

Theoretische Grundlagen der Informatik

Übung am 16.12.2010

INSTITUT FÜR THEORETISCHE INFORMATIK



- 5.Übungsblatt auf der Homepage
- Heute: Lehrevaluation
- Es werden jetzt Evaluationsbögen ausgeteilt
- Füllt diese bitte gleich aus
- Nach 10 Minuten werden die Bögen wieder eingesammelt und die Übung beginnt

Aufgabe 1

Betrachten Sie das Problem HAMILTONIAN CIRCUIT:

Definition (HC):

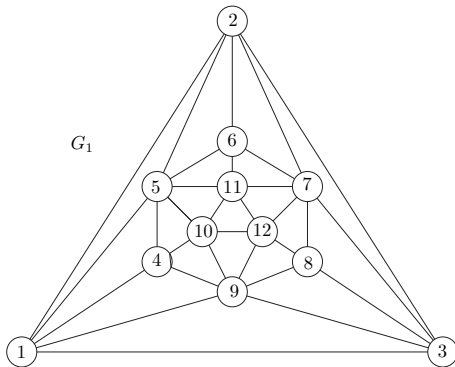
Gegeben: Ungerichteter Graph $G = (V, E)$.

Gesucht: Enthält G einen hamiltonschen Kreis?

Bemerkung: Ein hamiltonscher Kreis ist eine Permutation (v_1, v_2, \dots, v_n) der Knotenmenge, so dass $(v_i, v_{i+1}) \in E$ für alle $i = 1, \dots, n-1$ und $(v_n, v_1) \in E$. Ein hamiltonscher Kreis ist also ein zusammenhängender Kreis in G , der jeden Knoten genau einmal enthält.

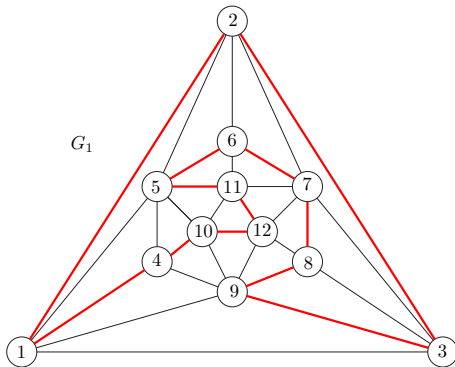
Aufgabe 1 a)

Geben Sie einen hamiltonschen Kreis in den folgenden Graphen an oder argumentieren Sie, warum der Graph keinen hamiltonschen Kreis enthalten kann:



Aufgabe 1 a)

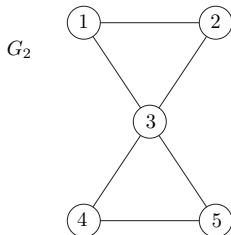
Geben Sie einen hamiltonschen Kreis in den folgenden Graphen an oder argumentieren Sie, warum der Graph keinen hamiltonschen Kreis enthalten kann:



- Beispiel für einen hamiltonschen Kreis:
(1, 2, 3, 9, 8, 7, 6, 5, 11, 12, 10, 4)

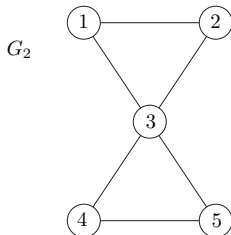
Aufgabe 1 a)

Geben Sie einen hamiltonschen Kreis in den folgenden Graphen an oder argumentieren Sie, warum der Graph keinen hamiltonschen Kreis enthalten kann:



Aufgabe 1 a)

Geben Sie einen hamiltonschen Kreis in den folgenden Graphen an oder argumentieren Sie, warum der Graph keinen hamiltonschen Kreis enthalten kann:



- Jeder Pfad, der von 1 nach 4 und wieder zurück führt, enthält den Knoten 3 zweimal. Dies gilt auch für jeden (zusammenhängenden) Kreis, der 1 und 4 enthält, also gibt es keinen hamiltonschen Kreis.

- Wir wollen zeigen, dass das HAMILTONIAN CIRCUIT-Problem \mathcal{NP} -vollständig ist
- Dazu: Umweg über das DIRECTED HAMILTONIAN CIRCUIT-Problem

Definition:

Gegeben ist ein gerichteter Graph $G = (V, E)$. Das DIRECTED HAMILTONIAN CIRCUIT-Problem besteht darin, zu entscheiden, ob G einen gerichteten Hamiltonkreis enthält.

