



Hausaufgaben zur Vorlesung »Analyse von Algorithmen und Komplexität«

Blatt 9

Definition 1 (Vertex Cover (VC)). Für einen (ungerichteten) Graphen $G = (V, E)$ ist ein Vertex Cover eine Menge von Knoten $C \subseteq V$, sodass $u \in C$ oder $v \in C$ für alle $\{u, v\} \in E$ gilt, also ist für alle Kanten mindestens ein Endpunkt in C .

Für das VERTEXCOVER Problem ist neben einem Graphen G eine Zahl $k \in \mathbb{N}_{\geq 0}$ gegeben und es wird gefragt, ob ein Vertex Cover mit Größe (Anzahl der Knoten) höchstens k in G existiert.

Definition 2 (Feedback Vertex Set (FVS)). Für einen gerichteten Graphen $G = (V, E)$ ist ein Feedback Vertex Set eine Menge von Knoten $X \subseteq V$, sodass $G \setminus X$ kreisfrei ist.

Für das FEEDBACKVERTEXSET Problem ist neben einem Graphen G eine Zahl $k \in \mathbb{N}_{\geq 0}$ gegeben und es wird gefragt, ob ein Feedback Vertex Set mit Größe (Anzahl der Knoten) höchstens k in G existiert.

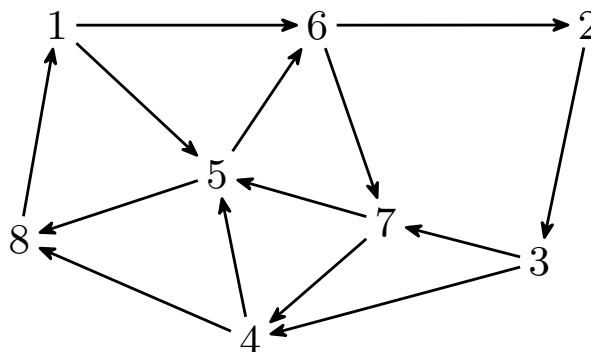


Abbildung 1: Beispiel Graph.

Hausaufgabe 9.1 (Nichtdeterministischer Algorithmus für VC)

(5 Punkte)

1. Finden Sie ein minimales Vertex Cover in Abbildung 1. Für ein Vertex Cover ist die Orientierung der Kanten nicht relevant.
2. Zeigen Sie $\text{VERTEXCOVER} \in \mathbf{NP}$, indem Sie einen nichtdeterministischen Algorithmus angeben, der VERTEXCOVER in Polynomialzeit löst. Begründen Sie die Korrektheit und die Laufzeit von Ihrem Algorithmus.

Lösung. 1. Ein minimales Vertex Cover ist zum Beispiel $\{3, 5, 6, 7, 8\}$.

2. Unser nichtdeterministischer Algorithmus geht wie folgt vor: Für jeden Knoten wird nichtdeterministisch entschieden, ob dieser im Vertex Cover sein soll. Nachdem dies für alle Knoten entschieden wurde, wird überprüft, ob nicht mehr als k Knoten gewählt wurden und ob für jede Kante mindestens ein Endpunkt im Vertex Cover liegt.

Existiert ein Vertex Cover im Graphen, existiert auch eine Reihe von nichtdeterministischen Entscheidungen, die diese Teilmenge von Knoten auswählt. Diese Teilmenge wird dann auch von dem Algorithmus akzeptiert. Damit ist der Algorithmus korrekt.

Beide diese Schritte können naiv in Polynomialzeit erledigt werden, also löst dieser Algorithmus das Vertex Cover Problem in Polynomialzeit. $\square$

Bewertung 1 Punkt für das Beispiel und 4 Punkte für den Algorithmus, dabei 2 Punkte für die Beschreibung und jeweils 1 Punkt für Korrektheit und Laufzeit.

Hausaufgabe 9.2 (Polynomieller Verifizierer für FVS)

(5 Punkte)

1. Finden Sie ein minimales Feedback Vertex Set in Abbildung 1.
2. Zeigen Sie $\text{FEEDBACKVERTEXSET} \in \mathbf{NP}$, indem Sie einen polynomiellen Verifizierer für FEEDBACKVERTEXSET angeben. Begründen Sie die Korrektheit und die Laufzeit von Ihrem Verifizierer.

Lösung. 1. Ein minimales FVS ist zum Beispiel $\{5, 6\}$.

2. Das Zertifikat hat die Form einer Teilmenge $X \subseteq V$, die beschreibt, welche Knoten das Feedback Vertex Set bilden sollen. Zum Beispiel über ein Bit pro Knoten. Nun wird überprüft, ob $|X| \leq k$ gilt und ob $G \setminus X$ azyklisch ist.

Hat der Graph ein Feedback Vertex Set von Größe maximal k , gibt es ein Zertifikat, welches dieses beschreibt. Dieses Zertifikat wird dann von unserem Verifizierer akzeptiert, damit ist der Verifizierer korrekt.

Die Größe von X zu überprüfen ist naiv in Polynomialzeit möglich. Für den zweiten Test bietet sich der Algorithmus zur topologischen Sortierung aus der Vorlesung an, welcher in Polynomialzeit eine topologische Sortierung findet, wenn der gegebene Graph azyklisch ist. Insgesamt ist dieser Verifizierer korrekt und auch ein polynomieller Verifizierer.

Bewertung 1 Punkt für das Beispiel und 4 Punkte für den Verifizierer, dabei 2 Punkte für die Beschreibung und jeweils 1 Punkt für Korrektheit und Laufzeit. $\square$

Abgabe: 16.06.2025, 10:00 in Moodle – bitte beachten Sie die Informationen zum Abgabeformat im Moodlekurs.

Hausaufgabe 9

Marek Lenczewski
Matrikelnummer: 1025252

22. Juni 2025

Hausaufgabe 9.1

Teil 1

Ein minimales Vertex Cover ist $C = \{2, 7, 5, 4, 1\}$.
Die Minimale Knotenanzahl ist 5.

Teil 2

Der Algorithmus arbeitet wie folgt:

- Zuerst wird für jeden Knoten $v \in V$ nichtdeterministisch entschieden, ob dieser in das Vertex Cover aufgenommen wird. Dabei wird drauf geachtet, dass die Anzahl k nicht übersteigt.
- Danach wird geprüft, ob für jede Kante $\{u, v\} \in E$ entweder $v \in C$ oder $u \in C$ oder beides zutrifft.
- Falls dies für alle Kanten gilt, akzeptieren wir. Andernfalls lehnen wir ab.

