Prof. Dr. M. Grohe E. Fluck, A. Riazsadri, J. Feith

Übungsblatt 8 mit Lösungen

Abgabetermin: Mittwoch, der 14. Dezember 2022 um 14:30

Aufgabe 4 (Entscheidungsvariante zu Optimierungsvariante) **5 Punkte**

Zeigen Sie: Falls die Entscheidungsvariante BPP-E des Bin Packing Problem (BPP, siehe Folie 400) in P ist, so kann auch die Optimierungsvariante BPP in polynomieller Zeit gelöst werden.

Lösung:

Wir nehmen also an, dass die Entscheidungsvariante von BPP in P ist und konstruieren einen Polynomialzeit-Algorithmus $\mathcal{A}_{\text{OPT}}$, der eine optimale Lösung des BPP berechnet und dabei den Polynomialzeit-Algorithmus für die Entscheidungsvariante $\mathcal{A}_{\text{ENT}}$ als Unterprogramm benutzt.

Wir nutzen aus, dass man zwei Objekte x und y "zusammenkleben" kann, indem man beide löscht und durch ein neues Objekt x' mit Gewicht $w(x') = w(x) + w(y)$ ersetzt. Damit wird erzwungen, dass x und y im gleichen Behälter landen. Falls sich dadurch der Wert der Lösung nicht ändert, so gibt es eine Verteilung, die bei gleichbleibender Qualität x und y dem gleichen Behälter zuordnet. Beachte, dass die Lösung durch das Zusammenkleben nicht besser werden kann.

Sei n die Anzahl an Elemente der BPP-Instanz. Durch eine binäre Suche können wir den Wert $b_{\text{opt}} \leq n$ der optimalen Lösung finden und damit b_{opt} in Polynomialzeit finden.

(Hinweis: Lineare Suche ist bei diesem Problem auch möglich. Bei anderen Problemen mit in der Eingabe exponentiellen Suchraum (da Größe n in der Eingabe binär kodiert ist), geht dies nicht.)

Der Algorithmus $\mathcal{A}_{\text{OPT}}$ zur Berechnung der optimalen Verteilung geht nun wie folgt vor:

- $\mathcal{A}_{\text{OPT}}$ iteriert über alle Paare (x, y) von Objekten.
- Dabei klebt $\mathcal{A}_{\text{OPT}}$ das Paar (x, y) zusammen und prüft mit Hilfe von $\mathcal{A}_{\text{ENT}}$, ob es eine Verteilung der Objekte auf b_{opt} Behälter gibt.
  - Falls ja, bleiben x und y zusammengeklebt und der Algorithmus wird rekursiv mit dem zusammengeklebten Objekt fortgesetzt, bis die Anzahl der Objekte b_{opt} ist.
  - Falls nein, dürfen x und y nicht zusammengeklebt werden und es wird das nächste Paar gewählt.

Die auf diese Art erhaltene Verteilung ordnet jedem Behälter ein zusammengeklebtes Objekt zu. Zerlegt man diese Objekte wieder in seine Bestandteile, d.h. die einzelnen Objekte, aus denen es zusammengeklebt worden ist, erhält man die genaue Verteilung.

In jedem Rekursionsschritt wird die Eingabe um ein Objekt verringert; also werden, wenn n die Anzahl der ursprünglichen Objekte ist, $n - b_{\text{opt}}$ Rekursionen durchgeführt. Dabei benötigt jeder Rekursionsschritt maximal $n \cdot (n-1)$ viele Aufrufe von $\mathcal{A}_{\text{ENT}}$. Folglich ist die Laufzeit von $\mathcal{A}_{\text{OPT}}$ polynomiell in der Eingabelänge beschränkt.

Aufgabe 5 (Zertifikat und Verifizierer) 6 (2 + 2 + 2) Punkte

Zeigen Sie, jeweils mit Hilfe eines Polynomialzeitverifizierers, dass die folgenden Entscheidungsprobleme in NP sind. Beschreiben Sie dazu im Detail die Kodierung und die Länge des Zertifikats, sowie die Arbeitsweise und die Laufzeit des Verifizierers (eine Beschreibung ähnlich zu Folie 417 ist **nicht** hinreichend detailiert).

a) Für Vektoren $c, d \in \mathbb{Z}^k$ sei $c \geq d$ falls für alle $i \in \{1, \ldots, k\}$ gilt, dass $c_i \geq d_i$. Wir betrachten folgendes Entscheidungsproblem:

> $\{-1, 0, 1\}$-Restricted Integer Programming
> **Eingabe:** Eine Matrix $A \in \{-1, 0, 1\}^{m \times n}$ und ein Vektor $b \in \{-1, 0, 1\}^m$.
> **Frage:** Gibt es einen Vektor $x \in \{0, 1\}^n$ mit $Ax \geq b$?

b) Sei $G = (V, E)$ ein Graph. Ein *Matching* ist eine Teilmenge $X \subseteq E$, sodass jeder Knoten in V inzident ist zu maximal einer Kante in X.

