Universität Stuttgart

Übungsblatt 8

Datenstrukturen und Algorithmen (SS 2022)

Abgabe: Montag, 20.06.2022, 15:30 Uhr — Besprechung: ab Montag, 27.06.2022

Abgabevorschriften: Die Aufgaben auf diesem Blatt sind unter Einhaltung der Abgabevorschriften[1] zu lösen und abzugeben.

Lernziele: Nach dem Tutorium zu diesem Blatt sollten Sie folgende Lernziele erreicht haben. Wenn nicht, zögern Sie nicht, Ihre:n Tutor:in anzusprechen um die Lücken zu füllen, Unklarheiten zu klären oder Fragen zu beantworten.

- Sie können Flüsse in Netzwerken erkennen und können das Problem des Maximalen Durchflusses erklären.
- Sie können den Ford-Fulkerson-Algorithmus durchführen und in Java implementieren.
- Sie können Bipartite Graphen erkennen, beschreiben und das Problem des Bipartiten Matchings erklären.
- Sie können die Ungarische Methode durchführen und in Java implementieren.

Punkte: Dieses Übungsblatt beinhaltet 5 Aufgaben mit einer Gesamtzahl von 30 Punkten. Zum Bestehen werden also 15 Punkte benötigt.

Aufgabe 1 Impl Algorithmus von Kruskal implementieren *[Punkte: 4]*

Ziel dieser Aufgabe ist es, einen Minimalen Spannbaum der Wege in der Wilhelma zu finden. Hierfür soll der Kruskal-Algorithmus verwendet werden, den Sie in der Vorlesung kennengelernt haben. Im Eclipse-Projekt sind folgende Codefragmente gegeben:

- Schnittstelle `IEdge` und bereits ausimplementierte Klasse `Edge`
- Schnittstelle `INode` und bereits ausimplementierte Klasse `Node`
- Schnittstelle `IWeightedGraph` und bereits ausimplementierte Klasse `WeightedGraph`
- Die unvollständige Klasse `MinimalSpanningTree`
- Klasse `Wilhelma`

Vervollständigen Sie die Klasse `MinimalSpanningTree`, indem Sie die `kruskal` Methode implementieren

Hinweise:

- Lesen Sie den vorgegebenen Code sorgfältig.
- Modifizieren Sie ausschließlich die zu implementierenden Methoden der Klasse `MinimalSpanningTree`. Alle anderen Klassen und Interfaces dürfen **nicht** verändert werden.
- IWeightedGraph ist ein Objekt zur Speicherung eines gerichteten Graphen. Da der Kruskal-Algorithmus aus der Vorlesung auf ungerichteten Graphen definiert ist, können Sie erwarten, dass es jede Kante zweimal gibt, $s \to t$ sowie $t \to s$. Für die Ausgabemenge der Kanten des minimalen Spannbaums ist jedoch nur ein Vertreter der Kante anzugeben, also entweder $s \to t$ oder $t \to s$.
- Wie bereits bei der Dijkstra-Aufgabe, können Sie sich auch den Minimalen Spannbaum der Wege in der Wilhelma visualisieren lassen. Gehen Sie dafür nachdem Sie den Kruskal-Algorithmus implementiert haben folgendermaßen vor:
  1. Führen Sie `Wilhelma.java` aus.
  2. Kopieren Sie die den gesamten Inhalt der Konsole.

[1] `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_file_2904210.html`

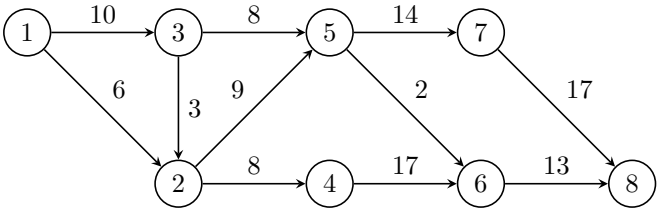
3. Öffnen Sie in einem Browser Ihrer Wahl die Webseite `www.geojson.io`.
4. Fügen Sie den kopierten Inhalt nun in das rechte Feld auf der Webseite ein.

