ÜBUNG 4 (ADVANCED LEVEL BEISPIEL 3)

1. Aufgaben

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

Beispiel Fahrplanauskunft

Implementieren Sie den Algorithmus Priority First Search, um den schnellsten Weg zwischen 2 Stationen in einem Verkehrsnetz (z.B. Wiener U-Bahn) zu finden. Das Verkehrsnetz wird als gewichteter Graph in Form einer Adjazenzliste abgespeichert: jede Station ist ein Knoten, jede Verbindung zwischen 2 Stationen ist eine Kante, die Fahrzeit in Minuten zwischen den beiden Stationen ist das Gewicht der Kante. Zusätzlich wird bei jeder Kante der Name der Linie abgespeichert, z.B. U1.

Das Programm liest den Plan des Verkehrsnetzes aus einer Textdatei ein (Dateiname als Parameter), baut die Adjazenzlisten des Graphen auf und fragt den Benutzer nach Start- und Zielstation.

Die Ausgabe des Programms ist die schnellste Verbindung zwischen diesen beiden Stationen (gesamte Fahrzeit in Minuten) und der vollständige Weg der gefundenen Verbindung mit allen Zwischenstationen und Linien.

Hinweise

- Um den kürzesten Weg am Ende der Berechnung ausgeben zu können, ist eine Hilfsdatenstruktur notwendig, z.B. ein Array das für jeden Knoten den Vorgänger im schnellsten Weg enthält. Die Werte dieses Feldes können während der Suche berechnet werden, indem beim Einordnen in den Heap für jeden Knoten der Vorgänger ebenfalls im Heap abgespeichert wird. Dadurch erhält man beim Entfernen eines Knotens aus den Heap dessen Vorgänger und kann ihn in das Hilfsfeld eintragen. Wenn nun der kürzeste Weg gefunden wurde, können aus diesem Hilfsfeld die einzelnen Knoten des kürzesten Weges in umgekehrter Reihenfolge ausgelesen werden.
- Der vorgestellte Algorithmus sucht strikt nach dem kürzesten Pfad in dem Graphen, berücksichtig aber nicht die Zeiteinbuße durch Umsteigen. Versuchen Sie die Suche dahingehend zu erweitern, dass beim Umsteigen von einer Linie zu einer anderen eine gewisse Zeitkonstante (z.B. 5 Minuten) zum minimalen Weg addiert wird und somit zusätzlich versucht wird, das Umsteigen zu minimieren.
- Eine grafische Oberfläche ist nicht unbedingt erforderlich.
- Das Dateifomat ist wie folgt definiert:
    - Eine Zeile beginnt mit dem Namen der Linie, gefolgt von einem Doppelpunkt, gefolgt von einer Liste der Stationen – zwischen den Stationensbezeichnungen ist jeweils die Fahrtdauer in ganzen Minuten angegeben (gehen Sie davon aus, dass die angegebene Fahrtdauer richtungsunabhängig ist). Der Doppelpunkt ist nicht Teil des Namens der Linie. Stationsbezeichungen sind prinzipiell in Anführungszeichen

eingeschlossen um das Parsing zu erleichtern. Jede Zeile wird mit einem Zeilenumbruch abgeschlossen (auch die letzte Zeile). Mehrere aufeinanderfolgende Leerzeichen und/oder Tabs sind wie 1 Leerzeichen zu interpretieren. Mehrere aufeinanderfolgende Zeilenumbrüche sind wie 1 Zeilenumbruch zu interpretieren.
  - Beispiel:
    U1: "Stephansplatz" 2 "Karlsplatz" 2 "Taubstummengasse"
    U2: „Karlsplatz" 1 "Museumsquartier" 1 "Volkstheater"
  - Sie können das beiliegende File ubahn.txt zu Testzwecken verwenden. Ihre Abgabe wird jedoch mit anderen Daten getestet welche auch komplexere Graphen enthalten können (z.B. Ringlinien mit Zyklen, etc.).

Optionale Erweiterungen

- Graphische Oberfläche, Web-Interface zur Suche, …

2. Abgabe

Abgabe über Moodle
Abzugeben ist eine zip/tgz Datei mit

a) Quellcode mit ausführlichen Kommentaren
b) Ausführbares Programm
c) kurze Installations- und Bedienungsanleitung
d) Doku mit Beschreibung des implementierten Algorithmus + Testprotokoll

Abgabeschluss ist der 6.Juni 2017 (23:55)

Weiters muss die Abgabe in Form eines eigenen ausführlichen Code Reviews am 6. bzw. 8.Juni 2017 präsentiert werden!

Dieses Beispiel ersetzt das 4.Übungsbeispiel (3. Code Review mit 25 Punkten) und die Vorlesungsprüfung!

ACHTUNG!!!
Bitte bis 16.5 für die „advanced" Beispiele per Mail an nimm@technikum-wien.at mit der Beispielnummer anmelden. Max. 3 Gruppen pro Verband (A,B) dürfen ein Advanced Beispiel (1,2,3) wählen, Vergabe nach First-Come-First-Served Algorithmus. Ein Nichtantreten zur Vorlesungsprüfung bei gleichzeitiger Anmeldung zum „advanced" Beispiel bedeutet automatisch, dass sie diesen Prüfungsmodus gewählt haben und negativ beurteilt werden, wenn das Beispiel dann doch nicht abgegeben wird.