Karlsruher Institut für Technologie

# Lösung zum Übungsblatt 3

Abgabe bis 4. Dezember 2024

## Aufgabe 1: Odd Subgraph

**3 + 5 Punkte**

Gegeben ist ein Graph $G$ und ein Parameter $k$. Das Problem Odd Subgraph fragt, ob es einen Subgraphen von $G$ mit $k$ Kanten gibt, in dem alle Knoten ungeraden Grad haben.

**Teilaufgabe (a)** Zeige: Wenn der maximale Knotengrad in $G$ höchstens $k$ ist und $G$ mindestens $k \cdot (2k-1)$ Kanten enthält, dann gibt es ein Matching in $G$ mit $k$ Kanten.

*Lösung 1a:*

Wir bestimmen greedy ein Matching, indem wir eine beliebige Kante ins Matching wählen und die Kante selbst sowie alle adjazenten Kanten löschen. Da der Knotengrad höchstens $k$ ist, ist jede Kante zu höchstens $2 \cdot (k-1)$ Kanten adjazent. Wir löschen also in jedem Schritt höchstens $2k-1$ Kanten. Wegen $|E(G)| \geq k \cdot (2k-1)$ wählen wir damit mindestens $k$ Kanten in das Matching.

**Teilaufgabe (b)** Zeige, dass für Odd Subgraph ein Kern der Größe $O(k^2)$ in polynomieller Zeit berechenbar ist.
*Tipp*: Wie löst du die Fälle, die von Teilaufgabe (a) nicht abgedeckt sind?

*Lösung 1b:*

Zunächst löschen wir alle isolierten Knoten. Wir unterscheiden zwei Fälle.

**Fall 1:** Es gibt einen Knoten $v$ mit Grad $> k$.
Falls $k$ ungerade ist, dann bilden $v$ und die Kanten zu $k$ beliebige Nachbarn von $v$ eine Lösung. Falls $k$ gerade ist und $G$ ein Stern ist, gibt es keine Lösung. Ansonsten gibt es eine Kante $e$, die nicht zu $v$ inzident ist. Da $\deg(v) > k$ gibt es noch $k-1$ weitere Nachbarn $V'$ von $v$, die nicht zu $e$ inzident sind. Zusammen mit der Kante $e$ bilden die Kanten zwischen $v$ und den Knoten in $V'$ eine Lösung mit $k$ Kanten. Eine solche Lösung können wir in Linearzeit bestimmen.

**Fall 2:** Alle Knoten haben Grad $\leq k$.
Wenn $G$ mindestens $k \cdot (2k-1)$ Kanten enthält, gibt es nach Teilaufgabe a) in $G$ ein Matching mit $k$ Kanten. Dieses können wir in Linearzeit bestimmen. Die $k$ Matchingkanten bilden dann eine Lösung für Odd Subgraph.

Ansonsten enthält $G$ weniger als $k \cdot (2k-1)$ Kanten, ist also ein Kern der Größe $O(k^2)$.

## Aufgabe 2: Edge Clique Cover **10 Punkte**

Das parametrisierte Problem Edge Clique Cover ist wie folgt definiert. Gegeben ist ein Graph $G$ sowie ein Parameter $k$. Gesucht ist eine Menge von maximal $k$ Cliquen, sodass jede Kante von $G$ in mindestens einer der Cliquen enthalten ist.

Gib sichere Reduktionsregeln an, die für Edge Clique Cover einen Kern mit maximal $2^k$ Knoten berechnen.

*Hinweis:* Zwei Knoten $u$ und $v$ sind *echte Zwillinge*, wenn $N(u) \cup \{u\} = N(v) \cup \{v\}$ (dabei bezeichnet $N(u)$ die Menge aller Nachbarn von $u$). Kannst du solche echten Zwillinge loswerden?

*Lösung 2:*

**Reduktionsregel 1.** Lösche Knoten ohne Kante
*Sicher:* Diese Knoten können in keiner Clique vorkommen, die eine Kante abdeckt.

