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

HS 2019

# Algorithmen und Komplexität
# Lösungsvorschlag zu Übungsblatt 11

## Lösungsvorschlag zu Aufgabe 1

a) Falls $G \in L$ so existiert ein Hamiltonkreis, welchen wir als Zertifikat $w$ benutzen. Da dieser aus $n$ Knoten und $n$ Kanten besteht, kann er offensichtlich so codiert werden dass $|w| \leq |G|^c + c$, wobei $|G|$ die Länge der Codierung des Graphen bezeichnet. Gegeben $G$ und $w$ können wir in polynomieller Zeit überprüfen, dass $G$ tatsächlich einen Hamiltonkreis enthält: Wir testen zuerst, ob $w$ jeden Knoten genau einmal besucht und dann ob alle von $w$ verwendeten Kanten in $G$ sind.

b) Als Erstes betrachten wir die Eingabelänge $|P|$. Sei $\ell$ die Anzahl von Monomen in $P$, also $P = P_1 \pm \ldots \pm P_\ell$ mit $P_j = a_j x_1^{b_{j,1}} \ldots x_k^{b_{j,k}}$. Sei $b = \max\{b_{j,i}\}$. Wir können jeden Exponenten in $\log b$ Bits kodieren. Dann ist die Eingabelänge $|P| = \ell(\log n + k \cdot \log b) + \log n + \log k$.

Falls $(P, n, k) \in L$ so existiert eine Nullstelle in $P \ mod \ n$ welches wir als Zertifikat verwenden. Sei $x = (x_1, \ldots, x_k) \in \{0, \ldots, n-1\}^k$ dieses Zertifikat. Da wir wissen $x_i \in \{0, \ldots, n-1\}$ für alle $i \in \{1, \ldots, k\}$ wissen wir, dass wir unser Zertifikat in $k \log n = O(|P|^2)$ Bits kodieren können. Dies zeigt, dass die Länge des Zertifikats polynomiell beschränkt ist in $|P|$.

Um nun zu testen ob $x$ tatsächlich eine Nullstelle $mod \ n$ für $P$ ist, berechnen wir $P(x) \ mod \ n$.

Addition und Subtraktion in $\mathbb{Z}/n\mathbb{Z}$ können in $O(\log n)$ berechnet werden. Zum Beispiel gilt

$$a + b \ mod \ n = \begin{cases} a + b & \text{falls } a + b < n, \\ a + b - n & \text{falls } a + b \geq n. \end{cases}$$

Die Multiplikation kann in $O(\log^2 n)$ berechnet werden. Angenommen wir haben zwei zahlen $c, d \in \mathbb{Z}/n\mathbb{Z}$, dann ist die Länge von $c \cdot d$ höchstens $2 \log n$. Es ist leicht zu sehen, dass wir die Division mit Rest zweier $O(\log n)$ Bit zahlen in $O(\log^2 n)$ berechnen können. (z.B. schriftliche Division). Wir berechnen $x_i^{b_{j,i}}$ durch *iteratives Quadrieren*. Wir beobachten dass

$$\begin{aligned} x_i^{b_{j,i}} &= (x^2)^{b_{j,i}/2} && \text{für } b_{i,j} \text{ gerade,} \\ x_i^{b_{j,i}} &= x \cdot (x^2)^{b_{j,i}-1/2} && \text{für } b_{i,j} \text{ ungerade.} \end{aligned}$$

Wir können sehen, dass wir $x_i^{b_{j,i}}$ in höchstens $2\lceil \log(b_{j,i}) \rceil$ Multiplikationen berechnen können. Wir schlussfolgern, dass wir $P(x)$ in

$$O(\underbrace{\ell \cdot k \log b \log^2 n}_{potentieren} + \underbrace{\ell \cdot k \log^2 n}_{multiplizieren} + \underbrace{\ell \log n}_{addieren}) = O(|P|^3)$$

berechnen können.

