Wintersemester 2008/09
Prof. Dr. J. Vygen
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Einführung in die Diskrete Mathematik
3. Übung

1. Sei n eine natürliche Zahl. Zeigen Sie, dass es genau $(n+1)^{n-1}$ Branchings auf der Knotenmenge $\{1,\dots,n\}$ gibt. (4 Punkte)

2. Sei G ein ungerichteter Graph, $v \in V(G)$, und $c : E(G) \to \mathbb{R}$. Sei S die Menge der

   (a) zusammenhängenden aufspannenden Teilgraphen von G.

   (b) aufspannenden Bäume in G, in denen v ein Blatt ist.

   (c) aufspannenden Bäume in G, in denen v kein Blatt ist.

   Zeigen Sie, wie man in polynomieller Zeit ein Element von S mit minimalem Gesamtgewicht berechnen kann. (4 Punkte)

3. Das Voronoi-Diagramm einer endlichen Menge $V \subset \mathbb{R}^2$ besteht aus den Gebieten

$$P_v := \left\{ x \in \mathbb{R}^2 : ||x-v||_2 = \min_{w \in V} ||x-w||_2 \right\}$$

   für alle $v \in V$. Die Delaunay-Triangulierung von V ist der Graph

$$(V, \{\{v,w\} \subseteq V : v \neq w, |P_v \cap P_w| > 1\}).$$

   Ein minimaler aufspannender Baum für V ist ein Baum T mit $V(T) = V$ und minimaler Länge $\sum_{\{v,w\} \in E(T)} ||v-w||_2$. Man beweise, dass jeder minimale aufspannende Baum für V ein Teilgraph der Delaunay-Triangulation ist. (4 Punkte)

4. Sei T ein Baum mit n Knoten. Ein Knoten $v \in V(T)$ heißt Zentralknoten, wenn die Zusammenhangskomponenten, die nach Herausnahme von v entstehen, jeweils höchstens $\frac{n}{2}$ Knoten enthalten. Zeigen Sie, dass ein Baum stets einen Zentralknoten enthält. Geben Sie ausserdem ein möglichst effizientes Verfahren an, um einen solchen Knoten zu finden (mit Laufzeitanalyse). (4 Punkte)

5. Implementieren Sie Kruskals Algorithmus zur Berechnung eines minimalen aufspannenden Baumes in einem zusammenhängenden Graphen mit einer Laufzeit von $O(m \log n)$. Das Programm muss insbesondere die in der Vorlesung beschriebenen Datenstrukturen zur Verwaltung der Zusammenhangskomponenten enthalten.

   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 einen Graphen beschreibt, hat das folgende Format:

   ```
   Knotenanzahl
   Kantenzahl
   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 den ersten beiden Zeilen steht jeweils eine einzelne natürliche Zahl (größer als 0), welche in der ersten Zeile die Anzahl der Knoten und in der zweiten die Anzahl der Kanten angibt. 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. Dabei nehmen wir an, dass, wenn wir n Knoten haben, die Knoten von 0 bis $n-1$ durchnumeriert sind. Der dritte Eintrag in der Zeile ist eine ganze Zahl, die das Gewicht der Kante bezeichnet. 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). Parallele Kanten können vorkommen.

   Ausgabeformat: Das Programm muss in der ersten Zeile der Ausgabe das Gewicht des berechneten Baums ausgeben. Die weiteren Zeilen müssen jeweils genau einen Index einer Kante des Baumes enthalten. Jeder Index einer Baumkante muss dabei in genau einer Zeile vorkommen, und die Indizes müssen aufsteigend sortiert sein.

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

```
4
6
0 3 15
1 3 25
1 2 -12
1 2 21
2 3 18
0 1 10
```

Die Ausgabe der Programms muss dann so aussehen:

```
13
0
2
5
```

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. Für die Sortierung dürfen Sie `qsort` oder `std::sort` verwenden. Die Laufzeit dieser Sortierroutinen soll hier nicht betrachtet werden. Andere Algorithmen aus externen Bibliotheken dürfen nicht verwendet werden.

Abgabe: Der Quelltext der Programms muss bis 11. 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.

Weitere Hinweise und ein Beispielprogramm, mit dem sich Graphen einlesen lassen, befinden sich auf der Seite

`http://www.or.uni-bonn.de/lectures/ws08/edm_08_uebung.html`

(16 Punkte)

6. Erweitern Sie Ihr Programm aus Aufgabe 5 so, dass es zusätzlich in dem berechneten Baum einen Zentralknoten (siehe Aufgabe 4) bestimmt. Dazu soll in der Ausgabe des Programms eine Zeile angehängt werden, die den Index eines Zentralknotens enthält. Für das Programm gelten ansonsten die Spezifikationen aus der vorigen Aufgabe. (8 Zusatzpunkte)

Abgabe: Dienstag, den 11.11.2008, **vor** der Vorlesung.