Satz:

Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

- DHC ist in \mathcal{NP} enthalten, da wir in polynomieller Zeit entscheiden können, ob eine Menge an Kanten ein gerichteter hamiltonscher Kreis ist.
- \mathcal{NP} -Schwere wird über eine Reduktion direkt von 3-SAT gezeigt
- Sei I mit Variablenmenge $X = \{x_1, \dots, x_n\}$ und Klauselmengemenge $C = (c_1, \dots, c_m)$ eine Eingabe für 3-SAT
- Wir konstruieren daraus in polynomieller Zeit einen gerichteten Graphen G , der genau dann einen HC enthält, wenn I lösbar ist

Satz:

Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

- Identifiziere jede Variable mit einem Knoten und jede Klausel mit einem Subgraphen (Klauselkomponente), der aus 6 Knoten besteht
- Jede Klauselkomponente hat 3 Teile mit 2 Knoten, die den einzelnen Literalen entsprechen
- Verbinde die Variablenknoten und die Klauselkomponenten durch zwei Kreise (einen grünen und einen roten)
- Der grüne Kreis verbindet die Variablenknoten mit den Teilen der Klauselkomponenten, die positive Literale enthalten

Satz:

Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

- Dazu: Jede Variable x_i hat genau eine ausgehende grüne Kante, diese führt zum ersten Auftreten von x_i in einer Klauselkomponente (oder zu x_{i+1} , falls x_i in keiner Klausel vorkommt)
- Jeder Teil x_i einer Klauselkomponente hat eine Kante zur nächsten Klauselkomponente, in der x_i auftaucht, oder zum Variablenknoten x_{i+1} , falls es keine solche Komponente mehr gibt (x_1 , falls $i = m$)
- Analog dazu verbindet der rote Kreis die Variablenknoten mit den Teilen der Klauselkomponenten, die negative Literale enthalten

Beispielreduktion

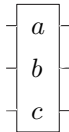
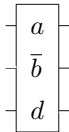
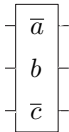
$$(\bar{a} \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee d) \wedge (a \vee b \vee c)$$

a

b

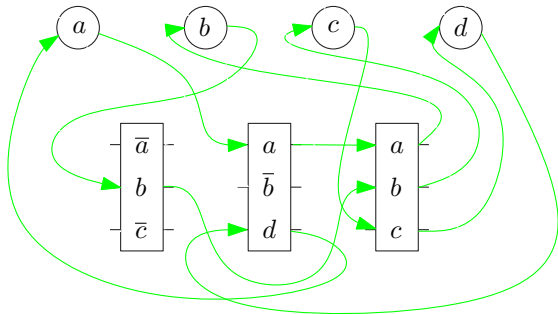
c

d



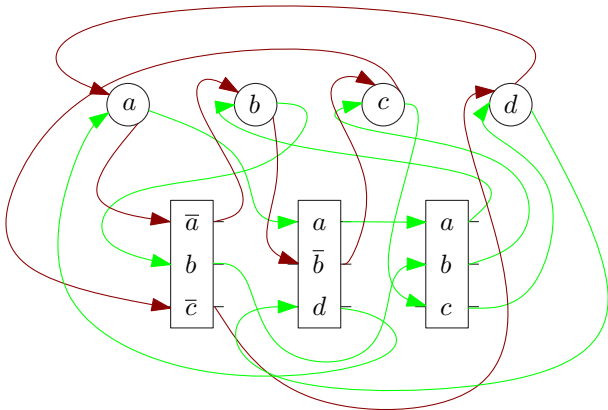
Beispielreduktion

$$(\bar{a} \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee d) \wedge (a \vee b \vee c)$$

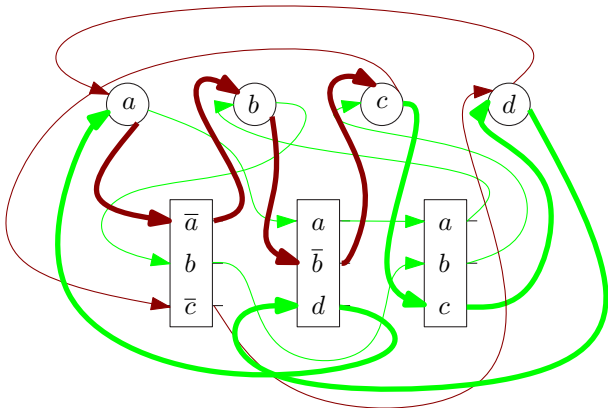


Beispielreduktion

$$(\bar{a} \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee d) \wedge (a \vee b \vee c)$$



$$(\bar{a} \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee d) \wedge (a \vee b \vee c)$$



Erfüllende Belegung: \bar{a}, \bar{b}, c, d

Satz:

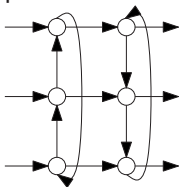
Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

Was müssen die Klauselkomponenten erfüllen?

