Prof. Dr. Volker Linnemann
Dr. Sven Groppe
Universität zu Lübeck
Institut für Informationssysteme

Lübeck, den 29.5.2008

Vorlesung „Algorithmen und Datenstrukturen“

Sommersemester 2008

7. Übungsblatt

1. Graphen (12 Punkte)

Gegeben sei die folgende Adjazenzmatrix:

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	1	0	0	1	1
3	0	0	0	0	0	0	1
4	1	0	0	0	0	1	0
5	0	1	0	1	0	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0

a) Zeichnen Sie den durch die Matrix definierten Graph G auf. (1 Punkt)
b) Geben Sie einen einfachen Weg in G vom Knoten 4 zum Knoten 7 an. (2 Punkte)
c) Ein spannender Baum eines Graphen ist ein Teilgraph des Graphen, der ein Baum ist und alle dessen Knoten enthält. Zeichnen Sie einen spannenden Baum B_G des Graphen G. Markieren Sie den Wurzelknoten. (2 Punkte)
d) Geben Sie eine topologische Sortierung von B_G an. (2 Punkte)
e) Gibt es eine topologische Sortierung von G? Begründen Sie Ihre Antwort. (1 Punkt)
f) Beschreiben Sie G durch eine Adjazenzliste. (2 Punkte)
g) Zeichnen Sie die reflexive, transitive Hülle G^* von G. (2 Punkte)

2. Programmieraufgabe zu Graphen (21 Punkte)

Ein Flugplan soll durch einen gerichteten und gewichteten Graphen repräsentiert werden und besteht aus Flughäfen (Knoten), die durch Routen (Kanten) miteinander verbunden sein können. Flughäfen und Routen besitzen jeweils einen Namen. Die Entfernung einer Route soll als Gewicht der entsprechenden Kante interpretiert werden. Entfernungen sind ganzzahlige Kilometerangaben.

Routen können in beide Richtungen (ungerichtet) oder nur in eine Richtung (gerichtet) nutzbar sein. Reale Flugrouten werden zwar i.A. in beide Richtungen angeboten, jedoch können gerichtete Routen so interpretiert werden, dass das Flugzeug in eine Richtung ausgebucht ist.

a) Implementieren Sie eine Klasse *Flughafen*, die einen Knoten des Graphen repräsentiert. Ein Flughafen hat einen Namen und eine Nummer. (1 Punkt)

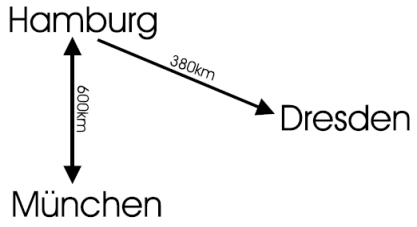
b) Implementieren Sie eine Klasse *Route*, die eine Kante des Graphen repräsentiert. Eine Route hat eine Nummer, einen Namen, einen Startort, einen Zielort, ein Gewicht (die Länge der Route) und eine Komponente, die angibt, ob die Route gerichtet oder ungerichtet ist. (2 Punkte)

c) Implementieren Sie eine Klasse *Flugplan*, die den Graphen repräsentiert. Die Klasse beinhaltet je eine Menge von Flughafen-Objekten und eine Menge von Routen-Objekten. Verwenden Sie die *LinkedList* aus der Java API, um eine Menge von Objekten zu realisieren. (2 Punkte)

d) Die Definition eines Flugplans soll aus einer Datei gelesen werden. Die Datei hat folgenden Aufbau:

n
m
1;Knotenname1
2;Knotenname2
..
n;Knotennamen
1;Kantenname1;StartNr1;ZielNr1;Richtung1;Gewicht1
2;Kantenname2;StartNr2;ZielNr2;Richtung2;Gewicht2
..
m;Kantennamem;StartNrm;ZielNrm;Richtungm;Gewichtm

wobei *n* die Anzahl der Flughäfen (Knoten) und *m* die Anzahl der Routen (Kanten) ist. Die Richtung wird durch die Schlüsselwörter *gerichtet* und *ungerichtet* definiert. Das Gewicht ist eine ganze Zahl. Die *StartNr* und *ZielNr* bezieht sich auf die vorher definierten Knotennummern. Die einzelnen Datenfelder sind immer durch Semikola (;) getrennt. Als Beispiel wird folgender Flugplan durch eine Datei beschrieben:



