Praktikum 3

Graphentheoretische Konzepte und Algorithmen

Bei der Aufgabe des dritten Praktikums handelt es sich um die Implementation zweier Algorithmen zur Findung vergrößernder Flüsse in unserer Graphen Implementation. Bei den implementierten Algorithmen handelt es sich um den Ford-Fulkerson und den Edmond-Karp Algorithmus

Steffen Giersch & Maria Lüdemann

Gruppe 12

HAW Hamburg

03.12.2013

Inhaltsverzeichnis

1. Aufgabenteilung: 2
2. Quellenangaben: 2
Begründung: 2
3. Bearbeitungszeitraum 2
4. Aktueller Stand 3
5. Skizze 3
6. Änderungen 5
7. Zugriffe 5

1. Aufgabenteilung:

Student	Aufgabe
Steffen Giersch	Entwurf, Implementation, Test
Maria Lüdemann	Entwurf, Implementation, Test

Da wir uns beim Programmieren und Planen immer zusammen setzten haben wir jeden Teil gemeinsam bearbeitet.

2. Quellenangaben:

- Ford-Fulkerson: Graphentheorie für Studierende der Informatik, Christoph Klauck & Christoph Maas,4. Auflage 2011 diente als Vorlage für den Pseudocode
- Edmond-Karp: Zur erklärung verwendeten wir die Beschreibung von Wikipedia

Begründung:

Wir übernahmen für diesen Aufgabenteil keinen Fremdcode doch zogen wir sehr anschauliche Algorithmen Beschreibungen zu rate

3. Bearbeitungszeitraum

Datum	Dauer	Aufgabe
18.11.2013	2 Stunde	Planung erste Implementation des Ford-Fulkerson
21.11.2013	2 Stunden	Implementation des Ford-Fulkerson
27.11.2013	1 Stunden	Fehlersuche und Brichtigung des Ford-Fulkerson
28.11.2013	2,5 Stunden	Planung und Implementation des EdmondKarp
3.12.2013	0,5 Stunde	Implementation der Dereferenzierung und beenden des Protokols

4. Aktueller Stand

- Fertig

5. Skizze

Ford Fulkerson:

```
fordFulkerson source target
        Setze den Fluss(e) aller Kanten auf 0
        Markiere jede Ecke als nicht inspiziert (0) und nicht markiert (0)
        Markiere die Quelle und setze maxFlow(v) auf unendlich

        while(true)
                allInspected = false

                while(true)
                        Wenn target markiert ist bricht die Schleife ab
                        Wähle eine zufällige markierte, aber nicht inspizierte Ecke vi
                        und markiere sie als inspiziert
                        Wenn keine Ecke gefunden wurde, setze allInspected = true und
                        brich die Schleife ab

                        Für jede ausgehende Kante ei aus vi
                                Wenn ei nicht markiert und nicht voll ausgelastet ist
                                        Markiere das Ziel von ei
                                        Setze den Vorgänger von diesem auf vi
                                        Markiere den Fluss als Positiv
                                        Setze maxFlow(ziel) auf min(kapazität(ei) -
                                        fluss(ei), maxFlow(vi))

                        Für jede eingehende Kante ei in vi
                                Wenn ei nicht markiert ist und einen Fluss > 0 hat
                                        Markiere die Quelle von ei
                                        Setze den Vorgänger von dieser auf vi
                                        Markiere den Fluss als Negativ
                                        Setze maxFlow(quelle) auf min(fluss(ei),
                                        maxFlow(vi)

                        endWhile

                Wenn allInspected = true dann brich die Schleife ab

                id = target
                moreFlow = maxFlow(target)

                while(id ist nicht die Quelle)
                        Wenn wir eine Vorwärtskante haben
                        Addiere zum Fluss in id  von dem Vorgänger aus moreFlow hinzu
                        Wenn wir eine Rückwärtskante haben
```

```
                    Subtrahiere vom Fluss aus id in den Vorgänger moreFlow
                    id = vorgänger(id)
                endWhile

                Entferne die Markierungen von allen Ecken außer der Quelle
            endWhile

            Gebe die Summe aller ausgehenden Flüsse aus source als Ergebnis aus
end
```

Edmond-Karp:

```
edmondsKarp source target
        Setze den Fluss(e) aller Kanten auf 0
        Markiere jede Ecke als nicht inspiziert (0) und nicht markiert (0)
        Markiere die Quelle und setze maxFlow(v) auf unendlich

        while(true)
                allInspected = false
                warteschlange = {}

                while(true)
                        Wenn target markiert ist bricht die Schleife ab
                        Füge alle neu markierten und nicht inspizierten Ecken hinten an
die warteschlange an
                        Wenn die warteschlange leer ist, setze allInspected = true und
                        brich die Schleife ab
                        Setze vi auf das vorderste Element der Warteschlange

                        Für jede ausgehende Kante ei aus vi
                                Wenn ei nicht markiert und nicht voll ausgelastet ist
                                        Markiere das Ziel von ei
                                        Setze den Vorgänger von diesem auf vi
                                        Markiere den Fluss als Positiv
                                        Setze maxFlow(ziel) auf min(kapazität(ei) -
fluss(ei), maxFlow(vi))

                        Für jede eingehende Kante ei in vi
                                Wenn ei nicht markiert ist und einen Fluss > 0 hat
                                        Markiere die Quelle von ei
                                        Setze den Vorgänger von dieser auf vi
                                        Markiere den Fluss als Negativ
                                        Setze maxFlow(quelle) auf min(fluss(ei),
                                        maxFlow(vi)

                        endWhile

                Wenn allInspected = true dann brich die Schleife ab
```

```
            id = target
            moreFlow = maxFlow(target)

            while(id ist nicht die Quelle)
                    Wenn wir eine Vorwärtskante haben
                    Addiere zum Fluss in id  von dem Vorgänger aus moreFlow hinzu
                    Wenn wir eine Rückwärtskante haben
                    Subtrahiere vom Fluss aus id in den Vorgänger moreFlow
                    id = vorgänger(id)
            endWhile

                Entferne die Markierungen von allen Ecken außer der Quelle
        endWhile

        Gebe die Summe aller ausgehenden Flüsse aus source als Ergebnis aus
end
```

6. Änderungen

Wir haben seit der letzten Aufgabe einige nötige Änderungen vorgenommen. Dabei haben wir den GraphReader um für die Tests eine einfachere Art und Weise haben den Graph einzulesen sowie eine vereinfachte Möglichkeit haben Attribute mit einzulesen.

7. Zugriffe

Graph 8:

Misst man die Zeit die die beiden Algorithmen anhand ihrer Dereferenzierung benötigen kommt man auf folgenden Vergleich:

Edmond Karp : 28335 Schritte

Ford Fulkerson: 17460, 18202, 24268,25722,23779,18228,17225 Schritte

Er benötigt also im Schnitt zwischen 18000 und 24000 Schritten zum Lösen. Diese Zahl schwankt da unsere Implementation des Ford Fulkersons mit Zufallswerten rechnet. Durch das verwenden einer Warteschlange bleibt die Zeit des Edmond Karp Algorithmus konstant.

Ein weiteres Beispiel anhand des Graphen 9:

Hier benötigt der Edmond Karp konstant 4211 Schritte

Und der Ford Fulkerson: 2973,2128,2958,2830,2626,2342,2711

Anhand dieser beiden Beispielgraphen ist ganz gut zu sehen, dass die Implementation mit Zufallswahl der Wege ein wenig perfomanter ist da sie im Schnitt immer unter der Zeit liegt als die Implementation mit Warteschlange