Existiert ein Vertex Cover der Größe höchstens k , dann gibt es auch eine, nichtdeterministisch ausgewählte, Knotenmenge, die dazu passt. Von dieser werden alle Kanten abgedeckt und der Algorithmus akzeptiert. Falls es kein Vertex Cover gibt, dann lehnt der Algorithmus ab.

Die Auswahl der Knoten und die Prüfung der Kanten läuft in polynomieller Zeit, also ist dies ein polynomieller nichtdeterministischer Algorithmus, der VERTEX COVER löst und damit ist $VERTEXCOVER \in NP$.

Hausaufgabe 9.2

Teil 1

Ein minimales Feedback Vertex Set ist $X = \{1, 5\}$

Teil 2

Der Verifizierer arbeitet wie folgt:

- Es wird eine Teilmenge $X \subseteq V$ übergeben, die das Feedback Vertex Set darstellen soll.
- Dann wird geprüft, ob $|X| \leq k$ gilt und es wird abgelehnt, falls es nicht gilt.
- Danach wird ein Graph $G' = G \setminus X$ konstruiert, in dem alle Knoten aus X und alle an X angrenzenden Kanten entfernt wurden.

- Auf dem Graphen G' wird eine Tiefensuche durchgeführt, dabei werden Knoten während der Bearbeitung grau markiert und beim Backtracking schwarz.
- Wird eine Kante zu einem grauen Knoten entdeckt, dann wird abgelehnt, da dies einen Kreis anzeigt.
- Falls die Tiefensuche den gesamten Graphen durchläuft ohne abzulehnen, dann wird akzeptiert.

Existiert ein Feedback Vertex Set der Größe höchstens k , dann gibt es auch ein Zertifikat, das genau diese Knotenmenge beschreibt. Alle Teilmengen, die die Tiefensuche überstehen, werden von dem Verifizierer akzeptiert und sonst abgelehnt.

Die Prüfung der übergebenen Teilmenge und die Konstruktion des Graphen laufen in polynomieller Zeit. Ebenfalls die Tiefensuche läuft in polynomieller Zeit, somit ist auch der Verifizierer polynomiell und verifiziert die Lösung für das FEEDBACK VERTEX SET in polynomieller Zeit. Also gilt $FEEDBACKVERTEXSET \in NP$.



Hausaufgaben zur Vorlesung »Analyse von Algorithmen und Komplexität«

Blatt 10

Definition 1 (Feedback Vertex Set (FVS)). Für einen gerichteten Graphen $G = (V, E)$ ist ein Feedback Vertex Set eine Menge von Knoten $X \subseteq V$, sodass $G \setminus X$ kreisfrei ist.

Für das FEEDBACKVERTEXSET Problem ist neben einem Graphen G eine Zahl $k \in \mathbb{N}_{\geq 0}$ gegeben und es wird gefragt, ob ein Feedback Vertex Set mit Größe (Anzahl der Knoten) höchstens k in G existiert.

Hausaufgabe 10.1 (FEEDBACKVERTEXSET ist **NP**-vollständig) (5 Punkte)
Zeigen Sie, dass FEEDBACKVERTEXSET **NP**-vollständig ist.

Hinweis: Reduzieren Sie VERTEXCOVER auf FEEDBACKVERTEXSET. Wenn ein Vertex Cover aus einem Graphen entfernt wird, hat dieser keine Kanten mehr.

Lösung. FEEDBACKVERTEXSET $\in$ **NP** ist bekannt aus der letzten Hausaufgabe. Nun zeigen wir eine Reduktion von VERTEXCOVER auf FEEDBACKVERTEXSET. Betrachte die folgende **Reduktion**:

Für eine gegebene VERTEXCOVER Instanz $(G = (V, E), k)$ erzeuge den Graphen $G' = (V, E')$ mit $E' := \{(u, v), (v, u) \mid \{u, v\} \in E\}$. Gebe (G', k) als FEEDBACKVERTEXSET Instanz aus.

Korrektheit:

$\Rightarrow$ (Falls (G, k) eine Ja-Instanz von VERTEXCOVER ist, dann ist (G', k) eine Ja-Instanz von FEEDBACKVERTEXSET.):

Sei $C \subseteq V$ ein Vertex Cover mit Größe kleiner oder gleich k . Dann enthält $G \setminus C$ keine Kanten. Damit hat $G' \setminus C$ ebenfalls keine Kanten und somit keine Kreise.

$\Leftarrow$ (Falls (G', k) eine Ja-Instanz von FEEDBACKVERTEXSET ist, dann ist (G, k) eine Ja-Instanz von VERTEXCOVER.):

Sei $X \subseteq V'$ ein Feedback Vertex Set mit Größe kleiner oder gleich k . Dann ist X ein Vertex Cover in G , da ansonsten eine Kante $\{u, v\} \in E$ existiert, für die $u \notin X \wedge v \notin X$ gilt. Damit wäre aber (u, v, u) ein Kreis in G' und X somit kein Feedback Vertex Set.

Laufzeit: Das Erzeugen der Menge E' geht in Zeit $O(|E|)$. Insgesamt ist die Reduktion demnach in Laufzeit $O(|E|)$, also in polynomieller Laufzeit, durchführbar.

Also existiert eine polynomielle Reduktion von VERTEXCOVER auf FEEDBACKVERTEXSET. Da VERTEXCOVER nach den aktuellen Präsenzaufgaben **NP**-vollständig ist und

FEEDBACKVERTEXSET $\in$ **NP** gilt, ist somit FEEDBACKVERTEXSET **NP**-vollständig. $\square$

Bewertung 2 Punkte für die Konstruktion, 1 Punkt für den 1. Fall, 1 Punkt für den 2. Fall, 0.5 Punkte für die Laufzeit, 0.5 Punkte für die Beweisführung im Ganzen

Hausaufgabe 10.2 (Turingmaschinen)

(5 Punkte)

Entwerfen Sie eine Turingmaschine für die Sprache $L = \{ w \in \Sigma^* \mid w \text{ ist ein Palindrom} \}$ über einem gegebenen Alphabet Σ und geben Sie die Laufzeit Ihrer Turingmaschine an. Begründen Sie dabei, warum Ihre Turingmaschine korrekt ist und die angegebene Laufzeit hat.

Lösung. Die Turingmaschine sieht folgendermaßen aus:

- (1) Falls kein Wort auf dem Band steht, akzeptiere.
- (2) Falls nur ein Buchstabe auf dem Band steht, akzeptiere.
- (3) Lese den ersten Buchstaben und bewege danach den Kopf zum letzten Buchstaben. Falls die beiden Buchstaben verschieden sind, verwerfe. Ansonsten ersetze beide Buchstaben durch ein neues Symbol x . Ersetze dafür zuerst den letzten Buchstaben und bewege dann den Kopf zum ersten Buchstaben.
- (4) Gehe zu Schritt (1).