## Lösungsvorschlag zu Aufgabe 2

a) Wir wissen, dass $\{0,1\}^n$ genau $2^n$ viele Elemente enthält. Anders gesagt gibt es für boolsche Variabeln $x_1, \ldots, x_n$ genau $2^n$ verschiedene Belegungen. Für eine Funktion $f$ kann bei jeder Belegung $x$ entweder $f(x) = 0$ oder $f(x) = 1$ auftreten. Mit jeder möglichen Belegung wird also die Anzahl Funktionen $f$ verdoppelt. Somit erhalten wir, dass die Anzahl solcher Funktionen genau $2^{\#\text{ Belegungen}} = 2^{2^n}$ ist.

b) Sei eine boolsche Funktion $f\colon \{0,1\}^n \to \{0,1\}$ gegeben; und sei $A := f^{-1}(1) \subseteq \{0,1\}^n$ die Menge ihrer erfüllenden Belegungen. Die Idee ist nun, für jede erfüllende Belegung $a \in A$ eine Klausel einzuführen, welche nur von der Belegung $a$ erfüllt wird. Sei z.B. $a = (1,0,1)$. Dann ist $a$ die einzige Belegung, welche die Klausel $C_a := x_1 \wedge \bar{x}_2 \wedge x_3$ erfüllt. Durch Disjunktion der so erzeugten Klauseln erhält man eine DNF-Formel $F = \vee_{a \in A} C_a$, die genau von den Belegungen in $A$ erfüllt wird, und somit genau der boolschen Funktion $f$ entspricht.

c) Sei eine boolsche Funktion $f\colon \{0,1\}^n \to \{0,1\}$ gegeben; und sei $A := f^{-1}(0) \subseteq \{0,1\}^n$ die Menge ihrer nichterfüllenden Belegungen. Die Idee ist nun, für jede nichterfüllende Belegung $a \in A$ eine Klausel einzuführen, welche genau diese Belegung ausschliesst. Sei z.B. $a = (1,0,1)$. Dann wird die Klausel $C_a := \bar{x}_1 \vee x_2 \vee \bar{x}_3$ von allen Belegungen ausser $a$ erfüllt. Allgemein ergibt sich für eine beliebige auszuschliessende Belegung $a \in A$ aus der Disjunktion der jeweils entgegengesetzten Literale eine Klausel $C_a$, die von allen Belegungen ausser $a$ erfüllt wird. Durch Konjunktion der so erzeugten Klauseln erhält man eine KNF-Formel $F = \wedge_{a \in A} C_a$, die von allen Belegungen ausser denjenigen in $A$ erfüllt wird, also genau der boolschen Funktion $f$ entspricht.

d) Wir betrachten die sogenannte *XOR*-Funktion auf $k+1$ booleschen Variabeln:

$$XOR(x_1, \ldots, x_{k+1}) := \sum_{i=1}^{k+1} x_i \mod 2$$

Sei $F = C_1 \wedge \ldots \wedge C_m$ eine $k$-KNF, welche die Funktion *XOR* darstellt. Nun unterscheiden wir zwei Fälle: entweder ist jede Klausel $C_i$ für jede Belegung der Variabeln 1 (dies geschieht dann, wenn eine Klausel z.B. $x_1 \vee \overline{x_1}$ enthält). Dann ist aber $F(x) = 1$ für jede Belegung $x$, und $F$ kann deshalb *XOR* sicher nicht darstellen.

Andernfalls gibt es mindestens eine Klausel $C_i$, die für gewisse Belegungen nicht erfüllt ist. Da $F$ eine $k$-KNF ist, enthält $C_i$ höchstens $k$ Literale. Also haben höchsten $k$ Variabeln einen Einfluss auf das Resultat dieser Klausel.

Nun können wir aber all diese (höchstens $k$) Variabeln so belegen, dass kein Literal von $C_i$ erfüllt ist, und somit 0 das Resultat von $C_i$ ist. Die Belegung dieser Variabeln ist dadurch fixiert, und wir wissen bereits, dass die KNF das Resultat 0 hat.

Da wir aber $k+1$ Variabeln zur Verfügung haben, bleibt eine nicht-leere Menge von Variabeln übrig, die wir noch nicht belegt haben. Hier können wir nun die Variabeln so mit 0 oder 1 belegen, dass $XOR(x_1, \ldots, x_{k+1}) = 1$ gilt.

Also gibt es für jede KNF $F$, die *XOR* darstellen soll, eine Belegung der Variabeln $x_1, \ldots, x_{k+1}$, so dass einerseits $F$ nicht erfüllt wird, aber gleichzeitig *XOR* 1 ist. Somit ist *XOR* nicht als $k$-KNF darstellbar.

