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

Zweites Übungsblatt

Ausgabe: 3. November 2006
Abgabe: 10. November

Problem 1: AlgoVis3D

Ihr habt alle schon die Projekte Graphlibs und JungDemo aus dem CVS ausgescheckt. Führt jetzt ein Update aus (*Rechtsklick→Team→Update* oder *Rechtsklick→Team→Synchronize with Repository, Rechtsklick→Update*). Es sollten drei neue Bibliotheken dazugekommen sein, darunter algovis3d.jar.

Legt in Eurem Projekt ein neues Package an (*Rechtsklick auf das Projekt→New→Class*), z.B. `praktikum.demo` in das ihr auch die Klasse vom letzten Übungsblatt kopiert (eclipse kümmert sich automatisch um die Korrektur der `package`-Klausel) und legt dort eine neue Klasse, z.B. `AlgoVis3DDemo` an (*Rechtsklick auf das Package→New→Class*). Diese soll nur folgende `main`-Methode enthalten:

```
public static void main(String[] args) {
  new AlgoVis3D();
}
```

Führt die Methode aus (*Rechtsklick auf die Klasse→Run as→Java Apllication*) und probiert den AlgoVisualizer aus.

Problem 2: Dijkstra

Implementiert den Algorithmus von Dijkstra (für gerichtete Graphen) und eine passende Visualisierung. Legt dazu eine Klasse `Dijkstra` in einem neuen Unterpaket `algo` an. Für eine einfaches Beispiel eines Algorithmus mit Visualisierung schaut Euch die Klasse DemoAlgorithm in der Bibliothek `algovis3d.jar` in Eclipse an. Benutzt so viele verschiedene Quanten wie möglich, z. B. ChangeVertexAppearance, ChangeVertexLocation, ChangeVertexPickInformation etc.

Es soll angenommen werden, dass der Startknoten ein `UserDatum` mit dem Schlüssel `"label"` und dem Wert `"s"` hat. Die Kantenlängen haben als Schlüssel `"label"` und als Wert die Kantenlänge (als String!).

Speichert in jedem Knoten sein aktuelles Distanzlabel als `UserDatum` `"distance"`. Markiert auch für jede Kante, ob sie zum Kürzeste-Wege-Baum (KWB) gehört, ebenfalls durch ein `UserDatum` namens `"treeEdge"`.

Problem 3: JUnit Tests

Für die Überprüfung der Korrektheit der Warteschlange, muss jeder Schleifendurchlauf einzeln ausgeführt werden können. Dafür sollen für die folgender Eigenschaften JUnit-Tests erstellt werden.

- Die Anzahl der ausgeführten Schritte plus die Länge der Knotenwarteschlange ist stets gleich der Anzahl der Knoten (plus minus eins ;-)
- Das Distanzlabel des aktuellen Knotens ist nicht größer als das Label des nächsten Knotens in der Warteschlange.

Am Ende des Algorithmus sollte außerdem folgende Eigenschaft getestet werden:

- Die als Baumkanten markierten Kanten bilden wirklich einen Baum (einen zusammenhängenden Graphen mit $n-1$ Kanten).
- Für jeden Knoten v mit seinen eingehenden Kanten (w, v) gilt

$$\text{dist}(w)+\text{length}(v,w) \geq \text{dist}(v) \ .$$