TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 2

Deadline für dieses Übungsblatt ist **Montag, 04.04.2022, 20:00 Uhr**. Um Aufgaben für diese Übung anerkannt zu bekommen, 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 2*
     Button *Abgabe hinzufügen*
     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 2*
     Button *Abgabe bearbeiten*
     Bearbeitete Aufgaben anhaken und *Änderungen speichern*.

Bitte beachten Sie:

- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft verändern. Nach der Deadline ist keine Veränderung mehr möglich. Es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.

- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen hochladen (beachten Sie die maximale Dateigröße).

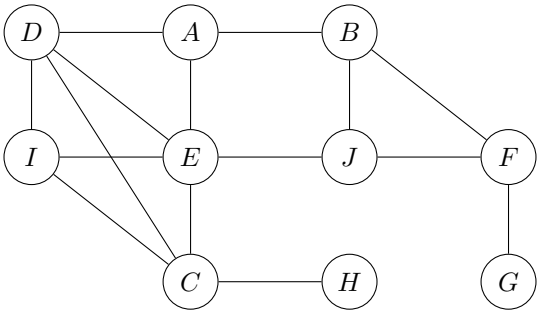
- Geben Sie Ihren Namen, Ihre Matrikelnummer und Ihre E-Mail-Adresse in den Ausarbeitungen an.

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

Aufgabe 1. Führen Sie auf dem nachfolgenden Graphen die Breiten- und Tiefensuche entsprechend den Algorithmen in den Vorlesungsfolien durch. Geben Sie dabei jeweils eine gültige Reihenfolge an, in der die Knoten besucht werden.

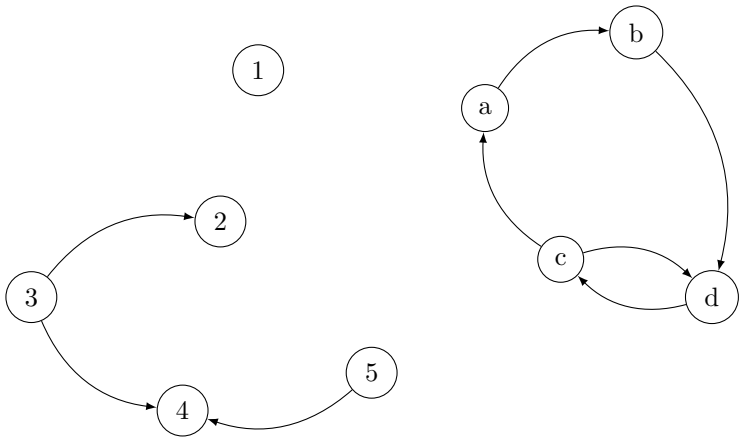
Zeichnen Sie zusätzlich den Breiten- und Tiefensuchbaum. Eine Kante $v \to w$ in diesen Bäumen drückt aus, dass Knoten w von Knoten v aus entdeckt wurde.

Verwenden Sie jeweils A als Startknoten. Haben Sie die Wahl zwischen mehreren Knoten, gehen Sie alphabetisch vor.



Aufgabe 2. Lösen Sie folgende Unteraufgaben zum Thema Zusammenhangskomponenten.

(a) Bestimmen Sie die starken und schwachen Zusammenhangskomponenten des unten abgebildeten Graphen.



(b) Wie viele Kanten müssen zum linken Teilgraphen (mit Knoten 1-5) mindestens hinzugefügt werden, um ihn stark zusammenhängend zu machen? Begründen Sie Ihre Antwort und geben Sie eine kleinstmögliche Kantenmenge an, durch deren Einfügen der Teilgraph stark zusammenhängend wird.

(c) Sei $\mathcal{S}$ die Menge aller gerichteten Graphen $G = (V, E)$, für die gilt: für jedes Knotenpaar $v, w \in V$ gibt es einen gerichteten Pfad von v nach w **oder**[1] von w nach v.

