Theoretische Grundlagen der Informatik II
Blatt 4

Markus Vieth, David Klopp, Christian Stricker

17. November 2015

Aufgabe 1

1. INTEGER PROGRAMMING

Eingabe: Matrix C aus ganzen Zahlen und ein ganzzahliger Vektor d
Beschreibung: Es gibt einen Vektor x bestehend aus 0 und 1, so dass Cx = d gilt.

2. CLIQUE:

Eingabe: Graph G, positive ganze Zahl k
Beschreibung: Es gibt eine Menge der Größe k von Knoten, die alle paarweise untereinander verbunden sind.

3. SET PACKING:

Eingabe: Familie von Mengen $\{S_j\}$, positive ganze Zahl l
Beschreibung: Beinhaltet $\{S_j\}$ l paarweise disjunkte Teilmengen.

4. NODE COVER:

Eingabe: Graph G', positive ganze Zahl l
Beschreibung: Es gibt eine Knotenmenge der Größe l, sodass alle Kanten mit mindestens einem Knoten aus dieser Menge verbunden sind.

5. SET COVERING:

Eingabe: endliche Familie von endlichen Mengen $\{S_j\}$, positive ganze Zahl k
Beschreibung: Es gibt eine Unterfamilie $\{T_h\} \subseteq \{S_j\}$ mit $\leq$ k Mengen, sodass $\cup T_h = \cup S_j$.

6. UNDIRECTED HAMILTON CIRCUIT:

Eingabe: Graph G
Beschreibung: G enthält einen Zirkel, der jeden Knoten genau einmal beinhaltet.

7. DIRECTED HAMILTON CIRCUIT:

Eingabe: gerichteter Graph H
Beschreibung: G enthält einen gerichteten Zirkel, der jeden Knoten genau einmal beinhaltet.

8. CLIQUE COVER:

Eingabe: Graph G', positive ganze Zahl l
Beschreibung: N' ist die Vereinigung von l oder weniger Cliquen.

9. Knapsack:

Eingabe: $(a_1, a_2, ..., a_r, b) \in Z^{n+1}$
Beschreibung: $\sum a_j x_j = b$ hat eine 0-1 Lösung.

10. PARTITION:

Eingabe: $(c_1, c_2, ..., c_s) \in Z^s$
Beschreibung: Es gibt eine Menge $I \subseteq \{1, 2, ..., s\}$, sodass $\sum_{h \in I} c_h = \sum_{h \notin I} c_h$.

Aufgabe 2

Sei $f(x_1, ..., x_n)$ ein boolscher Ausdruck mit den Klauseln $Y_1, ..., Y_k$.

Behauptung: SAT $\in$ NP

Guess

Es werden nichtdeterministisch die Wahrheitsbelegungen von x_1 bis x_n geraten.

Check

For Y_i in $f(x_1, .., x_n)$:
  KlauselIstWahr = true
  For x_i in Y_i:
// Falls ein Term 0 ist, wird die ganze Klausel false
    If $x_i == 0$:
      KlauselIstWahr = false
//Wenn eine Klausel wahr ist, ist der ganze boolsche Ausdruck wahr
  If KlauselIstWahr == true:
    return Ja
return Nein

Im worst-case beträgt die Laufzeit O(k*n) => $SAT \in NP$

Aufgabe 3

Reduktion 3-SAT $\leq$ CLIQUE

Seien $Y_1, ...Y_n$ die Klauseln des boolschen Ausdrucks.

Graph bilden:

Bilde die Menge aller Knoten V:
V = $\{< \sigma, i > \mid$ wobei σ ein Literal aus der Klausel Y_i meint $\}$
D.h jedes Literal pro Klausel ist eindeutig durch einen Knoten vertreten.

Bilde die Menge der Kanten E:
E = $\{\{< \sigma, i >, < \delta, j >\}\ i \neq j \vee \sigma \neq \delta\ \}$ D.h verbinde alle Kanten miteinander, die nicht in der selben

Klausel sind und bei denen das Literal nicht der Negation des Literals entspricht.

Es ergibt sich somit der Graph G(V,E). Wähle k = $\#\{Y_1, ...Y_n\}$. D.h k entspricht der Anzahl von Klauseln des boolschen Ausdrucks.
Diese Operationen lassen sich in poylnomieller Laufzeit erstellen.

Wende nun den CLIQUEN-Algorithmus an. Die Lösung von 3-SAT ist nun äquivalent zur Lösung von CLIQUE.

q.e.d