- Wenn ein HC eine Komponente über die i -te eingehende Kante betritt, muss er sie auch über die i -te ausgehende Kante verlassen
- Es muss möglich sein, dass ein Hamiltonkreis eine Komponente ein-, zwei- oder dreimal passiert

Die folgende Komponente erfüllt diese Anforderungen:



Satz:

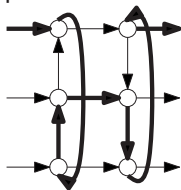
Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

Was müssen die Klauselkomponenten erfüllen?

- Wenn ein HC eine Komponente über die i -te eingehende Kante betritt, muss er sie auch über die i -te ausgehende Kante verlassen
- Es muss möglich sein, dass ein Hamiltonkreis eine Komponente ein-, zwei- oder dreimal passiert

Die folgende Komponente erfüllt diese Anforderungen:



einmal durchlaufen

Satz:

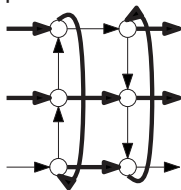
Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

Was müssen die Klauselkomponenten erfüllen?

- Wenn ein HC eine Komponente über die i -te eingehende Kante betritt, muss er sie auch über die i -te ausgehende Kante verlassen
- Es muss möglich sein, dass ein Hamiltonkreis eine Komponente ein-, zwei- oder dreimal passiert

Die folgende Komponente erfüllt diese Anforderungen:



zweimal durchlaufen

Satz:

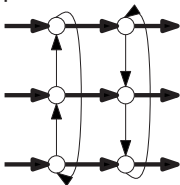
Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

Was müssen die Klauselkomponenten erfüllen?

- Wenn ein HC eine Komponente über die i -te eingehende Kante betritt, muss er sie auch über die i -te ausgehende Kante verlassen
- Es muss möglich sein, dass ein Hamiltonkreis eine Komponente ein-, zwei- oder dreimal passiert

Die folgende Komponente erfüllt diese Anforderungen:



dreimal durchlaufen

Satz:

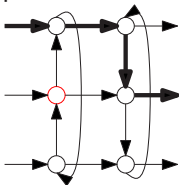
Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

Was müssen die Klauselkomponenten erfüllen?

- Wenn ein HC eine Komponente über die i -te eingehende Kante betritt, muss er sie auch über die i -te ausgehende Kante verlassen
- Es muss möglich sein, dass ein Hamiltonkreis eine Komponente ein-, zwei- oder dreimal passiert

Die folgende Komponente erfüllt diese Anforderungen:



Wenn die Komponente mit falscher Kante verlassen wird, werden Knoten isoliert

Satz:

Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

- Sei nun eine erfüllende Belegung der Variablen gegeben
- Starte Hamiltonkreis an der ersten Variable und beginne mit grüner ausgehenden Kante, falls die Variable erfüllt ist, sonst mit roter Kante
- Durchlaufe die Klauselkomponenten jeweils so, wie auf der letzten Folie illustriert (je nachdem, welche Literale erfüllt sind)
- Erreiche zweiten Variablenknoten und fahre entsprechend fort
- Konstruiere so Hamiltonkreis

Satz:

Das DIRECTED HAMILTONIAN CIRCUIT-Problem ist \mathcal{NP} -vollständig

Beweis:

- Sei nun Hamiltonkreis gegeben
- Für jede Variable prüfe, welche der beiden ausgehenden Kanten im Kreis sind
- Falls rote Kante, setze Variable auf falsch, falls grüne Kante, setze Variable auf wahr
- Der Kreis durchläuft dabei nur Klauselkomponenten, die erfüllte Literale enthalten
- Da wir einen Hamiltonkreis durchlaufen haben, müssen alle Klauseln erfüllt sein

Satz:

Das HAMILTONIAN CIRCUIT-Problem (HC) ist \mathcal{NP} -vollständig

Satz:

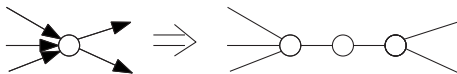
Das HAMILTONIAN CIRCUIT-Problem (HC) ist \mathcal{NP} -vollständig

Beweis:

