BuK WS 2020/21
Tutorium 04
30. Januar 2021

Til Mohr, 405959
Andrés Montoya, 405409
Marc Ludevid Wulf, 405401

Übungsblatt 09

Aufgabe 4

```
Input G=(V,E)

v_start := getFirst(V)
K := {v_start}
E' := E
weights := sumAllWeights(E')
tourWeights := 0

while !SetEquals(K,V):
    u := getLast(K)
    E'' := E'

    while true:
        if(E'' = ∅):
            return ∅ //Error


        weights' := weights
        tourWeights' := tourWeights
        E''' := E'

        v := getMinCostFrom(E'', u)
        cost := getCost(E'', (u,v))
        weights' −= cost
        tourWeights' += cost

        forall(k in K && k ≠ v_start):
            E''' = removeIfExists(E''', (k,v))

        if(TSP-E ((V',E'''), (tourWeights' + getCost(E'',(v,v_start)))))
            return add(K,v)
        else if(TSP-E ((V',E'''),weights))
            K = add(K,v)
            E' = E'''
            weights = weights'
            tourWeights = tourWeights'
            break
        else
            E'' = remove(E'', (u,v))


return K
```

TODO: Korrektheit + Laufzeit

Aufgabe 5

Hierzu nehmen wir uns eine Formel φ in 3-KNF mit den Klauseln k_i. Jedes k_i in φ ist daher der Form $(x_i \vee y_i \vee z_i)$. Nun teilen wir jedes k_i in zwei weitere Klauseln k_i' und k_i'', wobei wir auch noch für jede Klausel eine Variable c_i, und für alle Klauseln die Variable $f = 0$ einführen.
Aus k_i wird also $k_i' \wedge k_i'' = (x_i \vee y_i \vee c_i) \wedge (\bar{c}_i \vee z_i \vee f)$. Hierbei wird c_i auf $\neg(x_i \vee y_i)$

gesetzt.
Diese Konstruktion können wir in Linearer Zeit erstellen.

Korrektheit:
Sei φ in 3-KNF mit gegebener Belegung der Variablen. Dann ist jede Klausel k_i der Form $(x_i \vee y_i \vee z_i)$.
Hieraus konstruieren wir eine neue aussagenlogische Formel φ' wie oben beschrieben.

- Falls x_i oder y_i wahr ist, so setzen wir c_i auf 0. Damit ist zum einen die Klausel k_i' wahr, zum anderen gibt es in dieser Klausel mindestens sowohl ein wahres, als auch ein falsches Literal. Desweiteren ist dadurch (unabhängig von z_i) auch k_i'' wahr, da dort $\bar{c_i}$ enthalten ist. Auch diese Klausel besitzt sowohl ein wahres, als auch ein falsches Literal (f).
- Falls x_i und y_i falsch sind, aber z_1 wahr, dann setzen wir c_i auf 1. Damit ist die erste Klausel (k_i') wahr und besitzt auch wieder mindestens ein wahres, als auch ein falsches Literal. Die Klausel k_i' ist aufgrund von z_i auch wahr, und besitzt auch wieder ein wahres und ein falsches Literal (f).
- Falls keines der drei Literale wahr ist, so gibt es ja keine Erfüllende Belegung der Variablen, weshalb die gesamte Formel nicht in 3-SAT ist. Trotzdem setzen wir hier ja c_i auf 1, weshalb ja die Klausel k_i' war ist. Jedoch ist die Klausel k_i'' folglich nicht wahr, weshalb die resultierende Formel auch nicht in NOT-ALL-EQUAL-SAT ist.

Damit ist der Wahrheitsgrad von φ' immer genau derselbe wie von φ für jede Belegung. Hat also φ keine erfüllende Belegung (nicht in 3-SAT), so hat auch φ' keine erfüllende Belegung (nicht in NOT-ALL-EQUAL-SAT).
Hat aber φ eine erfüllende Belegung (ist in 3-SAT), so hat φ' eine erfüllende Belegung, die die gleichen Variablen gleich belegt, und weitere Variablen so belegt, dass jede Klausel mindestens ein wahres und ein falsches Literal besitzt (ist in NOT-ALL-EQUAL-SAT).

Aufgabe 6

(a) Hierzu bauen wir uns einen Verifizierer, welches als Zertifikat einen Vektor $x \in \{0,1\}^n$ übergeben bekommt, welches einfach codiert werden kann als $x_1...x_n$.
Unser Verifizierer überprüft nun Folgende Dinge:

- Eingabe korrekt formatiert (Linearer Zeitaufwand in n).
- Überstimmt die Vektordimension n von x überein mit der gegebenen Matrix A und dem Vektor b? (höchstens Quadratischer Zeitaufwand in $n \cdot m$, je nach Codierung der Matrix A)
- Berechnung von Ax (Quadratischer Zeitaufwand in $n \cdot m$)
- Abgleich, ob $Ax \geq b$ (Linearer Zeitaufwand in n).

Damit läuft der Verifizierer in polynomieller Zeit und $\{-1,0,1\}$-RESTRICTED INTEGER PROGRAMM ist in NP.

(b)