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

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 7

für die Übung am Montag den 11. bzw. Dienstag den 12. Juni 2018.

Geben Sie bis **spätestens Sonntag, 10.6.2018, 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 7*
    Button *Abgabe bearbeiten*
    Bearbeitete Beispiele anhaken und *Änderungen speichern.*
  - Link *Hochladen Lösungen Übungsblatt 7*
    Button *Abgabe hinzufügen*
    PDF-Datei mit Lösungen hochladen und *Änderungen sichern.*
  - Wenn Sie das Programmierbeispiel gelöst haben:
    Link *Hochladen Source-Code Übungsblatt 7*
    Button *Abgabe hinzufügen*
    Java-Datei (`E7.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 Lösen Sie die folgende Instanz des Rucksackproblems durch dynamische Programmierung. Geben Sie dazu die vollständige Belegung der 2-dimensionalen Lösungstabelle an wie auf Folie 50 des Foliensatzes „Dynamische Programmierung“. Geben Sie außerdem die Menge der für die optimale Lösung ausgewählten Gegenstände an.

Kapazität $G = 11$

#	Wert	Gewicht
1	2	2
2	6	3
3	9	4
4	15	6
5	19	8

Aufgabe 2 Gegeben sei ein Array A in dem jedem Element eine der Farben Rot (r), Gelb (y) oder Grün (g) zugeordnet ist. Die roten Elemente haben den Wert -1, die gelben Elemente haben den Wert 1 und die grünen Elemente haben den Wert 2. Betrachten Sie das Problem ein zusammenhängendes Teilarray mit mindestens einem Element zu finden, dessen Summe aller enthaltenen Werte maximal ist. Im Beispielarray

$$\{y, r, r, \mathbf{g}, \mathbf{g}, \mathbf{r}, \mathbf{r}, \mathbf{g}, \mathbf{y}, r\}$$

wäre die Lösung das Teilarray vom vierte bis zum neunten Element mit Gesamtsumme 5.

Entwerfen Sie einen Algorithmus in Pseudocode, der den Wert des besten Teilarrays mit Hilfe dynamischer Programmierung und Speicherung der Zwischenlösungen in Arrays bestimmt und dafür Laufzeit $O(n)$ benötigt. Beschreiben Sie, wie man aus den gespeicherten Zwischenlösungen und der Lösung das optimale Teilarray finden kann.

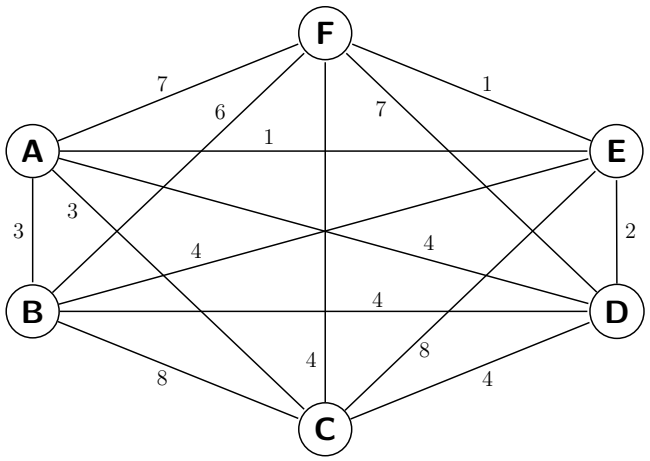
Aufgabe 3 Betrachten Sie das folgende Optimierungsproblem:

> Sei $\mathcal{K} = \{K_1, K_2, \ldots, K_m\}$ eine Menge von m Infektionskrankheiten und $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ eine Menge von n Impfstoffen. Jeder Impfstoff $I \in \mathcal{I}$ schützt gegen eine Teilmenge $K_I \subset \mathcal{K}$ der Krankheiten. Umgekehrt kann jede Krankheit von mindestens einem und höchstens vier verschiedenen Impfstoffen verhindert werden. Sie möchten sich mit einer minimalen Anzahl von Impfstoffen gegen alle Krankheiten in $\mathcal{K}$ schützen.

Entwerfen und beschreiben Sie einen polynomiellen Approximationsalgorithmus mit Gütegarantie 4 für dieses Problem. Erläutern Sie, weshalb die Approximationsgüte gilt und welche asymptotische Laufzeit der Algorithmus hat.

Hinweis: Orientieren Sie sich an den Ideen des 2-Approximationsalgorithmus `Approx-Vertex-Cover` für Vertex Cover aus der Vorlesung.

Aufgabe 4 Betrachten Sie die folgende Instanz des symmetrischen TSP:



Wenden Sie die Spanning-Tree-Heuristik aus der Vorlesung auf diese Instanz an und illustrieren Sie klar jeden Ihrer Schritte. Konstruieren Sie dabei Ihre Eulertour von A ausgehend und besuchen Sie die möglichen Knoten in alphabetischer Reihenfolge, wenn diese nicht eindeutig ist. Welche Länge hat Ihre Tour P? Ist hier die Gütegarantie von 2 aus der Vorlesung eingehalten?

Aufgabe 5 (Programmieraufgabe, 2 Punkte)

Implementieren sie eine naive, sowie eine dynamische Version des "Coin-Games".

Die **detaillierte Angabe** zu dieser Programmieraufgabe finden Sie **im TUWEL** (PDF-Dokument „Programmieraufgabe 7"). Dieses Dokument enthält auch eine Beschreibung der zu implementierenden Funktionen und die Theoriefragen dieser Aufgabe.

Sie bekommen das benötigte Framework („AlgoDat_E7.zip") ebenfalls im TUWEL zur Verfügung gestellt. Implementieren Sie darin in der Datei „exercise/E7.java" die geforderten Methoden.

Wichtig: Damit das Beispiel als positiv absolviert gilt, müssen **alle** geforderten Methoden korrekt implementiert und die Theoriefragen in der PDF Abgabe ausführlich beantwortet werden!