$\textsc{Matching} = \{(G, k) \mid k \in \mathbb{N}, G \text{ ist ein Graph und hat ein Matching der Größe } k\}$

c) $\textsc{MaxCut} = \{(G, k) \mid k \in \mathbb{N}, G \text{ ist ein Graph und es gibt eine Teilmenge von Knoten } S, \text{ sodass es mindestens } k \text{ Kanten gibt zwischen } S \text{ und } V \setminus S\}$

Lösung:

a) Als Zertifikat verwenden wir $x \in \{0, 1\}^n$, kodiert als String $x_1 x_2 \ldots x_n$. Dieses ist linear in der Eingabegröße.

Der Verifzierer berechnet Ax prüft ob $Ax \geq b$. Für den ersten Schritt braucht er maximal $O(m^2 n^2)$ für den zweiten maximal $O(m^2)$ also ist die Berechnung in polinomieller Zeit möglich.

Die Korrektheit des Verifizierers ist offensichtlich, da das Zertifikat eine gültige Lösung kodiert, wenn diese existiert.

b) Ohne Beschränkung der Allgemeinheit seien die Knoten $V = \{1, \ldots, n\}$, wobei n die Anzahl der Knoten von G ist.

Wenn $k < n/2$, dann verwenden wir als Zertifikat ein Matching X, kodiert als $\text{bin}(v_0)\#\,\text{bin}(u_0)\#\,\text{bin}(v_1)\#\,\text{bin}(u_1)\#\ldots\#\,\text{bin}(v_{k-1})\#\,\text{bin}(u_{k-1})$. Dieses hat Länge $O(k \log n) \in O(n \log n)$. Wenn $k \geq n/2$, dann verwenden wir ein leeres Zertifikat.

Der Verifizierer geht wie folgt vor:

1. Prüfe, ob $k < n/2$ und verwerfe sonst.
2. Prüfe, ob das Zertifikat $2k$ Knoten enthält und verwerfe sonst.

3. Prüfe, ob kein Knoten mehrfach im Matching vorkommt und verwerfe sonst.
4. Prüfe, ob $v_i u_i$ eine Kante in G ist, für alle $i < k$ und verwerfe sonst.
5. Akzeptiere.

Für den ersten Schritt berechnet man erst die Binärdarstellung vom n und vergleicht dann bitweise mit $\text{bin}(k)$. Dies ist in Zeit polynomiell in n und $\log k$ möglich. Für den zweiten Schritt berechnet wie # das Zertifikat enthält in Binärdarstellung. Dann prüft man ob die Zahl ungerade ist und berechnet dann die Hälfte der Zahl (ohne Rest). Zuletzt vergleicht man mit der Binärdarstellung von k. Dies ist in polynomieller Zeit möglich. Für den dritten Schritt zählt man für jeden Knoten ob dieser maximal einmal im Zertifikat vorkommt (indem man zuerst die Binärdarstellung berechnet und dann vergleicht). Auch dieser Schritt ist in polynomieller Zeit möglich. Für den vierten Schritt überprüft man ob der entsprechende Eintrag in der Adjazenzmatrix 1 ist, was auch in polynomieller Zeit möglich ist.

Die Korrektheit des Verifizierers ist offensichtlich, da das Zertifikat eine gültige Lösung kodiert, wenn diese existiert.

c) Ohne Beschränkung der Allgemeinheit seien die Knoten $V = \{v_1, \ldots, v_n\}$, wobei n die Anzahl der Knoten von G ist.

Als Zertifikat verwenden wir einen String $s \in \{0,1\}^n$ wobei $s_i = 1$, wenn $v_i \in S$, für alle $0 < i \leq n$. Die Länge des Zertifikats ist linear in der Eingabelänge.

Der Verifizierer geht wie folgt vor:

1. Initiere Zähler ℓ auf 0.
2. Für jede Kante $v_i v_j \in E$ überprüfe ob $s_i \neq s_j$, wenn ja erhöhe den Zähler um 1.
3. Prüfe ob $\ell \geq k$, verwerfe wenn nicht.
4. Akzeptiere.