- HC ist in \mathcal{NP} enthalten, da wir in polynomieller Zeit entscheiden können, ob eine Menge an Kanten ein ungerichteter hamiltonscher Kreis ist
- Reduziere von DHC
- Sei $G = (V, E)$ ein gerichteter Graph, also eine Instanz von DHC
- Konstruiere daraus ungerichteten Graphen $G' = (V', E')$, der genau dann einen HC enthält, wenn G einen DHC enthält
- Konstruiere G' so, dass jede gerichtete Kante durch eine ungerichtete Kante ersetzt wird und jeder Knoten durch einen Pfad, der aus drei Knoten besteht

Satz:

 Das HAMILTONIAN CIRCUIT-Problem (HC) ist \mathcal{NP} -vollständig

Beweis:


- Die eingehenden Kanten werden mit erstem Knoten des Pfades verbunden, die ausgehenden Kanten mit letztem Knoten des Pfades
- Ein DHC in G läßt sich direkt in einen ungerichteten HC in G' übersetzen
- Sei C ein HC in G'
- Betrachte die Kanten in C , die bereits in G waren
- Diese bilden DHC in G , denn: Würde man z.B. den linkesten Knoten im Beispiel von links erreichen und nach links wieder verlassen, wäre der mittlere Knoten nicht mehr auf einem HC passierbar

Erinnerung: TRAVELING SALESMAN-Entscheidungsproblem

Gegeben sei ein vollständiger Graph $G = (V, E, c)$, mit Kantengewichten, d.h.

- $V := \{1, \dots, n\}$
- $E := \{\{u, v\} \mid u, v \in V, u \neq v\}$
- $c: E \rightarrow \mathbb{Z}^+$.

Gegeben sei zusätzlich auch ein Parameter $k \in \mathbb{Z}^+$. Die Frage ist nun: Gibt es eine Tour (Rundreise), deren Länge höchstens k ist? Dabei entspricht eine Tour einer Permutation π auf V , das bedeutet, dass jeder Knoten genau einmal enthalten ist.

- Eine Tour entspricht also genau einem hamiltonschen Kreis in einem vollständigen, gewichteten Graphen

Aufgabe 1 b)

Zeigen Sie, dass das TRAVELING SALESMAN-Entscheidungsproblem \mathcal{NP} -vollständig ist. Benutzen Sie dazu die Tatsache, dass das HAMILTONIAN CIRCUIT-Problem \mathcal{NP} -vollständig ist.

Aufgabe 1 b)

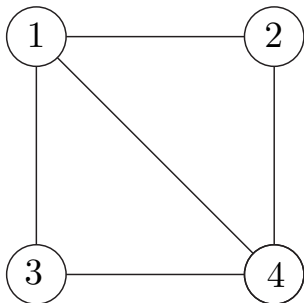
Zeigen Sie, dass das TRAVELING SALESMAN-Entscheidungsproblem \mathcal{NP} -vollständig ist. Benutzen Sie dazu die Tatsache, dass das HAMILTONIAN CIRCUIT-Problem \mathcal{NP} -vollständig ist.

- TSP liegt in \mathcal{NP} , denn in polynomieller Zeit kann man überprüfen, ob eine Eingabe eine Rundreise ist und die Länge dieser Rundreise berechnen (s. Skript).
- Transformiere HAMILTONIAN CIRCUIT auf TSP
- Sei $G = (V, E)$, $V = \{x_1, \dots, x_n\}$ eine Instanz von HC
- Konstruiere daraus Instanz I von TSP mit
 - n Orten $\{o_1, \dots, o_n\}$
 - Gewichten c_{ij} , die 1 sind, falls $(x_i, x_j) \in E$ und $n + 1$ sonst
 - Kostengrenze $k := n$
- Offensichtlich ist die Transformation polynomiell.

Aufgabe 1 b)

Zeigen Sie, dass das TRAVELING SALESMAN-Entscheidungsproblem \mathcal{NP} -vollständig ist. Benutzen Sie dazu die Tatsache, dass das HAMILTONIAN CIRCUIT-Problem \mathcal{NP} -vollständig ist.

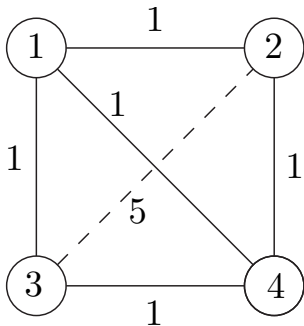
Beispiel für eine Konstruktion:



Aufgabe 1 b)

Zeigen Sie, dass das TRAVELING SALESMAN-Entscheidungsproblem \mathcal{NP} -vollständig ist. Benutzen Sie dazu die Tatsache, dass das HAMILTONIAN CIRCUIT-Problem \mathcal{NP} -vollständig ist.

