \subseteq V$ mit $|C| = k$, sodass jedes Paar von Knoten in C durch eine Kante aus $\overline{E}$ verbunden ist. Da E keine der Kanten aus $\overline{E}$ enthält, ist C im ursprünglichen Graphen G ein Independent Set der Größe k und es gilt $(G, k) \in$ IS.

Somit ist die Äquivalenz erfüllt und es gilt insgesamt IS $\leq_p$ Clique und Clique ist NP-schwer. Da $G = \overline{\overline{G}}$ gilt, folgt mit derselben Reduktion direkt auch Clique $\leq_p$ IS und Clique ist zusätzlich in NP. Zusammengenommen ist damit bewiesen, dass Clique NP-vollständig ist.

3 Dominating Set ist NP-vollständig

Zur Erinnerung hier nochmals die Problemdefinitionen von Dominating Set und Vertex Cover (VC):

> **Dominating Set**
>
> **Gegeben:** Ein Graph $G = (V, E)$, eine natürliche Zahl $k \leq |V|$.
>
> **Gefragt:** Gibt es eine Menge $D \subseteq V$ mit $|D| = k$, sodass jeder Knoten $v \in V$ entweder in D oder benachbart zu einem Knoten in D ist?

> **Vertex Cover (VC)**
>
> **Gegeben:** Ein Graph $G = (V, E)$, eine natürliche Zahl $k \leq |V|$.
>
> **Gefragt:** Gibt es eine Teilmenge $S \subseteq V$ mit $|S| = k$, sodass jede Kante des Graphen mindestens einen Endpunkt in S hat?

Wir zeigen zunächst, dass Dominating Set in NP ist, indem wir einen Polynomialzeitverifizierer angeben: Gegeben eine Instanz $(G = (V, E), k)$ sowie als Zertifikat eine Menge $D \subseteq V$ der Länge $|D| \in O(|V|)$ (wie zuvor als Bitstring codiert), überprüft unser Verifizierer für jeden Knoten $v \in V$, ob $v \in D$ gilt oder es eine Kante $\{u, v\} \in E$ mit $u \in D$ gibt. Ist dies für alle Knoten der Fall, so akzeptiert der Algorithmus die Eingabe, ansonsten verwirft er sie.

Damit prüft der Verifizierer exakt die Definition von Dominating Set und ist somit korrekt, da es exakt dann ein entsprechendes Zertifikat gibt, falls es eine Menge S gibt, die die Bedingung erfüllt. Pro Knoten kann der Test $v \in D$ in $O(|V|)$ Schritten erfolgen; der Test für jede an v anliegende Kante ist in $O(|E| \cdot |V|)$ Schritten möglich. Somit hat der Verifizierer polynomielle Laufzeit und Dominating Set ist in NP.

Bekannt ist, dass VC NP-vollständig ist. Wir zeigen daher, dass Dominating Set NP-schwer ist, indem wir VC in polynomieller Zeit darauf reduzieren.

Unsere Reduktionsfunktion f erhält als Eingabe also einen Graphen $G = (V, E)$ und eine natürliche Zahl $k \leq |V|$. Sie konstruiert aus G einen neuen Graphen G' wie folgt: Zu jeder Kante $e = \{u, v\} \in E$ wird ein neuer Knoten w_e zusammen mit den beiden Kanten $\{u, w_e\}$ und $\{v, w_e\}$ in den Graphen eingefügt. Somit wird jede Kante des ursprünglichen Graphen zu einem Dreieck erweitert, wie Abbildung 1 zeigt. Es gilt also $G' = (V', E')$ mit $V' = V \cup \{ w_e \mid e \in E \}$ und $E' = E \cup \{ \{x, w_e\} \mid e \in E \wedge x \in e \}$. Sei des weiteren $I(V)$ die Menge der isolierten Knoten in G (also die Menge der Knoten, die an keiner Kante beteiligt sind) und sei $k' = \min(k + |I(V)|, |V|)$, dann gibt die Reduktion $f(G, k) = (G', k')$ als Instanz für Dominating Set zurück.

Die Ergänzung der zusätzlichen Knoten und Kanten geschieht in $O(|E|)$ Schritten. Das Zählen der isolierten Knoten erfolgt in $O(|V|)$ Schritten. Somit hat die Reduktionsfunktion insgesamt polynomielle Laufzeit.

Sei für den Beweis der Äquivalenz nun $(G = (V, E), k) \in$ VC und sei $(G', k') = f(G, k)$.

