GTI Übungsblatt 11
Tutor: Marko Schmellenkamp
ID: MS1
Übung: Mi 16-18

Max Springenberg, 177792

11.1

11.1.1

Wir betrachten das Hilfs-Programm Prog in Pseudo-Code:
Eingaben: Knoten v, Array mit boolschen Werten für die Markierung von Knoten mk, $k \in \mathbb{N}$

1. IF $k > 0$ THEN
2. IF $\neg \bigvee_{u \in adj(v)} mk[u]$ THEN
3. return false
4. $V' = \{u \in adj(v) | \neg mk[u]\}$
5. return $\bigvee_{u \in V'} Prog(u, mk', k-1)$, mit mk' als mk mit $mk[u] = true$
6. return true

Dieses Programm testet, ob von einem Knoten aus alle möglichen Wege bis zur Länge k.
Dabei wird über die Adjazenzlisten in $O(n^2)$ iteriert und insegesamt $O(n^2)$ rekursive Aufrufe, je Rekursionsschritt getätigt, die jedoch nur maximal k mal ausgeführt werden, mit $n = |V|$.
$O(\prod_{i=1}^{k} n^2) = O(n^{2*k})$

Da k eine konstante ist, ist die Laufzeit für k-WEG-FORTSETZUNG noch polynomiell.

Ist k nun aber eine codierte Eingabe, die eine Zahl repräsentiert, so gilt für den Algorithmus, dass er $O(n^{2*2^{|k|}}) = O(n^{2^{|k|+1}})$ benötigt, da exponentiell große Zahle in logarithmischer Länge codiert werden. Ferner ist die Laufzeit dann auch nichtmehr polynomiell.

11.1.2

ALG_{WF} sei ein Algorithmus, der WEG-FORTSETZUNG löse.

(i) ALG_{WFW} sei wie folgt definiert:

1. FOR $k := |V|$ DOWNTO 1 DO
2. IF $ALG_{\text{WF}}(G, w, k)$ THEN
3. return k

Die maximale Länge eines Weges ist $|V|$, da jeder Knoten Höchstens einmal besucht werden kann. Wenn getestet werden kann, ob ein Weg der Länge k möglich ist, so kann von dem größt möglichem bis zu den je nächst grösßten in polynomieller Zeit getestet werden, welches k die Länge der maximalen Fortsetzung wäre.
Damit Kann WEG-FORTSETZUNG-O in polynomieller Zeit berechnet werden, wenn WEG-FORTSETZUNG in polynomieller Zeit berechnet werden kann.

(ii) ALG_{WFO} sei wie folgt definiert:

1. $k_{\max} = ALG_{\text{WFW}}$(G,w)
2. FOR $v \in V$ DO
3. $G' = (V - \{v\}, E - \{\{v, u\} | u \in V\})$
4. IF $ALG_{\text{WFW}}(G, w) = k_{\max}$ THEN
5. G = G'
6. return V

Wenn die Länge einer maximalen Fortsetzung ermittelt werden kann, so kann für jeden Knoten, sowie dann auch seinen inzidenten Kanten in polynomieller Zeit getestet werden, ob diese Auf dem Weg liegen, bzw. in dem Weg enthalten sind.
Wenn man alle Knoten und Kanten entfernt, die nicht auf dem Weg liegen, bzw. in diesem enthalten sind, so bleibt nur noch der Weg übrig.
Damit Kann WEG-FORTSETZUNG-O in polynomieller Zeit berechnet werden, wenn WEG-FORTSETZUNG-W in polynomieller Zeit berechnet werden kann.

11.2

11.2.1

Eine mögliche Lösung wäre das Wort $w \stackrel{\text{def}}{=} c_1 c_3 c_2$

11.2.2

Damit ein Problem in NP liegt müssen 3 Eigenschaften gelten:

1. Es gibt einen Suchraum von Lösungskandidaten
2. Die Lösungskandidaten sind polynomiell groß in der Eingabe
3. Jeder Lösungskandidat kann in polynomieller Zeit getestet werden

1.
Der Suchraum von Lösungen ist

$$L_{\text{Such}} = \{w \in \Sigma^* | \forall \sigma \in \Sigma : \#_\sigma(w) > 0\}$$

3.
Die Zusatzeingabe ist ein Wort w aus L_{Such}.
Nach Der Vorlesung kann zu jedem RE ein $\epsilon - NFA$ und zu jedem $\epsilon - NFA$ ein NFA in polynomieller Zeit konstruiert werden.
Der zugehörige NFA A_β testet dann in polynomieller Zeit, ob w in der Sprache $L(\beta)$ ist.
Damit kann ein Lösungskandidat in polynomieller Zeit getestet werden.

