Praktische Informatik II SS 2009

Übungsblatt 09

Abgabe: 15.06.09

Aufgabe 1 Eine Reise, die ist lustig, eine Reise, die ist schön

Ihr plant eine Reise in das schöne ABC-Land. Viel wisst ihr noch nicht über das Land, aber ihr kennt die fünf größten Städte. Ziel ist es, die kürzeste Route von A-Stadt nach Z-Stadt zu finden.
Der Graph in Abbildung 1 stellt die Distanz zwischen den Städten dar. Zeigt Schritt für Schritt wie der Dijkstra-Algorithmus vorgeht. Zeigt für jeden Knoten, wann er zu den "gewählten Knoten" gehört, wann er im "Rand" ist und wann er zu der Menge der "unerreichten Knoten" gehört.

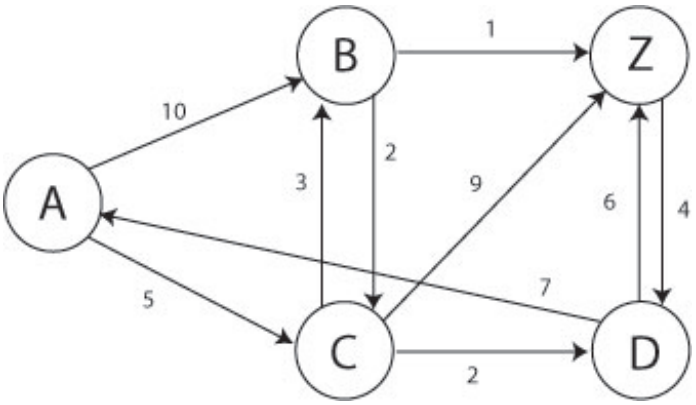

Abbildung 1: Entfernungen im ABC-Land

Aufgabe 2 Wieso in die Ferne schweifen

Nun ja, der Geldbeutel gibt keine große Reise her. Daher entscheidet ihr euch für eine Deutschlandtour. Hierzu braucht ihr einen Routenplaner, der euch immer den kürzesten Weg von einer Startstadt hin zu einer Zielstadt liefert und diesen Routenplaner programmiert ihr euch natürlich selber. Um die kürzesten Wege zu berechnen, benutzt ihr selbstverständlich den Dijkstra-Algorithmus, den ihr in der Vorlesung kennengelernt habt.

Aufgabe 2.1 Dijkstra - Algorithmus

Implementiert nun den Routenplaner in Form einer Methode, die die Namen zweier Städte als Parameter bekommt und daraufhin die kürzeste Route zwischen den beiden Städten als Zeichenkette zurückliefert. Den Verbindungsgraphen für einige

deutsche Städte erhaltet ihr in Form der auf der PI2-Webseite bereitgestellten Klasse `Neighbors`. Diese implementiert eine statische Methode `getNeighbors()`, die zu einer ihr bekannten Stadt alle Nachbarn, sowie die jeweiligen Entfernungen dorthin liefert.

Für die Implementierung des Dijkstra-Algorithmus wird eine Vorrangwarteschlange benötigt. Erklärt kurz was eine Vorrangwarteschlange ist und welche Methoden hierfür benötigt werden. Erklärt wieso diese Datenstruktur so wichtig für die Implementierung dieses Algorithmus ist. Benutzt in eurer konkreten Implementierung die Klasse `PriorityQueue<E>` aus der Java-Klassenbibliothek. Da die in dieser Warteschlange zu speichernden Knoten vergleichbar sein müssen, sollte eure Knotenklasse das Interface `Comparable` implementieren.

Aufgabe 2.2 Komplexität des Dijkstra-Algorithmus (Zusatzaufgabe +20%)

Analysiert die Komplexität des Dijkstra-Algorithmus. Inwiefern hängt die Komplexität von der Datenstruktur der Vorrangwarteschlange ab?