Aufgabe 2 Maximaler Fluss verstehen *[Punkte: 7]*

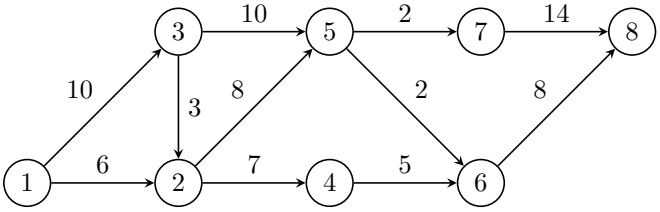
In dieser Aufgabe geht es um das Verständnis von Maximalen Flüssen und dem Ford-Fulkerson-Algorithmus.

(a) *(1 Punkt)* Gegeben sind das Netzwerk N_1 und der Fluss F_1:

Netzwerk N_1:

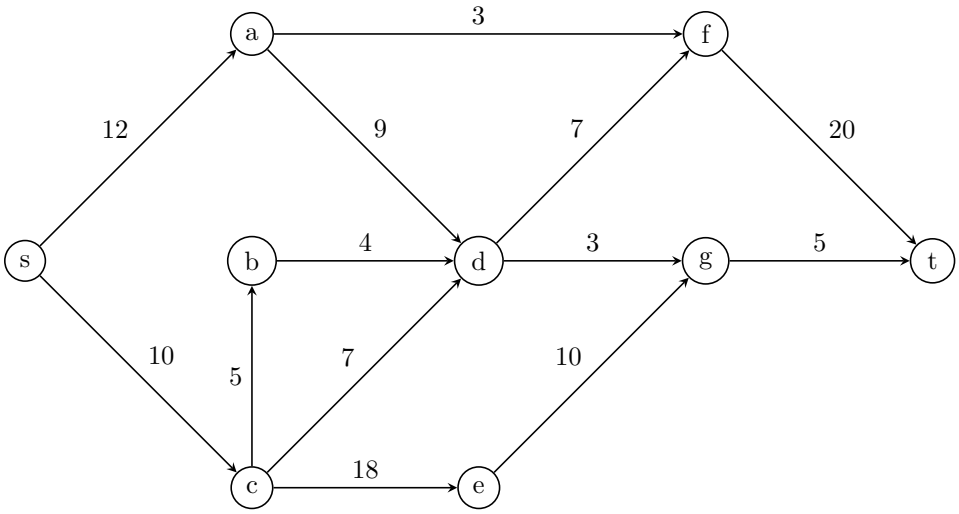


Fluss F_1:



Ist F_1 (mit Quelle 1 und Senke 8) ein korrekter Fluss des Netzwerks N_1? Begründen Sie Ihre Antwort.

(b) *(5 Punkte)* Gegeben sei das folgende Netzwerk N_2:



Führen Sie den Ford-Fulkerson Algorithmus durch um den maximalen Fluss im Netzwerk von s nach t zu bestimmen. Verwenden Sie in jedem Schritt eine Tiefensuche als Auswahlstrategie des nächsten Pfades. Das heißt, um den nächsten Pfad auszuwählen führen Sie auf dem aktuellen Restnetzwerk eine Tiefensuche ausgehend von Knoten s durch, um einen Pfad von s nach t zu identifizieren und fügen diesen Pfad als nächstes hinzu. Wenn ein Knoten mehrere adjazente Knoten hat, arbeiten Sie diese in lexikographischer Reihenfolge, d.h. A < B < C < ..., ab.

Geben Sie zusätzlich zu dem maximalen Fluss, den Sie identifiziert haben auch für jeden Schritt den Fluss, das Restnetzwerk und den Pfad an, den Sie hinzufügen. Wählen Sie hierfür eine geeignete Darstellung, wie Knoten-Kanten-Diagramme (s. Vorlesung) oder Adjazenzmatrizen.

(c) *(1 Punkt)* Können der A*-Algorithmus oder der Kruskal-Algorithmus auch als Auswahlstrategie des nächsten Pfades im Ford-Fulkerson-Algorithmus verwendet werden? Begründen Sie Ihre Antwort.

Aufgabe 3 Impl Maximaler Fluss implementieren *[Punkte: 8]*

In dieser Aufgabe sollen Sie den Ford-Fulkerson-Algorithmus zur Bestimmung des Maximalen Flusses in einem Graphen implementieren. Im Eclipse-Projekt sind folgende Codefragmente gegeben:

- Die unvollständige Klasse `TreeTraversal`
- Schnittstelle `IFordFulkerson` und unvollständige Klasse `FordFulkerson`

(a) *(3 Punkte)* Vervollständigen Sie die Klasse `TreeTraversal`, indem Sie die Methode `dfs` (Abkürzung für Tiefensuche) implementieren.

