TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 5

PDF erstellt am: 3. Mai 2024

Deadline für dieses Übungsblatt ist **Montag, 13.5.2024, 20:00 Uhr**. Damit Sie für diese Übung Aufgaben anerkannt bekommen können, gehen Sie folgendermaßen vor:

1. Öffnen Sie den TUWEL-Kurs der Lehrveranstaltung *186.866 Algorithmen und Datenstrukturen (VU 5.5)* und navigieren Sie zum Abschnitt *Übungsblätter*.
2. Teilen Sie uns mit, welche Aufgaben Sie gelöst haben **und** welche gelösten Aufgaben Sie gegebenenfalls in der Übungseinheit präsentieren können. Gehen Sie dabei folgendermaßen vor:
   - Laden Sie Ihre Lösungen in einem einzigen PDF-Dokument in TUWEL hoch.
     Link *Hochladen Lösungen Übungsblatt 5*
     Button *Abgabe hinzufügen* bzw. *Abgabe bearbeiten*
     PDF-Datei mit Lösungen hochladen und *Änderungen sichern*.
   - Kreuzen Sie an, welche Aufgaben Sie gegebenenfalls in der Übung präsentieren können. Die Lösungen der angekreuzten Aufgaben müssen im hochgeladenen PDF enthalten sein.
     Link *Ankreuzen Übungsblatt 5*
     Aufgaben entsprechend anhaken und *Änderungen speichern*.

Bitte beachten Sie:

- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft überschreiben. Sollte kurz vor der Deadline etwas schief gehen (Ausfall TUWEL, Internet, Scanner, etc.) und Sie die Endversion mit allen gelösten Aufgaben nicht mehr hochladen können, haben Sie zumindest Ihre Lösungen teilweise schon hochgeladen und angekreuzt. Nach der Deadline ist keine Veränderung mehr möglich. Die Deadline ist strikt – es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen und hochladen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen hochladen (beachten Sie die maximale Dateigröße).
- Beachten Sie die Richtlinien für das An- und Aberkennen von Aufgaben. Details dazu finden Sie in den Folien der Vorbesprechung.

Aufgabe 1.

(a) Welche Sortieralgorithmen werden für die nachfolgenden Funktionen in den angegebenen Programmiersprachen verwendet?

- `sort()` in Python.
- `sort()` in PHP.

(b) Die Funktion `sort()` der C++ Standard Library wird üblicherweise mit einer Kombination mehrere Sortieralgorithmen namens Introsort implementiert.

Welche Algorithmen werden bei Introsort kombiniert werden? Erklären Sie kurz wie diese Kombination funktioniert und welche Vorteile sie bietet.

(c) Ist die Insertionsort Implementierung aus den Vorlesungfolien stabil? Falls ja, argumentieren Sie kurz warum. Falls nein, konstruieren Sie ein Gegenbeispiel.

Aufgabe 2. Im Folgenden sind verschiedene Probleme mit einem Zertifikat und einem Zertifizierer gegeben. Überlegen Sie sich, ob das Zertifikat und der Zertifizierer geeignet sind, um zu zeigen, dass das gegebene Problem in NP ist. Begründen Sie Ihre Antwort.

(a)
- **Problem:** Gegeben eine natürliche Zahl n, ist n eine zusammengesetzte Zahl (also keine Primzahl)?
- **Zertifikat:** Ein Teiler t von n.
- **Zertifizierer:** Überprüfe, dass $\frac{n}{t}$ eine natürliche Zahl ist, und dass $t \notin \{1, n\}$.

(b)
- **Problem:** Gegeben sei ein Graph G und eine natürliche Zahl k. Ist k die minimale Anzahl an Farben, sodass es eine k-Färbung von G gibt?
- **Zertifikat:** Eine k-Färbung von G.
- **Zertifizierer:** Überprüfe, dass keine zwei benachbarten Knoten die selbe Farbe haben.

(c)
- **Problem:** Gegeben sei eine aussagenlogische Formel Φ in konjunktiver Normalform. Ist Φ unerfüllbar?
- **Zertifikat:** Eine Wahrheitsbelegung f für die n Boole'schen Variablen, die Φ nicht erfüllt.
- **Zertifizierer:** Überprüfe, ob f die Formel Φ nicht erfüllt.

(d)
- **Problem:** Gegeben sei ein Graph G. Ist dieser Graph zusammenhängend?
- **Zertifikat:** Ein leerer String.
- **Zertifizierer:** Führe eine Breitensuche von einem beliebigen Knoten aus und überprüfe, dass jeder Knoten erreichbar ist.