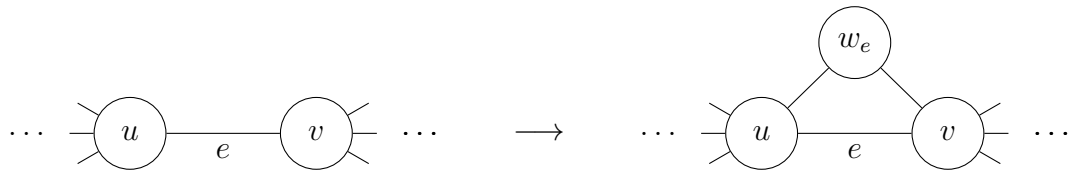

Abbildung 1: In der Reduktion von VC auf Dominating Set werden alle Kanten durch Dreiecke ersetzt.

Also gibt es nach Definition von Vertex Cover eine Teilmenge $S \subseteq V$ mit $|S| = k$, sodass jede Kante des Graphen G mindestens einen Endpunkt in S hat. Wir behaupten, dass $D = S \cup I(V)$ in G' ein Dominating Set der Größe $\leq k'$ ist. Per Definition beinhaltet D alle isolierten Knoten in G'. Alle anderen Knoten sind Teil von mindestens einem Dreieck wie in Abbildung 1 dargestellt. Da S ein Vertex Cover in G ist, muss in einem solchen Dreieck u, v, w_e zumindest einer der beiden Knoten u, v in S sein. Dieser Knoten ist per Konstruktion auch in D und dominiert dort alle Knoten des Dreiecks. Somit sind alle Knoten von G' entweder in D oder werden von einem Knoten aus D dominiert. Falls nun $|D| < k'$ gilt, können beliebige $|D| - k'$ zusätzliche Knoten zu D hinzugenommen werden, damit das Dominating Set genau k' Knoten enthält. Somit gilt $(G', k') \in$ Dominating Set.

Sei für die Rückrichtung $(G', k') = f(G, k) \in$ Dominating Set. Es gibt also eine Menge D von k' Knoten, sodass jeder Knoten $v \in V'$ entweder in D oder benachbart zu einem Knoten in D ist. Wir betrachten die Knotenmenge $S = D \setminus I(V)$. Da D alle isolierten Knoten von G' enthalten muss, hat S die Größe $|S| \leq k$. Wir zeigen nun, wie man aus S ein Vertex Cover der Größe k für G konstruieren kann. Per Konstruktion gibt es zu jeder Kante $e = \{u, v\} \in E$ ein Dreieck u, v, w_e in G'. Da D ein Dominating Set für G' ist, muss zumindest einer der drei Knoten des Dreiecks in D und somit auch in S sein, da die herausgenommene Knotenmenge $I(V)$ nur isolierte Knoten aus V enthält. Falls dieser Knoten w_e ist, entfernen wir ihn aus S und ergänzen stattdessen beliebig u oder v; im anderen Fall ist bereits u oder v in S enthalten. Somit wird jede Kante e durch einen Knoten aus (dem möglicherweise angepassten) S abgedeckt. Sollte nun $|S| < k$ gelten, so können beliebige $|S| - k$ zusätzliche Knoten aus V zu S hinzugenommen werden, damit das Vertex Cover genau k Knoten enthält. Somit gilt $(G, k) \in$ Vertex Cover.

Zusammengenommen ist damit bewiesen, dass Dominating Set NP-vollständig ist.

4 Partition ist NP-vollständig

Zur Erinnerung hier nochmals die Problemdefinitionen von Partition und Subset-Sum:

> **Partition**
>
> **Gegeben:** Eine Menge von n Objekten mit den Größen $a_1, a_2, \ldots, a_n \in \mathbb{N}$.
>
> **Gefragt:** Gibt es eine Teilmenge $P \subseteq \{1, 2, \ldots, n\}$ der Objekte mit der Gesamtgröße $\sum_{i \in P} a_i = \sum_{i \notin P} a_i$?

> **Subset-Sum**
>
> **Gegeben:** Eine Menge von n Objekten mit den Größen $a_1, a_2, \ldots, a_n \in \mathbb{N}$ sowie eine Zahl $b \in \mathbb{N}$.
>
> **Gefragt:** Gibt es eine Teilmenge $A' \subseteq \{1, 2, \ldots, n\}$ der Objekte mit der Gesamtgröße $\sum_{i \in A'} a_i = b$?