Korrektheit: Die Turingmaschine vergleicht immer den ersten und den letzten Buchstaben. Wenn die Buchstaben nicht gleich sind, ist das Wort kein Palindrom und es wird verworfen. Ansonsten werden die Buchstaben rausgestrichen (siehe Schritt (3)). Diese Prozedur wird solange wiederholt, bis man in der Mitte des Wortes angekommen ist und entweder nur noch ein Buchstabe oder kein Buchstabe vorhanden ist. In beiden Fällen ist das Wort ein Palindrom und es wird akzeptiert (siehe Schritte (1) und (2)).

Laufzeit: Die Schritte (1) bis (3) benötigen jeweils eine Laufzeit von $O(n)$, wobei n hier die Länge der Eingabe bezeichnet. Da in jedem Durchlauf der Schritte (1) bis (3) zwei Buchstaben raus gestrichen werden oder das Wort akzeptiert oder verworfen wird, gibt es maximal $\lfloor n/2 \rfloor$ Durchläufe, bis nur noch ein oder kein Buchstabe auf dem Band stehen und somit in den Schritten (1) oder (2) akzeptiert wird. Damit ergibt sich insgesamt eine Laufzeit von $O(n^2)$. $\square$

Bewertung 2 Punkte für die Turingmaschine, 1 Punkt für die Korrektheit und 2 Punkte für die Laufzeit: 1 Punkt Angabe, 1 Punkt Begründung

Abgabe: 23.06.2025, 10:00 in Moodle – bitte beachten Sie die Informationen zum Abgabeformat im Moodlekurs.

Hausaufgabe 10

Marek Lenczewski
Matrikelnummer: 1025252

22. Juni 2025

Hausaufgabe 10.1

Teil 1: FeedbackVertexSet $\in$ NP

Zuerst brauchen wir einen polynomiellen Verifizierer, um zu zeigen dass FeedbackVertexSet $\in$ NP.

Der Verifizierer bekommt einen Graphen G und eine Teilmenge X , dieser arbeitet wie folgt:

- Prüfe, ob $|X| \leq k$, sonst ablehnen
- Den Graphen $G' = G \setminus X$ konstruieren
- Tiefensuche auf G' durchführen, dabei werden Knoten vom aktuellen Pfad als grau markiert und nach vollständiger Bearbeitung als schwarz. Wird eine Kante zu einem grauen Knoten gefunden, zeigt dies einen Kreis an und wir lehnen ab
- Akzeptieren, wenn die Tiefensuche nichts findet

Die Schritte 1-4 laufen in polynomieller Zeit, somit läuft der Verifizierer in polynomieller Zeit. Damit ist *FeedbackVertexSet* $\in$ NP.

Teil 2: FeedbackVertexSet ist NP-schwer

Jetzt brauchen wir eine Reduktion von VertexCover auf FeedbackVertexSet, um zu zeigen, dass *FeedbackVertexSet* NP – Schwer ist.

Konstruktion: Sei dafür ein VertexCover gegeben mit einem Graphen G und einer Zahl k . Aus G wird ein G' , wobei jede ungerichtete Kante $\{u, v\} \in E$ durch zwei gerichteten Kanten (u, v) und (v, u) zu E' ersetzt wird. Das k wird einfach übernommen $k = k'$.

Beweis VertexCover nach FeedbackVertexSet:

- Sei C ein VertexCover von G mit $|C| \leq k$
- Da C ein VertexCover ist, wird jede Kante mit C verbunden
- Wenn C entfernt wird, dann hat $G \setminus C$ keine Kanten mehr
- Dann hat auch $G' \setminus C$ keine Kanten und ist somit Kreisfrei
- Damit ist C ein FeedbackVertexSet für G' mit $|C| \leq k'$

Beweis FeedbackVertexSet nach VertexCover:

- Sei X ein FeedbackVertexSet von G' mit $|X| \leq k'$
- Dann ist $G' \setminus X$ kreisfrei
- Angenommen X wäre kein VertexCover von G , dann gäbe es eine Kante $\{u, v\} \in E$ mit $u, v \notin X$
- Somit wären u, v noch in $G' \setminus X$ vorhanden
- Die Kanten (u, v) und (v, u) würden, durch die Konstruktion, einen Kreis erzeugen
- Das ist ein Widerspruch, somit ist $G' \setminus X$ kreisfrei und X ist ein VertexCover

Da $FeedbackVertexSet \in NP$ und $NP - schwer$ ist, folgt dass $FeedbackVertexSet$ NP-Vollständig ist. $\square$

Hausaufgabe 10.2

Lösung. Die Turingmaschine arbeitet wie folgt:

1. Falls das Band leer ist oder nur aus Markierungen besteht, akzeptiere
2. Merke das erste unmarkierte Zeichen und ersetze es durch X
3. Laufe zum letzten unmarkierten Zeichen oder verwefe
4. Falls dieses Zeichen nicht mit dem gemerkten übereinstimmt, verwefte
5. Ersetze es durch X und gehe zurück zum Anfang
6. Gehe zu 1

Korrektheit: Ein Wort ist genau dann ein Palindrom, wenn es sich von vorne und hinten gleich liest. Dies prüft die Turingmaschine, indem sie Schrittweise die äußeren Buchstaben verifiziert, bis nur noch einer oder keiner Vorhanden ist, dann akzeptiert ist und wenn eine Unstimmigkeit auftritt, dann lehnt sie ab.

Laufzeit: Die Turingmaschine hat eine Laufzeit von $\mathcal{O}(n^2)$. In jedem Durchlauf macht sie $\mathcal{O}(n)$ Schritte und das mit $\mathcal{O}(n)$ Durchläufen. $\square$



Hausaufgaben zur Vorlesung »Analyse von Algorithmen und Komplexität«

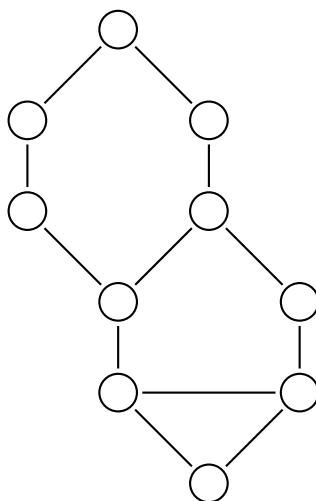
Blatt 11

Definition 1 (Dreiecksüberdeckung eines Graphen (Δ COVER)). Gegeben ist ein endlicher ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$. Dieser Graph hat eine Dreiecksüberdeckung, wenn eine Menge von Knoten $C_\Delta \subseteq V$ mit $|C_\Delta| \leq k$ existiert, so dass zu jedem Teilgraphen (Dreieck) $D = (V_D, E_D) \subseteq (V, E)$ mit $|V_D| = 3$ und $E_D = \{ \{v_1, v_2\} \in V_D \times V_D \mid v_1 \neq v_2 \}$ von G gilt, $v \in C_\Delta$ für mindestens ein $v \in V_D$.

