Programming C++

Project **Navigationssystem**

Status: 25. Mai 2021, 19:36

Supervisor: Prof.Dr. G. Haase, `gundolf.haase@uni-graz.at`

Implementieren Sie ein Programm, das ermöglicht, in einer Karte den kürzesten Weg zwischen zwei Punkten zu berechnen. (8 Pkt.)

- Definieren Sie zuerst eine Klasse `Graph`, die im Konstruktor eine Karte aus einer Datei (deren Name als Parameter übergeben wird) einliest. Eine Karte besteht aus einer Menge von Punkten (*Knoten*), die durch Straßen (*Kanten*) verbunden sind. Als Vereinfachung nehmen wir an, dass jede Straße gerade ist, d. h. ihre Länge der Euklidischen Distanz entspricht.
- Schreiben Sie nun eine Methode `edgeLength(const unsigned int i, const unsigned int j)`, die die Länge der Straße zwischen Knoten i und j ermittelt, bzw. -1 zurückgibt, wenn es zwischen diesen Knoten keine Straße gibt. Machen Sie anschließend eine Methode `pathLength(const vector<unsigned int> &path)`, die die Länge eines Pfades ermittelt (bzw. -1 zurückgibt, wenn dieser Pfad nicht zulässig ist, d. h. nicht immer eine Straße vorhanden ist).
- Erzeugen Sie eine Methode `navi`, die den kürzesten Weg zwischen zwei Knoten mithilfe des Dijkstra-Algorithmus[1] ermittelt.
- Testen Sie Ihr Programm mit der Datei Map.txt[2] (siehe auch Abbildung 1), die folgendes Format hat:

```
NODES:
Knoten-Index     x-Koordinate     y-Koordinate
...
EDGES:
Knoten-Index     Knoten-Index
...
```

  Zuerst werden die Knoten mit ihren Indexen und Koordinaten aufgelistet und danach kommen alle existierenden Kanten, von denen immer Anfangs- und Endknoten auf einer Zeile stehen.

  Berechnen Sie nun den kürzesten Weg zwischen folgenden Knotenpaaren: $(54, 56)$, $(32, 9)$, $(23, 8)$.

Zusatzaufg.: Ändern Sie die Methode `navi` so, dass sie auch für E-Fahrzeuge, die limitierte Reichweite (als Parameter übergeben) haben, funktioniert. Gehen Sie dabei davon aus, dass in jedem Knoten eine E-Ladestation vorhanden ist. (+1 Pkt.)

Zusatzaufg.: Um die Navigation BenutzerInnen-freundlicher zu machen, geben Sie außer der (+2 Pkt.)

[1] `https://de.wikipedia.org/wiki/Dijkstra-Algorithmus`

[2] `http://imsc.uni-graz.at/haasegu/Lectures/Kurs-C/SS21/projects/Navi/Map.txt`

Reihenfolge der Knoten auch die Fahrtrichtung in jedem Knoten (auf jeder Kreuzung) an. Dabei wird angenommen, dass eine Winkeländerung unter 10° immer noch als „geradeaus“ gilt.

Zusatzaufg.: Implementieren Sie den Algorithmus von Floyd und Warshall[3] und ermitteln Sie den kürzesten Weg zwischen allen Knotenpaaren. Machen Sie anschließend dieselbe Berechnung (Ermitteln der Kürzesten Distanz zwischen allen Knotenpaaren) auch mit Ihrem bereits implementierten Dijkstra-Algorithmus und vergleichen Sie die Laufzeiten. (+1 Pkt.)

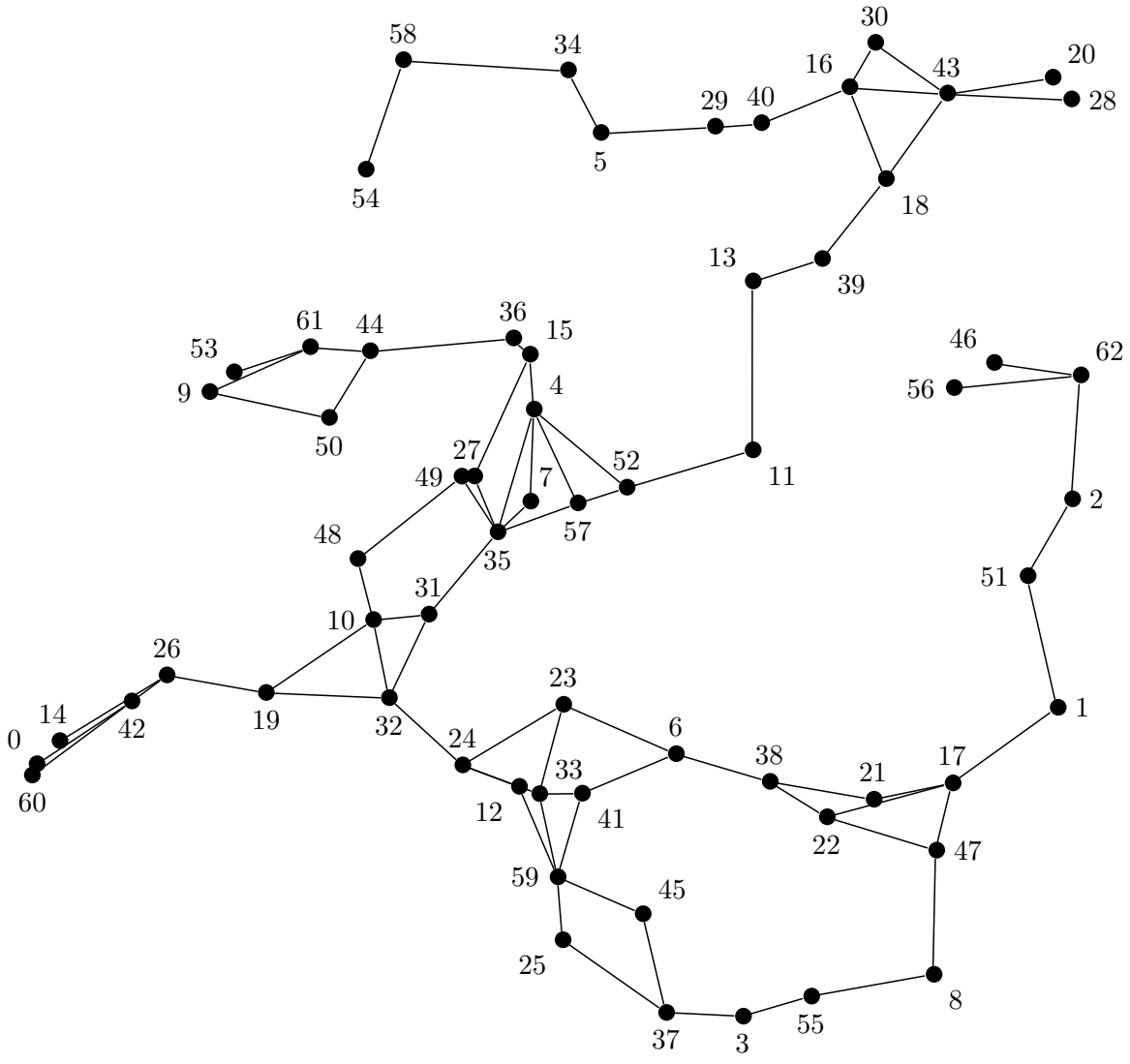

Abbildung 1: Karte

[3] https://de.wikipedia.org/wiki/Algorithmus_von_Floyd_und_Warshall