Wintersemester 2018/19
Professor Dr. S. Hougardy
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I
10. Übung

1. Es sei G ein gerichteter Graph, und G' derjenige einfache gerichtete Graph, dessen Knoten die starken Zusammenhangskomponenten von G sind und der genau dann eine Kante (X, Y) enthält, wenn $\delta^+_G(V(X)) \cap \delta^-_G(V(Y)) \neq \emptyset$. Zeigen Sie, dass G' eine topologische Ordnung besitzt. (3 Punkte)

2. (a) Zu einem gegebenen ungerichteten Graphen G soll eine inklusionsminimale Teilmenge $F \subseteq E(G)$ berechnet werden, so dass $(V(G), E(G) \setminus F)$ bipartit ist. Zeigen Sie, dass ein solches F in linearer Zeit berechnet werden kann.

   (b) Es sei G ein bipartiter Graph mit n Knoten und k Zusammenhangskomponenten. Wie viele Mengen $X \subseteq V(G)$ gibt es, so dass $\delta(X) = E(G)$ ist? (3+3 Punkte)

3. Es sei S eine endliche Menge mit einer partiellen Ordnung „$\preceq$“. Beweisen Sie, dass dann folgende Aussagen äquivalent sind:

   (a) $\preceq$ ist durch Schlüssel induziert;

   (b) $(a \not\preceq b \wedge b \not\preceq c \wedge a \neq c) \Rightarrow a \not\preceq c$ für alle $a, b, c \in S$;

   (c) $\{(x, y) \in S \times S \mid x = y \vee (x \not\preceq y \wedge y \not\preceq x)\}$ ist eine Äquivalenzrelation. (6 Punkte)

4. Implementieren Sie einen Algorithmus, der einen DFS-Baum in einem Graphen berechnet. Die Schnittstelle des Algorithmus soll genauso aussehen wie bei `bfs`, d.h. sie soll wie folgt aussehen:

   ```
   Graph dfs(const Graph & graph, Graph::NodeId start_node)
   ```

   Durch Implementieren einer rekursiven Funktion soll dabei vermieden werden, dass explizit ein Stack als Datenstruktur benutzt werden muss. Orientieren Sie sich dabei an der folgenden Beschreibung in Pseudocode, die ohne explizite Verwendung eines Stacks Q auskommt:

   ```
   DFS-vist(v)
   1   R := R ∪ {v}
   2   for w mit (v,w) ∈ E(G) bzw. {v,w} ∈ E(G)
   3       if w ∉ R then DFS-visit(w)
   ```

   Nach Setzen von $R := \emptyset$ berechnet sodann ein Aufruf von `DFS-visit(s)` einen DFS-Baum mit Wurzel s. Zum Testen Ihres Algorithmus schreiben Sie analog zu dem Programm `bfs.cpp` eine Funktion `main`, die einen Graphen aus einer als Parameter übergebenen Datei als *ungerichteten* Graphen einliest und einen im Knoten mit Index 0 gewurzelten DFS-Baum ausgibt. Der DFS-Baum soll natürlich alle vom Knoten mit Index 0 aus erreichbaren Knoten enthalten. (5 Punkte)

Abgabe: Montag, den 17.12.2018, bis 10:12 Uhr.

Abgabe der Programmierübungen:
Per E-Mail an `alma_prog_gr_XX@dm.uni-bonn.de`, wobei `XX` durch die Nummer Ihrer Übungsgruppe zu ersetzen ist, also z.B. `alma_prog_gr_07@dm.uni-bonn.de`, wenn Sie in Gruppe 7 sind, oder `alma_prog_gr_12@dm.uni-bonn.de`, wenn Sie in Gruppe 12 sind. Wenn Sie Ihre Übungsgruppe nicht kennen, schreiben Sie an `alma_prog_gr_unbekannt@dm.uni-bonn.de`.

Öffnungszeiten des Help Desks:
Montags, 16 – 19 Uhr und freitags, 12 – 15 Uhr, jeweils in Raum N1.002, Endenicher Allee 60, Nebengebäude.
`www.mathematics.uni-bonn.de/files/bachelor/help-desk.pdf`

Zusätzlich gibt es ab sofort einen **Help Desk für Programmierfragen**, und zwar immer freitags, 8 – 10 Uhr und 12 – 16 Uhr, im PC-Pool in der Wegelerstraße 6, Raum E02 (Hochschulrechenzentrum).