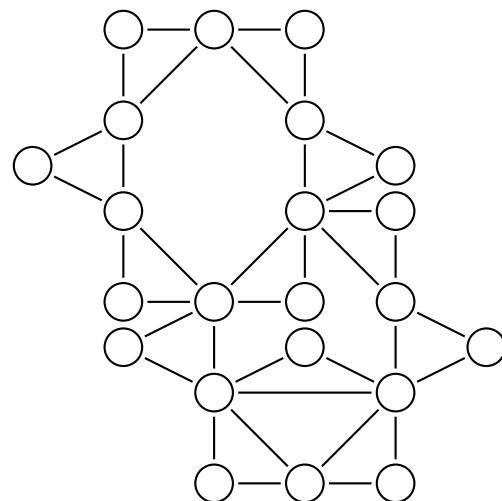
Hausaufgabe 11.1

(10 Punkte)

- (a) (1 Punkt) Markieren Sie im folgenden Graphen (Abbildung 1 (a)) ein VERTEXCOVER der Größe 5.
- (b) (1 Punkt) Markieren Sie im folgenden Graphen (Abbildung 1 (b)) eine Dreiecksüberdeckung der Größe 5.
- (c) (8 Punkte) Zeigen Sie, dass Δ COVER **NP**-vollständig ist.



(a)



(b)

Abbildung 1: Beispielgraph

Lösung.

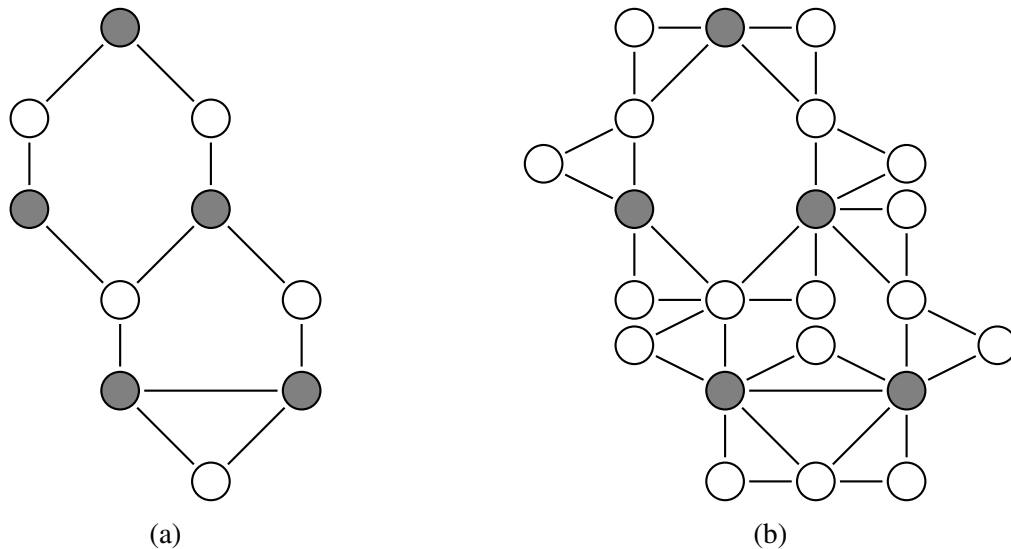


Abbildung 2: Beispielgraph Lösung

(c) Wir zeigen zunächst, $\Delta \text{ COVER} \in \mathbf{NP}$ mittels eines polynomiellen Verifizierers.

Der polynomielle Verifizierer erhält als Zertifikat eine Menge von Knoten $C_\Delta \subseteq V$. Zunächst überprüft er, ob $|C_\Delta| \leq k$. Falls dies nicht der Fall ist, verwirft die Instanz. Ansonsten überprüfe für alle Teilgraphen $D = (V_D, E_D) \subseteq (V, E)$ mit $|V_D| = 3$ und $E_D = \{ \{v_1, v_2\} \in V_D \times V_D \mid v_1 \neq v_2 \}$ von G , ob für mindestens ein $v \in V_D$ auch $v \in C_\Delta$ gilt. Falls dies gilt, akzeptiere. Ansonsten verwirft.

Korrektheit: Der Verifizierer überprüft die geforderten Eigenschaften. Falls ein Graph eine Dreiecksüberdeckung besitzt, so existiert ein entsprechendes Zertifikat C_Δ und der Verifizierer akzeptiert die Instanz. Falls ein Graph keine Dreiecksüberdeckung besitzt, existiert kein valides entsprechendes Zertifikat C_Δ und der Verifizierer verwirft die Instanz.

Laufzeit: Es gibt maximal $|V|^3$ verschiedene Teilmengen, die ein Dreieck bilden könnten und daher überprüft werden müssen. Die Überprüfung geht in $O(|V|)$ Zeit. Da alle anderen Schritte des Verifizierers in $O(1)$ Zeit erledigt werden können, ergibt sich somit insgesamt eine Laufzeit von $O(|V|^4)$, was einer polynomiellen Laufzeit entspricht.

Also gilt $\Delta \text{ COVER} \in \mathbf{NP}$.

Nun reduzieren wir VERTEXCOVER auf $\Delta \text{ COVER}$. Betrachte folgende Reduktion:

Sei $I = (G = (V, E), k)$ eine Instanz von VERTEXCOVER . Bilde den Graphen $G' = (V', E')$ mit $V' = V \cup \{v_e \mid e \in E\}$ und $E' = E \cup \{ \{v_e, e_1\}, \{v_e, e_2\} \mid e = \{e_1, e_2\} \in E \}$. Gebe $I' = (G', k)$ als $\Delta \text{ COVER}$ Instanz aus.

