Aufgabenblock 4

TECHNISCHE UNIVERSITÄT DARMSTADT

Softwarepraktikum, WS 17/18
Abgabetermin: 14.12.2017 23:59 Uhr
FordFulkerson

Hinweis

Wir messen der Einhaltung der Grundregeln der wissenschaftlichen Ethik größten Wert bei. Mit der Abgabe einer Lösung bestätigen Sie, dass Sie/Ihre Gruppe der alleinige Autor/die alleinigen Autoren des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen Quellen deutlich zitiert haben. Weiterführende Informationen finden Sie unter http://www.es.tu-darmstadt.de/lehre/plagiatshinweise.html.
Weiterhin dürfen **keine Klassen unterhalb** von org.sopra.internal.* referenziert werden, dies führt zu einer Bewertung mit **0 Punkten** für die betreffende Aufgabe. Jede Abgabe muss alle notwendigen Dateien zum Kompilieren enthalten. Das in der jeweiligen Teilaufgabe geforderte Schema der Bezeichnung der Dateien ist einzuhalten.
Zusätzlich soll in jeder Klasse das Interface org.sopra.api.exercises.ExerciseSubmission implementiert werden und die in der Dokumentation beschriebene Funktionalität besitzen.

Lerninhalte

- Kennenlernen der Funktionsweise des Ford-Fulkerson Algorithmus und Üben des Umgangs mit Generics durch das Erstellen einer generischen Klasse, die den Ford-Fulkerson Algorithmus implementiert.
- Umgang mit JUnit durch das Festlegen der Testkriterien und Testen des Algorithmus üben.

Aufgabe 4.1 - Ford-Fulkerson Algorithmus (25 Punkte)

In dieser Aufgabe sollen Sie die generische Klasse `FordFulkersonImpl` erstellen, die den in der Frontalveranstaltung vorgestellten Ford-Fulkerson Algorithmus implementiert. Die Klasse soll das Interface `org.sopra.api.exercises.exercise4.FordFulkerson` implementieren. Speichern Sie die Klasse im Paket `solutions.exercise4` ab.
Um generische Knotentypen zu unterstützen, verwenden Sie den Typparameter `V`. Beachten Sie zusätzlich zu den Angaben in den Javadocs und den Folien der Frontalveranstaltung folgende Hinweise:

a) Implementieren Sie die Methode `augmentPath`. Diese Methode durchläuft den als Parameter übergebenen Pfad und sucht dabei das Minimum der Kapazitäten der Kanten des Pfades. Ist dieses gefunden, wird der Fluss jeder Kante des Pfades mit der Methode `addFlow` um die ermittelte Kapazität erhöht.

b) Implementieren Sie die Methode `findPath`. Diese Methode soll mit Hilfe einer Breitensuche einen kürzesten Pfad mit Kapazitäten größer Null von einem gegebenen Start- zu einem gegebenen Zielknoten in einem Graphen finden. Ein Pfad ist dann kürzer als ein anderer Pfad, wenn er weniger Kanten enthält.

c) Implementieren Sie die Methode `findMaxFlow`. Diese Methode soll mit Hilfe der zuvor erstellten Methoden den maximalen Fluss für einen übergebenen Flussgraphen ermitteln und den Fluss der Kanten dieses Flussgraphen entsprechend aktualisieren. Verwenden Sie zur Erzeugung des Residualgraphen aus dem Flussgraphen die Implementierung aus Übungsblatt 3.

