Fakultät IV - Elektrotechnik und Informatik
Algorithmen und Datenstrukturen SoSe 2017
Brock / Mabrouk / Sieverling / Stahl

Aufgabenblatt 7

letzte Aktualisierung: 09. June, 15:41 Uhr

Ausgabe: 9.6.2017
Abgabe: 21.7.2016 19:59

Thema: *Flüsse, Ford-Fulkerson*

Abgabe

Die folgenden Datei muss für eine erfolgreiche Abgabe im svn Ordner
`Tutorien/txx/Studierende/deinname/Abgaben/`
eingecheckt sein:

Geforderte Dateien:
`Blatt07/src/Network.java`
`Blatt07/src/ResidualGraph.java`

Wichtige Ankündigungen

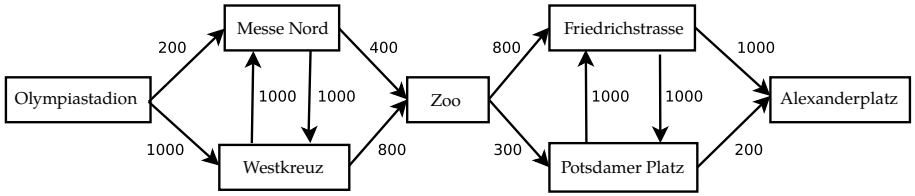
- Unit tests dürfen geteilt werden. Lösungen dürfen auf keinen Fall geteilt werden. Wir testen auf Plagiate.

1. Aufgabe: Flussgraphen - Einführung

Flussgraphen sind gerichtete Graphen mit Kantengewichten, wobei das Gewicht einer Kante die Kapazität des Flusses entlang der Kante angibt.
Flussgraphen modellieren verschiedene Transportprobleme wie z.B. Verkehr in einem Strassennetz, Produktionswege, Routing in einem Netzwerk oder die verfügbare Leistung im elektrischen Energieversorgungsnetz.
Auf diesem Zettel werdet ihr den Ford-Fulkerson Algorithmus benutzen, um maximale Flüsse zu berechnen.

1.1. (Tut) Begriffe - Grundlagen

Nach dem Pokalfinale fahren 75.000 Menschen mit U- und S-Bahn vom Olympiastadion Richtung Alexanderplatz zur Feier. Der Graph unten zeigt einen Ausschnitt des Verkehrsnetzes und die verschiedenen möglichen Wege, modelliert als gerichteter Graph. Die Kantengewichte geben an, wieviele Fahrgäste pro Stunde über jede Teilstrecke transportiert werden können.



Erläutert die folgenden Begriffe anhand des Beispiel-Flussgraphen

- Netzwerk, Kapazität
- Quelle, Senke
- Fluss, maximaler Fluss

1.2. (Tut) Restflussgraph

Was ist der Restflussgraph? Wie sieht der Restflussgraph für die beiden Beispielgraphen in Bildern 3 und 4 aus?

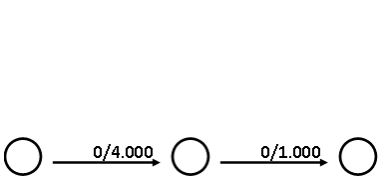


Bild 3

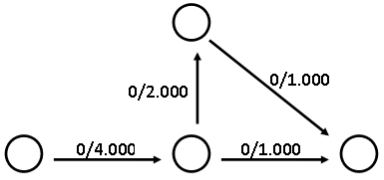


Bild 4

1.3. (Tut) Ford-Fulkerson Algorithmus

Erläutert die beiden Schritte des Ford-Fulkerson Algorithmus am Beispiel-Flussgraphen im Bild 4

- Finde einen augmentierenden Pfad im Restflussgraph
- Update den Restflussgraph mit dem augmentierenden Pfad.

Wendet den Ford-Fulkerson Algorithmus auf die beiden Graphen an.

2. Aufgabe: Edmonds-Karp

2.1. (Tut) Edmonds-Karp-Algorithmus: Theorie Wie verbessert der Algorithmus von Edmonds und Karp die Laufzeit vom Ford-Fulkerson-Algorithmus beim Maximalen-Fluss-Problem?

3. Aufgabe: Flussgraphen - Ford-Fulkerson

3.1. Ford-Fulkerson (100 Punkte) Implementiert die vorgestellte Variante des Ford-Fulkerson Algorithmus in der Methode `int fordFulkerson()` der Klasse `Network.java`.

Die Klasse `Network.java` besitzt eine Funktion `initializeResidualGraph`. Diese erstellt den Residualgraphen zum gegebenen Netzwerk. Implementiert in der Klasse `ResidualGraph.java` die vorgegebenen Methoden:

- `LinkedList<Node> findAugmentingPath(int start, int end)`
  findet einen flussverstärkenden (augmentierenden) Pfad im Residualgraph. Falls keiner existiert, gibt die Methode eine leere Liste zurück. Bei nicht-existierenden Start- oder Endknoten wirft die Methode eine Exception.
- `int findMinCapacity(LinkedList<Node> path)`
  findet die minimale Restkapazität auf dem Pfad und gibt diese zurück. Ein nicht existenter Pfad (leere Knotenliste) hat die Restkapazität 0.
- und `void updateResidualCapacity(int minCapacity, LinkedList<Node> p`
  updated die Restkapazität mit dem gebebenen Fluss über den gegebenen (augmentierenden) Pfad.

Implementiert danach den kompletten Algorithmus in `Network.java`

- `int fordFulkerson()`
  Wendet den Ford-Fulkerson Algorithmus an, um den maximalen Fluss zu berechnen. Diese Methode soll den Residualgraphen erstellen, und die oben implementierten Methoden bis zur Terminierung des Algorithmus ausführen. Die Methode gibt am Ende den Wert des Gesamtflusses zurück.

Hinweis:
- Die Kanten in `Network` und in `ResidualGraph` haben nur eine Variable `weight` für das Kantengewicht. Beim Network entspricht `weight` der Kapazität einer Kante und bei ResidualGraph der Restkapazität. Der aktuelle Fluss wird in unserer Implementierung nicht gespeichert, kann aber jederzeit aus dem Residualgraphen abgelesen werden.
- Kanten mit Restkapazität 0 müsst ihr nicht aus dem Residualgraphen löschen. Ihr könnt sie in findAugmentingPath getrennt behandeln.