(e)
- **Problem:** Gegeben sei ein Graph G und eine natürliche Zahl k. Gibt es eine maximale Clique der Größe k?
- **Zertifikat:** Eine Clique U der Größe k in G.
- **Zertifizierer:** Überprüfe, dass $|U| = k$, dass U eine Clique ist (also alle Knoten in U paarweise verbunden sind) und, dass keine Obermenge von U existiert, die eine Clique ist.

Aufgabe 3.

(a) Sei $\mathcal{P}$ ein Ja/Nein-Problem für Instanzen mit Größe n. Nehmen Sie an, dass es eine Reduktion von $\mathcal{P}$ auf ein Problem $\mathcal{Q}$ gibt, die $O(n^3 \cdot \log n)$ Zeit benötigt. Nehmen Sie weiterhin an, dass Sie Problem $\mathcal{Q}$ in Zeit $O(m^2)$ lösen können, wobei m nun die Eingabegröße einer Instanz von $\mathcal{Q}$ ist.

- Bedeutet dies, dass jede Instanz von $\mathcal{P}$ in polynomieller Zeit gelöst werden kann?
- Geben Sie eine bestmögliche obere Schranke für die Laufzeit für das Problem $\mathcal{P}$ an und begründen Sie Ihre Antwort kurz.

(b) Sei $\mathcal{P}$ ein Ja/Nein-Problem für Instanzen mit Größe n. Nehmen Sie an, dass es eine Reduktion von $\mathcal{P}$ auf ein Problem $\mathcal{Q}$ gibt, die $O(n)$ Zeit benötigt. Nehmen Sie weiterhin an, dass Problem $\mathcal{Q}$ NP-vollständig ist. Welche der folgenden Aussagen sind dann wahr? Begründen Sie Ihre Antwort.

- $\mathcal{P}$ ist in NP.
- $\mathcal{P}$ ist NP-vollständig.
- Falls SAT in polynomieller Zeit gelöst werden kann, kann auch $\mathcal{P}$ in polynomieller Zeit gelöst werden.
- Falls SAT in linearer Zeit gelöst werden kann, kann auch $\mathcal{P}$ in linearer Zeit gelöst werden.

Aufgabe 4. Sei $G = (V, E)$ ein Graph. Wir nennen eine Menge von Knoten $X \subseteq V$ ein Guarding Set, falls jeder Knoten in $V \setminus X$ zu mindestens einem Knoten in X adjazent ist. Das GUARDING SET Problem ist wie folgt definiert:

Gegeben ein Graph $G = (V, E)$ und eine ganze Zahl k. Gibt es ein Guarding Set X von G, sodass $|X| \leq k$?

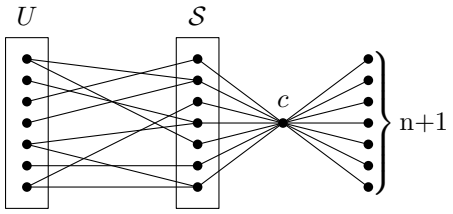
(a) Zeigen Sie, dass das Problem in NP liegt, indem Sie eine Zertifikat/Zertifizierer Kombination angeben.

Zeigen Sie, dass GUARDING SET „genauso schwer" ist wie das Problem SET COVER aus der Vorlesung, indem Sie wie folgt vorgehen.

(b) Geben Sie eine polynomielle Reduktion von GUARDING SET auf SET COVER an und begründen Sie die Korrektheit Ihrer Reduktion.

(c) Überprüfen Sie, ob folgender Beweis eine polynomielle Reduktion von SET COVER auf GUARDING SET liefert.

Beweis. Sei $(U, \mathcal{S}, k)$ eine Instanz von SET COVER mit $|U| = n$. Wir nehmen im Folgenden an, dass für alle $u \in U$ ein $S \in \mathcal{S}$ existiert mit $u \in S$. Anderenfalls ist es eine triviale Nein-Instanz und wir geben eine beliebige Nein-Instanz von GUARDING SET aus. Wenn $k \geq n$, ist es eine triviale Ja-Instanz (wähle für jedes $u \in U$ ein $S \in \mathcal{S}$ mit $u \in S$) und wir geben eine beliebige Ja-Instanz von GUARDING SET aus. Daher können wir weiters annehmen, dass $k < n$. Wir konstruieren nun eine Instanz $(G, k + 1)$ von GUARDING SET:



