Informatik III - Tutorium IX & X (SR -107)

Tut Nr. 10 – Üb8 & 10, Reduktionen

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
IAKS Beth

21. Januar 2009

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 09: Mittwochs 8:00 Uhr - Raum -107
Tutorium 10: Mittwochs 9:45 Uhr - Raum -107

Übungsblattabgabe Donnerstag.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Übungsblatt 10

Was wollen wir heute erreichen?

- Übungsblatt 10
- NP-Vollständigkeits-Beweise

Aufgabe 1

a) Sei $\mathcal{A} \subseteq \mathbb{N}^2$ eine Teilmenge der natürlichen Zahlen.
Seien $M = \{(a, b, c) | c \in \mathbb{N}, (a, b) \in \mathcal{A}$ und $a + b = c)\}$ und
$N = \{(b, n) | b \in \{0, 1\}^*, (n, n) \in$
$\mathcal{A}$ und b ist die Binärdarstellung von $2n\}$.
Beweisen Sie durch Reduktion:

1. Wenn M entscheidbar ist, dann ist auch N entscheidbar.

Aufgabe 1

a) Sei $\mathcal{A} \subseteq \mathbb{N}^2$ eine Teilmenge der natürlichen Zahlen.
Seien $M = \{(a, b, c) | c \in \mathbb{N}, (a, b) \in \mathcal{A}$ und $a + b = c)\}$ und
$N = \{(b, n) | b \in \{0, 1\}^*, (n, n) \in$
$\mathcal{A}$ und b ist die Binärdarstellung von $2n\}$.
Beweisen Sie durch Reduktion:

1. Wenn M entscheidbar ist, dann ist auch N entscheidbar.
2. Wenn N nicht rekursiv aufzählbar ist, dann ist auch M nicht rekursiv aufzählbar.

Aufgabe 1

a) Sei $\mathcal{A} \subseteq \mathbb{N}^2$ eine Teilmenge der natürlichen Zahlen.
Seien $M = \{(a, b, c) | c \in \mathbb{N}, (a, b) \in \mathcal{A}$ und $a + b = c)\}$ und
$N = \{(b, n) | b \in \{0, 1\}^*, (n, n) \in$
$\mathcal{A}$ und b ist die Binärdarstellung von $2n\}$.
Beweisen Sie durch Reduktion:

1. Wenn M entscheidbar ist, dann ist auch N entscheidbar.
2. Wenn N nicht rekursiv aufzählbar ist, dann ist auch M nicht rekursiv aufzählbar.

b) Sei $M = \{(i, j, k) \in \mathbb{N}^3 | \varphi_i(j + k) \downarrow\}$, dabei ist φ_i wie in der Vorlesung definiert die Funktion, die von dem **while**-Programm mit Index i berechnet wird. Beweisen Sie durch Reduktion einer geeigneten Menge, von der in der Vorlesung schon gezeigt wurde, dass sie nicht entscheidbar ist, dass M nicht entscheidbar ist.

Aufgabe 1 a)

Tafel.

Aufgabe 1 b)

Sei $H = \{i \in \mathbb{N} | \varphi_i(i) \downarrow\}$. Diese Menge ist laut Vorlesung nicht entscheidbar (ihre charakteristische Funktion würde das Halteproblem lösen). Um zu zeigen, dass M nicht entscheidbar ist, reduzieren wir die nicht entscheidbare Menge H auf M. Dazu definieren wir die totale, berechenbare Funktion

$$\begin{aligned} f: \quad \mathbb{N} &\to \mathbb{N}^3 \\ i &\mapsto (i, i, 0). \end{aligned}$$

Es gilt $f(H) \subset M$: Für alle $x \in H$ gilt $f(x) = (x, x, 0) \in M$, denn wenn $\varphi_x(x) \downarrow$, dann gilt auch $\varphi_x(x + 0) \downarrow$.

Außerdem gilt $f(\bar{H}) \subset \bar{M}$: Für alle $x \notin H$ gilt $\varphi_x(x) \uparrow$, also auch $\varphi_x(x + 0) \uparrow$.