   Beachten Sie beim Aktualisieren der Kanten im übergebenen Flussgraphen Folgendes:

   Eine Kante e_{ij} zwischen den Knoten i und j im Flussgraphen besitzt eine gegenläufige Kante e_{ji} mit gleicher Kapazität $c(e_{ij}) = c(e_{ji})$. Der neue Fluss $f(e_{ij})$ lässt sich dann aus den Residualkapazitäten $c_f(e^r_{ij})$ und $c_f(e^r_{ji})$ des Residualgraphen über folgende Formel bestimmen:

   $f(e_{ij}) = \text{if } c_f(e^r_{ij}) < c_f(e^r_{ji}) \text{ then } c_f(e^r_{ji}) - [c_f(e^r_{ij}) + c_f(e^r_{ji})]/2 \text{ else } 0$

Weiterführende Informationen: Generics, Collection-API, FordFulkerson mit ungerichteten Kanten

Aufgabe 4.2 - Ford-Fulkerson testen (20 Punkte)

In dieser Aufgabe sollen Sie eine Implementierung des `FordFulkerson<T>` Algorithmus auf ihre Funktionsfähigkeit testen. Erstellen Sie dazu die Klasse `FordFulkersonTest`, die von `AbstractFordFulkersonTest` erbt und das Interface `ExerciseSubmission` implementiert. Speichern Sie die Klasse im Paket `solutions.exercise4` ab.
Die Superklasse stellt die in Abbildung 1 dargestellten Graphen als vererbte Objektattribute bereit.
Die Objektattribute sind vom Typ `FlowGraph<String>`. Die Knoten des Graphen sind String-Objekte, die jeweils den Namen der Knoten entsprechen. Die zu testende Implementierung ist über das vererbte Attribut `sut` (*System Under Test*) vom Typ `FordFulkerson<String>` bereitgestellt und wird vor jedem Testfallaufruf neu initialisiert.
Verwenden Sie die zur Verfügung gestellten Graphen als Eingabedaten zur Implementierung folgender Testfälle:

a) Vervollständigen Sie den Testfall `test_findPath1`: Testen Sie die Methode `findPath` mit `flowGraph1` als Testdaten. Bestimmen Sie zunächst das erwartete Ergebnis anhand der Abbildung und vergleichen Sie anschließend das Soll-Verhalten mit der tatsächlichen Ausgabe nach Aufruf von `sut.findPath`.

b) Vervollständigen Sie den Testfall `test_findPath2`: Testen Sie die Methode `findPath` mit `flowGraph2` als Testdaten. Bestimmen Sie zunächst das erwartete Ergebnis anhand der Abbildung und vergleichen Sie anschließend das Soll-Verhalten mit der tatsächlichen Ausgabe nach Aufruf von `sut.findPath`.

c) Vervollständigen Sie den Testfall `test_findPath_IsNull`: Testen Sie die Methode `findPath` mit `flowGraph3` als Testdaten. Bestimmen Sie zunächst das erwartete Ergebnis anhand der Abbildung und vergleichen Sie anschließend das Soll-Verhalten mit der tatsächlichen Ausgabe nach Aufruf von `sut.findPath`.

d) Vervollständigen Sie die Testfälle `test_findPath_ParamStartIsNull`, `test_findPath_ParamTargetIsNull` und `test_findPath_ParamGraphIsNull` indem Sie für jeden Testfall die Methode `sut.findPath` mit den im Namen des Testfalls genannten fehlerhaften Parametern aufrufen. Testen Sie, ob die Methode `sut.findPath` das gewünschte Verhalten im Fehlerfall aufweist.

e) Vervollständigen Sie den Testfall `test_augmentPath1`: Testen Sie die Methode `augmentPath` mit `flowGraph1` als Testdaten. Verwenden Sie als Eingabe den Pfad von `s` nach `t` über `b`, `a` und `d`. Bestimmen Sie zunächst das erwartete Ergebnis anhand der Abbildung und vergleichen Sie anschließend das Soll-Verhalten mit der tatsächlichen Ausgabe nach Aufruf von `sut.augmentPath`.

f) Vervollständigen Sie den Testfall `test_augmentPath2`: Testen Sie die Methode `augmentPath` mit `flowGraph2` als Testdaten. Verwenden Sie als Eingabe den Pfad von `s` nach `t` über `a`, `c`, `b` und `e`. Bestimmen Sie zunächst das erwartete Ergebnis anhand der Abbildung und vergleichen Sie anschließend das Soll-Verhalten mit der tatsächlichen Ausgabe nach Aufruf von `sut.augmentPath`.