Beispiel für eine Konstruktion:



Aufgabe 1 b)

Zeigen Sie, dass das TRAVELING SALESMAN-Entscheidungsproblem \mathcal{NP} -vollständig ist. Benutzen Sie dazu die Tatsache, dass das HAMILTONIAN CIRCUIT-Problem \mathcal{NP} -vollständig ist.

- Es gilt: G enthält einen HC genau dann, wenn I eine Rundreise mit Kosten n zulässt, denn:
- Sei $(x_{i_1}, \dots, x_{i_n})$ ein HC in G dann sind die Kosten der Rundreise $(o_{i_1}, \dots, o_{i_n})$ gleich n
- Sei $(o_{i_1}, \dots, o_{i_n})$ eine Rundreise mit Kosten höchstens n , dann gilt für $i = 1, \dots, n - 1$ dass $(x_i, x_{i+1}) \in E$ und $(x_n, x_1) \in E$ ist
- Damit ist $(x_{i_1}, \dots, x_{i_n})$ ein HC in G .
- Also gibt es eine polynomielle Transformation vom HAMILTONIAN CIRCUIT-Problem in das TSP-Problem
- Damit ist das TSP-Problem \mathcal{NP} -vollständig.

Aufgabe 2

Zeigen oder widerlegen Sie: Zu jedem Problem $\Pi \in \mathcal{NP}$ gibt es ein Polynom p , sodass Π durch einen deterministischen Algorithmus mit Zeitkomplexität in $\mathcal{O}(2^{p(n)})$ gelöst werden kann, wobei n die Größe der Eingabe bezeichnet.

Aufgabe 2

Zeigen oder widerlegen Sie: Zu jedem Problem $\Pi \in \mathcal{NP}$ gibt es ein Polynom p , sodass Π durch einen deterministischen Algorithmus mit Zeitkomplexität in $\mathcal{O}(2^{p(n)})$ gelöst werden kann, wobei n die Größe der Eingabe bezeichnet.

- Nach Voraussetzung gibt es eine NTM M , die Π entscheidet und deren Zeitkomplexitätsfunktion durch ein Polynom $q(n)$ beschränkt ist.
- Damit gibt es zu jeder Instanz I mit $|I| = n$ ein Orakelwort der Länge maximal $q(n)$, so dass M I in Zeit $q(n)$ akzeptiert.
- Baue aus M eine DTM M' , die Π in Zeit $2^{p(n)}$ entscheidet
- M' arbeitet wie folgt:
 - Zunächst berechnet M' $q(|I|)$
 - Zähle dann alle Worte über Γ mit aufsteigender Länge auf und simuliere dabei M mit dem gegebenen Wort als Orakelwort.
 - Falls M die Eingabe akzeptieren würde, akzeptiere
 - Falls M' alle Worte der Länge kleiner gleich $q(n)$ aufgezählt hat, bricht M' die Berechnung ab und verwirft die Eingabe.

Aufgabe 2

Zeigen oder widerlegen Sie: Zu jedem Problem $\Pi \in \mathcal{NP}$ gibt es ein Polynom p , sodass Π durch einen deterministischen Algorithmus mit Zeitkomplexität in $\mathcal{O}(2^{p(n)})$ gelöst werden kann, wobei n die Größe der Eingabe bezeichnet.

- Die Zahl der Worte, die aufgezählt werden müssen, ist also $k^{q(n)} = 2^{q(n) \cdot \log k} = 2^{p_1(n)}$, wobei k die Anzahl der Symbole im Bandalphabet ist
- Jede Simulation geht in polynomieller Zeit, also insgesamt ist die Laufzeit durch $2^{p_1(n)} \cdot p_2(n)$ beschränkt
- Daher ist die Gesamtrechenzeit in $\mathcal{O}(2^{p_1(n)} \cdot p_2(n))$, also auch in $\mathcal{O}(2^{p_1(n)} \cdot 2^n) = \mathcal{O}(2^{p_1(n)+n}) = \mathcal{O}(2^{p(n)})$ für ein Polynom $p(n)$

Aufgabe 3

Betrachten Sie das folgende Entscheidungsproblem FREE EDGE:

Gegeben ist ein ungerichteter Graph $G = (V, E)$ und eine Zahl $K \in \mathbb{N}$ mit $K \leq |V|$. Gibt es für alle Teilmengen V' von V der Größe K eine Kante $\{u, v\} \in E$, so dass weder u noch v in V' enthalten sind?

- Formulieren Sie das komplementäre Problem co-FREE EDGE

Aufgabe 3

Betrachten Sie das folgende Entscheidungsproblem FREE EDGE:

Gegeben ist ein ungerichteter Graph $G = (V, E)$ und eine Zahl $K \in \mathbb{N}$ mit $K \leq |V|$. Gibt es für alle Teilmengen V' von V der Größe K eine Kante $\{u, v\} \in E$, so dass weder u noch v in V' enthalten sind?

