Diskrete Mathematik I

2. Übung

1. a) Sei G ein ungerichteter Graph, C ein Kreis und D ein Schnitt in G. Zeigen Sie, dass $|E(C) \cap D|$ geradzahlig ist. (1 Punkt)

   b) Zeigen Sie, dass jeder Schnitt in einem ungerichteten Graphen die disjunkte Vereinigung $\subseteq$-minimaler Schnitte ist. (2 Punkte)

   c) Zeigen Sie, dass es in jedem ungerichteten Graphen einen Schnitt gibt, der mindestens die Hälfte aller Kanten enthält. (2 Punkte)

2. Geben Sie einen Algorithmus mit linearer Laufzeit für das folgende Problem an: Gegeben die Adjazenzliste eines (gerichteten oder ungerichteten) Graphen G, ist die Adjazenzliste des (bis auf Isomorphie eindeutigen) maximalen einfachen Subgraphen von G zu bestimmen. Parallele Kanten können an beliebiger Position stehen und müssen nicht aufeinander folgen. Beweisen Sie, dass die Laufzeit Ihres Verfahrens tatsächlich linear in der Eingabegröße ist. (3 Punkte)

3. Sei G ein ungerichteter Graph, $s \in V(G)$ und T ein DFS-Baum, der als Ergebnis des Algorithmus' DFS für die Eingabe (G, s) erzeugt wird. Den Knoten s bezeichnen wir als die *Wurzel* von T. x ist ein Vorgänger von y in T, wenn x auf dem (eindeutigen) s-y-Pfad in T liegt. x ist ein *direkter Vorgänger* von y, wenn die Kante $\{x, y\}$ auf dem s-y-Pfad in T liegt. y ist ein (direkter) Nachfolger von x, wenn x ein (direkter) Vorgänger von y ist. Man beachte, dass nach dieser Definition jeder Knoten ein Nachfolger sowie ein Vorgänger von sich selbst ist. Jeder Knoten außer s hat genau einen direkten Vorgänger.

   Zeigen Sie:

   a) Für jede Kante $\{v, w\} \in E(G)$ ist v ein Vorgänger oder ein Nachfolger von w in T. (1 Punkt)

   b) Ein Knoten v ist ein Artikulationsknoten von G genau dann, wenn entweder

   (i) $v = s$ und $|\delta_T(v)| > 1$ oder

   (ii) $v \neq s$ und es einen direkten Nachfolger w von v gibt, so dass keine Kante in G einen echten Vorgänger von v (d.h. einen von v verschiedenen Vorgänger) mit einem Nachfolger von w verbindet. (2 Punkte)

4. Verwenden Sie Aufgabe 3, um einen Algorithmus mit linearer Laufzeit zu finden, der die *Blöcke* eines ungerichteten Graphen findet. Ein *Block* eines ungerichteten Grapen G ist ein maximaler zusammenhängender und keinen Artikulationsknoten enthaltender Teilgraph von G.

   Tipp: Es ist von Nutzen, während der Abarbeitung des DFS-Algorithmus' rekursiv Zahlen

$$\alpha(x) := \min\{f(w) : w = x \text{ oder } \{w,y\} \in E(G) \setminus T \text{ für einen Nachfolger } y \text{ von } x\}$$

zu berechnen. Hierbei sei (R, T) der DFS-Baum (mit Wurzel s), und die f-Werte sollen die Reihenfolge angeben, in der die Knoten zu R hinzugefügt werden (siehe GRAPH-SCANNING-ALGORITHMUS aus der Vorlesung). Wenn $\alpha(x) \geq f(w)$ für einen Knoten $x \in R \setminus \{s\}$, wobei w den direkten Vorgänger von x bezeichne, dann muss w entweder die Wurzel oder ein Artikulationsknoten sein. (5 Punkte)

5. **Programmieraufgabe:** Implementieren Sie den in der Vorlesung vorgestellten Algorithmus zum Auffinden der starken Zusammenhangskomponenten eines gerichteten Graphen.

   Ihr Programm soll den Eingabegraphen G aus einer ASCII-Textdatei mit folgendem Format einlesen:

   – Die erste Zeile enthält die Anzahl $|V(G)|$ der Knoten.

   – Die übrigen Zeilen definieren die Kantenmenge $E(G)$ durch Angabe eines durch Leerzeichen getrennten Knotenpaares auf jeder Zeile. Die Knoten werden durch Zahlen in $\{0, \ldots, |V(G)| - 1\}$ bezeichnet, und die in einer Zeile spezifizierte Kante soll vom ersten zum zweiten Knoten orientiert sein.

   Beispiel:

```
4
0 2
2 3
0 2
3 0
```

   Der so angegebene Graph enthält einen (gerichteten) Kreis der Länge 3, zwei parallele Kanten und einen isolierten Knoten (mit der Nummer 1).

   Ihr Programm soll die Anzahl der starken Zusammenhangskomponenten ausgeben sowie für jede solche Komponente die in ihr enthaltenen Knoten auflisten.

   Testinstanzen werden in Kürze auf der Vorlesungsseite im Internet zur Verfügung gestellt.

   Sie dürfen in ANSI-C, ANSI-C++ oder Java programmieren. Um die volle Punktzahl zu erlangen, ist es neben den beiden wichtigsten Kriterien – einer korrekten Implementierung und dem tatsächlichen Erreichen linearer Laufzeit – erforderlich, dass ihr Programmcode ohne Modifikation mit GCC 3.x (GNU-C-Compiler bzw. GNU-C++-Compiler Version 3.x) bzw. JDK 1.4.x übersetzbar ist.

   Schicken Sie Ihren Programmcode bitte bis zum angegebenen Abgabetermin an `mueller@or.uni-bonn.de`. **(16 Punkte)**

Abgabe bis spätestens Di, den 04.11.03 vor der Vorlesung.