g) Vervollständigen Sie den Testfall `test_augmentPath_ParamNull`: Testen Sie, ob die Methode `augmentPath` bei Aufruf mit einem fehlerhaften Parameter `null` das gewünschte Verhalten aufweist.

h) Vervollständigen Sie die Testfälle `test_findMaxFlow_flowGraphA`, `test_findMaxFlow_flowGraphB` und `test_findMaxFlow_flowGraphC` mit den Testdaten `flowGraphA`, `flowGraphB` bzw. `flowGraphC`. Ermitteln Sie zunächst das jeweils erwartete Ergebnis, indem Sie den maximalen Fluss zwischen `s` und `t` bestimmen.
Gehen Sie davon aus, dass das Ergebnis von `sut.findMaxFlow` korrekt ist, wenn sowohl die Summe der ausgehenden Flüsse von `s` als auch die Summe der eingehenden Flüsse nach `t` dem maximalen Fluss entsprechen.

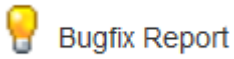
i) Vervollständigen Sie die Testfälle `test_findMaxFlow_ParameterGraphIsNull`, `test_findMaxFlow_ParameterStartIsNull` und `test_findMaxFlow_ParameterTargetIsNull`: Testen Sie, ob die Methode `findMaxFlow` bei einem Aufruf mit einem fehlerhaften Parameter `null` das gewünschte Verhalten aufweist.

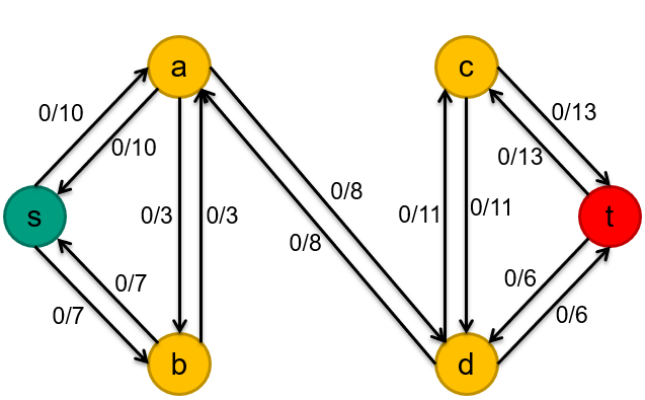
j) Vervollständigen Sie den Testfall `test_findMaxFlow_ParameterTargetNotInGraph`: Testen Sie, ob die Methode `findMaxFlow` bei einem Aufruf mit einem Zielknoten `Target`, der nicht im Graphen enthalten ist, das gewünschte Verhalten aufweist.

Weiterführende Informationen: JUnit

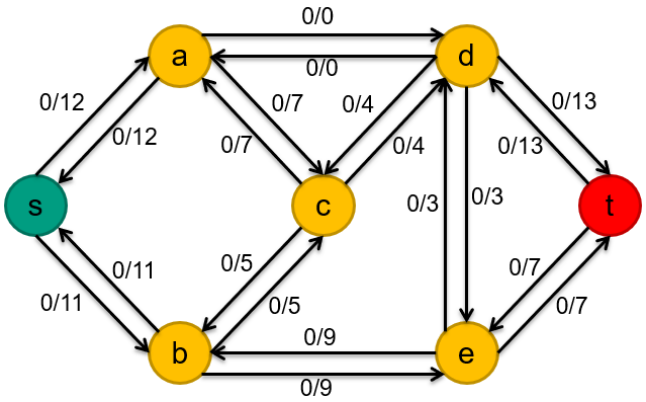
Kritik, Verbesserungsvorschläge und Bug-Report

Sollten Sie Kritik oder Verbesserungsvorschläge haben bzw. Bugs finden, dann nutzen Sie dafür bitte den Bug-Report Button im Moodle-Kurs.