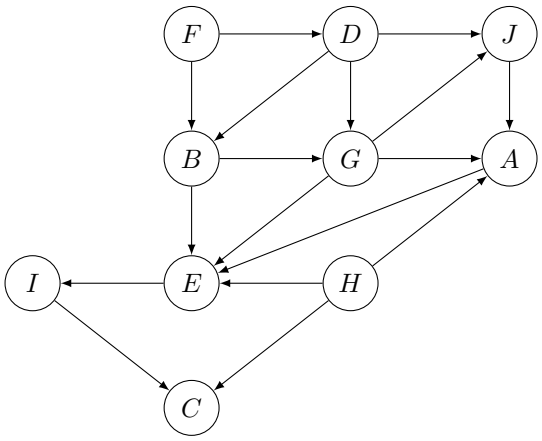
Sei $G = (V, E)$ ein beliebiger gerichteter Graph und seien $G_1 = (V_1, E_1), \ldots, G_n = (V_n, E_n)$ die starken Zusammenhangskomponenten von G. Wir konstruieren einen gerichteten Graphen $G' = (V', E')$ mit $V' = \{1, \ldots, n\}$ und $E' = \{(i, j) \mid \text{es gibt } v \in V_i \text{ und } w \in V_j \text{ sodass } (v, w) \in E\}$. Es gilt, dass $G \in \mathcal{S}$ genau dann wenn $G' \in \mathcal{S}$. Beweisen Sie eine der zwei Implikationen dieser Äquivalenz, nämlich: wenn $G \in \mathcal{S}$ dann $G' \in \mathcal{S}$.

---

[1] Wie üblich handelt es sich hierbei um ein inklusives Oder.

Aufgabe 3. Lösen Sie folgende Unteraufgaben zum Thema topologische Sortierung.

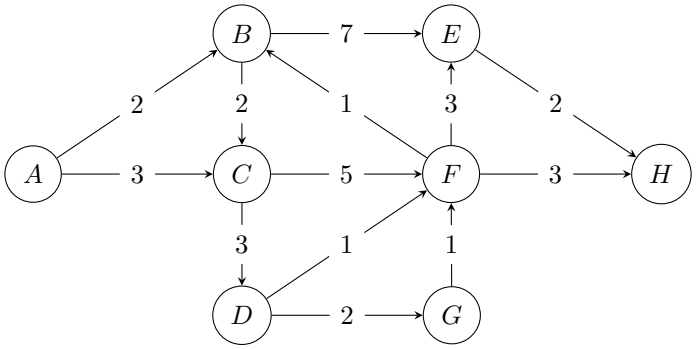
(a) Bestimmen Sie für nachfolgenden gerichteten Graphen eine topologische Sortierung. Wie viele unterschiedliche topologische Sortierungen gibt es?



(b) Sei $\mathcal{S}$ so definiert wie in Aufgabe 2(c). Geben Sie einen Algorithmus mit Laufzeit $O(|V| + |E|)$ an, der als Eingabe einen gerichteten azyklischen Graphen $G = (V, E)$ bekommt und feststellt, ob $G \in \mathcal{S}$.
Hinweis: Besitzt G eine topologische Sortierung? Wenn ja, überlegen Sie wie Sie diese für einen Algorithmus verwenden können.
Anmerkung für Interessierte: Mithilfe der Äquivalenz aus Aufgabe 2(c) lässt sich leicht ein Algorithmus finden, der für beliebige (nicht nur für azyklische) gerichtete Graphen entscheidet, ob sie in $\mathcal{S}$ liegen.

Aufgabe 4. Lösen Sie folgende Unteraufgaben zum Thema Dijkstra-Algorithmus.

(a) Führen Sie den Algorithmus von Dijkstra auf dem nachfolgenden Graphen mit A als Startknoten aus. Geben Sie die Reihenfolge an, in der die Knoten als untersucht/discovered markiert werden. Bestimmen Sie für jeden Knoten die Länge des kürzesten Pfades ausgehend von A. Geben Sie außerdem den kürzesten Pfad von A nach H an.