Wir beweisen, dass Partition in NP ist, indem wir einen Polynomialzeitverifizierer angeben: Gegeben eine Instanz $(a_1, a_2, \ldots, a_n)$ sowie als Zertifikat eine als Bitstring codierte Teilmenge $P \subseteq \{1, 2, \ldots, n\}$ der Länge $|P| \in O(n)$, überprüft unser Verifizierer, ob $\sum_{i \in P} a_i = \sum_{i \notin P} a_i$ gilt. Ist dies der Fall, akzeptiert der Algorithmus die Eingabe, ansonsten verwirft er sie.

Damit prüft der Verifizierer exakt die Definition von Partition und ist somit korrekt, da es exakt dann ein entsprechendes Zertifikat gibt, falls es eine Menge P gibt, die die Bedingung erfüllt. Die Berechnung der Summen benötigt lediglich $O(n)$ Schritte. Somit hat der Verifizierer polynomielle Laufzeit und Partition ist in NP.

Bekannt ist, dass Subset-Sum NP-vollständig ist. Wir zeigen daher, dass Partition NP-schwer ist, indem wir Subset-Sum in polynomieller Zeit darauf reduzieren.

Unsere Reduktionsfunktion f erhält als Eingabe eine Subset-Sum Instanz $(a_1, a_2, \ldots, a_n, b)$. Sie berechnet die Summe $T = \sum_{i=1}^{n} a_i$ und gibt die Instanz $(a_1, a_2, \ldots, a_n, |2b - T|)$ für Partition zurück.

Das Berechnen der Summe sowie der zusätzlichen Objektgröße ist in $O(n)$ möglich und die Reduktion somit in Polynomialzeit berechenbar.

Sei für den Beweis der Äquivalenz nun $(a_1, a_2, \ldots, a_n, b) \in$ Subset-Sum und sei T wie zuvor beschrieben definiert. Dann gibt es eine Teilmenge $A' \subseteq \{1, 2, \ldots, n\}$ der Objekte mit der Gesamtgröße $\sum_{i \in A'} a_i = b$. Wir unterscheiden zwei Fälle:

1. Sei $2b \geq T$. Dann gilt $|2b - T| = (2b - T)$ und wir erhalten

$$\begin{aligned}(2b - T) + \sum_{i \notin A'} a_i &= 2\left(\sum_{i \in A'} a_i\right) - \left(\sum_{i \in A'} a_i + \sum_{i \notin A'} a_i\right) + \sum_{i \notin A'} a_i \\ &= \sum_{i \in A'} a_i.\end{aligned}$$

Somit ist die Aufteilung in, auf der einen Seite, die Objekte $\overline{A'}$ zusammen mit dem neuen Objekt mit Größe $|2b - T|$, und auf der anderen Seite die Objekte A' eine Partitionierung von $(a_1, a_2, \ldots, a_n, |2b - T|)$ in zwei gleich große Teile.

2. Sei $2b < T$. Dann gilt $|2b - T| = (T - 2b)$ und wir erhalten

$$\begin{aligned}(T - 2b) + \sum_{i \in A'} a_i &= \left(\sum_{i \in A'} a_i + \sum_{i \notin A'} a_i\right) - 2\left(\sum_{i \in A'} a_i\right) + \sum_{i \in A'} a_i \\ &= \sum_{i \notin A'} a_i.\end{aligned}$$

   Somit ist die Aufteilung in, auf der einen Seite, die Objekte A' zusammen mit dem neuen Objekt mit Größe $|2b - T|$, und auf der anderen Seite die Objekte $\overline{A'}$ eine Partitionierung von $(a_1, a_2, \ldots, a_n, |2b - T|)$ in zwei gleich große Teile.

In beiden Fällen gilt $(a_1, a_2, \ldots, a_n, |2b - T|) \in$ Partition.

Sei für die Rückrichtung nun $(a_1, a_2, \ldots, a_n, |2b - T|) \in$ Partition. Dann gibt es eine Teilmenge $P \subseteq \{1, 2, \ldots, n\}$ der Objekte mit der Gesamtgröße $\sum_{i \in P} a_i = \sum_{i \notin P} a_i$. Per Konstruktion haben beide Partitionen eine Gesamtgröße von $(T + |2b - T|)/2$. Wir unterscheiden dieselben zwei Fälle wie zuvor:

1. Sei $2b \geq T$. Dann gilt $|2b - T| = (2b - T)$ und beide Partitionen haben die Gesamtgröße b. Da das hinzugefügte Element mit Größe $|2b - T|$ nur entweder in P oder in $\overline{P}$ enthalten sein kann, enthält die jeweilige andere Partition nur Objekte aus der ursprünglichen Subset-Sum Instanz und ist zudem genau b groß.

