TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 3

PDF erstellt am: 9. April 2024

Deadline für dieses Übungsblatt ist **Montag, 22.4.2024, 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 3*
     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 3*
     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).

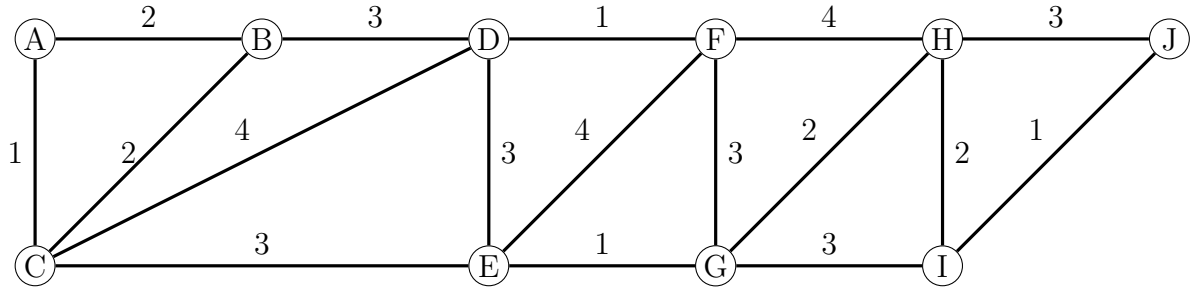
- Beachten Sie die Richtlinien für das An- und Aberkennen von Aufgaben. Details dazu finden Sie in den Folien der Vorbesprechung.

Aufgabe 1. Gegeben seien n Listen, die jeweils m Zahlen in aufsteigender Reihenfolge enthalten. Betrachten Sie den folgenden Lösungsalgorithmus, der daraus eine einzelne, geordnete Liste von $N = nm$ Zahlen erzeugt: Zunächst wird die erste Liste mit der zweiten so verschmolzen, dass eine geordnete Liste der Länge $2m$ entsteht. Danach wird die so erhaltene Liste mit der dritten Liste verschmolzen, sodass nun eine geordnete Liste mit $3m$ Elementen entsteht. Dieses Prozedere wird mit jeder weiteren noch nicht verschmolzenen Liste wiederholt bis man eine Liste der Länge nm erhält, die dann zurückgegeben wird.

(a) Berechnen Sie die Worst-Case Laufzeit des oben skizzierten Algorithmus.

(b) Entwerfen Sie einen effizienteren Algorithmus, der einen Heap beim Vergleich von Elementen aus verschiedenen Listen verwendet. Geben Sie die Worst-Case Laufzeit von dem entworfenen Algorithmus an.

Aufgabe 2. Gegeben sei der nachfolgend abgebildete ungerichtete, kantengewichtete Graph mit 10 Knoten.



(a) Verwenden Sie den Algorithmus von Kruskal, um einen minimalen Spannbaum für diesen Graphen zu finden. Notieren Sie die Reihenfolge, in der Sie die Kanten hinzufügen.

(b) Verwenden Sie den Algorithmus von Prim, um einen minimalen Spannbaum für diesen Graphen zu finden. Beginnen Sie mit Knoten A und notieren Sie die Reihenfolge, in der Sie die Kanten hinzufügen.

(c) Vergleichen Sie die Ergebnisse der beiden Algorithmen. Sind die resultierenden minimalen Spannbäume identisch? Kann es korrekte Ausführungen beider Algorithmen auf der obigen Probleminstanz geben, sodass die zurückgegebenen minimalen Spannbäume übereinstimmen?

Aufgabe 3. Sei $G = (V, E)$ ein Graph mit nicht-negativen Kantengewichten. Gegeben sind ein MST T von G, sowie ein kürzester Pfad P von s nach t in G mit $s, t \in V$. Sei G' der gleiche Graph wie G mit dem Unterschied, dass das Gewicht jeder Kante in E nun um 1 höher ist. Beweisen oder widerlegen Sie (z.B. mit Gegenbeispiel) folgende Aussagen:

(a) T ist auch ein MST von G'.

(b) P ist auch ein kürzester Pfad von s nach t in G'.

Aufgabe 4. Gegeben sind n Projekte $p_1, \ldots, p_n$. Für $1 \leq i \leq n$ bezeichnet s_i die Kosten für das Projekt p_i. Nehmen Sie an, es liegt ein Budget $B \in \mathbb{N}$ mit $B < \sum_{i=1}^{n} s_i$ vor.

(a) Maximiert ein Greedy-Algorithmus, der die Projekte nach *aufsteigenden* Kosten unter Berücksichtigung von B auswählt, die Anzahl der ausgewählten Projekte? Geben Sie einen Beweis oder ein Gegenbeispiel an.

(b) Maximiert ein Greedy-Algorithmus, der die Projekte nach *absteigenden* Kosten unter Berücksichtigung von B auswählt, die Auslastung des Budgets? Geben Sie einen Beweis oder ein Gegenbeispiel an.

Aufgabe 5. Betrachten Sie *Meanquicksort*, eine Variante von Quicksort zum Sortieren von Zahlen, die immer das arithmetische Mittel der jeweils zu sortierenden Subfolge als Pivot-Element verwendet. Ist das arithmetische Mittel kein Element der Subfolge, dann erfolgt das Aufteilen trotzdem durch Vergleich mit dem arithmetischen Mittel.

(a) Führen Sie Meanquicksort auf das folgende Array aus und implementieren Sie den Schritt des Aufteilens so, dass die Elemente einer Subfolge immer in der gleichen Reihenfolge angeordnet werden wie in der Originalfolge. Geben Sie die Zwischenschritte wie im Foliensatz „Divide-and-Conquer“ an.

$$[7, 3, 18, 0, 5, 9, 11]$$

(b) Erklären Sie, wie Sie für ein beliebiges n eine Beispielsequenz der Länge n erstellen können, die zeigt, dass die Worst-Case Laufzeit von Meanquicksort auch nicht besser als quadratisch sein kann. Die Beispielsequenz darf dabei keine Duplikate enthalten.

Hinweis: Es reicht nicht, einzelne Instanzen fester Größe als Beispielinstanzen anzugeben.