(b) *(5 Punkte)* Vervollständigen Sie die Klasse `FordFulkerson`, die `IFordFulkerson` implementiert.

Hinweise:

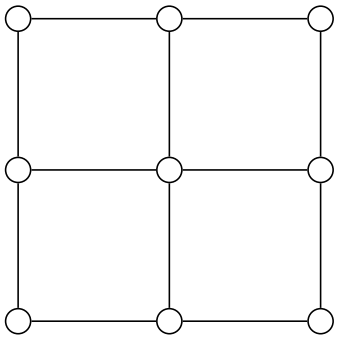
- Lesen Sie den vorgegebenen Code sorgfältig.
- Modifizieren Sie ausschließlich die zu implementierenden Methoden der Klassen `TreeTraversal` und `FordFulkerson`. Alle anderen Klassen und Interfaces dürfen **nicht** verändert werden.
- Achten Sie darauf, dass bei dieser Aufgabe die Graphen in Form einer Adjazenzmatrix (bzw. `ArrayList<ArrayList<Integer>>` gegeben sind).
- Verwenden Sie die Tiefensuche oder einen anderen passenden Algorithmus bei der Implementierung des Ford-Fulkerson-Algorithmus, um den nächsten Pfad auszuwählen. Wenn Sie die Tiefensuche als Auswahlstrategie verwenden, können Sie Ihre Lösung aus Aufgabe 2 verwenden, um Ihre Implementierung zu testen.

Aufgabe 4 Bipartites Matching *[Punkte: 7]*

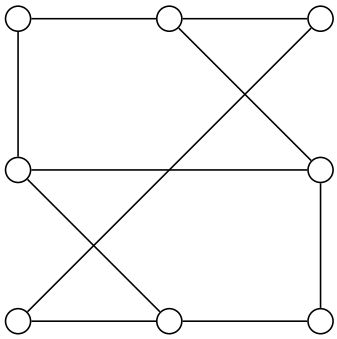
In dieser Aufgabe geht es um das Verständnis von (bipartiten) Matchings und der ungarischen Methode.

(a) *(2 Punkte)* Bestimmen Sie für jeden der folgenden Graphen ein perfektes Matching, oder zeigen Sie, dass kein perfektes Matching existiert.

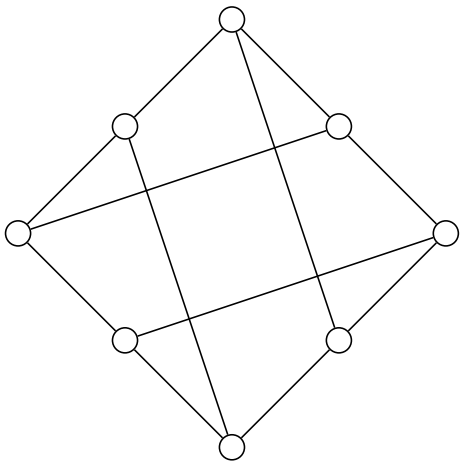
i)



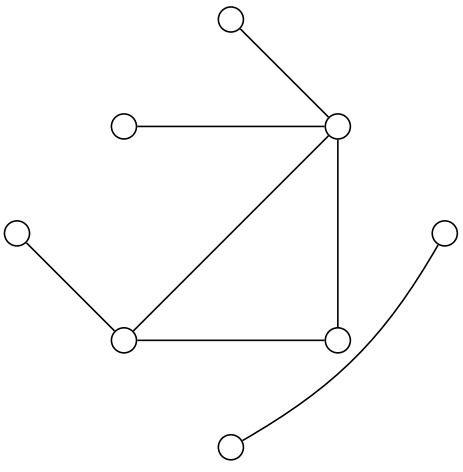
ii)



iii)



iv)



(b) *(4 Punkte)* Anna, Bob, Cleo, Dan, Emma und Flo wohnen zusammen in einer WG, in der wieder der wöchentliche Putztag ansteht. Weil sie mehr oder weniger Lust auf die einzelnen Haushaltsaufgaben haben oder sie mehr oder weniger gründlich erledigen, benötigen sie dafür auch unterschiedlich viel Zeit. Das haben sie in Tabelle 1 festgehalten. Für den anstehenden Putztag wollen sie jetzt die einzelnen Aufgaben so auf die Mitbewohner:innen verteilen, dass jede:r genau eine Aufgabe erledigt und die Aufgaben insgesamt so schnell wie möglich erledigt sind, damit alle das Wochenende genießen können. Bestimmen Sie diese Aufgabenverteilung.

