Team: Nummer 09, Sebastian Ewert, Steffen Windrath

Aufgabenaufteilung:

Wir haben alle Aufgaben zusammen besprochen und bearbeitet.

Quellenangaben: Welche Quelle wurde für den Algorithmus verwendet ???

Begründung für Codeübernahme:

1. kein Code übernommen

Bearbeitungszeitraum:

Aktueller Stand: 16.05., etwa 2 Stunden

25.05., etwa 1,5 Stunden

Skizze:

Ford-Fulkerson Algorithmus

Dieser Algorithmus bestimmt den maximalen Fluss eines Graphen durch die Bestimmung von vergrößernden Wegen. Hierzu wird zuerst ein vergrößernder Weg bestimmt. Dieser Fluss wird anschließend im Graphen vermerkt. Anschließend wird der nächste Weg gesucht. Dies wird wiederholt, bis kein Weg mehr gefunden wurde.

Zum Aufruf der Funktion wird die Quelle, die Senke und der Graph benötigt.

1. Kontrolle:

Es wird kontrolliert,

- ob der übergebene Graph schlicht, gerichtet und schwach zusammenhängend ist.
- ob alle Kanten den Wert "maxis" besitzen, und dieser eine $Zahl \in \Re +$ ist
- ob der übergebene Graph die angegebene Quelle und Senke enthält.

Sollte dies nicht der Fall sein, bricht der Algorithmus ab und gibt "nil" zurück.
Sonst wird Schritt 2 ausgeführt.

2. Initialisierung:

1. Allen Ecken des Graphen wird der Wert "vorg" zugewiesen, in dem die Vorgängerecke eines vergrößernden Weges vermerkt wird.
2. Außerdem erhalten die Ecken den Wert "d", in dem, beim ermitteln des Weges, der momentane maximale Fluss des Weges festgehalten wird.
3. Zudem wird den Ecken der Wert "richtung" zugewiesen.
4. Anschließend wird der "maxis"-Wert aller Kanten durch die Werte "max" = "maxis" und "is" = 0 ersetzt.

Der Initialwert von "vorg" und ist "unbekannt", der Initialwert von "d" ist "0". Eine Ausnahme ist hier die Quelle, welche mit dem Wert "undefiniert" für "vorg" und "unendlich" für den "d"-Wert initialisiert wird. Der Initialwert von "richtung" ist "nil".

Anschließend wird Schritt 3 ausgeführt.

3. Wege bestimmen:

Die "findFlowPath" Funktion wird mit folgenden Parametern aufgerufen:

- Liste der markierten Ecken: Eine Liste, die nur die Quelle enthält.
- Liste der inspizierten Ecken: Eine leere Liste.
- Senke: die übergebene Senke.
- Quelle: die übergebene Quelle.
- Graph: der übergebene Graph.

4. Min-Cut bestimmen:

Die “cut” Funktion wird mit folgenden Parametern aufgerufen:

- Menge von Ecken: die von der Funktion “flindFlowPath” zurückgegebene Liste von Ecken.
- Graph: der von der Funktion “findFlowPath” zurückgegebene Graph.

Anschließend werden die “max”-Werte aller Kanten der Form {plus, Kante} und {minus, Kante}, die von der “cut” Funktion zurückgegeben wurden, getrennt aufsummiert (plusSumme und minusSumme genannt). Das Ergebnis von (plusSumme - minusSumme) wird als minimaler Schnitt zurückgegeben.

“findFlowPath” Funktion:

Diese Funktion wird mit einer Liste von markierten Ecken, einer Liste von inspizierten Ecken der Senke, der Quelle und dem Graphen aufgerufen. Anschließend werden folgende Punkte, sofern nicht anders angegeben, nacheinander abgearbeitet.

1. Sollte die Liste der markierten Ecken leer sein wird der übergebene Graph und die Liste der inspizierten Ecken zurückgegeben und die Funktion wird beendet.

2. Wenn die Senke in der Liste der markierten Ecken enthalten ist wird die Funktion “raiseFlow” mit folgenden Parametern aufgerufen:
   - aktuelle Ecke des Weges: Die Senke
   - Wert “d”: der Wert “d” der Senke
   - Graph: der übergebene Graph

   Anschließend wird die Funktion “findFlowPath” mit mit folgenden Parametern rekursiv aufgerufen:

   - Liste der markierten Ecken: Eine Liste, die nur die Quelle enthält.
   - Liste der inspizierten Ecken: Eine leere Liste.
   - Senke: die übergebene Senke.
   - Quelle: die übergebene Quelle.
   - Graph: der von der Funk Graph.

   Sonst werden die Punkte 3 & 4 ausgeführt.