Durch die Funktion f ist H also auf M reduzierbar. Wäre M entscheidbar, so wäre ihre charakteristische Funktion χ_M berechenbar. Damit wäre auch $\chi_M \circ f$ berechenbar. Die Funktion $\chi_M \circ f$ wäre aber genau die charakteristische Funktion von H. Damit wäre auch H entscheidbar, was nicht sein kann. Deshalb kann M nicht entscheidbar sein.

Bemerkung: *Man kann statt f auch andere Reduktionsfunktionen nehmen. Hauptsache man nimmt eine Funktion von $\mathbb{N}$ nach $\mathbb{N}^3$, die ein i so auf (i, j, k) abbildet, dass $j + k = i$.*

Aufgabe 1

Gegeben seien ein ungerichteter Graph $G = (V, E)$ mit Knoten $v \in V$ und Kanten $e = (v_1, v_2) \in E$ mit $v_1, v_2 \in V$ und zwei Farben A und B.
Beweisen Sie: Die Sprache
2-COLOR $= \{G | G$ ist ein ungerichteter Graph, für den eine totale Funktion (die *2-Färbung*) $g : V \to \{A, B\}$ existiert, so dass $\forall_{(v_1,v_2)\in E} : g(v_1) \neq g(v_2)\}$
liegt in der Komplexitätsklasse **P**.

Aufgabe 1

Wir geben einen Algorithmus mit poly. Laufzeit an:
IDEE:

Aufgabe 1

Wir geben einen Algorithmus mit poly. Laufzeit an:
IDEE:

Färbe einen Knoten v_1 mit Farbe A.

Aufgabe 1

Wir geben einen Algorithmus mit poly. Laufzeit an:
IDEE:

Färbe einen Knoten v_1 mit Farbe A.

alle Knoten v_2 mit $\{v_1, v_2\} \in E$ färbe mit Farbe B.

Aufgabe 1

Wir geben einen Algorithmus mit poly. Laufzeit an:
IDEE:

Färbe einen Knoten v_1 mit Farbe A.

alle Knoten v_2 mit $\{v_1, v_2\} \in E$ färbe mit Farbe B.

Fahre nun so für jeden Knoten v_2 fort, bis der komplette Graph gefärbt ist.

Aufgabe 1

Wir geben einen Algorithmus mit poly. Laufzeit an:
IDEE:

> Färbe einen Knoten v_1 mit Farbe A.
>
> alle Knoten v_2 mit $\{v_1, v_2\} \in E$ färbe mit Farbe B.
>
> Fahre nun so für jeden Knoten v_2 fort, bis der komplette Graph gefärbt ist.

Weil dieses Vorgehen einer Breitensuche entspricht und jede Kante einmal betrachtet wird, hat man Aufwand in $\mathcal{O}(|V| \cdot |E|)$

Aufgabe 2

Die Komplexitätsklasse **co-P** sei definiert als die Menge der Sprachen $\mathcal{L}$, deren Komplementsprache $\mathcal{L}^C$ in der Komplexitätsklasse **P** liegt. **Erinnerung:** *Zu einer Sprache $\mathcal{L}$ über einem Alphabet $\mathcal{A}$ ist die Komplementsprache $\mathcal{L}^C = \mathcal{A}^* \setminus \mathcal{L}$.*
Beweisen Sie: **P** $=$ **co-P**.

Aufgabe 2

$\mathbf{P} \subseteq$ co-P:
Sei $\mathcal{L} \in \mathbf{P}$. Dann gibt es eine polynomiale deterministische Turingmaschine T mit Zustandsmenge Q und Endzustandsmenge $F \subseteq Q$, die $\mathcal{L}$ akzeptiert, die also für alle Eingaben w genau dann nach polynomialem Aufwand in einem Endzustand $q \in F$ hält, wenn $w \in \mathcal{L}$. Konstruiere daraus die polynomiale deterministische Turingmaschine T', die genauso aussieht wie T, nur dass die Endzustandsmenge nicht F sondern $Q \setminus F$ ist. T' akzeptiert genau die Komplementärsprache zu $\mathcal{L}$, damit ist $\mathcal{L}$ auch in **co-P**.