2.
Der Ausdruck, das Alphabet und die Zusatzeingabe haben alle polynomielle Größe in ihrer Eingabe.

Da alle Eigenschaften erfüllt werden können gilt: $ALLEZEICHEN \in NP$

11.2.3

$\phi = (\neg x_2 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_3 \vee x_1) \wedge (x_2 \vee x_3 \vee \neg x_3)$

$f(\phi) = (\beta_\phi, \Sigma_\phi)$, mit:

$\Sigma_\phi = \{c_1, c_2, c_3\}$

$\beta_\phi = (p(x_1) + n(x_1))(p(x_2) + n(x_2))(p(x_3) + n(x_3))$
$= (\epsilon\epsilon\epsilon + c_1c_2\epsilon)(c_1\epsilon\epsilon + \epsilon c_2c_3)(\epsilon c2c_3 + \epsilon\epsilon c_3)$
$= (\epsilon + c_1c_2)(c_1 + c_2c_3)(c_2c_3 + c_3)$

11.2.4

Es existieren m Variablen und k Klauseln in ϕ.

Für β_ϕ:
Es wird für jede Variable eine der String von Klauseln-repräsentierenden Zeichen erstellt die sie erfüllt, wenn sie positiv - und der String von Klauseln-repräsentierenden Zeichen erstellt, die sie erfüllt wenn sie neativ belegt wird.

Insgesamt werden also $2 * m * k$ Teilausdrücke erstellt, verodert und konkateniert.
Dies geht in $O(m * k)$

Für Σ_ϕ:
k Klauseln-repräsentierende Zeichen können in $O(k)$ erstellt und mit einem leer initialisiertem Alphabet vereinigt werden.

Damit ist der Aufwand der Funktion in $O(m * k)$ und insbesondere polynomiell.

11.2.5

w_i sei der $i-te$ reguläre Teilausdruck von β_ϕ mit $w_i = (p(x_i) + n(x_i)), 1 \leq i \leq m$, β_ϕ kann also auch als Konkatenation aller w_i geschrieben werden.

1. $\phi \in SAT \Rightarrow f(\phi) \in ALLEZEICHEN$

Wir wissen:
(i) Wenn die Variable $x_i, 1 \leq i \leq m$ die Klausel $\psi_j, 1 \leq j \leq k$ unter einer Belegung α erfüllt, so ist das Klausel-repräsentierende Zeichen c_j in w_i durch $p(x_i)$ wählbar.
(ii) Wenn die Variable $\neg x_i, 1 \leq i \leq m$ die Klausel $\psi_j, 1 \leq j \leq k$ unter einer Belegung α erfüllt, so ist das Klausel-repräsentierende Zeichen c_j in w_i durch $n(x_i)$ wählbar.

Wenn es eine Belegung α gibt, die ϕ erfüllt, so gilt, dass die Variablen unter der Belegung auch alle Klauseln der KNF erfüllen, demnach sind alle Klauseln-repräsentierenden Zeichen und ferner alle Zeichen des Alphabets durch (i) oder (ii) in β wählbar und ein Wort mit allen Zeichen des Alphabets existiert.
Damit gilt dann auch $f(\phi) \in ALLEZEICHEN$

2. $f(\phi) \in ALLEZEICHEN \Rightarrow \phi \in SAT$
$f(\phi) \in ALLEZEICHEN \Rightarrow \exists w \in L(f(\phi)) : \#_\sigma(w) > 1, \forall \sigma \in \Sigma$
Ferner wissen wir, dass β_ϕ auch als Konkatenation der Teilausdrücke w_i geschrieben werden kann. w_i enthält Strings mit Klauseln-repräsentierenden Zeichen für ein positive oder negative Belegung der jeweiligen Variablen x_i.
Wenn nun alle Zeichen in einem Wort w vorkommen, so gibt es eine Möglichkeit aus den einzelnen Teilausdrücken w_i dieses zu bilden. Ferner existiert damit die Möglichkeit alle x_i so zu belegen, dass alle Klauseln-repräsentierenden Zustände gewählt werden und ferner alle Klauseln erfüllt sind.
Wenn eine Belegung alle Klauseln erfüllt, so erfüllt sie auch ϕ.
Damit gilt auch $f(\phi) \in ALLEZEICHEN \Rightarrow \phi \in SAT$.