3. Die Funktion wählt eine zufällige Ecke $e \in Liste\ der\ markierten\ Ecken$ und fügt sie der Liste der inspizierten Ecken hinzu.

4. Der Liste der markierten Ecken werden alle Ecken a hinzugefügt, die Adjazent zur Ecke e sind und für die gilt $a \notin Liste\ der\ inspizierten\ Ecken$ und $a \notin Liste\ der\ markierten\ Ecken$, wenn

Vorwärtskanten
   - es eine Kante von e nach a gibt, deren “is”-Wert kleiner ist als dessen “max”-Wert. In diesem Fall wird der Wert “d” der Ecke a auf das Minimum von
     - dem Wert “d” der Ecke e und
     - der Differenz des “is”-Werts und “max”-Werts der Kante von e nach a

     gesetzt. Der Wert “vorg” der Ecke a wird auf e gesetzt. Außerdem wird der Wert “richtung” auf “plus” gesetzt.

Rückwärtskanten
   - es eine Kante von a nach e gibt, deren “is”-Wert größer als “0” ist.
     In diesem Fall wird der Wert von “d” der Ecke a auf das Minimum von
     - dem Wert “d” der Ecke e und
     - des “is”-Werts der Kante von a nach e

     gesetzt. Der Wert “vorg” der Ecke a wird auf e gesetzt. Außerdem wird der Wert “richtung” auf “minus” gesetzt.

5. Die Funktion “findFlowPath” wird mit folgenden Parametern rekursiv aufgerufen:

    - Liste der markierten Ecken: Die modifizierte Liste der markierten Ecke ohne die Ecke e
    - Liste der inspizierten Ecken: Die modifizierte Liste der inspizierten Ecken
    - Senke: die übergebene Senke.
    - Quelle: die übergebene Quelle.
    - Graph: der übergebene Graph.

“raiseFlow” Funktion:

Diese Funktion wird mit der aktuellen Ecke des Weges, dem Wert “d” der Senke und dem Graphen aufgerufen.

1. Die Vorgängerecke e der aktuellen Ecke a wird ermittelt, indem der Wert “vorg” der aktuellen Ecke abgerufen wird.
2. Wenn die Vorgängerecke e “undefiniert” ist wird der übergebene Graph zurückgegeben und die Funktion beendet.
3. Wenn der Wert “richtung” der aktuellen Ecke “plus” ist, wird der “is”-Wert der Kante von e nach a um den Wert “d” erhöht. Sonst wird der “is”-Wert der Kante von a nach e um den Wert “d” reduziert.
4. Die Funktion “raiseFlow” wird mit folgenden Parametern rekursiv aufgerufen:

    - aktuelle Ecke des Weges: Die Vorgängerecke e
    - Wert “d”: der übergebene Wert “d”
    - Graph: der übergebene Graph.

“cut” Funktion:

Diese Funktion wird mit einem Graphen und einer Menge von Ecken x aufgerufen.

1. Es werden alle Kanten des Graphen bestimmt, dessen Anfangsecken in x enthalten sind und dessen Endecken nicht in x enthalten sind. Diese werden in der Form {plus, Kante} in einer Liste zusammengefasst.
2. Es werden alle Kanten des Graphen bestimmt, dessen Endecken in x enthalten sind und dessen Anfangsecken nicht in x enthalten sind. Diese werden in der Form {minus, Kante} der Liste hinzugefügt.
3. Die Liste wird zurückgegeben.

Edmonds - Karp - Algorithmus

Dieser Algorithmus ist eine verbesserte Version des Ford-Fulkerson Algorithmus. Er arbeitet analog wie der Ford-Fulkerson mit folgenden Unterschieden.

- Ein zunehmender(augmentierender) Weg wird hierbei durch Anwendung der Breitensuche gesucht, dazu wird eine Queue verwendet
- Prinzipell muss bei dem Implementierung des Algorithmus folgendes gewährleistet sein:
    - Startpunkt der Funktion ist die Quelle s
    - es werden alle Kanten zu den benachbarten Ecken von s inspiziert und in der Queue aufgenommen
    - Sind alle von s benachbarten und Kanten abgearbeitet, wird das erste Element aus der Queue genommen und als neuer Ausgangspunkt festgelegt
    - um sicher zu gehen, dass eine Ecke und deren anliegende Kanten nicht doppelt begutachtet werden, wird wie oben beschrieben, eine Liste der bereits inspizierten Kanten geführt
- Jeder Knoten erhält zusätzlich ein weiteres Attribut “distance”, indem die Entfernung zur Quelle s gespeichert wird. Dabei ist Entfernung die Anzahl von der Quelle s
- um die “distance” zu ermitteln, wird eine Unterfunktion aufgerufen, welche über den Graphen läuft und die Kanten vom der aktuell bearbeiteten Ecke und der Quelle s durchzählt