- Formulieren Sie das komplementäre Problem co-FREE EDGE

Lösung:

- Gegeben ist ein ungerichteter Graph $G = (V, E)$ und eine Zahl $K \in \mathbb{N}$ mit $K \leq |V|$. Gibt es eine Teilmenge V' von V der Größe K , so dass für alle Kanten in E mindestens einer der Endknoten in V' enthalten ist?

Aufgabe 3

Betrachten Sie das folgende Entscheidungsproblem FREE EDGE:

Gegeben ist ein ungerichteter Graph $G = (V, E)$ und eine Zahl $K \in \mathbb{N}$ mit $K \leq |V|$. Gibt es für alle Teilmengen V' von V der Größe K eine Kante $\{u, v\} \in E$, so dass weder u noch v in V' enthalten sind?

- Zeigen Sie, dass FREE EDGE in $\text{co-}\mathcal{NP}$ liegt.

Aufgabe 3

Betrachten Sie das folgende Entscheidungsproblem FREE EDGE:

Gegeben ist ein ungerichteter Graph $G = (V, E)$ und eine Zahl $K \in \mathbb{N}$ mit $K \leq |V|$. Gibt es für alle Teilmengen V' von V der Größe K eine Kante $\{u, v\} \in E$, so dass weder u noch v in V' enthalten sind?

- Zeigen Sie, dass FREE EDGE in $\text{co-}\mathcal{NP}$ liegt.

Lösung:

- Zu zeigen: co-FREE EDGE liegt in \mathcal{NP}
- Zu zeigen ist, dass in polynomieller Zeit entschieden werden kann, ob alle Kanten einen Endknoten in V' haben
- Sei V' eine beliebige Teilmenge von V der Größe K .
- Für eine Kante $\{u, v\}$ kann offensichtlich in polynomieller Zeit entschieden werden, ob u oder v in V' liegt.
- Insgesamt müssen also $|E|$ Kanten überprüft werden, dies geht in polynomieller Zeit.

Aufgabe 4

Zeigen Sie: Aus $\mathcal{P} = \mathcal{NP}$ folgt $\mathcal{P} \setminus \{\emptyset, \Sigma^*\} = \mathcal{NPC}$. Warum sind \emptyset und Σ^* (insbesondere unter dieser Annahme) nicht \mathcal{NP} -vollständig?

Aufgabe 4

Zeigen Sie: Aus $\mathcal{P} = \mathcal{NP}$ folgt $\mathcal{P} \setminus \{\emptyset, \Sigma^*\} = \mathcal{NPC}$. Warum sind \emptyset und Σ^* (insbesondere unter dieser Annahme) nicht \mathcal{NP} -vollständig?

Beweis:

- Annahme: Sei $\mathcal{P} = \mathcal{NP}$
- “ \Rightarrow ”:
 - Sei L_1 in $\mathcal{P} \setminus \{\emptyset, \Sigma^*\}$ und L_2 in \mathcal{NP}
 - Zeige: Es gibt eine polynomielle Transformation von L_2 in L_1 gibt.
 - Daraus folgt direkt, dass L_1 \mathcal{NP} -vollständig ist.
 - Da $\mathcal{P} = \mathcal{NP}$ ist, gibt es eine DTM M , die L_2 in polynomieller Zeit entscheidet
 - Modifiziere M wie folgt: Falls die Eingabe akzeptiert wird, schreibt M ein beliebiges Wort aus L_1 auf das Band, falls nicht, schreibt M ein beliebiges Wort aus $\Sigma^* \setminus L_1$ auf das Band
 - Damit gilt offensichtlich, dass die Ausgabe der DTM genau dann in L_1 liegt, wenn die Eingabe in L_2 lag.

Aufgabe 4

Zeigen Sie: Aus $\mathcal{P} = \mathcal{NP}$ folgt $\mathcal{P} \setminus \{\emptyset, \Sigma^*\} = \mathcal{NPC}$. Warum sind \emptyset und Σ^* (insbesondere unter dieser Annahme) nicht \mathcal{NP} -vollständig?