**Reduktionsregel 2.** Seien Knoten $a$ und $b$ *echte Zwillinge*. Kontrahiere die Kante (a, b) und verschmelze die Knoten. Wenn $a$ und $b$ keine weiteren Nachbarn als sich selbst haben, dann reduziere $k$ um 1,
*Sicher:* Sei $C$ eine Clique, die o.B.d.A. $a$ enthält. Es ist nie verkehrt, $C$ zu vergrößern. Da die Nachbarschaften von $a$ und $b$ identisch sind, können wir auch $b$ noch zu $C$ hinzufügen. Wenn wir einen von zwei *Zwillingen* also in einer Clique haben, können wir den anderen auch immer zu dieser hinzufügen. Hatten wir vor dem Anwenden der Regel eine Ja-Instanz, haben wir danach auch noch eine, da wir die Kanten von $b$ immer mit der vergrößerten Version von $C$ abdecken können und wir so also auch einfach $b$ weglassen können. Hatten wir vor dem Verschmelzen eine Nein-Instanz, haben wir auch nach dem Verschmelzen eine Nein-Instanz, da das Verschmelzen von $a$ und $b$ keine benötigten Cliquen entfernt, da $b$ immer auch durch die erweiterte $C$-Clique abgedeckt werden kann, ohne dass zusätzliche Cliquen benötigt werden. Falls $a$ und $b$ keine weiteren Nachbarn haben, müssen wir ihre Kante immer abdecken und die Instanz ist genau dann eine Ja-Instanz, wenn die verkleinerte auch eine ist.
*Polynomiell ausführbar:* Für alle $n^2$ Knotenpaare die Nachbarschaften zu vergleichen, um zu entscheiden ob sie verschmolzen werden können, sowie das Updaten des Graphen sind in polynomieller Zeit möglich.

Um den Beweis der Korrektheit zu vereinfachen, zeigen wir zunächst folgendes Lemma:

**Lemma 1.** *Seien a und b zwei beliebige verschiedene Knoten im Kern. Die Cliquen, in denen a vorkommt, sind nicht die selben, in denen auch b vorkommt.*

*Proof.* Nach Reduktionsregel 1 kommen beide in mindestens einer Clique vor. Angenommen

das Lemma gilt nicht und es wären die selben Cliquen, dann hätten sie in jeder der Cliquen die selbe Nachbarschaft und dann wäre folglich auch die komplette Nachbarschaft der beiden Knoten identisch und sie wären *echte Zwillinge*. Das ist aber nach Reduktionsregel 2 ausgeschlossen, was ein Widerspruch ist.

□

Nun können wir die eigentlich gewünschte Aussage zeigen.

**Theorem 2.** *Nach dem wiederholten Anwenden der beiden Reduktionsregeln auf einer lösbare Edge Clique Cover Instanz mit k Cliquen, ist der Kern maximal $2^k$ Knoten groß.*

*Proof.* Da es $k$ Cliquen gibt, gibt es maximal $2^k$ verschiedene Kombinationen, in welchen Cliquen ein Knoten enthalten ist. Da diese Kombination an Cliquen nach unserem Lemma paarweise unterschiedlich sind, kann es maximal $2^k$ Knoten im Kern geben. □

## Aufgabe 3: Anwendung von Lenstra

**4 + 8 Punkte**

Erstelle eine geeignete ILP Formulierung, um zu zeigen, dass folgende Probleme in FPT sind:

**Variety Subset Sum:** Gegeben eine Multimenge $A$ von Zahlen in $\mathbb{N}$ ("Multi" heißt, Zahlen können mehrfach in $A$ auftauchen) und ein Wert $b \in \mathbb{N}$. Gibt es eine Teilmultimenge $X \subseteq A$ mit $\sum_{x \in X} x = b$? Betrachte als Parameter die Anzahl unterschiedlicher Zahlen in $A$.

*Lösung 3:*

Seien $z_1, \dots, z_k$ die uunterschiedlichen Zahlen in $A$ und sei $m_i$ die Anzahl Vorkommnisse von $z_i$. Wir stellen folgendes LP auf, wobei die Variable $x_i$ dafür steht, wie oft $z_i$ in der Lösung vorkommt (für $i \in 1, \dots, k$).

$$\text{Maximiere: } \sum_{i=1}^{i \leq a} x_i$$

$$\text{Nebenbedingungen: für alle } x_i \text{ mit } 1 \leq i \leq k\text{: } 0 \leq x_i \leq m_i$$

$$\sum_{i=1}^{i \leq k} x_i \cdot z_i = b$$

Mit der ersten Menge an Nebenbedingungen wird sichergestellt, dass jede Zahl nur so oft gewählt werden kann, wie sie in $A$ vorhanden ist. Die zweite Nebenbedingung sorgt dafür, dass die Werte der gewählten Zahlen in Summe genau $b$ ergeben.

