Technische Universität München
Institut für Informatik
Prof. Esparza
R. Neumann, F. Emmert, C. Denis

SS 2012
Aufgabenblatt Pre
Mi., 11. April 2012

Programmierpraktikum LEGO Mindstorms

Dieses Semester ist die Zulassung zum Praktikum auch an das Lösen einer Programmieraufgabe geknüpft. Diese Programmieraufgabe wird im Folgenden vorgestellt.

Aufgabe 1 Dijkstras Algorithmus

Aus Vorlesungen wie Grundlagen Algorithmen und Datenstrukturen ist das Problem der Kürzeste-Wege-Findung bekannt: Bei diesem ist ein Graph gegeben, bei dem Kanten nicht-negative Kosten zugewiesen sind. In diesem sucht man einen Weg von einem Knoten x zu einem Knoten y, so dass die dabei entstehenden Kosten minimal sind, es also keinen Weg mit geringeren Kosten gibt.
Einer der Algorithmen, die dieses Problem lösen, ist der Algorithmus von Dijkstra. Dieser ist in Java zu implementieren: Das Programm liest einen Graphen ein und bekommt als Parameter die Knoten zwischen denen ein Pfad zu finden ist. Anschließend gibt es den minimalen Pfad aus – sofern einer existiert.
Gegeben ist eine Datei `Dijkstra.java`, in der die Funktionen zum Einlesen des Graphen und das Formatieren der Ausgabe bereits implementiert ist. Sie müssen nun die Klasse `Graph` ausfüllen, so dass die Konstruktion des Graphen möglich ist (Methode `addEdge`) und anschließend der kürzeste Pfad ausgegeben werden kann (Methode `getShortestPath`). Dabei dürfen beliebig viele eigene Klassen und Methoden verwendet werden. Bibliotheken jenseits der Standardbibliothek von Java 6 sind nicht zulässig.

Termin Einen Tag vor der Vorbesprechung, also 10. Juni 2012 23:59 Uhr.

Modus Die Aufgabe ist von jedem Studenten in *Einzelarbeit* anzufertigen. Java-Quellcode aus anderen Quellen darf nicht verwendet werden.

Abgabe Als Abgabe ist der Quellcode an `lego@lists.model.in.tum.de` zu senden. Bitte den Header in der Datei entsprechend ausfüllen um die Zuordnung zu erleichtern.

Benotung Die Aufgabe dient nur der Zulassung und Vorbereitung. In die Benotung des Praktikums geht sie nicht ein.

Weitere Hinweise:

- Weder das Abändern der Exception-Klassen noch der Klasse `Dijkstra` ist zulässig. Es ist aber erlaubt beliebig viele eigene Klassen zu erstellen.
- Wenn ein Knoten zur Pfadsuche nicht bekannt ist, muss `getShortestPath` die Exception `NoSuchNodeException` werfen.
- Wenn kein Pfad gefunden werden konnte, muss `getShortestPath` die Exception `NoPathFoundException` werfen.
- Das Eingabeformat für den Graphen besteht aus Zeilen der Form $x\ y\ w$, wobei $x, y \in \mathbb{Z}$ und $w \in \mathbb{N}$, und beschreibt eine Kante mit dem Gewicht w von x nach y.

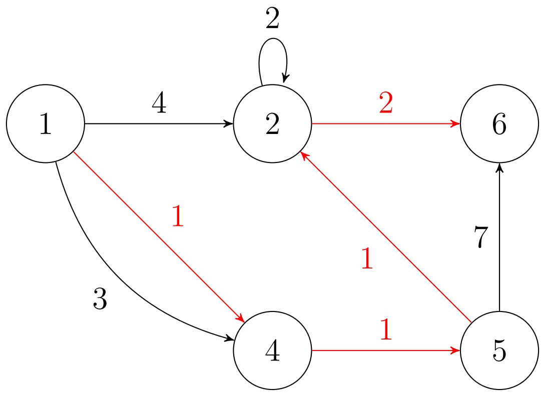

Abbildung 1: Beispielgraph

Beispiel

Angenommen, wir haben in einer Datei `graph.txt` den folgenden Graph gegeben, welcher dem Graphen in Abb. 1 entspricht:

```
1 2 4
1 4 1
1 4 3
2 6 2
2 2 2
4 5 1
5 6 7
5 2 1
```

Der kürzeste (im Sinne der Kosten) Pfad von 1 nach 6 ist der in Abb. 1 rot markierte. Die Implementierung sollte eben diesen liefern, dass heißt die Ausgabe sollte wie folgt sein:

```
> java Dijkstra 1 6 < graph.txt
1 --> 4 --> 5 --> 2 --> 6
```

oder

```
> java Dijkstra graph.txt 1 6
1 --> 4 --> 5 --> 2 --> 6
```