Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Komplexität

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 8

für die Übung am Montag den 18. bzw. Dienstag den 19. Juni 2018.

Geben Sie bis **spätestens Sonntag, 17.6.2018, 13:00 Uhr** über TUWEL an, welche Beispiele Sie bearbeitet und gelöst haben. Gehen Sie dabei folgendermaßen vor:

- TUWEL (`https://tuwel.tuwien.ac.at`)
  Kurs *186.866 Algorithmen und Datenstrukturen (VU 5.5)* im Abschnitt *Übungsblätter*
- Bearbeitete Beispiele ankreuzen **und** abgeben
  - Link *Ankreuzen Übungsblatt 8*
    Button *Abgabe bearbeiten*
    Bearbeitete Beispiele anhaken und *Änderungen speichern.*
  - Link *Hochladen Lösungen Übungsblatt 8*
    Button *Abgabe hinzufügen*
    PDF-Datei mit Lösungen hochladen und *Änderungen sichern.*
  - Wenn Sie das Programmierbeispiel gelöst haben:
    Link *Hochladen Source-Code Übungsblatt 8*
    Button *Abgabe hinzufügen*
    Java-Datei (`E8.java`) hochladen und *Änderungen sichern.*

Bitte beachten Sie:

- Sie können **vor** der Deadline beliebig oft ihre Auswahl an Beispielen und das zugehörige Lösungs-PDF verändern, aber **nach** der Deadline gibt es **keine** Veränderung ihrer angekreuzten Beispiele **und** der abgegebenen Dateien!
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen einreichen.
- Bitte geben Sie Ihren Namen, Matrikelnummer und E-Mail-Adresse in den Ausarbeitungen an.
- Wenn Sie das Programmierbeispiel kreuzen, muss sowohl die theoretische Ausarbeitung im PDF, als auch der Source-Code im Tuwel abgegeben werden.
- Beachten Sie die Richtlinien für das An- und Aberkennen von Beispielen in den Vorbesprechungsfolien. Neben der Überprüfung in der Übungseinheit werden danach stichprobenartig weitere Abgaben auf Spekulation und Plagiate überprüft.

Aufgabe 1 Betrachten Sie das Problem *Lastverteilung* aus der Vorlesung zu Approximationsalgorithmen und den zugehörigen Algorithmus `List-Scheduling`, der, wie in der Vorlesung gezeigt, im Allgemeinen eine Approximationsgüte von 2 garantiert.

Wir schränken die zulässigen Probleminstanzen nun wie folgt ein:

1. Es gibt mindestens 100 Jobs.
2. Es gibt höchstens 5 Maschinen.
3. Jeder Job j hat eine Bearbeitungszeit $1 \leq t_j \leq 10$.

Liefert der Algorithmus `List-Scheduling` auf solchen Instanzen eine bessere Approximationsgüte als 2? Falls ja, geben Sie eine möglichst kleine Schranke an, die für alle oben beschriebenen Instanzen gilt und begründen Sie Ihre Antwort. Falls nein, geben Sie ein Gegenbeispiel an, für das die Approximationsgüte weiterhin bei 2 liegt.

Aufgabe 2 Betrachten Sie das aus der Vorlesung bekannte Problem Set Cover als Minimierungsproblem mit einer Grundmenge $U = \{u_1, \ldots, u_n\}$ und einer Menge $\mathcal{S} = \{S_1, \ldots, S_m\}$ von Teilmengen von U unter der Einschränkung, dass jedes Element der Menge U nur in vier oder fünf Teilmengen aus $\mathcal{S}$ enthalten ist.

Geben Sie einen polynomiellen Approximationsalgorithmus mit einer konstanten Gütegarantie für dieses Problem an. Geben Sie an, für welche Konstante die Gütegarantie gilt und warum.

Hat Ihr Algorithmus auch eine konstante Gütegarantie für beliebige Instanzen von Set Cover (also ohne die obige Einschränkung)? Falls ja, begründen Sie warum, falls nein, geben Sie ein Gegenbeispiel an.

Aufgabe 3 Im MaxPart-Problem sucht man für einen Graphen $G = (V, E)$ nach einer Partitionierung der Knoten V in zwei disjunkte Mengen A und B mit $A \cup B = V$, so dass die Anzahl der Kanten (a, b) mit $a \in A$ und $b \in B$ (sog. *Verbindungskanten*) maximiert wird.