Korrektheit:

$\Rightarrow$ (Falls (G, k) eine Ja-Instanz von VERTEXCOVER ist, dann ist (G', k) eine Ja-Instanz von $\Delta \text{ COVER}$.):

Sei C ein Vertex Cover mit Größe kleiner oder gleich k von G . Dann ist C ebenfalls eine Dreiecksüberdeckung mit Größe kleiner oder gleich k . Offensichtlich gilt $|C| \leq k$. Da C ein Vertex Cover von G ist, gilt für jede Kante $\{u, v\} \in E$, dass $u \in C$ oder $v \in C$. Für alle Teilgraphen $D = (V_D, E_D) \subseteq (V, E)$ mit $|V_D| = 3$ und $E_D = \{ \{v_1, v_2\} \in V_D \times V_D \mid v_1 \neq v_2 \}$ von G gilt damit direkt, dass mindestens ein $v \in V_D$ auch in C enthalten ist. Der Graph G' besitzt neben allen solchen Teilgraphen in G zusätzlich solche Teilgraphen (Dreiecke), die durch die Knotenmenge

$\{v_e \mid e \in E\}$ und die entsprechenden Kanten $\{e, \{v_e, e_1\}, \{v_e, e_2\} \mid e = \{e_1, e_2\} \in E\}$ entstehen. Für diese Dreiecke gilt aber auch, dass mindestens ein Knoten, nämlich e_1 oder e_2 , in C enthalten sein muss, da C ein Vertex Cover ist.

$\Leftarrow$ (Falls (G', k) eine Ja-Instanz von Δ COVER ist, dann ist (G, k) eine Ja-Instanz von VERTEXCOVER.):

Sei C_Δ eine Dreiecksüberdeckung von G' der Größe $|C_\Delta| \leq k$. Ohne Beschränkung der Allgemeinheit können wir davon ausgehen, dass $C_\Delta \subseteq V$ gilt: Falls für einen Knoten $v_e \in C_\Delta$ gilt, können wir e_1 zu C_Δ hinzufügen und v_e aus C_Δ löschen. Dadurch bleibt die Überdeckungseigenschaft erhalten und die Menge wird höchstens kleiner. Nun gilt also $C_\Delta \subseteq V$. Die Menge C_Δ ist auch ein Vertex Cover mit Größe kleiner oder gleich k von G . Offensichtlich gilt $|C_\Delta| \leq k$. Da für alle Teilgraphen $D' = (V'_D, E'_D) \subseteq (V', E')$ mit $|V'_D| = 3$ und $E'_D = \{\{v_1, v_2\} \in V'_D \times V'_D \mid v_1 \neq v_2\}$ von G' gilt, dass mindestens ein Knoten $v \in V'_D$ ebenfalls in C_Δ enthalten sein muss, gilt dies auch für die Dreiecke $\{v_e, e_1, e_2\}$. Da außerdem $C_\Delta \subseteq V$ gilt, gilt $e_1 \in C_\Delta$ oder $e_2 \in C_\Delta$ für alle Kanten $e \in E$.

Laufzeit: Das Hinzufügen von einem neuen Knoten und zwei neuen Kanten pro Kante geht in $O(|E|)$. Damit geht die gesamte Reduktion in $O(|E|)$, also in polynomieller Zeit.

Also existiert eine polynomielle Reduktion von VERTEXCOVER auf Δ COVER. Da VERTEXCOVER **NP**-vollständig aus den Präsenzaufgaben bekannt ist und Δ COVER $\in$ **NP** gilt, ist somit Δ COVER **NP**-vollständig. $\square$

Bewertung (a) und (b) jeweils 1 Punkt, (c) gibt 8 Punkte: 1 Punkt Angabe Verifizierer/NDTM, 0,5 Punkte Korrektheit Verifizierer/NDTM, 0,5 Punkte Laufzeit Verifizierer/NDTM, 1 Punkt Angabe Reduktion, 3 Punkte Korrektheit (je Richtung 1,5 Punkte), 1 Punkt Laufzeit Reduktion, 1 Punkt Beweisführung

Abgabe: 30.06.2025, 10:00 in Moodle – bitte beachten Sie die Informationen zum Abgabeformat im Moodlekurs.

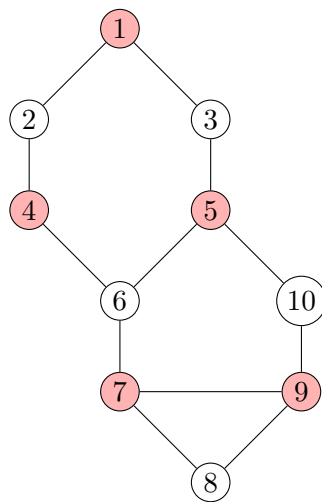
Hausaufgabe 11

Marek Lenczewski
Matrikelnummer: 1025252

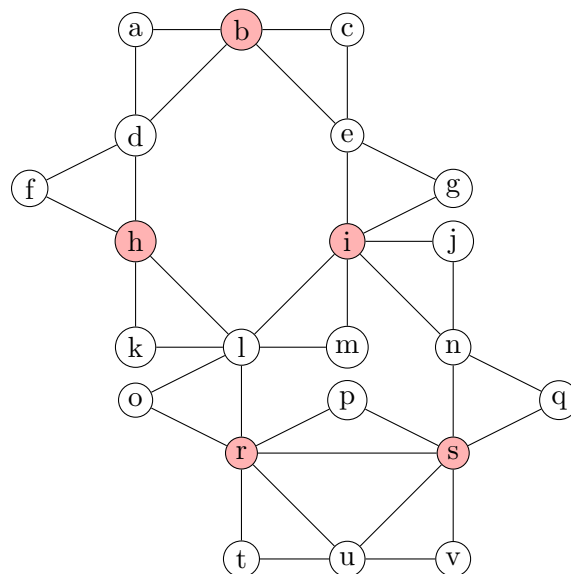
28. Juni 2025

Hausaufgabe 11.1

a



b



c

Teil 1: Δ -Cover $\in$ NP

Zuerst brauchen wir einen polynomiellen Verifizierer, um zu zeigen, dass $\Delta\text{-COVER} \in \text{NP}$. Der Verifizierer bekommt einen Graphen $G = (V, E)$, Zahl k und ein Zertifikat $C_\Delta \subseteq V$ und arbeitet wie folgt:

1. Prüfe, ob $|C_\Delta| \leq k$
2. Prüfe alle Knotentripel $\{a, b, c\} \subseteq V$, ob sie ein Dreieck bilden (also ob $\{a, b\}$, $\{b, c\}$ und $\{a, c\} \in E$ zutrifft) und ob a, b oder $c \in C_\Delta$
3. Ablehnen, falls ein Dreieck den test nicht besteht, sonst akzeptieren

Die Schritte 1 und 3 laufen in $\mathcal{O}(1)$ und Schritt zwei läuft in $\mathcal{O}(|V|^3)$, das ergibt eine polynomielle Laufzeit. Damit ist $\Delta\text{-COVER} \in \text{NP}$.