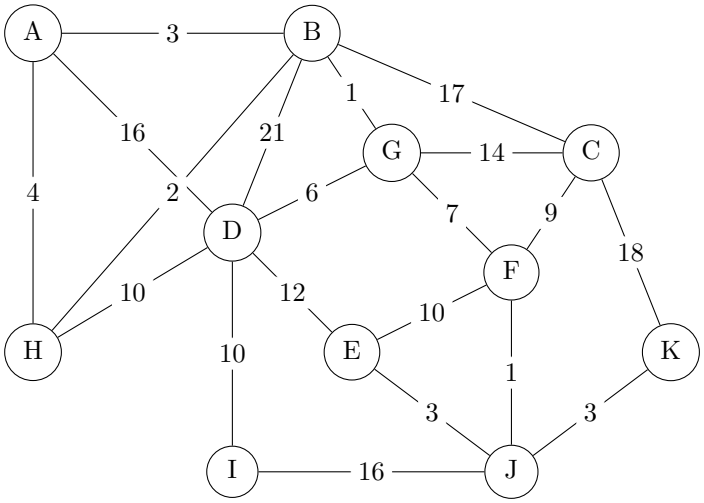


(b) Für den Korrektheitsbeweis des Dijkstra-Algorithmus wurde in der Vorlesung angenommen, dass alle Kanten nicht-negative Gewichte haben. Tatsächlich kann ohne dieser Annahme nicht garantiert werden, dass der Algorithmus von Dijkstra den kürzesten Pfad zwischen zwei Knoten findet. Geben Sie einen gerichteten Graphen an, sodass der Algorithmus von Dijkstra den korrekten kürzesten Pfad zwischen zwei von Ihnen gewählten Knoten in diesem Graphen nicht findet. Dazu muss zumindest eine Kante des Graphen ein negatives Gewicht haben.

Aufgabe 5. Lösen Sie folgende Unteraufgaben zum Thema minimale Spannbäume.

(a) Bestimmen Sie einen minimalen Spannbaum für den unten abgebildeten Graphen. Verwenden Sie dazu den Algorithmus von Prim und beginnen Sie bei **Knoten F**. Haben Sie danach die Wahl zwischen mehreren Knoten, gehen Sie alphabetisch vor.

Geben Sie nach jedem Schleifendurchlauf den aktuell ausgewählten Knoten, den Inhalt der Priority Queue Q, die Knotenmenge S und das Gewicht des aktuellen Spannbaums an.



(b) Sei T ein minimaler Spannbaum eines Graphen G und L die sortierte Folge der Kantengewichte von T. Angenommen T' ist ein anderer minimaler Spannbaum von G. Zeigen Sie, dass die sortierte Folge der Kantengewichte von T' mit L übereinstimmt.

Aufgabe 6. Gegeben sind n Programme $P_1, \ldots, P_n$, welche auf einer Festplatte mit einer Kapazität von D Megabyte gespeichert werden sollen. Der Platz, der für ein Programm P_i benötigt wird, ist s_i Megabyte. Nehmen Sie an, dass es nicht möglich ist, alle Programme gleichzeitig auf der Festplatte zu speichern (es gilt also $D < \sum_{i=1}^{n} s_i$).

(a) Maximiert ein Greedy-Algorithmus, der die Programme in aufsteigender, nach Größe geordneter, Reihenfolge auswählt, die Anzahl der zu speichernden Programme? Geben Sie einen Beweis oder ein Gegenbeispiel an.

(b) Maximiert ein Greedy-Algorithmus, der die Programme in absteigender, nach Größe geordneter, Reihenfolge auswählt, die Auslastung der Festplatte? Geben Sie einen Beweis oder ein Gegenbeispiel an.