3
2
1;Hamburg
2;München
3;Dresden
1;LH912;1;3;gerichtet;380
2;LH72;1;2;ungerichtet;600

Implementieren Sie in der Klasse Flugplan eine Methode, die eine solche Datei einliest und die Knoten und Kantenmengen korrekt füllt. Um die Datenfelder aus einer Zeile zu extrahieren, nutzen Sie bitte den *java.utilStringTokenizer*. Eine Anleitung finden Sie in der Java API. Testen Sie Ihre Methode anhand des Flugplans *Flugplan.txt*, die Sie von der Übungsseite herunterladen können. (6 Punkte)

e) Zeichnen Sie den Graphen, der durch die Datei *Flugplan.txt* definiert ist. Als kleine Geographie-Übung und um den Korrekteuren die Arbeit leichter zu machen, sollen die Flughäfen geographisch halbwegs korrekt gezeichnet werden. *Bemerkung:* Diese Teilaufgabe braucht nicht implementiert zu werden! (1 Punkt)

f) Implementieren Sie eine Methode *gibNachfolger*, die alle direkten Nachfolger eines Startflughafens zurückgibt. Ein direkter Nachfolger ist ein Zielflughafen (Knoten), der über genau eine Flugroute (Kante) vom Startflughafen erreichbar ist. Was sind die direkten Nachfolger des Flughafens mit der Nummer 37 aus *Flugplan_gross.txt*? (3 Punkte)

g) Implementieren Sie eine Methode *gibVorgaenger*, die alle direkten Vorgänger eines Zielflughafens zurückgibt. Ein direkter Vorgänger ist analog zu a) ein Startflughafen (Knoten), von dem aus der Zielflughafen über genau eine Flugroute (Kante) erreichbar ist. Was sind die direkten Vorgänger des Flughafens mit der Nummer 37 aus *Flugplan_gross.txt*? (3 Punkte)

h) Implementieren Sie eine Methode *gibErreichbareFlughaefen*, die alle erreichbaren Flughäfen eines Startflughafens zurückgibt (entspricht der reflexiven, transitiven Hülle eines Knotens). Welche Flughäfen sind vom Flughafen mit der Nummer 37 aus *Flugplan_gross.txt* erreichbar? (3 Punkte)

Bemerkungen:

- Jede Seite soll oben rechts den Namen der Abgebenden und die Übungsgruppennummer (wichtig!) enthalten.
- Lösungen für die Übungsaufgaben sind (in der Regel) zu zweit abzugeben.
- Kommentieren Sie Ihre Lösungen! Besteht eine Lösung aus mehreren Zetteln, so sind diese zusammen zu heften. Bitte keine Hüllen, Mappen, o.ä..
- Bitte schicken Sie *Programmieraufgaben zusätzlich zur Abgabe auf Papier in elektronischer Form per Email* an Ihren jeweiligen Tutor.
- Kommentieren Sie ihren Quelltext bei Programmieraufgaben. Dabei sollen keine Trivialitäten kommentiert werden, also bitte keine Kommentare wie

  ```
  x=5; // Wir weisen nun der Variablen x den Wert 5 zu
  ```

  sondern sinnvolle Kommentare, die Ideen des Quelltextabschnittes beschreiben oder auf Unteraufgaben (z. B. a), b), …) hinweisen.
- **Hinreichende Bedingung für die Zulassung zur Klausur:** 50% der erreichbaren Punkte bei jedem Übungszettel (bis auf zwei) und einmaliges Vorrechnen in der Übung
- **Zertifikatskriterium:** Das Bestehen der Klausur am Ende des Semesters

Abgabetermin: Donnerstag, 5.6.2008, nach der Vorlesung