*Alternativer Lösungsweg:* Wir zählen für festes $k \geq 1$ die Anzahl boolescher Funktionen über $n$ Variablen und die Anzahl $k$-KNF Formeln über $n$ Variablen.

- $\{0,1\}^n$ enthält genau $2^n$ viele Elemente. Anders gesagt gibt es für boolsche Variabeln $x_1, \ldots, x_n$ genau $2^n$ verschiedene Belegungen. Für eine Funktion $f$ kann bei jeder Belegung $x$ entweder $f(x) = 0$ oder $f(x) = 1$ auftreten. Mit jeder möglichen Belegung wird also die Anzahl Funktionen $f$ verdoppelt. Somit erhalten wir, dass es $2^{\#\text{ Belegungen}} = 2^{2^n}$ verschiedene boolesche Funktionen über $n$ Variablen gibt.

- Es gibt $\binom{n}{1} + \cdots + \binom{n}{k} = \sum_{i=1}^{k} \binom{n}{i} \leq \sum_{i=1}^{k} n^i \leq k \cdot n^k = \mathcal{O}(n^k)$ Möglichkeiten, die $k$ Variablen für eine Klausel mit höchstens $k$ Literalen aus den $n$ Variablen auszuwählen. Jede der ausgewählten Variablen kommt auf eine von zwei möglichen Arten in der Klausel vor (nämlich als $x$ oder $\bar{x}$). Es gibt also $2^k\mathcal{O}(n^k) = \mathcal{O}(n^k)$ verschiedene Klauseln mit höchstens $k$ Literalen über $n$ Variablen. Eine $k$-KNF Formel ist eine Teilmenge der Menge aller möglichen Klauseln mit höchstens $k$ Literalen. Für die Potenzmenge $\mathfrak{P}(X) = \{Y \mid Y \subseteq X\}$ einer endlichen Menge $X$ gilt $|\mathfrak{P}(X)| = 2^{|X|}$. Somit gibt es $2^{\mathcal{O}(n^k)}$ mögliche $k$-KNF Formeln.

Da $2^n = \omega(n^k)$ für festes $k$, ist auch $2^{O(n^k)} = o(2^{2^n})$. Für $n$ gross genug gibt es also mehr boolesche Funktionen über $n$ Variablen als $k$-KNF Formeln über $n$ Variablen. Also muss es eine solche Funktion $f\colon \{0,1\}^n \to \{0,1\}$ geben, die nicht als $k$-KNF Formel geschrieben werden kann.

## Lösungsvorschlag zu Aufgabe 3

a) Wir zeigen zuerst, dass ALMOST 3-SAT in $\mathcal{NP}$ liegt. Sei $F = C_1 \wedge \ldots \wedge C_m$ eine gegebene 3-KNF, welche in der Sprache ALMOST 3-SAT liegt, das heisst es gibt eine Belegung $\phi$ für die Variabeln $x_1, \ldots, x_n$, sodass genau eine Klausel nicht erfüllt ist. Sei das Zertifikat $|\phi|$ eine solche Formel. Da jede Variable $x_i$ mindestens einmal in $F$ vorkommt ist $|\phi|$ polynomiell beschränkt in $|F|$. Durch Einsetzten von $\phi$ in $F$ kann in $O(m)$ überprüft werden, ob genau eine Klausel nicht erfüllt ist.

b) Es bleibt zu zeigen, dass 3-SAT $\leq_p$ ALMOST 3-SAT. Für jede Formel $F$ über $x_1, \ldots, x_n$ können wir durch Hinzunahme von $x_{n+1}$ in konstanter Zeit die Formel $F' := F \wedge (x_{n+1}) \wedge (\overline{x_{n+1}})$ konstruieren. Falls es eine Belegung $\phi$ gibt, welche $F$ erfüllt, so erfüllt $\phi$ zusammen mit $x_{n+1} = 1$ genau eine Klausel von $F'$ nicht. Umgekehrt gilt, dass für jede Belegung der Variablen $x_1 \ldots, x_{n+1}$ genau eine der zwei Klauseln $(x_{n+1})$ oder $(\overline{x_{n+1}})$ nicht erfüllt ist. Gibt es also eine Belegung welche genau eine Klausel von $F'$ nicht erfüllt, so erfüllt diese Belegung alle Klauseln in $F$.