Teil 2: $\Delta\text{-Cover}$ ist NP-schwer

Jetzt brauchen wir eine Reduktion von VertexCover auf $\Delta\text{-COVER}$, um zu zeigen, dass $\Delta\text{-COVER}$ NP-schwer ist.

Konstruktion: Sei dafür ein VertexCover gegeben mit einem Graphen $G = (V, E)$ und einer Zahl k . Aus G wird $G' = (V', E')$, wobei für jede Kante $e = \{u, v\} \in E$ ein neuer Knoten $w_e \in V'$ hinzugefügt wird und mit beiden Knoten u, v verbunden wird, sodass die Kanten $\{v, w_e\}$ und $\{u, w_e\}$ zu E' dazu kommen. Das $k = k'$ wird übernommen. So werden aus allen Kanten Dreiecke.

Beweis VertexCover nach $\Delta\text{-Cover}$

- Sei C ein VertexCover für G mit $|C| \leq k$
- Zu zeigen sei, dass C auch ein $\Delta\text{-Cover}$ für G' mit $|C| \leq k$ ist
- Da C ein VertexCover ist, gilt $u \in C$ oder $v \in C$
- Jedes Dreieck in G' hat die Form $\{u, v, w_e\}$ für eine Kante $e = \{u, v\} \in E$
- Somit hat jedes Dreieck mindestens einen Knoten in C
- Also ist C ein $\Delta\text{-Cover}$ für G'

Beweis $\Delta\text{-Cover}$ nach VertexCover

- Sei C_Δ ein $\Delta\text{-Cover}$ für G' mit $|C_\Delta| \leq k'$
- Jede Kante $e = \{u, v\} \in E$ bildet durch $\{u, v, w_e\}$ ein Dreieck in G'
- Also gilt u, v oder $w_e \in C_\Delta$ für alle Kanten in G'
- Falls $w_e \in C_\Delta$, dann kann man w_e durch u oder v tauschen ohne die Eigenschaft von $\Delta\text{-Cover}$ zu verletzen, sodass u oder $v \in C_\Delta$ für alle Kanten in G' gilt
- Da sich die Knoten aus C_Δ nur vertauschen gilt weiter $|C| = |C_\Delta| \leq k$
- Weiter sind in C_Δ nur noch die Knoten aus V , somit gilt $u \in C$ oder $v \in C$ für alle Kanten E
- Also ist C ein VertexCover für G

Da $\Delta\text{-COVER} \in \text{NP}$ und NP-schwer ist, folgt dass $\Delta\text{-COVER}$ NP-vollständig ist. □



Hausaufgaben zur Vorlesung »Analyse von Algorithmen und Komplexität«

Blatt 12

Hausaufgabe 12.1 (Lower Bound: VERTEXCOVER)

(5 Punkte)

Betrachten Sie die Reduktion aus der Musterlösung zu Präsenzsreihe 10 von CLIQUE auf VERTEXCOVER.

1. Geben Sie die Anzahl der Knoten, die Anzahl der Kanten und den Wert k für die resultierende VERTEXCOVER Instanz in Abhängigkeit von der originalen CLIQUE Instanz an.
2. Zeigen Sie Lower Bounds für VERTEXCOVER bezüglich Anzahl der Knoten, Anzahl der Kanten und k basierend auf der ETH und der Reduktion aus der Musterlösung zu Präsenzsreihe 10.

Lösung. 1. Sei $I = (G = (V, E), k)$ eine Instanz von CLIQUE und $I' = (G' = (V', E'), k')$ die aus der Reduktion resultierende Instanz von VERTEXCOVER. Es gilt $|V'| = |V|$, $|E'| = \frac{1}{2}|V|(|V| - 1) - |E| \leq |V|^2$ und $k' = |V| - k \leq |V|$.

2. • Angenommen es gäbe einen Algorithmus, der VERTEXCOVER in $2^{o(|V'|)} \cdot |I|^{O(1)}$ löst. Durch die Kombination der Reduktion von CLIQUE auf VERTEXCOVER und diesem Algorithmus, existiert dann auch ein $2^{o(|V|)} \cdot |I|^{O(1)}$ Algorithmus für CLIQUE, da $|V'| = |V|$ nach Aufgabenteil 1. Aus der Musterlösung der Präsenzsreihe 12 folgt somit, dass die ETH falsch ist.

Damit kann es so einen Algorithmus für VERTEXCOVER nur geben, wenn die ETH falsch ist.

- Angenommen es gäbe einen Algorithmus, der VERTEXCOVER in $2^{o(\sqrt{|E'|})} \cdot |I|^{O(1)}$ löst. Durch die Kombination der Reduktion von CLIQUE auf VERTEXCOVER und diesem Algorithmus, existiert dann auch ein $2^{o(|V|)} \cdot |I|^{O(1)}$ Algorithmus für CLIQUE, da $|E'| \leq |V|^2$ nach Aufgabenteil 1. Aus der Musterlösung der Präsenzsreihe 12 folgt somit, dass die ETH falsch ist.

Damit kann es so einen Algorithmus für VERTEXCOVER nur geben, wenn die ETH falsch ist.

- Angenommen es gäbe einen Algorithmus, der VERTEXCOVER in $2^{o(k')} \cdot |I|^{O(1)}$ löst. Durch die Kombination der Reduktion von CLIQUE auf VERTEXCOVER und diesem Algorithmus, existiert dann auch ein $2^{o(|V|)} \cdot |I|^{O(1)}$ Algorithmus für CLIQUE, da $k' \leq |V|$ nach Aufgabenteil 1. Aus der Musterlösung der Präsenzsreihe 12 folgt somit, dass die ETH falsch ist.

Damit kann es so einen Algorithmus für VERTEXCOVER nur geben, wenn die ETH falsch ist. □

Bewertung 1 Punkt für den ersten Teil und 1.5 + 1.5 + 1 für den zweiten.

Definition 1 (HITTINGSET). Bei diesem Problem ist ein Universum U , Teilmengen des Universums $F_1, \dots, F_r \subseteq U$ und eine Zahl $k \in \mathbb{N}_{\geq 0}$ gegeben. Gibt es eine Menge $S \subseteq U$ mit $|U| \leq k$ und $S \cap F_i \neq \emptyset$ für alle $i \in [r]$?

Anschaulicher wird bei diesem Problem nach maximal k Elementen aus dem Universum gefragt, sodass diese Elemente alle gegebenen Mengen F_i „treffen“.

Hausaufgabe 12.2 (Lower Bound: HITTINGSET) (5 Punkte)
Betrachten Sie folgende Reduktion von 3-SAT auf HITTINGSET.