Die Maximierungsfunktion ist egal, da jede gültige Lösung des ILP auch eine gültige Lösung für Variety Subset Sum darstellt. Da die Anzahl der Variablen genau $k$ ist, folgt mit dem Theorem von Lenstra, dass Variety Subset Sum mit Parameter $k$ in FPT ist.

**Makespan Scheduling:** Gegeben $m \in \mathbb{N}$ Maschinen, $n$ Jobs mit Bearbeitunszeiten $p_1, \dots, p_n \in \mathbb{N}$ und eine Schranke für die maximale Bearbeitungsdauer $k \in \mathbb{N}$. Gibt es eine Verteilung der Jobs auf die Maschinen, sodass keine Maschine länger als $k$ Zeit benötigt? (Ganz formal: Gibt es eine Abbildung $f : \{1, \dots, n\} \to \{1, \dots, m\}$ sodass $\sum_{j : f(j)=i} p_j \leq k$ für alle $1 \leq i \leq m$?)
Betrachte als Parameter die Schranke $k$ auf die Bearbeitungsdauer.

*Hinweis:* Makespan Scheduling benötigt eine ähnliche Strategie wie das ILP für Closest String aus der Übung (am 28.11). Folgende Überlegung könnte nützlich sein: Welche Kosten können die Jobs haben und wie lassen sich die möglichen Jobzuweisungen auf eine Maschine mit einer Funktion in $k$ beschränken?

*Lösung 3:*

Der Schedule einer Maschine ist $k$ Zeitschritte lang und kann nur zwischen 0 bis $k$ Jobs beinhalten. Jeder dieser Jobs kann nur eine Länge von 1 bis $k$ haben. Es gibt also weniger als $(k+1)^k$ verschiedene Schedules $S$ und für jedes $s \in S$ können wir eine Variable $x_s$ erzeugen.

Nun müssen wir wieder mithilfe von Nebenbedingungen sicherstellen, dass nur alle gültigen Lösungen des ILPs auch valide Schudulings sind.

**Eigenschaft 1: nur $m$ Maschinen** Um zu gewährleisten, dass zu jedem Zeitpunkt nur maximal $m$ Maschinen arbeiten, legen wir fest, dass nur maximal $m$ Schedules gewählt werden dürfen. Dies erfüllen wir, indem wir die Summe unserer $(k+1)^k$ Variablen mit $m$ begrenzen, und jede einzelne nicht negativ sein darf:

$$0 \leq \sum_{s \in S} x_s \leq m$$

**Eigenschaft 2: Abarbeiten der $n$ Jobs** Hierfür setzen wir zunächst alle Variablen auf 0, deren Schedules nicht realisierbar sind, weil ihre Gesamtlänge $k$ übersteigt, oder die Anzahl Jobs einer Größe die durch $p_1, \dots, p_n$ verfügbare Anzahl Jobs dieser Größe übersteigt. Die verbleibenden Variablen $S'$ repräsentieren nun alle Schedules, die auch theoretisch umsetzbar sind. Wir schreiben $n_t$ für die Anzahl Jobs mit Länge $t$ und $n_{(s,t)}$ für die Anzahl Jobs mit Länge $t$ in Schedule $s$. Dann fügen wir für jede der $k$ Jobgrößen eine weitere Nebenbedingung ein, die sicherstellt, dass die Anzahl Vorkommnisse dieser Jobgröße genau $n_t$ ergibt:

$$\text{Für Jobgröße } 1 \leq t \leq k\text{: } n_t = \sum_{s \in S'} n_{(s,t)} \cdot x_s$$

**Maximierungsfunktion** Die Maximierungsfunktion ist egal, da jede gültige Lösung eine Lösung für Makespan Scheduling darstellt.

Da die Anzahl Variablen durch eine Funktion in $k$ beschränkt ist, folgt mit Lenstra, dass das Problem in FPT liegt.

## Aufgabe 4: Closest String

**5 Bonus-Punkte**

In dieser Aufgabe sollst du in einer Programmiersprache deiner Wahl ein Programm implementieren, das das Problem Closest String löst. Gegeben ist eine Menge von Strings, gesucht ist ein minimales $k$ und ein String $s$, sodass $s$ zu jedem gegebenen String Hamming-Distanz höchstens $k$ hat.

Nutze dazu den Algorithmus aus der Aktiv-Session 1. Du darfst aber gerne auch weitere Regeln implementieren und Optimierungen machen.

Beschreibe in der PDF-Abgabe, welche Verfahren du implementiert hast und was du noch weiter optimiert hast. Gib außerdem in der PDF-Abgabe für jede Instanz das kleinste $k$ an, für das du eine Lösung gefunden hast.