Der erste Schritt ist in konstanter Zeit möglich. Der zweite Schritt ist in polynomieller Zeit möglich, da $\ell \leq m$ immer gilt. Der dritte Schritt ist ebenfalls in polynomieller Zeit möglich, da $\ell \leq m$ und damit insbesondere $\text{bin}(\ell)$ polynomielle Länge hat.

Die Korrektheit des Verifizierers ist offensichtlich, da das Zertifikat eine gültige Lösung kodiert, wenn diese existiert.

Aufgabe 6 (nicht zusammenhängende Graphen) **4 Punkte**

Es sei $G = (V, E)$ ein ungerichteter Graph.
Wir definieren nun das Problem GraphNichtZusammenhang:

$$\textsc{GraphNichtZusammenhang} = \{G \mid G \text{ ist ein Graph, der nicht zusammenhängend ist}\}.$$

Zeigen Sie $\textsc{GraphNichtZusammenhang} \leq_p \text{SAT}$. Sie dürfen in der Reduktion *nicht* verwenden, dass Graphzusammenhang in P liegt.

Lösung:

Sei $G = (V, E)$ ein Graph mit n Knoten und m Kanten. Ohne Beschränkung der Allgemeinheit sei $V = \{v_0, \dots, v_{n-1}\}$.

Um eine solche Formel zu konstruieren überlegen wir uns zuerst eine andere Formulierung für das Problem. Ein Graph ist genau dann nicht zusammenhängend, wenn man die Knoten in zwei nicht-leere Mengen aufteilen kann, sodass für alle Kanten gilt, dass ihre Endpunkte nur in einer der beiden Mengen liegen. Diese Formulierung ist sehr ähnlich zu der Formulierung von k-Färbbarkeit. Wir führen also für jeden Knoten zwei Variablen ein, die jeweils angeben ob der entsprechende Knoten in der entsprechenden Menge liegt. Dann formulieren wir eine SAT-Formel, ähnlich zur Färbung. Allerdings müssen wir zusätzlich sicherstellen, dass die Teilmengen nicht leer sind.

Wir definieren eine Formel φ_G mit $2n$ Variablen $x_0^1, x_0^2, x_1^1, \dots, x_{n-1}^2$, sodass

$$G \in \textsc{Nicht Zusammenhängend} \Leftrightarrow \varphi_G \in \text{SAT},$$

wie folgt:

$$\begin{aligned}\varphi_G &= \left(\bigvee_{i=0}^{n-1} x_i^1\right) \wedge \left(\bigvee_{i=0}^{n-1} x_i^2\right) \wedge \left(\bigwedge_{i=0}^{n-1} (x_i^1 \vee x_i^2)\right) \wedge \left(\bigwedge_{v_i v_j \in E} \left(\neg(x_i^1 \wedge x_j^2) \wedge \neg(x_i^2 \wedge x_j^1)\right)\right) \\ &= \left(\bigvee_{i=0}^{n-1} x_i^1\right) \wedge \left(\bigvee_{i=0}^{n-1} x_i^2\right) \wedge \left(\bigwedge_{i=0}^{n-1} (x_i^1 \vee x_i^2)\right) \wedge \left(\bigwedge_{v_i v_j \in E} \left((\overline{x}_i^1 \vee \overline{x}_j^2) \wedge (\overline{x}_i^2 \vee \overline{x}_j^1)\right)\right).\end{aligned}$$

φ_G lässt sich in polinomieller Zeit aus G konstruieren, da φ_G $2n$ Variablen und $2+n+2m$ Klauseln von maximal Länge n hat. Insgesamt ist die Länge von φ_G in $O(n+m)$.

Wenn G nicht zusammenhängend ist, dann gibt es eine Partition der Knoten in zwei nicht-leere Mengen V_1 und V_2, sodass es keine Kante gibt, die die beiden Mengen miteinander verbindet. Setzt man nun x_i^j auf 1, genau dann wenn $x_i \in V_j$, für alle $i < n$ und $j = 1, 2$, dann ist dies eine erfüllende Belegung für φ_G.

Wenn φ_G erfüllbar ist mit einer Belegung α, so sei, für $j = 1, 2$, $X_j = \{v_i \mid x_i^j = 1\}$. Dann sind X_1 und X_2 zwei (nicht zwangsweise) disjunkte Mengen von Knoten. Die ersten

beiden Klauseln von φ_G erzwingen dass die Mengen nicht-leer sind, die Knotenklauseln erzwingen, dass $X_1 \cup X_2 = V$ und die Kantenklauseln erzwingen dass es keine Kante zwischen X_1 und X_2 gibt. Also ist G nicht zusammenhängend. Wir bemerken außerdem dass alle Knoten in $X_1 \cap X_2$ isoliert sind, also keine ausgehende Kante erhalten.