11.2.6

Wir haben zwei Bedingungen:
1. Jede Kante hat einen Knoten in S
2. $|S| \leq k$

Idee:
f konstruiert unter anderem für einen Graphen einen RE, der für jeden Knoten einen Teilausdruck

aus der Konkatenation eines Repräsentaten für die existens eines Knoten und Repräsentaten für alle zum Knoten inzidenten Kanten bereitstellt. Alle Teilausdrücke werden verodert, da Knoten gewählt werden können.
Für jeden für $VERTEXCOVER$ gewählten Knoten, kann also ein Teilausdruck mit einem Repräsentanten für die existenz eines Knoten und Repräsentanten für dessen inzidente Kanten gewählt werden.
Danach kann überprüft werden wie viele Repräsentanten für die existenz eines Knotens vorkommen (Anzahl der Knoten) und ob auch Repräsentanten für alle Kanten aus E vorkommen (jede Kante einen Knoten hat).
Σ_i, m_i befassen sich je mit der $i-ten$ oben genannten Bedingung.

Wir betrachten die Funktion f mit:
$G = (V, E)$
$f(G, k) = (\alpha, m_1, m_2, \Sigma_1, \Sigma_2)$, mit:

$\Sigma_1 = \{\sigma_{i,j} | \{i, j\} \in E\}$
$m_1 = 1$

$\Sigma_2 = \{\$\}$
$m_2 = k$

$\bigcirc$ sei das Äquivalent der Summenformel $\sum$ zur Adition, für die Konkatenation.Also Eine Konkatenation über Eleme
$\alpha = \alpha_1 + \ldots + \alpha_{|V|}$
$\forall l, 1 \leq l \leq |V| : \alpha_l = \$ \bigcirc_{(i,j)\in E} \ \sigma_{i,j}\sigma_{j,i}$

f ist total, da wir keine Definitionslücken definiert haben, und polynomiell berechenbar, da lediglich über Mengen iteriert wird.

Zu Zeigen bleibt:

(1) $(G, k) \in VERTEXCOVER \Rightarrow f(G, k) \in MINMAXZEICHEN$
Aus $(G, K) \in VERTEXCOVER$ folgt, dass nur maximal k Knoten gewählt wurden und alle Kanten einen Knoten in S haben.
Für $f(G, k)$ bedeutet das insbesondere, dass es k Teilausdrücke $\alpha_l, 1 \leq l \leq |V|$ gibt, die Repräsentanten aller Kanten als Teilwörter enthalten, da für jeden Knoten Repräsentanten aller seiner Kanten im ihm designierten Teilausdruck vorhanden sind.
So gilt für das Wort w der zu den Knoten aus S korrespondierenden konkatenierten Teilausdrücke:

$$\forall \sigma \in \Sigma_1 : \#_\sigma(w) \geq 1$$

, da Σ_1 die Repräsentanten aller Kanten umfasst und wie gesagt alle Kanten einen Knoten haben müssen.

Ferner gilt für w, dass nur k Teilausdrücke gewählt wurden, da auch nur k Knoten gewählt wurden,

ferner enthält jeder Teilausdruck genau ein \$.
also gilt auch

$$\#_{\$}(w) \leq k$$

Damit gilt $(G,k) \in VERTEXCOVER \Rightarrow f(G,k) \in MINMAXZEICHEN$.

(2) $f(G,k) \in MINMAXZEICHEN \Rightarrow (G,k) \in VERTEXCOVER$.
Wenn $f(G,k) \in MINMAXZEICHEN$, dann wissen wir, dass:
$\exists w \in L(\beta)$, sodass:
(i) $\#_{\$}(w) \leq k$ und ferner weniger als k Knoten gewählt wurden.
(ii) $\forall \sigma \in \Sigma_2 : \#_{\sigma}(w) \geq 1$ und ferner jede Kante zu einem der Knoten führt.
Dies sind die zwei oben genannten Bedingungen die $VERTEXCOVER$ erfüllen.

Damit gilt $f(G,k) \in MINMAXZEICHEN \Rightarrow (G,k) \in VERTEXCOVER$.
Und durch (1) auch $f(G,k) \in MINMAXZEICHEN \Leftrightarrow (G,k) \in VERTEXCOVER$.

f ist also eine polynomielle Reduktion von $VERTEXCOVER$ auf $MINMAXZEICHEN$.