

## ÜBUNG 3 (GRAPHEN)

### Aufgabe

Arbeiten Sie in 2er Gruppen an folgendem Programmierbeispiel. Die Wahl der Programmiersprache bleibt Ihnen überlassen (C, C++, C#, Java).

#### **Aufgabenstellung: Shortest Path im Wiener Verkehrsnetz**

Gegeben ist ein Graph mit gewichteten Kanten. Entwickeln und implementieren Sie einen Algorithmus, der den kostengünstigsten Weg zwischen zwei Knoten findet.

Der zu durchsuchende Graph ist durch eine Textdatei mit folgendem Aufbau gegeben. Jede Zeile beginnt mit dem Namen einer Linie im Wiener Verkehrsnetz (nur U-Bahn und Straßenbahn), gefolgt von einem Doppelpunkt. Danach folgen abwechseln ein String unter Anführungszeichen (Stationsname) und eine Zahl (Fahrzeit zur nächsten Station=Kosten der Kante). Es kann auch zyklische Linien geben (d.h. Stationen können mehrfach angefahren werden).

Beispiel:

```
U1: "Leopoldau" 2 "Grossfeldsiedlung" 1 "Aderklaaer Strasse" 1
"Rennbahnweg" 2 "Kagraner Platz" 2 "Kagran" 1 "Alte Donau" 2
"Kaisermuehlen-VIC" 1 "Donauinsel" 2 "Vorgartenstrasse" 1
"Praterstern" 1 "Nestroyplatz" 1 "Schwedenplatz" 1
"Stephansplatz" 2 "Karlsplatz" 2 "Taubstummengasse" 1
"Suedtirolerplatz" 2 "Keplerplatz" 1 "Reumannplatz"
```

Die vollständige Datei mit dem Wiener Verkehrsnetz ist in Moodle zu finden.

Ihre Aufgabe ist es ein Programm mit dem Namen "find\_path" zu schreiben, das drei Argumente in der folgenden Struktur entgegennimmt:

```
find_path filename_graph start ziel
```

- |                  |                                     |
|------------------|-------------------------------------|
| • filename_graph | File mit Graphen nach obigem Format |
| • start          | Startstation                        |
| • ziel           | Zielstation                         |

Ihr Programm soll zuerst den Graphen aus dem Input-File lesen und in eine geeignete Datenstruktur speichern. Anschließend wird der Algorithmus zur Suche des kürzesten Pfades vom Start zum Ziel ausgeführt und der gefundenen Pfad übersichtlich ausgegeben. Das Output-Format ist nicht fix vorgegeben, soll aber zumindest darstellen

- welche Kanten der Reihe nach durchlaufen werden und welche Linien dabei jeweils verwendet werden,
- wo umgestiegen werden muss,
- und wie hoch die Gesamtkosten (=Fahrzeit) sind.

**Protokoll**

Erstellen Sie ein Protokoll mit der Beschreibung des verwendeten Algorithmus. Beschreiben und begründen Sie weiters den Aufwand ihrer Algorithmen mittels O-Notation und versuchen sie die Laufzeitabschätzung mit eignen Messungen zu überprüfen.

**Vertiefung für 3er-Gruppen**

3er-Gruppen müssen zusätzlich eine der folgenden Erweiterungen nach ihrer Wahl implementieren:

- Implementieren Sie einen zweiten anderen Algorithmus Ihrer Wahl (z.B. Dijkstra, A\*, ...) der die Graphensuche durchführt und vergleichen Sie die Ergebnisse (Speicherbedarf, O-Notation).
- Implementieren Sie einen Algorithmus, welcher das TSP für das U-Bahnnetz oder Teile des U-Bahnnetzes löst.