Für eine gegebene 3-SAT Instanz I mit n Variablen $x_1, \dots, x_n$ und m Klauseln $C_1, \dots, C_m$: Als Universum wählen wir $U = \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$. Für jede Variable x_i fügen wir die Menge $F_i := \{x_i, \bar{x}_i\}$ hinzu. Für jede Klausel C_j fügen wir die Menge $F_{n+j} := C_j$ hinzu. Wir setzen $k := n$ und erhalten damit eine HITTINGSET Instanz I .

1. Geben Sie die Größe des Universums, die Anzahl der Mengen und die Zahl k aus der resultierenden HITTINGSET Instanz in Abhängigkeit von der Anzahl der Variablen n und der Anzahl der Klauseln m aus der originalen 3-SAT Instanz an.
2. Beweisen Sie Lower Bounds für HITTINGSET bezüglich Universumsgröße, Anzahl der Mengen und der Zahl k basierend auf der obigen Reduktion und der ETH.

Lösung. 1. Für die Universumsgröße gilt $|U| = 2n$, für die Anzahl der Mengen gilt $r = n + m$ und es gilt $k = n$.

2. • Angenommen es gäbe einen Algorithmus, der HITTINGSET in $2^{o(|U|)} \cdot |I|^{O(1)}$ löst. Durch die Kombination der Reduktion von 3-SAT auf HITTINGSET und diesem Algorithmus, existiert dann auch ein $2^{o(n)} \cdot |I|^{O(1)}$ Algorithmus für 3-SAT, da $|U| = 2n$ nach Aufgabenteil 1. Damit ist die ETH falsch.

Damit kann es so einen Algorithmus für HITTINGSET nur geben, wenn die ETH falsch ist.

- Angenommen es gäbe einen Algorithmus, der HITTINGSET in $2^{o(r)} \cdot |I|^{O(1)}$ löst. Durch die Kombination der Reduktion von 3-SAT auf HITTINGSET und diesem Algorithmus, existiert dann auch ein $2^{o(m)} \cdot |I|^{O(1)}$ Algorithmus für 3-SAT, da $r = n + m$ nach Aufgabenteil 1 und $n \leq 3m$ für 3-SAT gilt. Aus dem Sparsification Lemma folgt damit, dass die ETH falsch ist.

Damit kann es so einen Algorithmus für HITTINGSET nur geben, wenn die ETH falsch ist.

- Angenommen es gäbe einen Algorithmus, der HITTINGSET in $2^{o(k)} \cdot |I|^{O(1)}$ löst. Durch die Kombination der Reduktion von 3-SAT auf HITTINGSET und diesem Algorithmus, existiert dann auch ein $2^{o(n)} \cdot |I|^{O(1)}$ Algorithmus für 3-SAT, da $k = n$ nach Aufgabenteil 1. Damit ist die ETH falsch.

Damit kann es so einen Algorithmus für HITTINGSET nur geben, wenn die ETH falsch ist. □

Bewertung 1 Punkt für den ersten Teil und 1.5 + 1.5 + 1 für den zweiten.

Abgabe: 07.07.2025, 10:00 in Moodle – bitte beachten Sie die Informationen zum Abgabeformat im Moodlekurs.

Hausaufgabe 12

Marek Lenczewski
Matrikelnummer: 1025252

6. Juli 2025

Hausaufgabe 12.1.1

Die Knotenzahl bleibt gleich: $|V'| = |V| = n$

Die Kantenanzahl entspricht den nicht vorhandenen Kanten: $|E'| = |\overline{E}|$

k' entspricht allen Knoten, die vorher nicht in der Clique waren: $k' = n - k$

Hausaufgabe 12.1.2

Wir wissen, dass 3-SAT laut ETH mindestens $2^{\Omega(n)}$ braucht, wobei n für die Anzahl der Variablen steht.

In der Vorlesung wurde eine Reduktion von 3-SAT auf Clique gezeigt. In der Präsenzaufgabe 10.2 wurde eine Reduktion von Clique auf VertexCover gezeigt. Also gibt es eine Reduktion von 3-SAT zu VertexCover. Somit kann VertexCover nicht schneller lösbar sein als 3-SAT, weil sonst 3-SAT schneller werden würde, was nach ETH nicht geht.

So ergeben sich die folgenden Lower Bounds:

- **Bezüglich Anzahl der Knoten:** $2^{\Omega(n)}$, wobei $n = |V'|$. Die Knotenzahl verändert sich nicht bei den Reduktionen.
- **Bezüglich Anzahl der Kanten:** $2^{\Omega(\sqrt{m})}$, wobei $m = |E'|$. Die Kanten können bei den Reduktionen zu einem quadratischen Wachstum der Kanten führen.
- **Bezüglich k' :** $2^{\Omega(k')}$. Bei der Reduktion gilt $k' = n - k$, wobei k' linear von n abhängt.

Hausaufgabe 12.2.1

- **Größe des Universums:** $|U| = 2n$. Für jede der n Variablen gibt es zwei Elemente (x_i und $\bar{x}_i$)
- **Anzahl der Mengen:** $n + m$. n sind die Variablenmengen und m die Klauselmengen. (F_i und F_{n+j})
- **Parameter:** $k = n$. Wird in der Reduktion so übernommen.

Hausaufgabe 12.2.2

Wir wissen, dass 3-SAT laut ETH mindestens $2^{\Omega(n)}$ braucht, wobei n für die Anzahl der Variablen steht. Somit kann HittingSet nicht schneller sein.

Es ergeben sich die folgenden Lower Bounds für HittingSet:

- **Bezüglich Größe des Universums:** $2^{\Omega(|U|/2)}$, da für jede Variable zwei Elemente ins Universum aufgenommen werden
- **Bezüglich Anzahl der Mengen:** $2^{\Omega(n)}$, wobei die Anzahl der Variablen- und Klauselmengen ($m + n$) mindestens n ist
- **Bezüglich Zahl k :** $2^{\Omega(k)}$, da in der Reduktion $k = n$ gilt.



Hausaufgaben zur Vorlesung »Analyse von Algorithmen und Komplexität«

Blatt 13

Definition 1 (MAX-3-SAT). Bei dem Problem MAX-3-SAT ist eine Formel ϕ in konjunktiver Normalform gegeben, wobei jede Klausel 3 Literale enthält. Gesucht ist eine Belegung β der Variablen, die die Anzahl $v(\beta)$ der erfüllten Klauseln maximiert.