co-P $\subseteq$ P:
Sei $\mathcal{L} \in$ **co-P**. Dann ist die Komplementsprache $\mathcal{L}^C \in \mathbf{P}$. Damit gibt es eine deterministische Turingmaschine T^C, die $\mathcal{L}^C$ in polynomialer Zeit entscheidet. Analog zu oben kann man daraus eine polynomiale deterministische Turingmaschine konstruieren, die $\mathcal{L}$ akzeptiert. Damit ist $\mathcal{L} \in \mathbf{P}$.

Wir haben nun gezeigt, dass $\mathbf{P} \subseteq$ **co-P** und **co-P** $\subseteq \mathbf{P}$, damit erhalten wir insgesamt $\mathbf{P} =$ **co-P**.

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \leq L$ ($=$ NP-hart $=$ NP-schwer)

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \leq L$ ($=$ NP-hart $=$ NP-schwer)

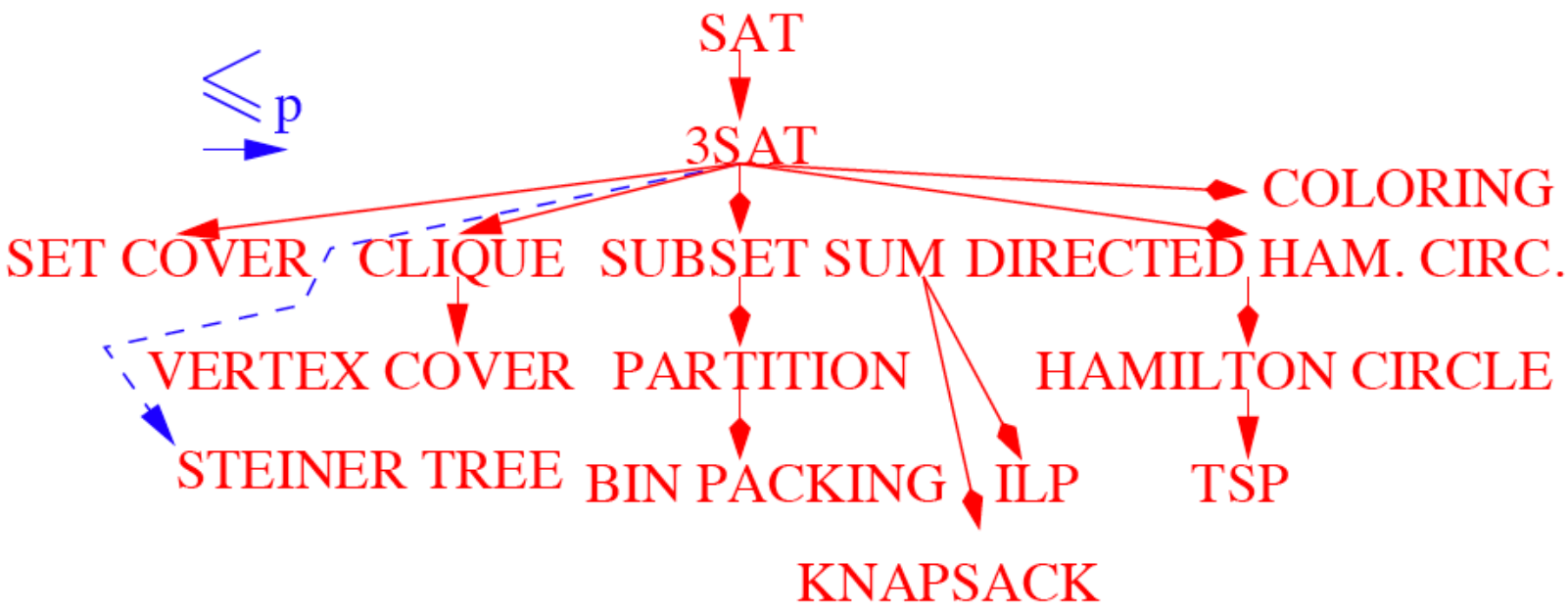
Definition: polynomiale Transformation

Eine polynomiale Transformation einer Sprache $L_1 \subseteq \Sigma_1^*$ in eine Sprache $L_2 \subseteq \Sigma_2^*$ ist eine Funktion $f : \Sigma_1^* \to \Sigma_2^*$ mit den Eigenschaften:

