Gruppe $\boxed{\mathbf{17}}$ Philipp Hochmann, 356148
Berechenbarkeit & Komplexität Felix Kiunke, 357322

Blatt 11

Aufgabe 11.1

Reduktionsabbildung: Jede Klausel $(a \vee b \vee c)$ der Eingabe wird transformiert zu

$$(*) \qquad (a \vee b \vee \bar{x}) \wedge (a \vee c \vee \bar{y}) \wedge (c \vee b \vee \bar{z}) \wedge (x \vee y \vee z)$$

wobei x, y, z o.B.d.A. keine Literale in der Eingabe sind. Dies ist sicherlich polynomiell (linear in der Anz. der Klauseln).

Beweis. *Korrektheit*

$w \in$ 3SAT
$\Longrightarrow$ In jeder Klausel $(a \vee b \vee c)$ von w kommt mindestens ein wahres Literal vor
$\Longrightarrow$ Es können x, y, z so gewählt werden, dass im Ausdruck $(*)$ in jeder Klausel mindestens ein Literal wahr und ein Literal falsch auftritt:
(a) Sind mehr als ein Literal von a, b, c wahr, so setze x und y jeweils auf *wahr* und z auf $falsch$
(b) Ist nur eines wahr, so setze x auf $falsch$, wenn es c ist, y auf $falsch$, wenn es b ist, und z auf $falsch$, wenn es a ist.
$\Longrightarrow$ Jede Klausel von $(*)$ hat mind. ein wahres und mind. ein falsches Literal, insb. ist die Belegung dann erfüllend
$\Longrightarrow$ $w \in$ Not-all-equal-SAT

$w \notin$ 3SAT
$\Longrightarrow$ In mind. einer Klausel $(a \vee b \vee c)$ von w kommt kein wahres Literal vor
$\Longrightarrow$ $(*)$ könnte dann nur erfüllend gemacht werden, wenn x, y, z jeweils auf $falsch$ gesetzt werden, dies ist wegen der letzten Klausel von (*) allerdings nicht erlaubt
implies Es gibt keine erfüllende Belegung von $(*)$
$\Longrightarrow$ $w \notin$ Not-all-equal-SAT

Es gilt daher 3SAT $\leq_p$ Not-all-equal-SAT.

Aufgabe 11.2

Algorithmus:
Solange der Graph noch Kanten enthält, wähle zufällig eine Kante, füge ihre Endknoten zum Ergebnis hinzu, und entferne alle zu diesen ihren Endknoten inzidenten Kanten.

Beweis 2-Approximation:
Ein Endknoten ist auf jeden Fall auch Teil der optimalen Lösung. Der Andere nicht notwenigerweise. Deshalb ist die berechnete Approximation schlimmstenfalls doppelt so groß.

Aufgabe 11.3

Wenn LongestPath in P liegt, so können wir, um HC zu lösen, auf die Eingabe von HC $G = (V, E)$ LongestPath mit $b = |V| - 1$ ausführen. Wenn kein Pfad der Länge b existiert, kann es auch keinen Hamiltonkreis geben. Wenn einer existiert, so kann es einen Hamiltonkreis geben, wenn eine Kante die zwei Endknoten des Pfades verbindet. Um das in polynomieller Zeit zu testen, gehen wir folgendermaßen vor:

Für jede Kante $e \in E$: Entferne e aus dem Graphen und füge zwei neue Knoten v_1 und v_2 ein. Füge außerdem zwei neue Kanten e_1 und e_2 hinzu, die zwischen den beiden Endknoten von e und v_1 bzw. v_2 verlaufen. Führe LongestPath auf $G' = (V \cup \{v_1, v_2\}, (E \setminus \{e\}) \cup \{e_1, e_2\})$ mit $b' = b + 2$ durch. Wenn nun ein Pfad der Länge b' existiert, so muss er die beiden Knoten v_1 und v_2 enthalten. Da diese Knoten nur über eine Kante mit dem restlichen Graphen verbunden sind, müssen sie Endknoten sein. Also existiert auch ein Pfad der Länge b in G, dessen Endknoten die beiden Enden von e sind. Da wir die Kante e in G' entfernt haben, können wir sie auch wieder zum Pfad hinzufügen, um ihn zu schließen und zu einem Hamiltonkreis zu machen. Wir geben also Ja zurück. Wenn kein Pfad der Länge b' existiert, fahren wir mit der nächsten Kante fort. Wenn wir alle Kanten abgearbeiten haben, geben wir Nein zurück, dann existiert kein Hamiltonkreis in G.

Da wir LongestPath höchstens $|E|+1$-mal durchführen und alle anderen Schritte offensichtlich auch polynomiell sind, können wir HC in Polynomialzeit berechnen, wenn LongestPath in P liegt. Da HC in NP liegt, kann LongestPath also auch nicht in P liegen (unter der Annahme, dass $\mathsf{P} \neq \mathsf{NP}$). $\square$