ETH Zürich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger, Dr. Johannes Lengler
Übungsleitung: Florian Meier

HS 2018

Algorithmen und Komplexität
Lösungsvorschlag zu Übungsblatt 12

Lösungsvorschlag zu Aufgabe 1

a) Zu jedem ungerichteten Graph $G = (V, E)$ konstruieren wir einen gerichteten Graphen $G' = f(G)$, sodass f in polynomieller Zeit berechnet werden kann und G genau dann einen ungerichteten Hamiltonkreis enthält, wenn G' einen gerichteten Hamiltonkreis enthält.

G' besteht aus der selben Knoten Menge V wie G. Für jede Kante $\{u, v\} \in E$ fügen wir die beiden gerichteten Kanten (u, v) und (v, u) in E' ein.

Angenommen es gibt einen Hamiltonkreis $(u_1, \ldots, u_n)$ in G so ist dies ein gerichteter Hamiltonkreis in G'. Umgekehrt ist jeder gerichtete Hamiltonkreis in G' ein ungerichteter Hamiltonkreis in G. Offensichtlich kann f in polynomieller Zeit berechnet werden.

b) Zu jedem gerichteter Graph G konstruieren wir einen ungerichteten Graph $G' = f(G)$, sodass f in polynomieller Zeit berechnet werden kann und G genau dann einen gerichteten Hamiltonkreis enthält, wenn G' einen ungerichteten Hamiltonkreis enthält.

In einem gerichteten Graphen wird jeder Knoten über eine eingehende Kante betreten und über eine ausgehende Kante verlassen. Um dies auch in einem ungerichteten Graphen zu erzwingen, benutzen wir folgende Konstruktion. Für jeden Knoten v_i von G fügen wir drei Knoten v_i^{in}, v_i^{mid} und w_i^{out} in G' ein. Für jede gerichtete Kante $e = (v_i, v_j)$ von von G fügen wir die ungerichtete Kante $\{v_i^{out}, v_j^{in}\}$ in G' ein. Zusätzlich fügen wir für alle i die Kanten $\{v_i^{in}, v_i^{mid}\}$ und $\{v_i^{mid}, v_i^{out}\}$ in G' ein.

Angenommen es gibt einen gerichteter Hamiltonkreis $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ in G dann ist per Konstruktion $(v_{i_1}^{in}, v_{i_1}^{mid}, v_{i_1}^{out}, v_{i_2}^{in}, v_{i_2}^{mid}, v_{i_2}^{out}, \ldots, v_{i_n}^{in}, v_{i_n}^{mid}, v_{i_n}^{out})$ ein ungerichteter Hamiltonkreis in G'.

Angenommen es gibt einen ungerichteten Hamiltonkreis in G'. Da jedes v_i^{mid} Grad 2 hat, müssen v_i^{in} und v_i^{out} vor oder nach v_i^{mid} besucht werden. Da es keine Kanten zwischen zwei out-Knoten gibt und keine Kanten zwischen zwei in-Knoten gibt, muss jeder out-Knoten zwischen einem mid-Knoten und einem in-Knoten auf dem Hamiltonkreis liegn. Genauso muss jeder in-Knoten zwischen einem mid-Knoten und eiem out-Knoten liegen. Der ungerichtete Hamiltonkreis muss also die Form $(v_{i_1}^{in}, v_{i_1}^{mid}, v_{i_1}^{out}, v_{i_2}^{in}, v_{i_2}^{mid}, v_{i_2}^{out}, \ldots, v_{i_n}^{in}, v_{i_n}^{mid}, v_{i_n}^{out})$ oder $(v_{i_1}^{out}, v_{i_1}^{mid}, v_{i_1}^{in}, v_{i_2}^{out}, v_{i_2}^{mid}, v_{i_2}^{in}, \ldots, v_{i_n}^{out}, v_{i_n}^{mid}, v_{i_n}^{in})$ haben. Im ersten Fall befinden sich die Kanten $(v_{i_k}, v_{i_{k+1}})$ und somit der gerichtete Hamiltonkreis $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ in G. Im zweiten Fall befinden sich die Kanten $(v_{i_{k+1}}, v_{i_k})$ und somit der gerichtete Hamiltonkreis $(v_{i_n}, v_{i_{n-1}}, \ldots, v_{i_1})$ in G.

Da G' lediglich dreimal so viele Knoten hat wie G lässt sich G' sicherlich in polynomieller Zeit in $|G|$ erzeugen.

Lösungsvorschlag zu Aufgabe 2

Wir nehmen an, dass die Variablenmenge der Formel F durch die Menge $\{x_1, x_2, \ldots, x_n\}$ gegeben ist. Ausserdem sei $m \geq n$ die Anzahl Literale der Formel F. Da jedes Literal einmal abgespeichert

werden muss, ist die Eingabegrösse $|F|$ mindestens $\Omega(m)$. Genauer gesagt ist die Eingabegrösse $\Theta(m \log(n))$ im logarithmischen Kostenmass, aber $\Omega(m)$ reicht für unsere Zwecke aus.

