Technische Universität München
Institut für Informatik
Prof. Dr. Angelika Steger

WS 2000/2001
Aufgabenblatt 1

Praktikum Algorithmen-Entwurf

(Abgabetermin: Montag, den 30.10.2000, $14.^{00}$ Uhr)

Aufgabe 1 LEDA (First Steps)

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.C und control.h von der Praktikumswebpage

`http://wwwmayr.in.tum.de/lehre/2001SS/optprak/data.html`

(siehe Hinweise zur Benutzung von LEDA) in dieses Verzeichnis. Das Programm dfs.C 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.C 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 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 Numerierung existiert übrigens genau dann, wenn der Graph keinen Zyklus enthält. Falls der Eingabegraph doch einen Zyklus enthält, soll Ihr Programm dies erkennen.
Bemühen Sie sich um eine möglichst effiziente Implementierung!
Als Eingabe für Ihr Programm sind in der Problemdatei mehre gerichtete Graphen gegeben. Das Programm soll für jedes Problem entweder "cycle" ausgeben, falls der Graph einen Kreis enthält oder die Knotennummern in der Reihenfolge ihrer topologischen Sortierung.

Eingabeformat:
#Probleme
#Knoten #Kanten
#Nr. Startknoten #Nr. Endknoten #Gewicht
...
#Knoten #Kanten
#Nr. Startknoten #Nr. Endknoten #Gewicht
...

Beispiel:
2
2 1
1 0 1
4 4
0 1 1
1 2 1
2 3 1
3 0 1

Ausgabeformat:
für jedes Problem:
"cycle" oder Folge der Knotennummern in der topologischen Sortierung

Ausgabe zu obigem Beispiel:
1 0
cycle

Hinweise

Die Abgabe der Lösungen muß jeweils bis Montag 14.00 Uhr per E-Mail an die Adresse `optprak@hpmayr1.informatik.tu-muenchen.de` erfolgen. Jede Zweiergruppe braucht nur eine Lösung abzugeben. Es sind nur die C++-Sourcen der

erstellten Programme abzugeben, also diesmal lediglich der Source-Code zu Aufgabe 2 und 3.
Aktuelle Informationen, Testdaten für Ihre Programme und PostScript-Dateien mit den Praktikumsunterlagen mit sie im WWW unter folgender URL verfügbar:

`http://wwwmayr.in.tum.de/lehre/2001SS/optprak/data.html`

Anforderungen an abgegebene Programme

Die abgegebenen Lösungen werden nach den folgenden Kriterien bewertet:

- Korrektheit der berechneten Ergebnisse (bei den Beispielgraphen und beliebigen weiteren Eingaben)
- Effizienz der Implementierung (Vermeidung ineffizienter Konstrukte, so daß bei Entfernung der Animationsroutinen die geforderte Worst-Case-Laufzeit eingehalten wird)
- Qualität der Animation (die Arbeitsweise des Algorithmus soll anschaulich visualisiert werden)
- Lesbarkeit des Quelltexts (ausreichend Kommentare, die das Verständnis erleichtern)

Außerdem legen wir großen Wert darauf, daß die Lösungen selbst erarbeitet und nicht Programme anderer Gruppen als "Vorlage" verwendet werden. Falls Probleme bei der Implementierung einer Lösung auftreten, sollte keinesfalls auf die evtl. schon fertigen Lösungen einer anderen Gruppe zurückgegriffen werden. (Diskussionen über Implementierungsmöglichkeiten sind natürlich erlaubt, lediglich die direkte Weitergabe von Source-Codes ist nicht gestattet.) Stattdessen können Verständnis- oder Implementierungsprobleme immer mit dem jeweiligen Betreuer besprochen werden.

Praktikumsbetreuer

- Ingo Rohloff, Raum SS2238, Tel. 289-28495, rohloff@in.tum.de
- Mark Scharbrodt, Raum S2238, Tel. 289-28494, scharbro@in.tum.de