1. es existiert eine **polynomial deterministische** Turing-Maschine, die f berechnet
2. für alle $x \in \Sigma_1^*$ gilt: $x \in L_1 \Leftrightarrow f(x) \in L_2$

Wir schreiben dann $L_1 \leq L_2$ (L_1 ist polynomial transformierbar in L_2).

Reduktionen von SAT



Problem **DOUBLE-SAT**:
Gegeben: Eine Menge $U = \{u_1, \ldots, u_m\}$ von booleschen Variablen, eine Menge C von Klauseln über U.

Frage: Existieren *zwei verschiedene* Wahrheitsbelegungen von U, sodass C erfüllt wird, d.h., dass alle Klauseln aus C den Wahrheitswert $wahr$ annehmen?

Problem **DOUBLE-SAT**:
Gegeben: Eine Menge $U = \{u_1, \ldots, u_m\}$ von booleschen Variablen, eine Menge C von Klauseln über U.

Frage: Existieren *zwei verschiedene* Wahrheitsbelegungen von U, sodass C erfüllt wird, d.h., dass alle Klauseln aus C den Wahrheitswert $wahr$ annehmen?

Aufgabe 1

Zeige, dass das Problem **DOUBLE-SAT** $\mathcal{NP}$-vollständig ist.

Aufgabe 2

Zeige, dass TRAVELING SALESMAN $\mathcal{NP}$-vollständig ist.
Reduziere hierbei von HAMILTONIAN CIRCUIT.

Aufgabe

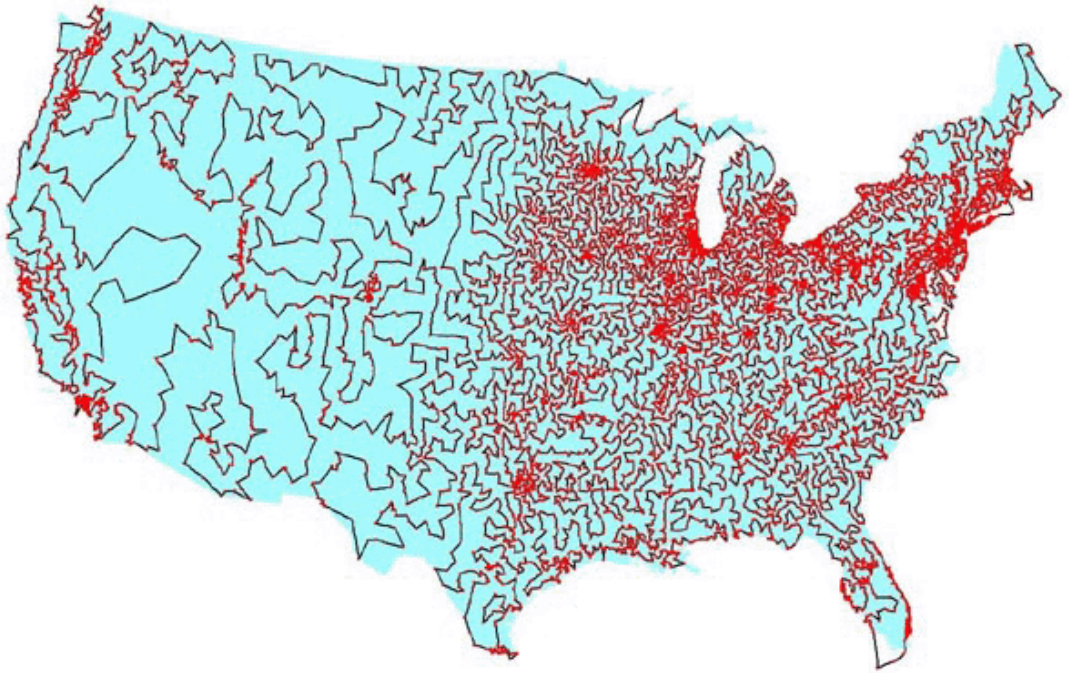
Problem **HAMILTONIAN CIRCUIT**:
Gegeben: Ein Graph $G = (V, E)$
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten $v_i \in V (i = 1, \ldots, n)$ mit $n := |V|$ und $\{v_j, v_{j+1}, \{v_n, v_1\} \in E (j = 1, \ldots, n-1)\}$?
Hinweis: HAMILTONIAN CIRCUIT ist $\mathcal{NP}$-vollständig.

