Wintersemester 2014/15
Prof. Dr. J. Vygen
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Einführung in die Diskrete Mathematik
4. Übung

1. Sei G ein ungerichteter Graph mit Kantengewichten $c : E(G) \to \mathbb{R}$.
   Wie lassen sich die folgenden Probleme möglichst effizient lösen?

   (a) Sei $v \in V(G)$ ein Knoten. Gesucht ist ein aufspannender Baum, in dem v kein Blatt ist und der unter allen aufspannenden Bäumen, in denen v kein Blatt ist, minimales Gewicht hat.

   (b) Man bestimme die Menge aller Kanten $e \in E(G)$, für die es einen aufspannenden Baum T_e mit minimalem Gewicht gibt, so dass e in T_e enthalten ist.

   (c) Man bestimme einen aufspannenden zusammenhängenden Teilgraphen von G mit minimalem Gewicht.

   (d) Man bestimme einen aufspannenden Baum T in G, dessen maximales Kantengewicht minimal ist. (2+2+2+2 Punkte)

2. Berechnen Sie in dem Graphen, der in Abbildung 1 dargestellt ist, mit Hilfe des Algorithmus von Edmonds ein gewichtsmaximales Branching. Geben Sie auch die Graphen an (mit den zugehörigen Kantengewichten), die während des Algorithmus durch Kontraktion entstehen. (2 Punkte)

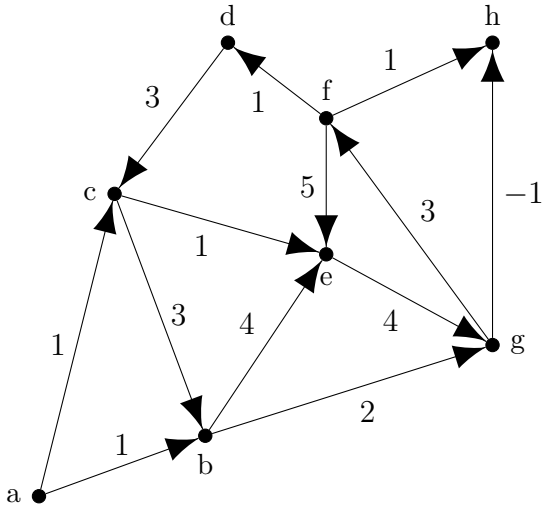

Abbildung 1: Instanz zur Berechnung eines maximal gewichteten Branchings.

3. Betrachten Sie die folgende Variante des Moore-Bellman-Ford-Algorithmus: Nummeriere die Knoten des gegebenen Graphen G in einer beliebigen Reihenfolge, es sei also $V(G) = \{v_1, \ldots, v_n\}$. Betrachte nun in jeder Iteration die Kanten in folgender Reihenfolge: Durchlaufe die Knoten von v_1 nach v_n und betrachte für jeden dabei besuchten Knoten v_i alle Kanten $(v_i, v_j) \in E(G)$ mit $i < j$, gegebenenfalls um $l(v_j)$ neu zu setzen. Durchlaufe anschließend alle Knoten von v_n nach v_1 und betrachte für jeden dabei besuchten Knoten v_i alle Kanten $(v_i, v_j) \in E(G)$ mit $j < i$, um gegebenenfalls $l(v_j)$ neu zu setzen. Zeigen Sie, dass, wenn man in jeder Iteration alle Kanten in dieser Reihenfolge betrachtet, $\lceil \frac{n}{2} \rceil + 1$ Iterationen ausreichend sind. (4 Punkte)

Abgabe der Theorieaufgaben: Donnerstag, den 6.11.2014, vor der Vorlesung.

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. Implementieren Sie Ihr Programm so, dass in jeder Iteration nur noch die Kanten (v, w) betrachtet werden, für die $l(v)$ in der vorigen Iteration verändert wurde. (12 Punkte)

Abgabe der Programmierübung: Donnerstag, den 13.11.2014, vor der Vorlesung.

Hinweise zur Programmierübung:

Das Programm muss in C oder C++ geschrieben sein. Es wird empfohlen, C++ zu verwenden. In diesem Fall kann man zum Einlesen und Speichern der Graphen die Klasse `Weighted_Graph` aus der Vorlesung "Algorithmische Mathematik I" aus dem Wintersemester 2012/2013 verwenden. Alle Datenstrukturen, die in dieser Vorlesung vorgestellt wurden, finden Sie hier:

`http://www.or.uni-bonn.de/~vygen/lectures/alma1ws12.html`

Sie können alle diese Programme und Datenstrukturen 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 positive ganze Zahl, welche die Anzahl der Knoten angibt. Wir nehmen an, dass, wenn wir n Knoten haben, diese von 0 bis $n-1$ durchnummeriert sind. Jede weitere Zeile spezifiziert genau eine Kante. Die ersten beiden Einträge einer Zeile sind zwei verschiedene nichtnegative

ganze Zahlen, welche die Nummern von Anfangs- und 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 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 13. 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 auf der Seite

`http://www.or.uni-bonn.de/lectures/ws14/edm_14_uebung.html`