Übung 05: Graphen

Abgabetermin: 20.05.2008 12:00

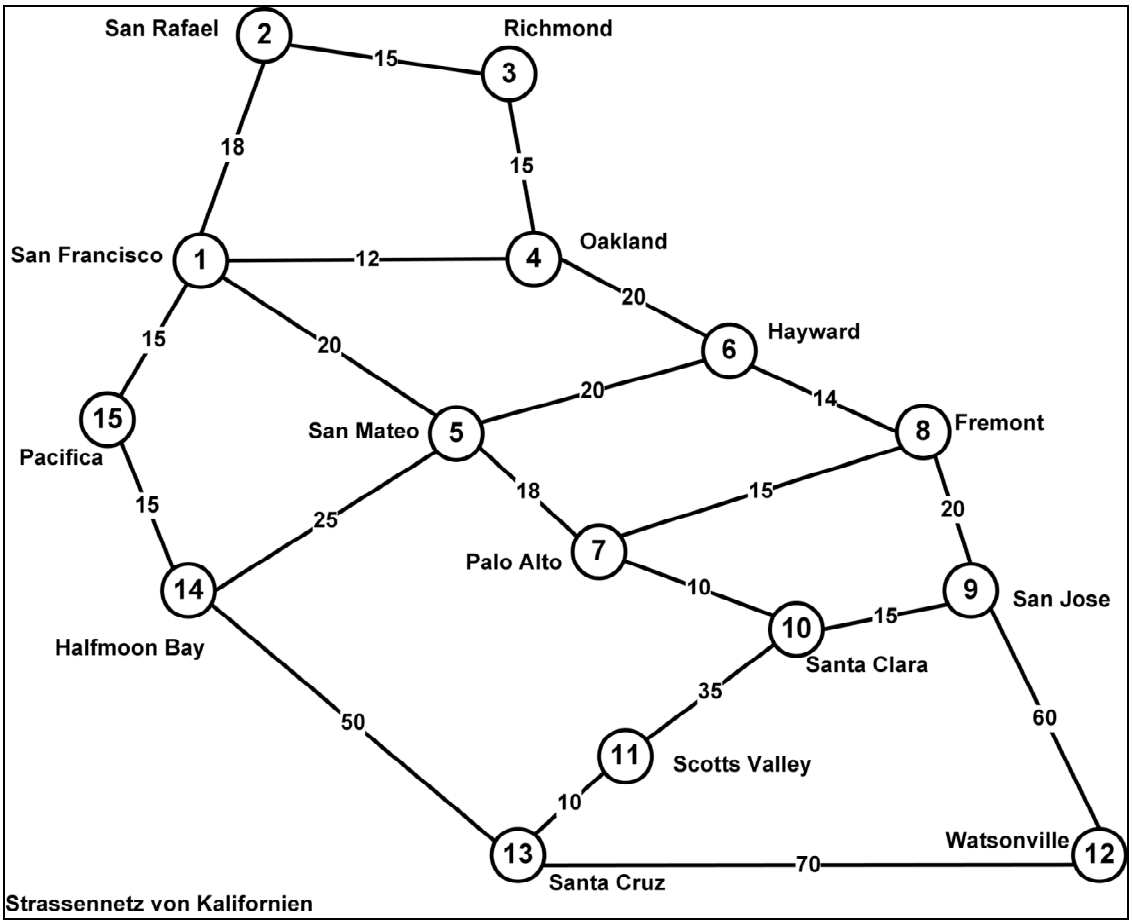
Name: _____ **Matrikelnummer**: _____

Gruppe: ❒ G1 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Tutor	Pkte
Aufgabe 05.1	24	❑	Java-Programm Testfälle und Ergebnisse	Java-Programm		

Aufgabe 05.1: Strassennetz, Kleinster spannender Baum, Kürzester Pfad

Strassennetze sind klassische Beispiele für Graphen. In dieser Übung soll das Strassennetz zwischen den Städten in der Bucht von San Francisco als ungerichteter gewichteter Graph mit Hilfe einer Adjazenzliste abgebildet werden. Kanten repräsentieren direkte Strassenverbindungen zwischen den Städten. Die Kanten sind mit den durchschnittlichen Reisezeiten zwischen den Städten gewichtet. Jede Stadt, also jeder Knoten, hat einen Namen und eine Nummer.



Strassennetz von Kalifornien

1) Implementieren Sie in der Klasse `RoadNetworkImpl` den gewichteten Graphen des Strassennetzes als Adjazenzliste. Verwenden Sie für die Knoten die Klasse `City` (Vorgabedatei) und ergänzen diese um benötigte Felder und Methoden. Führen Sie bei Bedarf weitere Klassen ein.

```
class RoadNetworkImpl
  implements RoadNetwork {
  public RoadNetworkImpl () {… }
  …
}
```

```
class City {
  int id;  // Nummer der Stadt
  String name;
  public String toString() {… }
  …
}
```

2) Lesen Sie Städte (`cities.txt`) und Strassenverbindungen (`roads.txt`) aus den Vorgabedateien ein. Die Vorgabedatei haben folgenden Aufbau:

cities.txt:
```
  CityId "CityName"
```
Beispiel:
```
  1 "San Francisco"
  2 "San Rafael"
  3 "Richmond"
  ...
```

roads.txt
```
  FromCityId ToCityId Time
```
Beispiel:
```
  1 2 18
  1 4 12
  1 5 20
  ...
```

3) Implementieren Sie in der Klasse `RoadNetworkImpl` die Methoden des Interface `RoadNetwork` (Vorgabedatei):

```
public interface RoadNetwork {
  City findCity(int id);
  void printDFS(City root);
  void printBFS(City root);
  void printMST(City root);
  void printShortestPath(City from, City to);
}
```

- `findCity` sucht im Graphen das zur gegebenen `CityId` passende `City`-Objekt.
- `printDFS` traversiert den Graphen ausgehend vom übergebenen Knoten mit Depth-First-Search und gibt die Knoten in der besuchten Reihenfolge auf der Konsole aus.

  ```
  01:San Francisco
  02:San Rafael
  03:Richmond
  ...
  ```

- `printBFS` traversiert den Graphen ausgehend vom übergebenen Knoten mit Breadth-First-Search und gibt die Knoten in der besuchten Reihenfolge auf der Konsole aus.

  ```
  01:San Francisco
  02:San Rafael
  04:Oakland
  ...
  ```

  Als Queue können Sie die Klasse `java.util.LinkedList` verwenden. Beispiel:

  ```
  Queue q = new LinkedList();
  q.offer(...); // einfügen
  City c = (City) q.poll(); // entnehmen
  ```

- `printMST` gibt alle Kanten des kleinsten spannenden Baums auf der Konsole aus. Implementieren Sie den benötigten Heap selbst, verwenden Sie dafür keine Klasse aus dem Java API.

  ```
  04:Oakland - 01:San Francisco
  15:Pacifica - 01:San Francisco
  03:Richmond - 04:Oakland
  ...
  ```

- `printShortestPath` ermittelt den kürzesten Pfad und die Gesamtreisedauer zwischen zwei Städten.

  ```
  Beispiel:
    network.printShortestPath(
      network.findCity(4), network.findCity(10))
  Ausgabe:
    Shortest Path from 04:Oakland to 10:Santa Clara
    10:Santa Clara-07:Palo Alto-08:Fremont-06:Hayward-04:Oakland=59 minutes
  ```

4) Definieren Sie Testfälle und testen Sie Ihr Programm.

Abzugeben ist:

- Das Java-Programm
- Testfälle und die Ergebnisse