Wintersemester 2010/11
Prof. Dr. J. Vygen
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Einführung in die Diskrete Mathematik

9. Übung

1. Man betrachte eine Verallgemeinerung des Minimum-Cost-Flow-Problems, bei der unendliche Kapazitäten erlaubt sind (d.h. $u(e) = \infty$ für manche Kanten e). Eine Instanz (G, u, b, c) heißt *unbeschränkt*, wenn es für jedes $\gamma \in \mathbb{R}$ einen b-Fluss f in (G, u) gibt mit $c(f) < \gamma$.

   (a) Man zeige, dass eine Instanz genau dann unbeschränkt ist, wenn es einen b-Fluss in (G, u) gibt und ein negativer Kreis existiert, dessen Kanten alle unendliche Kapazität haben.

   (b) Man zeige, wie man in $O(n^3+m)$-Zeit entscheiden kann, ob eine Instanz unbeschränkt ist.

   (c) Man zeige, dass in einer nicht unbeschränkten Instanz jede unendliche Kapazität auf äquivalente Weise durch eine endliche Kapazität ersetzt werden kann. (4 Punkte)

2. Sei (G, u, c, b) eine Instanz des Minimum-Cost-Flow-Problems, für das eine zulässige Lösung existiert. Zeigen Sie, dass es dann eine kostenminimale Lösung f gibt, für die eine Kantenmenge $F \subseteq E(G)$ existiert, so dass der $(V(G), F)$ zugrundeliegende ungerichtete Graph kreisfrei ist und auf allen Kanten $e \in E(G) \setminus F$ gilt: $f(e) \in \{0, u(e)\}$. (4 Punkte)

3. Implementieren Sie ein Verfahren, das in Zeit $O(n^3)$ ein kostenminimales perfektes Matching in einem bipartiten Graphen G mit Kantengewichten berechnet, wenn ein solches existiert. Falls kein perfektes Matching existiert, soll das Programm eine Knotenmenge X ausgeben, die die Hall-Bedingung verletzt (d.h. X soll ganz in einer Seite der Bipartition liegen, und es soll $|\Gamma(X)| < |X|$ gelten). Es empfiehlt sich, zur Lösung des Problems den Sukzessive-kürzeste-Wege-Algorithmus zu verwenden. Ein genauere Beschreibung der Aufgabe finden Sie umseitig. (16 Punkte)

Abgabe:
Aufgaben 1 und 2: Dienstag, den 14.12.2010, **vor** der Vorlesung.
Aufgabe 3: Dienstag, 11.1.2011, **vor** der Vorlesung (s.u.).

Hinweise zur Programmierübung:

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

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

Das Programm muss in C oder 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.
Die Einträge der Datei sind ausschließlich nichtnegative 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 Zahl n, welche die Anzahl der Knoten angibt. Diese Zahl ist stets gerade, und die beiden Seiten der Bipartition haben jeweils $\frac{n}{2}$ Elemente, wobei die eine Seite der Bipartition aus den Knoten $0, \ldots, \frac{n}{2} - 1$ besteht und die andere aus den Knoten $\frac{n}{2}, \ldots, n-1$ (die Knoten seien also wieder von 0 bis $n-1$ durchnumeriert). Die nächste Zeile enthält genau eine Zahl, und diese gibt die Zahl der Kanten an. Jede weitere Zeile besteht aus drei Einträgen und kodiert genau eine Kante. Die ersten beiden Einträge sind dabei die Nummern der Endknoten (wobei die kleinere Nummer zuerst kommt) und der dritte Eintrag in der Zeile bezeichnet die Kosten der Kante. Parallele Kanten können in dieser Aufgabe nicht vorkommen.
Der Index einer jeden Kante ist durch ihre Zeilennummer in der Eingabedatei gegeben: Zeile i kodiert die Kante mit Index $i-3$ (für $i = 3, \ldots, m+2$, wobei m die Zahl der Kanten sei).
Ausgabeformat: Das Programm muss in der ersten Zeile der Ausgabe eine 1 ausgeben, wenn es ein perfektes Matching gefunden hat, andernfalls muss es eine 0 ausgeben.
Wenn die erste Zeile eine 1 enthält, muss die nächste Zeile das Gewicht des berechneten Matchings enthalten. Jede weitere Zeile muss dann den Index von genau einer Matching-Kante enthalten (es muss in diesem Fall also $n+2$ Zeilen geben).
Wenn die erste Zeile eine 0 enthält, muss anschließend eine Menge X von Knoten angegeben werden, die die Hall-Bedingung verletzt. Dazu soll jede folgende Zeile den Index eines Knotens aus X enthalten.
Beispiel 1: Eine Eingabedatei für einen Graphen mit acht Knoten und sieben Kanten kann so aussehen:

```
8
7
3 7 1
2 7 0
2 6 8
1 5 6
0 6 5
1 4 1
0 4 2
```

Da der Graph ein perfektes Matching enthält, sieht eine korrekte Lösung so aus (die Ausgabe ist hier bis auf Sortierung der Matching-Kanten eindeutig):

```
1
17
0
2
3
6
```

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

```
6
4
0 3 9
1 3 1
2 4 5
2 5 2
```

Da der Graph kein perfektes Matching enthält, sieht eine korrekte Lösung so aus:

```
0
0
1
```

Weitere Testinstanzen befinden sich auf der Seite

`http://www.or.uni-bonn.de/lectures/ws10/edm_uebung_ws10.html`

Abgabe: Der Quelltext der Programms muss bis **11. Januar 2011, 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.