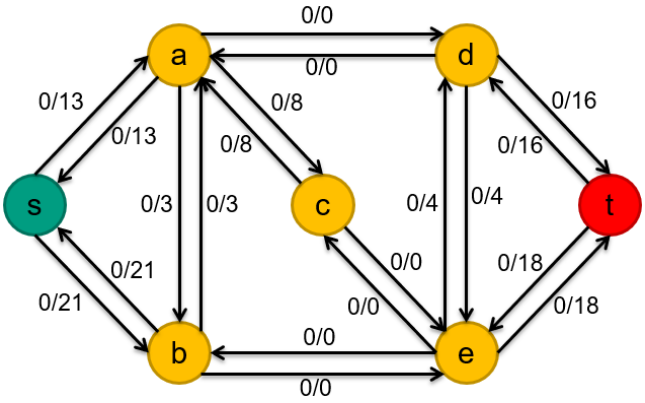




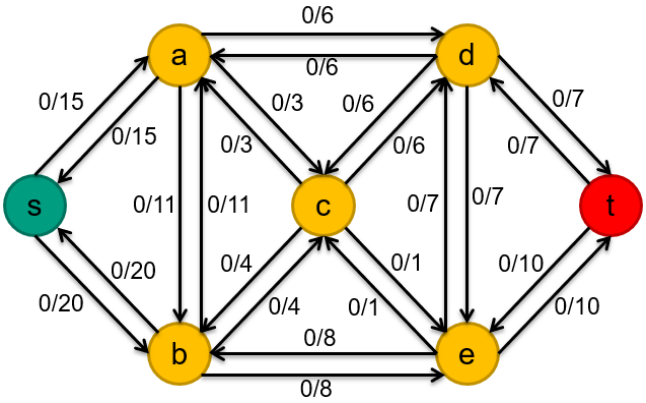
(a) `flowGraph1`



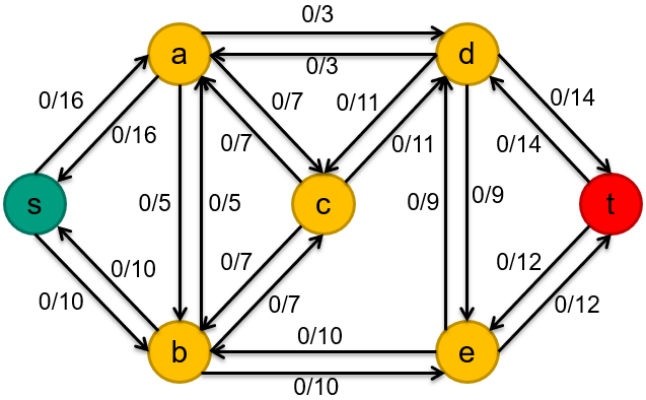
(b) `flowGraph2`



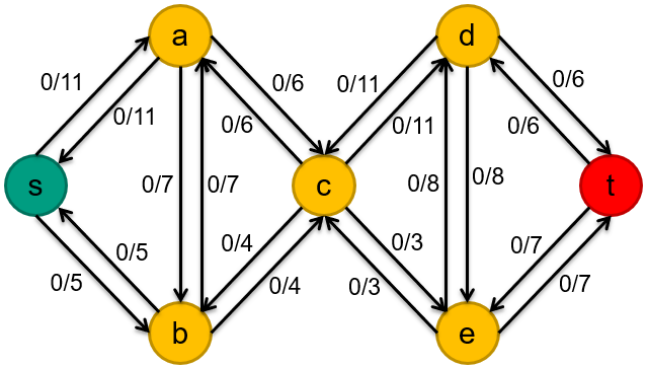
(c) `flowGraph3`



(d) `flowGraphA`



(e) `flowGraphB`



(f) `flowGraphC`

Abbildung 1: Durch die Klasse `AbstractFordFulkersonTest` bereitgestellte Testdaten.