2. Sei $2b < T$. Dann gilt $|2b - T| = (T - 2b)$ und beide Partitionen haben die Gesamtgröße $T - b$. Eine der beiden Partitionen P oder $\overline{P}$ enthält das hinzugefügte Element mit Größe $|2b - T|$ und ansonsten nur Objekte aus der ursprünglichen Subset-Sum Instanz mit einer Gesamtgröße von genau b.

In beiden Fällen haben wir somit eine Menge von Objekten der ursprünglichen Instanz mit Gesamtgröße b identifiziert und somit gilt $(a_1, a_2, \ldots, a_n, b) \in$ Subset-Sum.

Zusammengenommen ist damit bewiesen, dass Partition NP-vollständig ist.

5 Knapsack ist NP-vollständig

Zur Erinnerung hier nochmals die Problemdefinitionen von Knapsack und Subset-Sum:

> **Knapsack**
>
> **Gegeben:** Eine Menge von n Objekten mit den Gewichten $g_1, g_2, \ldots, g_n \in \mathbb{N}$ und den Werten $w_1, w_2, \ldots, w_n \in \mathbb{N}$ sowie zwei Zahlen $G, W \in \mathbb{N}$.
>
> **Gefragt:** Gibt es eine Teilmenge $M \subseteq \{1, 2, \ldots, n\}$ der Objekte mit dem Gesamtgewicht $\sum_{i \in M} g_i \leq G$ und dem Gesamtwert $\sum_{i \in M} w_i \geq W$?

> **Subset-Sum**
>
> **Gegeben:** Eine Menge von n Objekten mit den Größen $a_1, a_2, \ldots, a_n \in \mathbb{N}$ sowie eine Zahl $b \in \mathbb{N}$.
>
> **Gefragt:** Gibt es eine Teilmenge $A' \subseteq \{1, 2, \ldots, n\}$ der Objekte mit der Gesamtgröße $\sum_{i \in A'} a_i = b$?

Wir beweisen, dass Knapsack in NP ist, indem wir einen Polynomialzeitverifizierer angeben: Gegeben eine Instanz $(g_1, \ldots, g_n, w_1, \ldots, w_n, G, W)$ sowie als Zertifikat eine Indexmenge $M \subseteq \{1, 2, \ldots, n\}$ der Länge $|M| \in O(n)$, überprüft unser Verifizierer, ob sowohl $\sum_{i \in M} g_i \leq G$ als auch $\sum_{i \in M} w_i \geq G$ gilt. Ist dies der Fall, akzeptiert der Algorithmus die Eingabe, ansonsten verwirft er sie.

Damit prüft der Verifizierer exakt die Definition von Knapsack und ist somit korrekt, da es exakt dann ein entsprechendes Zertifikat gibt, falls es eine Menge M gibt, die die Bedingung erfüllt. Sind die Objekte als Array gegeben und ist die Teilmenge als Bitstring der Länge n codiert (das Objekt (g_i, w_i) ist genau dann in M, falls an Stelle i im Bitstring eine 1 steht), benötigt die Berechnung der Summe lediglich $O(n)$ Schritte. Somit hat der Verifizierer polynomielle Laufzeit und Knapsack ist in NP.

Bekannt ist, dass Subset-Sum NP-vollständig ist. Wir zeigen daher, dass Knapsack NP-schwer ist, indem wir Subset-Sum in polynomieller Zeit darauf reduzieren.

Unsere Reduktionsfunktion f erhält als Eingabe eine Subset-Sum Instanz $(a_1, a_2, \ldots, a_n, b)$. Sie gibt als Instanz für Knapsack $(a_1, \ldots, a_n, a_1, \ldots, a_n, b, b)$ zurück, übernimmt also die Anzahl n von Objekten und setzt für jedes $1 \leq i \leq n$ das Objektgewicht $g_i = a_i$ und ebenso den Objektwert $w_i = a_i$; außerdem wählt die Funktion $G = W = b$.

Die Reduktionsfunktion kopiert lediglich zweimal die Eingabewerte und ist somit in Polynomialzeit berechenbar.

Da Wert und Gewicht von allen Objekten jeweils identisch sind, gilt für jede Teilmenge $M \subseteq \{1, 2, \ldots, n\}$, dass $\sum_{i \in M} g_i = \sum_{i \in M} w_i$ ist. Somit sind die Bedingungen von Knapsack und Subset-Sum äquivalent und wir können direkt $(a_1, a_2, \ldots, a_n, b) \in$ Subset-Sum $\Leftrightarrow f(a_1, a_2, \ldots, a_n, b) \in$ Knapsack folgern.