Beweis:

- Annahme: Sei $\mathcal{P} = \mathcal{NP}$
- “ \Leftarrow ”:
 - Sei L in \mathcal{NPC}
 - Da $\mathcal{P} = \mathcal{NP}$ ist, liegt L auch in \mathcal{P} .
 - Falls L die leere Menge ist, so kann eine beliebige nicht-leere Sprache L' offensichtlich nicht in L transformiert werden (Ein Wort in L' müsste ja auf ein Wort in der leeren Menge abgebildet werden).
 - Falls $L = \Sigma^*$ ist, so kann eine beliebige Sprache ungleich Σ^* offensichtlich nicht in L transformiert werden (Ein Wort nicht in L' müsste ja auf ein Wort in $\Sigma^* \setminus L = \emptyset$ abgebildet werden).
 - Beides ist ein Widerspruch zur \mathcal{NP} -Vollständigkeit von L
 - Also ist $L \in \mathcal{P} \setminus \{\emptyset, \Sigma^*\}$.

Aufgabe 5

Formulieren Sie das Problem EXACT COVER als *Integer Program*.
Konstruieren Sie dazu zu einer allgemeinen Instanz I von EXACT COVER eine Instanz von INTEGER PROGRAMMING, die genau dann eine Ja-Instanz ist, wenn I eine Ja-Instanz ist.

Aufgabe 5

Formulieren Sie das Problem EXACT COVER als *Integer Program*.
Konstruieren Sie dazu zu einer allgemeinen Instanz I von EXACT COVER eine Instanz von INTEGER PROGRAMMING, die genau dann eine Ja-Instanz ist, wenn I eine Ja-Instanz ist.

Lösung:

- Sei $O = \{o_1, \dots, o_m\}$ und $S = \{S_1, \dots, S_n\}$ mit $S_i \subseteq O$ eine Instanz von EXACT COVER
- Zu entscheiden ist, ob es eine Teilmenge S' von S gibt, so dass jedes Element aus O in genau einer der Mengen aus S' enthalten ist
- Idee: Benutze Vektor $\{x_1, \dots, x_n\} \in \{0, 1\}^n$, um beliebige Teilmengen von S zu kodieren
- Interpretation: Falls $x_i = 1$ ist, ist S_i in S' enthalten, falls $x_i = 0$ ist, ist S_i nicht in S' enthalten

Problem INTEGER PROGRAMMING

Gegeben: $a_{ij} \in \mathbb{N}_0, b_i, c_j \in \mathbb{N}_0, 1 \leq i \leq m, 1 \leq j \leq n, B \in \mathbb{N}_0.$

Frage: Existieren $x_1, \dots, x_n \in \mathbb{N}_0$, so dass

$$\sum_{j=1}^n c_j \cdot x_j = B \text{ und}$$

$$\sum_{j=1}^n a_{ij} \cdot x_j \leq b_i \text{ für } 1 \leq i \leq m?$$

$$\underbrace{\hspace{10em}}_{A \cdot \bar{x} \leq \bar{b}}$$

Formulieren Sie das Problem EXACT COVER als *Integer Program*.
Konstruieren Sie dazu zu einer allgemeinen Instanz I von EXACT COVER eine Instanz von INTEGER PROGRAMMING, die genau dann eine Ja-Instanz ist, wenn I eine Ja-Instanz ist.

Formulierung als Integer Program:

- Einschränkung 1: Die Variablen x_i müssen in $\{0, 1\}$ liegen
 - Sei E die $n \times n$ -Einheitsmatrix und $\vec{1}_n$ der Vektor der Länge n , der nur Einsen enthält.
 - Dann sind die $x_i \in \{0, 1\}$ genau dann, wenn $E \cdot x \leq \vec{1}_n$ ist
- Einschränkung 2: Jedes Objekt in O darf in höchstens einer Menge aus S' vorkommen
 - Sei A eine $m \times n$ -Matrix mit Koeffizienten $A_{ij} := 1$, falls o_i in S_j liegt und $A_{ij} := 0$, wenn o_i nicht in S_j liegt
 - Dann ist jede Variable genau dann in höchstens einer Menge aus S' vorhanden, falls $A \cdot x \leq \vec{1}_m$ ist

Aufgabe 5

Formulieren Sie das Problem EXACT COVER als *Integer Program*.
Konstruieren Sie dazu zu einer allgemeinen Instanz I von EXACT COVER eine Instanz von INTEGER PROGRAMMING, die genau dann eine Ja-Instanz ist, wenn I eine Ja-Instanz ist.