	Bad putzen	Staub saugen	Wäsche bügeln	Geschirr spülen	Kehrwoche machen	Einkaufen gehen
Anna	90	30	50	60	15	90
Bob	30	60	40	30	15	120
Cleo	40	45	60	55	20	75
Dan	120	120	60	40	15	60
Emma	60	50	90	90	15	75
Flo	75	50	75	30	20	80

Tabelle 1: Beobachtete Zeiten der Mitbewohner:innen für die Haushaltsaufgaben.

Wenden Sie hierfür die ungarische Methode an. Geben Sie zusätzlich zur finalen Aufteilung der Aufgaben auf die Mitbewohner:innen auch den Zustand der Adjazenzmatrix nach jedem Schritt (auch jeder Iteration) an und markieren Sie jeweils die durchgestrichenen Linien und minimalen Werte, die für die weitere Berechnung wichtig sind.

Hinweise:
Falls Sie bei der Lösung der Aufgabe an einer Stelle landen, bei der die Anzahl der Linien $= |A|$ ist, aber es nicht möglich ist, mit der aktuellen Adjazenzmatrix ein perfektes Matching zu bestimmen, gibt es eine Bedeckung der Matrix mit weniger Linien.
Falls Sie am Schluss mehr als $|A|$ Nullen in der Adjazenzmatrix finden, müssen Sie diese noch auswählen bzw. reduzieren, sodass Sie ein perfektes Matching erhalten.

(c) *(1 Punkt)* Der in der Vorlesung vorgestellte Algorithmus bezieht sich explizit auf das Minimum Weight Matching Problem. Wie könnte man den Algorithmus anpassen, um das Maximum Weight Matching Problem zu lösen, wenn z.B. die Mitbewohner:innen der WG jeweils eine Aufgabe erledigen, aber insgesamt so viel Zeit wie möglich mit Hausarbeiten verbringen möchten? Begründen Sie Ihre Antwort.

Aufgabe 5 Einsatz von Graphen und Graphalgorithmen *[Punkte: 4]*

Graphen und Graphalgorithmen können für verschiedenste Szenarien zur Lösung von scheinbar sehr unterschiedlichen Problemen eingesetzt werden. Diskutieren Sie die Verwendung von Graphen und passender Algorithmen in den folgenden Teilaufgaben.

(a) *(2 Punkte)* Das "Kleine-Welt-Phänomen"[2] besagt, dass Menschen zu anderen Menschen über erstaunlich kurze Beziehungswege (also über Ketten von Bekannten, die wiederum Personen kennen, die Personen kennen, ...) verbunden sind. Eine Theorie besagt sogar, dass in den USA eine Beziehungskette von einem Bürger zu jedem anderen Bürger aus höchstens 6 Personen besteht. Wie kurz ist wohl die maximale Beziehungskette zwischen zwei Student:innen der Uni Stuttgart?

Wie würden Sie die Situtation der Student:innen als Graph darstellen? Welche Kanten, Knoten, Kantengewichte, etc. würden Sie wählen? Wie könnten Sie die nötigen Informationen für die Erstellung des Graphen sammeln? Gibt es einen in der Vorlesung vorgestellten Algorithmus, mit dem man anschließend die Länge der maximalen Beziehungskette ermitteln könnte? Begründen Sie Ihre Antwort.

(b) *(2 Punkte)* Der Verkehr im Kern einer Stadt soll vermessen werden, um zu bestimmen, wie viele Autos auf einmal hindurchfahren können. Wie würden Sie die Situation als Graph darstellen? Welche Kanten, Knoten, Kantengewichte, etc. würden Sie wählen? Wo würden Sie Vereinfachungen einführen? Gibt es einen in der Vorlesung vorgestellten Algorithmus, mit dem man anschließend den Verkehrsfluss in der Stadt ermitteln könnte? Begründen Sie Ihre Antwort.

[2] `https://de.wikipedia.org/wiki/Kleine-Welt-Ph%C3%A4nomen`