Hausaufgabe 13.1 (MAX-3-SAT)

(10 Punkte)

Betrachten Sie folgenden Algorithmus A:

- Sei β_0 die Belegung, die alle Variablen auf `false` setzt.
 - Sei β_1 die Belegung, die alle Variablen auf `true` setzt.
 - Falls $v(\beta_0) \geq v(\beta_1)$, gib β_0 zurück.
 - Sonst gib β_1 zurück.
- (a) (6 Punkte) Zeigen Sie, dass obiger Algorithmus Güte 2 hat, d.h. $v(A(\phi)) \geq \frac{1}{2}v(OPT(\phi))$ gilt für alle Eingaben ϕ .
- (b) (4 Punkte) Geben Sie eine Formel an, bei der der Algorithmus eine Belegung ausgibt, die genau die Hälfte der maximal erfüllbaren Klauseln erfüllt. Begründen Sie kurz, warum Ihre Formel geeignet ist.

Hinweis: Sie brauchen nur höchstens zwei Klauseln und vier Variablen.

Lösung. (a) Für jede Klausel C gilt, dass sie von mindestens einer der Belegungen β_0 und β_1 erfüllt wird. Wir zeigen nun per Widerspruch, dass der obige Algorithmus eine Güte von 2 hat:

Angenommen, es gilt $v(A(\phi)) < \frac{1}{2}v(OPT(\phi))$ für eine Formel ϕ . Sei m die Anzahl an Klauseln in ϕ . Dann gilt $v(OPT(\phi)) \leq m$. Da $v(A(\phi)) < \frac{1}{2}v(OPT(\phi)) \leq \frac{1}{2}m$, gilt $v(\beta_0) < \frac{1}{2}m$ und $v(\beta_1) < \frac{1}{2}m$. Da jede Klausel C allerdings von mindestens einer der Belegungen β_0 und β_1 erfüllt wird, gilt $v(\beta_0) \geq \frac{1}{2}m$ oder $v(\beta_1) \geq \frac{1}{2}m$. Dies ist ein Widerspruch.

Also gilt $v(A(\phi)) \geq \frac{1}{2}v(OPT(\phi))$ für alle Eingaben ϕ und obiger Algorithmus hat Güte 2.

- (b) Betrachte folgende Formel: $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_0 \vee \neg x_2 \vee \neg x_3)$ Mit $\beta = \{x_0 \rightarrow \text{false}, x_1 \rightarrow \text{true}, x_2 \rightarrow \text{true}, x_3 \rightarrow \text{true}\}$ gilt $v(\beta) = 2$. Da ϕ nur zwei Klauseln hat, gilt somit $v(OPT(\phi)) = 2$. Es gilt allerdings $v(\beta_0) = 1$, da hier nur die zweite Klausel erfüllt ist, und $v(\beta_1) = 1$, da hier nur die erste Klausel erfüllt ist. Somit gilt $v(A(\phi)) = 1 = \frac{1}{2} \cdot 2 = \frac{1}{2}v(OPT(\phi))$.

Bewertung (a) 2 Punkte korrekte Beweisführung im Allgemeinen, 2 Punkte Abschätzung Algorithmus, 2 Punkte Abschätzung Optimum; (b) 2 Punkte für Angabe einer korrekten Formel, 1 Punkt Ausgabe Algorithmus, 1 Punkt Optimum □

Abgabe: 14.07.2025, 10:00 in Moodle – bitte beachten Sie die Informationen zum Abgabeformat im Moodlekurs.

Hausaufgabe 13

Marek Lenczewski
Matrikelnummer: 1025252

13. Juli 2025

Hausaufgabe 13.1.a

Zu zeigen ist, dass der Algorithmus die Güte 2 hat, also die Lösung vom Algorithmus mindestens halb so gut ist wie die optimale Lösung. Also $\nu(A(\phi)) \geq \frac{1}{2}\nu(OPT(\phi))$ für alle Eingaben ϕ .

Sei ϕ eine beliebige 3-SAT-Formel mit m Klauseln $C_1, C_2, \dots, C_m$, wobei jede Klausel genau 3 Literale enthält.

Jede Klausel C_i ist wahr, wenn β_0 wahr ist oder wenn β_1 wahr ist oder beides. Da die Variablen in C_i nur positiv oder negativ sein können, muss dabei mindestens β_0 oder β_1 wahr werden.

Der Algorithmus wählt die Belegung β_0 oder β_1 aus, die die meisten Klauseln C_i erfüllt. Somit ist die Anzahl der ausgewählten Klauseln C_i mindestens $\frac{m}{2}$, da eine bei jeder Klausel mindestens eine Belegung wahr werden muss und somit hat mindestens eine Belegung mindestens $\frac{m}{2}$ wahre Klauseln.

Sei ν_0 = Anzahl der von β_0 erfüllten Klauseln und ν_1 = Anzahl der von β_1 erfüllten Klauseln.

So ergibt sich $\nu(A(\phi)) = \max\{\nu_0, \nu_1\} \geq \frac{\nu_0 + \nu_1}{2} \geq \frac{m}{2}$.

Da die optimale Lösung höchstens alle m Klauseln erfüllen kann, gilt $\nu(OPT(\phi)) \leq m$.

Daraus folgt: $\nu(A(\phi)) \geq \frac{m}{2} \geq \frac{\nu(OPT(\phi))}{2}$.

Somit hat der Algorithmus eine Güte von 2. □

Hausaufgabe 13.1.b

Der Algorithmus erfüllt genau die Hälfte der maximal erfüllbaren Klauseln, wenn jede Klausel nur aus positiven oder negativen Variablen besteht und sich diese genau gleich aufteilen.

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

Unter β_0 (alle false):

- $C_1 = (x_1 \vee x_2 \vee x_3) = (\text{false} \vee \text{false} \vee \text{false}) = \text{false}$
- $C_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3) = (\text{true} \vee \text{true} \vee \text{true}) = \text{true}$
- $\nu(\beta_0) = 1$

Unter β_1 (alle true):

- $C_1 = (x_1 \vee x_2 \vee x_3) = (\text{true} \vee \text{true} \vee \text{true}) = \text{true}$
- $C_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3) = (\text{false} \vee \text{false} \vee \text{false}) = \text{false}$
- $\nu(\beta_1) = 1$

Der Algorithmus wählt $\nu(A(\phi)) = \max\{\nu(\beta_0), \nu(\beta_1)\} = \max\{1, 1\} = 1$.

Eine optimale Lösung wäre zum Beispiel $x_1 = \text{true}, x_2 = \text{false}, x_3 = \text{false}$.

- $C_1 = (\text{true} \vee \text{false} \vee \text{false}) = \text{true}$
- $C_2 = (\text{false} \vee \text{true} \vee \text{true}) = \text{true}$

Daraus ergibt sich $\nu(OPT(\phi)) = 2$, also $\nu(A(\phi)) = \frac{\nu(OPT(\phi))}{2}$. Somit hat es die Güte 2.