Aufgabe

Problem **HAMILTONIAN CIRCUIT**:
Gegeben: Ein Graph $G = (V, E)$
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten $v_i \in V (i = 1, \ldots, n)$ mit $n := |V|$ und $\{v_j, v_{j+1}, \{v_n, v_1\} \in E (j = 1, \ldots, n-1)\}$?
Hinweis: HAMILTONIAN CIRCUIT ist $\mathcal{NP}$-vollständig.

Problem **TSP**:
Gegeben: Ein Graph $G = (V, E)$ und Parameter k
Frage: Existiert ein einfacher Kreis in G, der alle Knoten in V enthält, also eine Folge $(v_1, \ldots, v_n)$ von paarweise verschiedenen Knoten mit Kostenfunktion $c \leq k$?

Beispiel:



Rundreise durch die USA mit Besuch von jedem Ort, mit mehr als 500 Einwohnern.

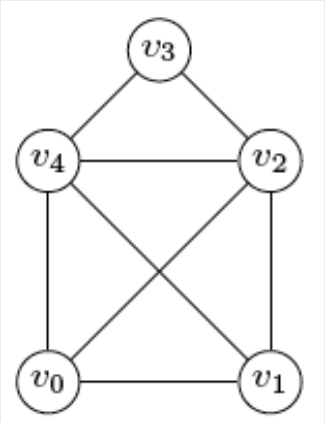
1. Schritt: zu zeigen: **TSP** liegt in $\mathcal{NP}$
Rate Lösung (Folge von Knoten) und überprüfe. $\mathcal{O}(n)$.

1. Schritt: zu zeigen: **TSP** liegt in $\mathcal{NP}$
Rate Lösung (Folge von Knoten) und überprüfe. $\mathcal{O}(n)$.

2. Schritt: **HAMILTONIAN CIRCUIT** $\leq$ **TSP**
Instanz von **HC**: $G = (V, E), V = \{v_1, \ldots, v_n\}$
Die korrespondierende Instanz von **TSP** enthält n Orte, die Gewichte c_{ij} sind 0, wenn $(v_i, v_j) \in E$ und 1 sonst. Kostengrenze k ist n.
Offensichtlich ist diese Transformation polynomial.
Damit gilt:
G enthält einen **HC** genau dann, wenn c eine Rundreise mit Kosten kleiner n zulässt. Sei $(v_{i_1}, \ldots, v_{i_n}, v_{i_1})$ ein **HC** in G, dann sind dessen Kosten gleich n.
Sei $(v_{i_1}, \ldots, v_{i_n}, v_{i_1})$ eine Rundreise mit Kosten n, dann gilt für $i = 1, \ldots, n-1$, dass $(v_i, v_{i+1}) \in E$ und $(v_n, v_1) \in E$. Damit ist $(v_{i_1}, \ldots, v_{i_n}, v_{i_1})$ ein **HC** in G.

Aufgabe 3

Gegeben sei folgender Graph:



Gibt es einen Hamiltonkreis? Wandeln Sie hierzu das Problem in ein TSP um und finden Sie eine optimale Rundtour.