Der Graph G (siehe Zeichnung) wird dabei wie folgt erzeugt: Wir starten mit der Knotenmenge $U \cup \mathcal{S}$ und verbinden die Knoten $u \in U$ und $S \in \mathcal{S}$ genau dann, wenn $u \in S$. Zusätzlich enthält G einen Knoten c, der mit jedem Knoten in $\mathcal{S}$ und weiteren $n + 1$ anonymen Knoten verbunden ist. G lässt sich offensichtlich in Polynomialzeit erzeugen. Es reicht nun zu zeigen, dass $(U, \mathcal{S}, k)$ eine Ja-Instanz von SET COVER ist, genau dann, wenn $(G, k + 1)$ eine Ja-Instanz von GUARDING SET ist.

($\Rightarrow$): Angenommen $(U, \mathcal{S}, k)$ ist eine Ja-Instanz von SET COVER. Sei $X \subseteq \mathcal{S}$ eine Menge der Größe höchstens k, die dies verifiziert. Daher hat in G jeder Knoten aus U einen Nachbarn in X. Des Weiteren sind alle Knoten außerhalb von U mit c benachbart (oder gleich c). Die Menge $X \cup \{c\}$ ist also ein Guarding Set von G. Daher ist $(G, k + 1)$ eine Ja-Instanz von GUARDING SET.

($\Leftarrow$): Angenommen $(G, k+1)$ ist eine Ja-Instanz von Guarding Set. Sei $X \subseteq V(G)$ eine Menge der Größe höchstens $k+1$ die dies verifiziert. Weil wir oben $k < n$ annahmen, gilt $|X| \leq n$. Daher $c \in X$, denn sonst können die $n+1$ Knoten rechts in der Zeichnung nicht „geguardet“ werden. Also werden alle Knoten in $\mathcal{S}$ von $c \in X$ „geguardet“. Weiters können wir wegen $c \in X$ ohne Beschränkung der Allgemeinheit annehmen, dass X keine anonymen Knoten enthält, da diese einfach entfernt werden können und X immer noch ein Guarding Set von G bleibt.

Da alle Knoten in $\mathcal{S}$ von $c \in X$ „geguardet“ werden, können wir ohne Beschränkung der Allgemeinheit annehmen, dass $U \cap X = \emptyset$, denn ist $u \in U \cap X$, erhalten wir wieder ein Guarding Set von G, wenn wir u aus X entfernen und stattdessen einen beliebigen Nachbarn von u zu X hinzufügen.

Daher muss jeder Knoten in U mindestens einen Nachbarn in $X \cap \mathcal{S}$ haben, also ist $X \cap \mathcal{S}$ ein geeignetes Set Cover. Es verbleibt nur noch $|X \cap \mathcal{S}| \leq k$ zu zeigen: Da $|X| \leq k+1$, $X \subseteq \mathcal{S} \cup \{c\}$ und $c \notin \mathcal{S}$, folgt $|X \cap \mathcal{S}| \leq k$. Also ist $(U, \mathcal{S}, k)$ eine Ja-Instanz von Set Cover. □

Aufgabe 5. Wir nennen einen Graphen $G = (V, E)$ *dreiecksfrei*, wenn V keine drei paarweise unterschiedlichen Knoten u, v, w enthält, sodass $(u, v), (v, w), (u, w) \in E$. Nun betrachten wir das Problem Triangle-Free Vertex Cover, eine Variante des aus der Vorlesung bekannten, NP-vollständigen Vertex Cover Problems:

Gegeben sei ein dreiecksfreier Graph G und eine ganze Zahl k. Gibt es ein Vertex Cover S von G, sodass $|S| \leq k$?

Zeigen Sie formal, dass Triangle-Free Vertex Cover NP-vollständig ist.

Hinweis: Zeigen Sie, dass $(G = (V, E), k)$ genau dann eine Ja-Instanz von Vertex Cover ist, wenn $(G', k + |E|)$ eine Ja-Instanz von Triangle-Free Vertex Cover ist. Hierbei ist G' der Graph der aus G hervorgeht, indem jede Kante (u, v) von G durch einen Pfad $u, x_{u,v}, x_{v,u}, v$ ersetzt wird, wobei $x_{u,v}$ und $x_{v,u}$ neue Knoten sind. Die folgende Abbildung zeigt ein Beispiel der beschriebenen Operation: