Wintersemester 2013/14
Prof. Dr. J. Vygen
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Einführung in die Diskrete Mathematik
4. Übung

1. Wie kann man in linearer Zeit entscheiden, ob ein gerichteter Graph eine aufspannende Arboreszenz enthält? (3 Punkte)

2. Sei G ein gerichteter Graph mit Kantengewichten $c : E(G) \to \mathbb{R}_+$. Seien $s, t \in V(G)$, $L \subseteq V(G)$, $L \neq \emptyset$, so dass von jedem Knoten aus jedes Element von L erreichbar ist, und $\pi(v) := \min\left\{0, \min_{l \in L}\left(\text{dist}_{(G,c)}(t,l) - \text{dist}_{(G,c)}(v,l)\right)\right\}$ für $v \in V(G)$. Beweisen Sie die folgenden Aussagen:

   (a) π ist ein zulässiges Potential.

   (b) Jeder kürzeste s-t-Weg in (G, c_π) ist ein kürzester s-t-Weg in (G, c).

   (c) $\left\{v \in V(G) \mid \text{dist}_{(G,c_\pi)}(s,v) < \text{dist}_{(G,c_\pi)}(s,t)\right\} \subseteq \left\{v \in V(G) \mid \text{dist}_{(G,c)}(s,v) < \text{dist}_{(G,c)}(s,t)\right\}$.

   Bemerkung: Wenn man eine große Anzahl von Kürzeste-Wege-Berechnungen im selben Graphen aber mit unterschiedlichen Start- und Endknoten durchführen muss, kann es sich lohnen, vorher Abstände zu einer gewissen Menge L von Knoten zu berechnen, die als Orientierungspunkte dienen. Unter Ausnutzung der in diesem Satz bewiesenen Eigenschaften kann man damit die Aufrufe des Dijkstra-Algorithmus in der Praxis beschleunigen. (2+1+2 Punkte)

3. Sei G ein gerichteter Graph mit konservativen Kantengewichten $c : E(G) \to \mathbb{R}$. Seien $s, t \in V(G)$, wobei t in G von s aus erreichbar sei. Man zeige: Die minimale Länge eines s-t-Weges in G ist gleich dem maximalen Wert von $\pi(t) - \pi(s)$, wobei π ein zulässiges Potential von (G, c) sei. (2 Punkte)

4. Implementieren Sie einen Algorithmus (basierend auf dem Moore-Bellman-Ford-Algorithmus), der zu einem gegebenen gerichteten Graphen G und $c : E(G) \to \mathbb{R}$ entweder ein zulässiges Potential oder einen negativen Kreis berechnet. Ihr Programm soll Laufzeit $O(nm)$ haben. (10 Punkte)

Abgabe: Dienstag, den 12.11.2013, **vor** der Vorlesung.

b.w.

Hinweise zur Programmierübung:

Zum Einlesen und Speichern der Graphen ist die Klasse `Weighted_Graph` aus der Vorlesung "Algorithmische Mathematik I" aus dem Wintersemester 2012/2013 zu verwenden.

Einlesen der Daten: Dem Programm muss beim Aufruf der Name einer Datei übergeben werden. Ein Aufruf hat also die Form

```
<programmname> <dateiname>
```

Eine gültige Datei, die eine Instanz beschreibt, hat das folgende Format:

```
Knotenanzahl
Knoten0a Knoten0b Gewicht0
Knoten1a Knoten1b Gewicht1
...
```

Die Einträge der Datei sind ausschließlich ganze Zahlen. Sie können voraussetzen, dass die Summe der Absolutbeträge aller Zahlen in der Eingabe kleiner als 2^{31} ist. In der ersten Zeile steht eine einzelne natürliche Zahl (größer als 0), welche die Anzahl der Knoten angibt. Wir nehmen an, dass, wenn wir n Knoten haben, die Knoten von 0 bis $n-1$ durchnumeriert sind. Jede weitere Zeile spezifiziert genau eine Kante. Die ersten beiden Einträge einer Zeile sind zwei verschiedene nichtnegative ganze Zahlen, welche die Nummern der Endknoten der Kante sind. Der dritte Eintrag in der Zeile ist eine ganze Zahl, die das Gewicht der Kante bezeichnet. Es können parallele Kanten vorkommen, und der Graph muss nicht zusammenhängend sein.

Ausgabeformat: Die erste Zeile der Ausgabe muss genau eine Zahl enthalten, nämlich 0, wenn ein negativer Kreis gefunden worden ist, und 1 sonst.

Wenn ein negativer Kreis gefunden wurde, soll der Rest der Ausgabe einen solchen Kreis kodieren. In jeder Zeile soll dabei der Index eines Knoten eingetragen werden, so dass aufeinanderfolgende Knoten durch eine Kante des Kreises verbunden sind.

Wenn kein negativer Kreis gefunden wurde, sollen die weitere Zeilen der Ausgabe $\pi(0), \ldots, \pi(n-1)$ enthalten, so dass π ein zulässiges Potential ist.

Beispiel 1: Eine Eingabedatei für einen Graphen mit 4 Knoten und 5 Kanten kann so aussehen:

```
4
0 1 2
1 3 4
0 3 7
2 0 -1
2 3 2
```

Die Ausgabe der Programms kann dann so aussehen:

```
1
-1
0
0
0
```

Beispiel 2: Eine Eingabedatei für einen Graphen mit 4 Knoten und 4 Kanten kann so aussehen:

```
4
0 3 -3
1 0 -1
3 1  2
1 2  1
```

Die Ausgabe der Programms kann dann so aussehen:

```
0
3
1
0
```

Das Programm muss in C++ geschrieben sein. Es muss korrekt arbeiten und ohne Fehlermeldung kompiliert werden können. Der Code muss auf einem gängigen Linuxsystem funktionieren. Algorithmen aus externen Bibliotheken dürfen nicht verwendet werden.

Abgabe: Der Quelltext der Programms muss bis 12. November, 16:15 Uhr, per E-Mail beim jeweiligen Tutor eingegangen sein. Außerdem ist bis zu diesem Zeitpunkt ein Ausdruck des Quelltextes zusammen mit den Theorieaufgaben abzugeben.

Testinstanzen befinden sich ab dem 6.11.2013 auf der Seite

`http://www.or.uni-bonn.de/lectures/ws13/edm_13_uebung.html`