Lösung:

Eine optimale Rundtour ist z.B.: $(v_0, v_1, v_2, v_3, v_4, v_0)$ mit Kosten 5, also ist diese optimale Rundtour (da ihre Kosten nicht grösser als 5 sind) auch ein Hamiltonkreis.

Aufgabe 4

Betrachte das Problem $\mathrm{HALFCLIQUE}$:
Gegeben: Graph $G = (V, E)$ mit $|V|$ gerade.
Frage: Gibt es eine Clique der Grösse $|V|/2$ in G?
Zeige, dass $\mathrm{HALFCLIQUE}$ $\mathcal{NP}$-vollständig ist.

Aufgabe 4

Betrachte das Problem $\mathrm{HALFCLIQUE}$:
Gegeben: Graph $G = (V, E)$ mit $|V|$ gerade.
Frage: Gibt es eine Clique der Grösse $|V|/2$ in G?
Zeige, dass $\mathrm{HALFCLIQUE}$ $\mathcal{NP}$-vollständig ist.

Problem **CLIQUE**:
Gegeben: Ein Graph $G = (V, E)$ und einen Parameter $k \leq |V|$
Frage: Gibt es in G eine Clique der Grösse mindestens k? Hinweis: CLIQUE ist $\mathcal{NP}$-vollständig.

Aufgabe

The problem is NP-complete.

To show that it is in NP, we give a non-deterministic polynomial-time algorithm for recognizing "yes" instances. This algorithm non-deterministically guesses a set of $n/2$ vertices, and then checks whether it is a clique.

To show that it is NP-hard, we give a reduction from the NP-complete problem **Clique**. Let (G, j) be an instance of **Clique**. (Check your lecture notes to remind yourself about this problem.) We will construct an instance G' of **HalfClique** such that (G, j) is a "yes" instance of **Clique** iff G' is a "yes" instance of **HalfClique**.

We do the construction in two cases. Let $V(G)$ be the set of vertices of G, so $|V(G)|$ is the number of vertices of G.

Aufgabe

Case 1: If $j \leq |V(G)|/2$ then we will construct a graph G' with $2|V(G)| - 2j$ vertices by adding $|V(G)| - 2j$ vertices to G. Connect each of these new vertices to every vertex in G' (except itself). We claim that G has a clique of size j iff G' has a clique of size $|V(G)| - j$. So (G, j) is a "yes" instance of **Clique** iff G' is a "yes" instance of **HalfClique**. Here is why the claim is true.

1. If G' has a clique of size $|V(G)| - j$ then at least j of these are from G and these are a clique.

2. If G has a clique of j then these plus the new vertices make a clique of size $|V(G)| - j$ in G'.

Case 2: If $j > |V(G)|/2$ then construct a graph G' with $2j$ vertices by adding $2j - |V(G)|$ new vertices and no new edges. We claim that G has a clique of size j iff G' has a clique of size j. So (G, j) is a "yes" instance of **Clique** iff G' is a "yes" instance of **HalfClique**.

Problem **PARTITION**:
Gegeben: Natürliche Zahlen $a_1, a_2, ..., a_k \in \mathbb{N}$.
Frage: Gibt es eine Teilmenge $J \subseteq \{1, 2, ..., k\}$ mit $\sum_{i \in J} a_i = \sum_{i \notin J} a_i$?
Hinweis: PARTITION ist $\mathcal{NP}$-vollständig.

Problem **PARTITION**:
Gegeben: Natürliche Zahlen $a_1, a_2, ..., a_k \in \mathbb{N}$.
Frage: Gibt es eine Teilmenge $J \subseteq \{1, 2, ..., k\}$ mit $\sum_{i \in J} a_i = \sum_{i \notin J} a_i$?
Hinweis: PARTITION ist $\mathcal{NP}$-vollständig.

Problem **BIN PACKING**:
Gegeben: Eine Behältergrösse $b \in \mathbb{N}$, die Anzahl der Behälter $k \in \mathbb{N}$, Objekte $a_1, a_2, ..., a_n \leq b$
Frage: Können die Objekte so auf die k Behälter verteilt werden, dass kein Behälter überläuft? (Das heisst: gesucht ist, ob eine Abbildung $f : \{1, ..., n\} \longrightarrow \{1, ..., k\}$ existiert, sodass für alle j=1,...k gilt: $\sum_{f(i)=j} a_i \leq b$).

