Universität Karlsruhe
Fakultät für Informatik
ITI Wagner

Praktikum Algorithmen für planare Graphen
WS 06/07
Prof. Dr. D. Wagner, M. Baur, M. Holzer, S. Mecke

# Drittes Übungsblatt

**Ausgabe:** 17. November 2006
**Abgabe:** 24. November 2006

## Einführung von Einbettungen

In Dateien werden Graphen üblicherweise mit ihrer *geometrischen Einbettung* gespeichert, d. h. für die Knoten werden Koordinaten in der Ebene (im Raum etc.) angegeben. Für Algorithmen ist typischerweise die *kombinatorische Einbettung* praktischer. Eine solche Einbettung definiert für jeden Knoten die zirkulären Ordnung seiner inzidenten Kanten und damit auch seiner Nachbarn und seiner inzidenten Facetten.

| Für den Verlauf des Praktikums sei diese Ordnung stets im Gegenuhrzeigersinn. |
|---|

Die erste Aufgabe besteht darin, aus einer GraphML-Datei einen Graphen mit geometrischer Einbettung einzulesen und daraus eine Graphen mit kombinatorischer Einbettung zu erzeugen.

### **Problem 1:** GraphML-Datei einlesen

Im ersten Teil soll eine GraphML-Datei in einen JUNG-Graph eingelesen werden. Die Datei enthält die Knotenpositionen als GraphML-Attribute mit Namen `x` und `y`. [1] Diese sollen, zu einem `java.awt.geom.Point2D` zusammengefasst, als `UserDatum` mit Namen `position` an die Knoten anhängt werden.

### **Problem 2:** `PlanarGraph`-Objekt erzeugen

Zur Repräsentation eines Graphen in seiner kombinatorischen Einbettung stellen wir eine Klasse `UndirectedPlanarGraph` zur Verfügung, die alle nötigen Funktionen bereithält. Aufgabe ist es, aus dem oben eingelesenen Graphen eine `UndirectedPlanarGraph` zu erzeugen. Dazu müssen die Kanten in der richtigen Reihenfolge ihren inzidenten Knoten hinzugefügt werden.

### **Problem 3:** Weitere Möglichkeiten

Falls Euch langweilig wird, oder Ihr das Gefühl habt, noch nicht alles verstanden zu haben, könntet Ihr zum Beispiel folgende Dinge tun:

- Schreibt einige weitere Tests für Eure Klassen (sowieso immer eine gute Idee).

[1] Die Methode `de.uka.algo.io.JungGraphMLFileHandler.loadGraphFromFile()` sollte dabei wieder gute Dienste leisten. Sie speichert die GraphML-Attribute in `UserDatum`s ab. Die $x$-Koordinate eines Knotens kann zum Beispiel mit `Float.parseFloat(vertex.getUserDatum("x").toString())` ausgelesen werden.

- Implementiert eine Right-First-Tiefensuche für Planare Graphen (das brauchen sowieso einige Gruppen).
- Schreibt eine Methode, die zu einem Planaren Graphen seinen Dualgraphen als Planar-Graph berechnet (auch das werden einige Gruppen brauchen).
- Schreibt eine Methode, die zu einem Planaren Graphen eine Triangulierung berechnet (dito).