Zusammengenommen ist damit bewiesen, dass Knapsack NP-vollständig ist.

6 TSP ist NP-vollständig

Zur Erinnerung hier nochmals die Problemdefinitionen von TSP und HC:

> **Travelling Salesperson Problem (TSP)**
>
> **Gegeben:** Eine Distanzmatrix C der Größe $n \times n$ von natürlichen Zahlen, wobei für alle $1 \leq i, j \leq n$ der Eintrag $C[i, j]$ die Distanz zwischen zwei Städten i, j angibt, sowie eine Maximaldistanz $d \in \mathbb{N}$.
>
> **Gefragt:** Gibt es eine Rundfahrt durch alle n Städte, deren Gesamtstrecke maximal d ist?

> **Hamiltonian Cycle (HC)**
>
> **Gegeben:** Ein Graph $G = (V, E)$.
>
> **Gefragt:** Gibt es einen Kreis in G, der jeden Knoten genau einmal besucht?

Wir beweisen, dass TSP in NP ist, indem wir einen Polynomialzeitverifizierer angeben: Gegeben eine Instanz (C, n, d) sowie als Zertifikat eine Permutation π der Zahlen $(1, 2, \ldots, n)$ der Länge $|\pi| \in O(n)$, überprüft unser Verifizierer, ob $C[\pi(n), \pi(1)] + \sum_{i=1}^{n-1} C[\pi(i), \pi(i+1)] \leq d$ gilt. Ist dies der Fall, akzeptiert der Algorithmus die Eingabe, ansonsten verwirft er sie.

Die beschriebene Summe ist die Gesamtlänge der durch π beschriebenen Rundfahrt. Damit prüft der Verifizierer exakt die Definition von TSP und ist somit korrekt, da es exakt dann ein entsprechendes Zertifikat gibt, falls es eine Rundfahrt, d.h. eine Permutation der Städte, gibt, die die Bedingung erfüllt. Die Berechnung der Summe und der Vergleich benötigen lediglich $O(n)$ Schritte, wenn die Permutation als Array gegeben ist. Somit hat der Verifizierer polynomielle Laufzeit und TSP ist in NP.

Gegeben ist, dass HC NP-vollständig ist. Wir zeigen daher, dass TSP NP-schwer ist, indem wir HC in polynomieller Zeit darauf reduzieren.

Unsere Reduktionsfunktion f erhält als Eingabe eine HC Instanz $G = (V, E)$ mit $n = |V|$ Knoten und $m = |E|$ Kanten. Sie konstruiert daraus eine Distanzmatrix C der Größe $n \times n$ mit

$$\forall 1 \leq i, j \leq n \colon C[i, j] = \begin{cases} 1, & \text{falls } \{i, j\} \in E; \\ 2, & \text{sonst.} \end{cases}$$

Als Maximalstrecke d wählt die Reduktionsfunktion $d = n$.

Das Berechnen der n^2 Matrixeinträge kostet pro Eintrag maximal $O(n + m)$ Schritte, somit ist die Reduktionsfunktion in polynomieller Zeit berechenbar.

Sei nun für den Beweis der Äquivalenz $G \in \text{HC}$ und seien C, n wie beschrieben daraus konstruiert. Dann gibt es einen Kreis in G, der jeden Knoten genau einmal besucht und somit aus n Kanten besteht. Die zu diesen Kanten korrespondierenden Matrixeinträge

in C sind nach Konstruktion alle 1 und dieselbe Rundreise in der konstruierten TSP Instanz hat Gesamtdistanz n. Also ist $(C, d) \in \text{TSP}$.

Sei umgekehrt $(C, d) \in \text{TSP}$. Dann gibt es eine Rundfahrt durch alle n Städte, deren Gesamtstrecke maximal $d = n$ ist. Diese Rundfahrt besteht aus insgesamt n Teilabschnitten zwischen zwei Städten. Da nach Konstruktion jeder Teilabschnitt mindestens Distanz 1 zur Gesamtstrecke beiträgt, kann kein Teilabschnitt Distanz 2 haben, ohne dass die Gesamtdistanz größer als n würde. Somit korrespondiert jeder Teilabschnitt der Rundfahrt mit einer Kante im ursprünglichen Graphen G. Da die Rundfahrt jede Stadt genau einmal besucht, bilden diese Kanten einen Hamiltonkreis in G und es gilt $G \in \text{HC}$.

Zusammengenommen ist damit bewiesen, dass TSP NP-vollständig ist.