Gib zusätzlich den Quellcode sowie deine gefundenen Lösungen (im unten beschriebenen Format) als eine ZIP-Datei ab.

Für jede gelöste Instanz kannst du einen Punkt bekommen.

*Dateiformat Eingabe:* In der ersten Zeile steht $n$, die Anzahl der Strings. In den nächsten $n$ Zeilen ist jeweils ein String gegeben. Die Strings haben alle die gleiche Länge.

*Dateiformat Ausgabe:* Eine Zeile mit einem String, der minimale Hamming-Distanz zu den gegebenen Strings hat.

*Hinweis:* Mithilfe der Datei `validator.py` kannst du die Hamming-Distanz deiner Lösung zu den gegebenen Strings bestimmen. Dabei wird das Maximum über alle Hamming-Distanzen ausgegeben.

## Aufgabe 5: Kontaktvermeidung

**8 nicht übertragbare Bonuspunkte**

Ein Virus geht um und muss dringend an der Ausbreitung gehindert werden. Laut Modellrechnungen kann dies bewerkstelligt werden, indem sich jeder nur noch mit (höchstens) $d$ anderen Leuten trifft. Zum Glück können bis zu $k$ Personen geimpft werden, was dazu führt, dass deren Kontakte sofort ignoriert werden können.

Es muss nun also entschieden werden, ob es möglich ist $k$ Knoten aus einem gegebenen Graphen zu löschen, so dass anschließend der Maximalgrad höchstens $d$ ist.

Gib sichere Reduktionsregeln an, die für dieses Problem einen Kern mit Größe polynomiell in

$d + k$ berechnen.

*Lösung 5:*

Zuerst stellen wir fest, dass das Problem für $d = 0$ Vertex Cover entspricht. Dies inspiriert uns zu folgender Kernbildung.

**Reduktionsregel 1.** Falls ein Knoten $v$ mit Grad mehr als $d + k$ existiert, muss er zur Lösung hinzugefügt werden. Lösche $v$ und dekrementiere $k$ um 1.
*Sicherheit:* Wenn $v$ nicht zur Lösung hinzugefügt wird, dann müssten mehr als $k$ seiner Nachbarn zur Lösung hinzugefügt werden, damit $v$ einen Knotengrad von höchstens $d$ erreicht. Jede JA-Instanz muss also $v$ zur Lösung hinzufügen.

**Reduktionsregel 2.** Falls ein Knoten $v$ keine Nachbarn hat, lösche $v$.
*Sicherheit:* Da $v$ keine inzidenten Kanten hat ist er für die Lösbarkeit der Instanz nicht relevant.

**Reduktionsregel 3.** Falls es eine Kante $e = \{u, v\}$ zwischen zwei Knoten $u$, $v$ mit $\deg(u) \leq d$ und $\deg(v) \leq d$ gibt, lösche $e$. *Sicherheit:* Die Regel beeinflusst nicht, welche Knoten Grad über $d$ haben und auch nicht die Adjazenzen solcher Knoten. Folglich ist die resultierende Instanz genau dann lösbar, wenn die ursprüngliche es war.

**Kerngröße.** Wir zeigen nun, dass nach erschöpfender Anwendung der obigen Regeln ein polynomieller Kern vorliegt. Hierzu nehmen wir an, dass JA-Instanz vorliegt und zeigen, dass diese nicht zu groß sein kann.

Sei $X$ mit $|X| \leq k$ eine Lösung, d.h. nach Löschen von $X$ hat jeder Knoten in $G$ höchstens Grad $d$. Dann ist $|N(X)| \leq k \cdot (k + d)$, da Regel 1 nicht anwendbar ist und kein Knoten Grad mehr als $d + k$ hat. Gleichzeitig muss aber jeder Knoten $v$ mit mehr als $d$ Nachbarn hatte und nicht in $X$ ist, Nachbarn in $X$ haben. Knoten die mit höchstens $d$ Nachbarn haben wegen Regel 2 und 3 genau einen Nachbar der dann Grad mindestenz $d$ haben muss. Jeder Knoten hat also höchstens Distanz 2 zu einem Knoten in $X$. Da kein Knoten mehr als $d + k$ Nachbarn hat und $|X| \leq k$, ist die Größe der Instanz polynomiell durch $d + k$ beschränkt.

Da die Reduktionsregeln offensichtlich in Polynomialzeit ausführbar sind, ergibt sich so die gewünschte Kernbildung.