Sei $v_1, \ldots, v_n$ eine beliebige Ordnung der Knoten von G. Betrachten Sie den folgenden Greedy-Algorithmus. Zunächst weisen wir v_1 der Menge A zu. Jeden weiteren Knoten v_i, $2 \leq i \leq n$, weisen wir der jeweiligen Knotenmenge zu, für die die Anzahl neuer Verbindungskanten größer ist (bei Gleichheit ist die Zuweisung beliebig).

Weisen Sie nach, dass dieser Algorithmus eine Partitionierung der Knoten findet, die mindestens halb so viele Verbindungskanten hat wie eine optimale Partitionierung.

Hinweis: Weisen Sie jeder Kante den Endknoten mit größerem Index als *verantwortlichen* Knoten zu. Sei r_i die Anzahl der Kanten, für die v_i verantwortlich ist. Argumentieren

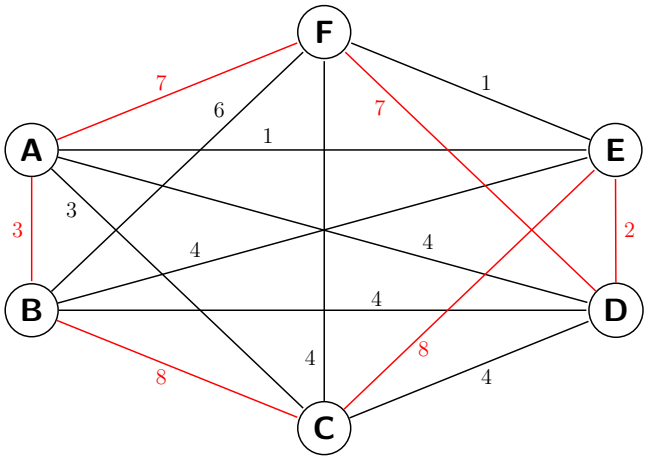
Sie, dass bei der Zuweisung von v_i im Algorithmus mindestens $r_i/2$ Verbindungskanten hinzukommen und vervollständigen Sie das Argument.

Aufgabe 4 Betrachten Sie die folgenden Nachbarschaftsstruktur auf Graphen:

Sei $G = (V, E)$ ein Graph und sei W eine Menge von Knoten, dann ist W' in der Nachbarschaft $N(W)$ von W enthalten, falls W' aus W erzeugt werden kann, indem zwei Knoten u, v aus W durch einen Knoten (entweder u, v oder einen neuen Knoten) ersetzt werden.

- Geben Sie einen Graphen an, auf dem eine lokale Suche nach einem Vertex Cover mit dieser neuen Nachbarschaftsstruktur immer ein optimales Ergebnis liefert, das lokale Optimum einer lokale Suche mit der Nachbarschaftsstruktur aus den Vorlesungfolien aber im Worst-Case um den Faktor 6 größer ist, als das globale Optimum.
- Liefert eine lokale Suche nach einem Vertex Cover mit der neuen Nachbarschaftsstruktur auf jedem Graphen ein lokales Optimum, das maximal um den Faktor 3 schlechter ist als das globale Optimum? Falls ja, begründen Sie, warum die Aussage wahr ist, falls nein, geben Sie ein Gegenbeispiel an.

Aufgabe 5 Betrachten Sie die folgende Instanz des TSP und die durch rote Kanten markierte Tour.



Führen Sie einen Schritt der 2-Opt Lokalen Suche durch, der die größtmögliche Verbesserung erzielt, und zeichnen Sie die resultierende Tour. Falls es mehrere Möglichkeiten gibt, die größtmögliche Verbesserung zu erziehlen, führen Sie alle durch und zeichnen Sie für alle die resultierende Touren.

Aufgabe 6 Programmieraufgabe (1 Punkte)

Ihre Aufgabe ist es, den **Approxiamtions Algorithmus für Vertex Cover** zu implementieren.

Die **detaillierte Angabe** zu dieser Programmieraufgabe finden Sie **im TUWEL** (PDF-Dokument „Programmieraufgabe 8“). Dieses Dokument enthält auch eine Beschreibung der zu implementierenden Funktionen und die Theoriefragen dieser Aufgabe.

Sie bekommen das benötigte Framework („AlgoDat_E8.zip“) ebenfalls im TUWEL zur Verfügung gestellt. Implementieren Sie darin in der Datei „exercise/E8.java“ die geforderten Methoden.

Wichtig: Damit das Beispiel als positiv absolviert gilt, müssen die geforderte Methode korrekt implementiert, der erzeugte Plot in die Abgabe eingefügt, und die Theoriefragen in der PDF-Abgabe ausführlich beantwortet werden!