Wir testen zuerst mit dem Algorithmus $\mathcal{A}$, ob F erfüllbar ist. Wenn nicht, gibt es nichts zu tun.

Andernfalls setzen wir $F_0 := F$ und testen mit $\mathcal{A}$, ob $F_0 \wedge x_1$ noch erfüllbar ist. Falls ja, setzen wir $F_1 := F_0 \wedge x_1$; sonst $F_1 := F \wedge \bar{x}_1$. Wir iterieren dieses Vorgehen für jede Variable x_i und konstruieren so $F_1, \ldots, F_n$ (siehe Algorithmus 1). Am Ende ist dann die erfüllende Belegung gegeben durch die Ergebnisse von $\mathcal{A}(F_{i-1} \wedge x_i)$, die wir jeweils in y_i abspeichern.

Zur Korrektheit: Falls F nicht erfüllbar ist, so erkennt dies unser fiktiver Algorithmus $\mathcal{A}$ und wir müssen keine Belegung finden. Es bleibt zu zeigen, dass im anderen Fall $x_1 = y[1], \ldots, x_n = y[n]$ tatsächlich eine erfüllende Belegung für F ist. Wir beweisen per Induktion über i, dass eine erüllende Belegung mit $x_1 = y[1], \ldots, x_i = y[i]$ existiert, und dass dies für jede erfüllende Belegung gilt.

Im Fall $i = 1$ wissen wir bereits, dass $F = F_0$ erfüllbar ist. Die Formel $F_0 \wedge x_1$ ist nur dann erfüllbar wenn $x_1 = 1$. Also gibt es dafür eine erfüllende Belegung genau dann wenn F_0 eine erfüllende Belegung aufweist in der $x_1 = 1$ gilt. In diesem Fall setzen wir $y[1] = 1$. Falls $F_0 \wedge x_1$ nicht erfüllbar ist, bleibt als einzige Möglichkeit, dass jede erfüllende Belegung von F_0 die Eigenschaft $x_1 = 0$ hat. In diesem Fall setzen wir auch tatsächlich $y[1] = 0$. In beiden Fällen ergänzen wir F um eine zusätzliche Klausel (entweder x_1 oder $\bar{x}_1$), so dass alle erfüllbaren Belegungen jeweils den gleichen Wert bei x_1 aufweisen.

Für den Induktionsschritt können wir danach das gleiche Argument wieder verwenden: Wir wissen, dass eine erfüllende Belegung für die Formel F_{i-1} existiert, und wir wissen auch per Induktion, dass für jede erfüllende Belegung $x_1 = y[1], \ldots, x_{i-1} = y[i-1]$ gilt. Nun iterieren wir den Test und finden heraus, ob wir $x_i = 0$ oder $x_i = 1$ setzen sollen. Nach n Iterationen steht schliesslich in $y[1], \ldots, y[n]$ eine erfüllende Belegung.

Zur Laufzeit: Per Annahme ist die Laufzeit von Algorithmus $\mathcal{A}$ polynomiell, also gibt es ein $c > 0$, so dass $\mathcal{A}$ auf Eingabe F Laufzeit $O(|F|^c + c)$ aufweist. Unser Algorithmus ruft nun $\mathcal{A}$ in jedem der n Schritte einmal auf, mit einer Eingabe die maximal $\mathcal{O}(n)$ länger ist als $|F|$. Wir haben bereits oben gesehen dass $|F| = \Omega(m)$, und da $m > n$ folgt $|F| + n = \Theta(|F|)$. Somit erhalten wir Laufzeit $O(n((|F| + n)^c + c)) = O(|F|^{c+1})$. Wir sehen dass unser Vorgehen tatsächlich polynomielle Laufzeit hat.

Algorithm 1 FINDASSIGNMENT(KNF-Formel $F = F(x_1, \ldots, x_n)$)

```
if A(F) = NO then
    return "no satisfying assignment"
end if
F_0 := F
for i = 1, ..., n do
    if A(F_{i-1} ∧ x_i) = YES then
        F_i := F_{i-1} ∧ x_i
        y[i] := 1
    else
        F_i := F_{i-1} ∧ x̄_i
        y[i] := 0
    end if
end for
return y[1], ..., y[n]
```

Lösungsvorschlag zu Aufgabe 3

Wir zeigen zuerst, dass VERTEX COVER in $\mathcal{NP}$ liegt, dies benötigen wir für die $\mathcal{NP}$-vollständigkeit

in den Aufgaben a) und b). Sei (G, k) eine Eingabe, welche in der Sprache VERTEX COVER liegt. Dann gibt es k Knoten V' in G, welche ein Vertex Cover bilden. Wir benutzen V' als Zertifikat. Offensichtlich gilt für die Codierungslängen $|V'| \leq |G|$. Gegeben V', so können wir alle Kanten von G durchgehen und überprüfen, ob jede davon mindestens einen Endpunkt in V' hat. Dies hat Laufzeit $|E| \cdot |V'|$, was polynomiell in $|G|$ ist.

a) Gegeben eine Eingabe (G, k) für das Entscheidungsproblem INDEPENDENT SET. Wir konstruieren daraus eine Eingabe (G', k') für VERTEX COVER, indem wir $G' = G$ und $k' = n - k$ setzen, wobei n die Anzahl Knoten in G ist.

