Übung 3: Wiener Verkehrsnetz Protokoll:

Datentypen:

Station: Ein Struct mit einem String „Name" als Datentyp, repräsentiert die Knoten im Graphen.

ConnInfo: Ein Struct mit int distance und string linie als Member.

Connection: Eine Klasse mit 3 Membern, Station, Info und Pre. Info ist vom Datentyp ConnInfo und Pre speichert den Namen des Vorgängers.

Graph: Eine Klasse mit l (Adj-Liste) als member, m ein map Container der STL namespace der allen einzigartigen Stationen eine ID zuweist, int size der die Anzahl der Stationen speichert und ein boolean Array vis welcher dazu dient die bereits besuchten Knoten zu markieren.

Datenstrukturen:

Verkettete Liste, um alle Verbindungen zu speichern.

Map um jeder Station eine einzigartige ID zuzuweisen.

Priority Queue, um alle entdeckten Pfade im Dijkstra Algorithmus zu speichern, diese werden aufsteigend nach der Größe der Gewichtung eingefügt.

Set der alle besuchten Pfade abspeichert, um den kürzesten Pfad nach der Terminierung des Dijkstra Algorithmus zu rekonstruieren.

Methoden:

readFile: liest die Daten aus der Datei heraus. Aufwand **O(n)** mit n = Anzahl der Strings in der Datei

Graph: Der Graph Konstruktor nimmt die eingelesenen Stationen und Verbindungen als Parameter und erzeugt anhand dieser einen Graphen. Um das schnelle Einfügen und Lesen von Daten zu gewährleisten wird die Hashmap m mit den Stationen als Key und IDs als Wert initialisiert Aufwand O(n) mit n = Anzahl der Stationen. Danach werden in der Adjazenz Liste l die IDs als Keys verwendet und für jede Verbindungen der jeweiligen Station ein Element in die Liste appendet. Aufwand O(2(1 + n)) = **O(n)** wobei n die Anzahl der bereits vorhandenen Verbindungen ist. (2 weil eine Verbindung jeweils einmal für 2 Stationen eingefügt werden muss, 1 weil der Zugriff auf das Array mit den gespeicherten Listen Konstant ist)

initVis: initialisiert das visited Array mit false. Aufwand **O(n)**

Dijkstra: Aufwand **O(V^2)** wobei V die Anzahl der Stationen ist. Dies ist aber nur der Fall wenn der Graph vollständig ist und jeder Knoten V mit jedem anderen Knoten im Graph verbunden ist.