Algorithmen und Datenstrukturen

Übung 4 (Advanced Level 3)

Sebastian Stricker – If16b038  
Johannes Fessler – If16b042

# Algorithmus

Das Netzwerk wird als ungerichteter Graph eingelesen und gespeichert.

Alle Stationen werden als „Vertex“ eingelesen.   
Die Station wird mit dem Namen als Key auch in eine Hashmap „m\_stations“ mit einem Vertex-Pointer als Value eingespeichert.

Zusätzlich werden zwischen zwei Vertices auch jeweils zwei Edges erzeugt, in jede Richtung eine, wobei ein Vertex-Pointer in der Edge auf den jeweils anderen Vertex zeigt.

In jedem Vertex ist ein Vector aus Edges gespeichert, die auf alle Nachbarn zeigen.

Jede neue Linie wird in einen String-Vector eingespeichert, die Reihenfolge des Einlesens ergibt den Index der Linie, der dann im gesamten Algorithmus immer wieder als Linien-ID verwendet  
wird und auch in den Edges hinterlegt ist.

Die Datenstruktur ist nun erzeugt, der Algorithmus fängt an.

Der Algorithmus ist eine Variation von Dijkstra, wir verwenden hierbei keine Adjazenzliste,

Dijkstra -

Initialisieren -

Abstrakter

Ausgehend von der Anfangsstation „origin“, die angegeben werden muss, besuchen wir alle Nachbarn und überprüfen, ob bereits ein anderer, kürzerer Pfad dorthin existiert.

Sobald ein Nachbar besucht wird, wird überprüft, ob der eigene Pfad schneller ist, anhand der Membervariable Pfadlänge des Nachbarn. Wenn der Pfad kürzer ist, wird die neue Pfadlänge eingetragen und ein Pointer „previous“, der auf den vorherigen, besten Pfad-Vertex zeigt.

Sobald das Ziel erreicht wurde, wird abgebrochen. - > bis der kürzeste

Die previous-Pointer in jedem betroffenen Vertex werden umgedreht, zeigen also vom ehemaligen Ziel nun zur ehemaligen Quelle.

Somit kann eine einwandfreie, schöne Ausgabe gewährleistet werden.

Die Ausgabe an sich erfolgt durch traversieren durch die previous-Pointer bis das Ziel erreicht ist.