   Angenommen $G = (V, E)$ hat ein Vertex Cover V' der Grösse $n - k$. Dann gibt es per Definition keine Kante in $V \setminus V'$ und somit ist dies ein Independent Set der Grösse k. Umgekehrt, falls es in G ein Independent Set S der grösse k gibt, so ist $V \setminus S$ sicherlich ein Vertex Cover der Grösse $n - k$, da keine Kante beide Endpunkte in S haben kann.

   Trivialerweise kann (G', k') in polynomieller Zeit aus (G, k) berechnet werden.

b) In folgendem Abschnitt gehen wir davon aus, dass jede Klausel der Formel genau drei Literale hat. Bitte beachten Sie, dass jede **3-Sat** Formel F immer in eine solche Formel F' transformiert werden kann, so dass F genau dann erfülbar ist, wenn auch F' erfüllbar ist. Dazu muss jede Klausel, welche nur zwei Literalen enthält, dubliziert werden und es muss eine neue Variable eingefügt werden, welche in einer der zwei Klauseln positiv und in der anderen negativ vorkommt.

   Wir konstruieren einen Graphen G wie folgt aus der 3-SAT Formel F:

   - Nehme zwei Knoten x und $\bar{x}$ für jede Variable x, die in F vorkommt, und verbinde sie mit einer Kante. (Wir nennen diese Knoten *Variablen-Knoten*.)
   - Für jede Klausel von F: Nehme drei (neue) Knoten für die Literale $y_1 \vee y_2 \vee y_3$. Verbinde sie zu einem Dreieck. Verbinde jeden Knoten y_i mit dem zugehörigen Variablen-Knoten x oder $\bar{x}$ (je nachdem ob das Literal negiert ist). Wir nennen diese Knoten *Literal-Knoten*.

   Wir erhalten einen Graphen mit $3m + 2n$ Knoten (für m Klauseln und n Variablen). Siehe Abbildung 1 für ein Beispiel. Diese Transformation ist offensichtlich polynomiell: wir generieren $2n + 3m$ Knoten und $6m + n$ Kanten (aus einer Eingabe der Länge $3m$, wobei $n \leq 3m$ angenommen wird).

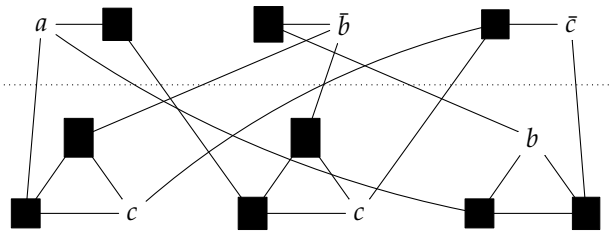

Abbildung 1: Transformation von $(a \vee \bar{b} \vee c) \wedge (\bar{a} \vee \bar{b} \vee c) \wedge (a \vee b \vee \bar{c})$.

   In diesem Graphen definiert jedes $(2m + n)$-Vertex-Cover (Kästchen in Abbildung 1) eine erfüllende Belegung von F: Zuerst überlegt man sich, dass jedes $(2m + n)$-Vertex-Cover genau $2m$ Literal-Knoten und n Variablen-Knoten enthält. (Angenommen es wären weniger als n Variablen-Knoten, dann können die Kanten $\{x, \bar{x}\}$ nicht alle abgedeckt werden; angenommen es wären mehr, dann können die Literal-Dreiecke nicht abgedeckt werden.) Wir müssen also noch die Kanten zwischen den Variablen-Knoten und den Literal-Knoten abdecken. Das ist aber nur dann möglich, wenn das Cover die Variablen-Knoten genau so enthält, dass mindestens eine Kante zu einem Literal hin schon abgedeckt ist. Aus Konstruktion geben dann die Variablen-Knoten im Cover eine erfüllende Belegung für F an.

Umgekehrt definiert jede erfüllende Belegung ein $(2m + n)$-Cover: nehme für jede Variable den Variablen-Knoten x und $\bar{x}$, wenn sie in der Belegung 1 bzw. 0 ist. Wähle dann aus jedem Literal-Dreieck zwei Knoten, so dass ein Vertex Cover entsteht; aus Konstruktion und da die Belegung erfüllend ist, können wir dies immer tun.

Somit haben wir 3-SAT auf k-Vertex-Cover (für $k = 3m + 2n$) reduziert.