Formulierung als Integer Program:

- Einschränkung 3: Die Summe der Anzahl der Variablen in den Mengen in S' muss genau n sein
 - Das ist genau dann erfüllt, wenn $\sum_{j=1}^n |S_j| \cdot x_j = n$
- Insgesamt also: Existiert $x := (x_1, \dots, x_n)$ mit $x_j \in \mathbb{N}_0$, so dass
- $\sum_{j=1}^n |S_j| \cdot x_j = n$ und
- $\begin{pmatrix} A \\ E \end{pmatrix} \cdot x \leq \vec{1}_{m+n}$

Aufgabe 6

Geben Sie den Pseudocode eines pseudopolynomialen Algorithmus für SUBSET SUM an (Tipp: dynamische Programmierung). Geben Sie eine obere Schranke für die Laufzeit an, die polynomiell in der Anzahl der Zahlen in der Eingabe und der größten vorkommenden Zahl ist. Dazu müssen Sie nicht auf Turingmaschinenebene argumentieren, sondern können sich am RAM-Modell orientieren. Das bedeutet, dass Sie zum Beispiel annehmen dürfen, dass die Addition und der Vergleich von zwei Zahlen in $O(1)$ Zeit ausgeführt werden können.

Geben Sie den Pseudocode eines pseudopolynomialen Algorithmus für SUBSET SUM an (Tipp: dynamische Programmierung).

Pseudopolynomialer Algorithmus (Dynamisches Programm):

- Idee: Berechne iterativ alle möglichen Zahlen kleiner gleich K , die sich als Summe der Gewichte einer Teilmenge $\{x_1, \dots, x_i\}$ schreiben lassen
- Eingabe: Instanz $M = \{x_1, \dots, x_n\}$, $w : M \rightarrow \mathbb{N}_0$, $K \in \mathbb{N}_0$ von SUBSET SUM
- Ausgabe: „ja“, falls die Instanz lösbar ist und „nein“ sonst
- $A := \{0\}$
- Für alle $x \in \{x_1, \dots, x_n\}$
 - Für alle $a \in A$: Falls $w(x) + a \leq K$, setze $A := A \cup \{w(x) + a\}$
- Falls $K \in A$, gib „ja“ aus, sonst gib „nein“ aus

Geben Sie den Pseudocode eines pseudopolynomialen Algorithmus für SUBSET SUM an (Tipp: dynamische Programmierung).

Beispiel:

- Menge $M := \{m_1, m_2, m_3\}$ mit Gewichten $[1, 3, 4]$
- Frage: Existiert eine Teilmenge $M' \subseteq M$, so dass $\sum_{m \in M'} w(m) = 6$ ist?

Teilmenge $\{\}$ mit Gewichten $[\]$

$A := \{0\}$

Teilmenge $\{m_1\}$ mit Gewichten $[1]$

$A := \{0, 1\}$

Teilmenge $\{m_1, m_2\}$ mit Gewichten $[1, 3]$

$A := \{0, 1, 3, 4\}$

Teilmenge $\{m_1, m_2, m_3\}$ mit Gewichten $[1, 3, 4]$

$A := \{0, 1, 3, 4, 5\}$

- 6 ist nicht in A , also ist die Instanz nicht lösbar

Geben Sie eine obere Schranke für die Laufzeit an, die polynomiell in der Anzahl der Zahlen in der Eingabe und der größten vorkommenden Zahl ist. Dazu müssen Sie nicht auf Turingmaschinenebene argumentieren, sondern können sich am RAM-Modell orientieren. Das bedeutet, dass Sie zum Beispiel annehmen dürfen, dass die Addition und der Vergleich von zwei Zahlen in $O(1)$ Zeit ausgeführt werden können.

Laufzeit des Algorithmus:

- Wie viele Elemente kann die Menge A zu jedem Zeitpunkt maximal enthalten?
- Sei g_{\max} die größte Zahl in der Eingabe
- A enthält natürliche Zahlen, die nicht größer als k sind, also ist $|A| \leq k + 1 \leq g_{\max} + 1$

Geben Sie eine obere Schranke für die Laufzeit an, die polynomiell in der Anzahl der Zahlen in der Eingabe und der größten vorkommenden Zahl ist. Dazu müssen Sie nicht auf Turingmaschinenebene argumentieren, sondern können sich am RAM-Modell orientieren. Das bedeutet, dass Sie zum Beispiel annehmen dürfen, dass die Addition und der Vergleich von zwei Zahlen in $O(1)$ Zeit ausgeführt werden können.

Laufzeit des Algorithmus:

- Insgesamt wird die innere Schleife im Algorithmus also $O(n \cdot g_{\max})$ mal durchlaufen
- In jedem Schleifendurchlauf wird eine Addition ausgeführt und es muss geprüft werden, ob ein Element in A liegt
- Dies geht naiv in $O(|A|)$, also in $O(g_{\max})$
- Insgesamt also Aufwand $O(n \cdot g_{\max}^2)$