Aufgabe 5

a) Zeigen Sie: Bin Packing ist NP-hart.

Aufgabe 5

a) Zeigen Sie: Bin Packing ist NP-hart.
b) Gegeben seien die Objekte der PARTITION Probleminstanz $(a_1, a_2, a_3, a_4, a_5, a_6) = (1, 1, 2, 3, 4, 5)$. Zeigen oder widerlegen Sie, ob das entsprechend transformierte und das ursprüngliche Problem eine Lösung besitzen.

Lösung 5 a):

Hier kann man die Reduktion PARTITION$\leq_p$BIN PACKING verwenden: $(a_1, ..., a_k) \mapsto$

$$\begin{cases} \text{Behältergrösse:} & b = \sum_{i=1}^{k} a_i/2 \\ \text{Zahl der Behälter:} & k = 2 \\ \text{Objekte:} & a_1, ..., a_k \end{cases}$$

Falls jetzt eine Lösung des PARTITION Problems existiert, dann gilt $\sum_{i \in J} a_i = \sum_{i \notin J} a_i$. Dann folgt:
$\sum_{i=1}^{k} a_i = \sum_{i \in J} a_i + \sum_{i \notin J} a_i = 2 * \sum_{i \in J} a_i \Rightarrow \sum_{i=1}^{k} a_i/2 = \sum_{i \in J} a_i$
So kann man die Menge J auch in einen Behälter tun, der dann das Gewicht $\sum_{i \in J} a_i = \sum_{i=1}^{k} a_i/2$ hat und die restlichen Objekte (die nicht in der Menge J sind) in den anderen Behälter.

Lösung 5 b):

Für die Objekte des BIN PACKING Problems kann man eine Lösung finden, wenn man wie oben die Abbildung anwendet: $k = 2, b = \sum_{i=1}^{6} a_i/2 = (1+1+2+3+4+5)/2 = 8$. Somit ist das BIN PACKING Problem mit der Menge $J = \{1, 4, 5\}$ (Indizes!!!) gelöst, also $\sum_{i \in J} a_i \leq b = 8$ sowie $\sum_{i \notin J} a_i \leq b = 8$. Deshalb soll die ursprüngliche Probleminstanz auch für PARTITION Problem eine Lösung besitzen: $\sum_{i \in J} a_i = \sum_{i \notin J} a_i = 8$

Aufgabe 6

Zeigen Sie, dass die polynomielle Reduzierbarkeit transitiv ist.
Aus $A \leq_p B$ und $B \leq_p C$ folgt $A \leq_p C$.

Aus $A \leq_p B$ folgt, dass es eine Abbildung $f : A \to B$ gibt, die

$$w \in A \text{ genau dann, wenn } f(w) \in B$$

erfüllt. Der Aufwand, $f(w)$ zu ermitteln, sei durch das Polynom p beschränkt. Ebenso erhalten wir aus $B \leq_p C$, dass es eine Funktion $g : B \to C$ gibt, die

$$x \in B \text{ genau dann, wenn } g(x) \in C$$

erfüllt. Der Aufwand, $g(x)$ zu ermitteln, sei durch das Polynom q beschränkt. Aus den beiden Abbildungeigenschaften erhalten wir

$$w \in A \text{ geanu dann, wenn } g(f(w)) \in C$$

Bei der Aufwandabschätzung von $g \circ f$ beachten wir, dass der Aufwand aus der Summe des Aufwands für die Anwendung der beiden Abbildungen ist. Er wird beschränkt durch $p(|w|) + q(|f(w)|)$. Da die Summe zweier Polynome wieder ein Polynom ist, ist A polynomiell auf C reduzierbar.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Polynomiale Transformierbarkeit kennen gelernt

Noch Fragen?

Vorschau

Vorschau

Vorschau

- .
- .

Bis zum nächsten Mal