Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Johannes Krugel, Jeremias Weihmann

Wintersemester 2010/11
Aufgabenblatt 1
19.10.2010

Praktikum Algorithmen-Entwurf

Letzter Abgabetermin: Dienstag, den 26.10.2010, 14:00 Uhr

Aufgabe 1 (Erste Schritte mit LEDA)

Machen Sie sich mit der Benutzung der LEDA-Bibliothek bei der Erstellung von C++-Programmen vertraut. Legen Sie ein Verzeichnis an, in dem Sie im Verlauf des Semesters Ihre Praktikumsaufgaben bearbeiten und Ihre Lösungen erstellen werden, und kopieren Sie das Makefile und die Quellen `dfs.cpp` und `control.h` von der Praktikumswebpage. Das Programm `dfs.cpp` sollte sich nun mittels `make dfs` übersetzen lassen.

Aufgabe 2 (Breitensuche)

Implementieren und animieren Sie Breitensuche in zusammenhängenden, ungerichteten Graphen unter Verwendung von LEDA. Verwenden Sie zur Darstellung des Graphen auf dem Bildschirm die Klasse `GraphWin`. Benutzen Sie das Programm `dfs.cpp` als Vorlage. Der Benutzer soll per Mausklick einen Startknoten s auswählen können, und Ihr Programm soll dann mittels einer Breitensuche den Graphen durchlaufen und dabei für alle anderen Knoten v die Länge eines kürzesten Pfades (bzgl. Anzahl der Kanten) von s zu v berechnen.

Es soll am Bildschirm gut mitverfolgt werden können, in welcher Reihenfolge das Programm die Knoten besucht, welche Knoten bereits besucht wurden, und welche Knoten gegenwärtig in der Queue gespeichert sind. Verwenden Sie zu diesem Zweck die Möglichkeiten, die `GraphWin` bzgl. der Darstellung von Knoten und Kanten anbietet.

Die Abstände der Knoten vom Startknoten s sollen als User-Labels auf dem Bildschirm dargestellt werden. Die Kanten, die im Breitensuch-Baum enthalten sind, sollen ebenfalls hervorgehoben werden.

Aufgabe 3 (Topologisches Sortieren)

Implementieren und animieren Sie einen Algorithmus, der einen beliebigen gerichteten Graphen $G = (V, E)$ als Eingabe bekommt und der versucht, jedem Knoten $v \in V$ eine eindeutige Nummer $f(v) \in \{1, 2, \ldots, |V|\}$ zuzuordnen, so dass für jede Kante $(u, v) \in E$ gilt: $f(u) < f(v)$. (Das heißt, der Anfangsknoten jeder Kante muss eine kleinere Nummer haben als der Endknoten der Kante.)
Eine solche Nummerierung existiert übrigens genau dann, wenn der Graph keinen Zyklus enthält. Falls der Eingabegraph doch einen Zyklus enthält, soll Ihr Programm dies erkennen und einen Zyklus am Bildschirm farbig markieren. Bemühen Sie sich um eine möglichst effiziente Implementierung!