Übung 7: Graphen

Abgabetermin: 15.05.2018

Name: **Matrikelnummer:**

Gruppe: ☐ **G1** Di 10:15-11:00 ~~☐ **G2** Di 11:00-11:45~~ ☐ **G3** Di 10:15-11:00

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	24	☐	Java-Programm	Projekt Archiv	☐	

Aufgabe 1: Graphenalgorithmen (24 Punkte)

In dieser Aufgabe müssen Sie verschiedene Graphalgorithmen implementieren. Gegeben ist ein Graph der aus **Knoten (vertex)** und **Kanten (edge)** besteht. Kanten können, siehe VO, **gerichtet** oder **ungerichtet** sein. Weiters kann den Kanten ein **Gewicht** zugewiesen werden.

Implementieren sie alle mit **TODO** markierten Methoden in den Klassen *GraphUtil*.

Aufgabe 1.1: DFS & BFS (8 Punkte)

Implementieren Sie eine Tiefensuche (DFS) und eine Breitensuche (BFS) für den gegebenen Graphen in den Methoden **depthFirstSearch** und **breadthFirstSearch**. Die Implementierungen von **depthFirstSearch** und **breadthFirstSearch** durchlaufen den Graphen und fügen jeden besuchten Knoten in eine Liste ein. Ein Beispiel finden Sie im Anhang.

Aufgabe 1.2: Minimal Spanning Tree (8 Punkte)

Implementieren Sie die Methode ***makeMinimalSpanningTree*** die einen minimalen Spannbaum laut Vorlesung erzeugt. Erzeugen Sie dabei einen neuen Graphen der nur mehr die Kanten des minimalen Spannbaums enthält. Verwenden Sie dafür die vorgegebene Prioritätswarteschlange *VertexArrayPriorityQueue,* der Sie eine Instanz von *MinWeightVertexComparator* übergeben, um die Knoten nach *minWeight* zu gewichten.

Aufgabe 1.3: Shortest Path (8 Punkte)

Implementieren Sie die Methode ***makeShortestPath*** die einen neuen Graphen erzeugt der nur mehr die Kanten des kürzesten Pfades enthält. Verwenden Sie dafür die vorgegebene Prioritätswarteschlange *VertexArrayPriorityQueue,* die Sie mit einer Instanz von *DistanceVertexComparator* erzeugen um die Knoten im Heap nach *distance* zu gewichten.

Abzugeben ist: Projekt Archiv

Implementierungshinweise:

- Verwenden Sie das Vorgabeprojekt **PI2_UE07.zip**.
- Sie können die Klasse *ArrayStack* und *ArrayQueue* als Hilfsdatenstruktur verwenden.
- Sie können die Methode ***DotMaker.makeDotForGraph(Graph graph)*** verwenden um eine GraphViz Darstellung eines Graph zu generieren. Dabei werden gerichtete Kanten mit Pfeilen dargestellt und Kanten deren Gewicht !=0 werden mit Kantengewicht gezeichnet.
- In der Implementierung des minimalen Spannbaums und der kürzesten Pfad Suche kopieren Sie den Graph mit der Hilfsmethode ***Graph.clone()*** und ersetzten die Kanten (***Graph.edges***) durch einen neue, leere Liste.
- Fügen Sie Ihre Implementierung in den mit **TODO** markierten Teilen den Klassen *GrahUtil* ein.
- Halten Sie sich an die **Codierungsrichtlinien** auf der Kurs Website.

Anhang