Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Y. Maus, O. Saukh

24 Juni, 2016

Algorithmen und Datenstrukturen
Sommersemester 2016
Übungsblatt 9

Abgabe bis 12:00, Freitag, 01 Juli, 2016

Achtung: Bitte fügen Sie Ihrer Lösung eine (kurze) Datei *erfahrungen.txt* mit Ihren Erfahrungen mit dem jeweiligen Aufgabenblatt hinzu. Das Forum können (und sollen) Sie nicht nur für Fragen zur Übung, sondern auch für Fragen zur Vorlesung benutzen.

Aufgabe 1: Dijkstra Algorithmus mit Heuristik

In der Vorlesung wurde der Dijkstra Algorithmus vorgestellt um alle kürzesten Wege von einem bestimmten Knoten zur allen anderen Knoten in einem Graph mit positiven Kantengewichten zu finden. Sei $w : E \to \mathbb{N}$ die Gewichtsfunktion der Kanten und für zwei Knoten $u, v \in V$ sei $d(u, v)$ die Länge eines kürzesten Weges von u nach v. Im Folgenden bezeichne $s \in V$ einen Start- und $t \in V$ einen Zielknoten.

a) (2 Punkte) Wenn man Dijkstras Algorithmus abbricht, sobald der Knoten t aus der Prioritätswarteschlange entfernt wird, so erhält man einen kürzesten Weg zwischen s und t ohne unbedingt alle kürzesten Wege von s aus zu berechnen.

   Welche Knoten werden in diesem Fall besucht und von welchen Knoten und Kanten hängt in dem Fall die Laufzeit des Algorithmus ab?

b) (4 Punkte) Der $s-t$-Dijkstra Algorithmus (welcher einen kürzesten Weg von s nach t findet) speichert in der Prioritätswarteschlange zu jedem Knoten u immer den bisher besten gefundenen Abstand $\delta(s, u)$ vom Startknoten. In einer Iteration entnimmt man den Knoten u mit dem minimalen Wert in der Prioritätswarteschlange und updated ggf. die Werte in der Warteschlange für alle seine Nachbarn v. D.h. falls $d(s, u) + w(u, v) < \delta(s, v)$ ist, so setzt man

$$\delta(s, v) := d(s, u) + w(u, v). \tag{1}$$

   Dies liefert zwar immer den kürzesten Weg (Beweis Vorlesung), jedoch werden dabei oft sehr viele Knoten besucht. Die Prioritätswarteschlange legt dabei fest, welcher Knoten als nächstes besucht wird. Im Folgenden wollen wir zeigen, wie man die besuchten Knoten besser priorisieren kann, wenn man mehr Informationen über den Graph zur Verfügung hat.

   Sei t der Zielknoten und nehmen Sie an, dass eine Funktion $h : V \to \mathbb{R}_+$ mit den folgenden Eigenschaften existiert (die Funktion h ist abhängig vom Ziel t).

$$0 \leq h(u) \leq d(u, t) \qquad \forall u \in V, \tag{2}$$

$$0 \leq h(u) \leq w(u, v) + h(v) \qquad \forall (u, v) \in E \tag{3}$$

   Nun wird der Dijkstra-Algorithmus so modifiziert, dass in der Prioritätswarteschlange statt $\delta(s, v)$ der Wert

$$\delta(s, v) + h(v) \tag{4}$$

gespeichert wird (auch die Initialisierung wird angepasst).

Beweisen Sie, dass der neue Algorithmus auch den kürzesten Pfad von s nach t findet. Laden Sie Ihre Lösung als PDF ins SVN hoch.

Hinweis: Betrachten Sie den gewichteten Graphen $G' = (V, E)$, der aus den Knoten und Kanten von G besteht und die Kantengewichtsfunktion $w'(u, v) := w(u, v) + h(v) - h(u)$ hat.

Aufgabe 2: Kürzeste Pfad Algorithmen

Im *public* SVN Verzeichnis liegt eine Datei **Freiburg.txt** - eine Abstraktion des Freiburger Straßenverkehrsnetzes. Das Format wurde bereit in Übungsblatt 7 verwendet. Die Gewichte der Kanten in dem gespeicherten Graph sind die echten Distanzwerte in Kilometern. In der Datei **points.txt** sind zwei Knoten IDs gespeichert, welche den IDs der Knoten s und t entsprechen.

a) (5 Punkte) Implementieren Sie den $s - t$-Dijkstra Algorithmus, welcher einen kürzesten Weg von einem beliebigen Knoten s zu einem Knoten t findet. Wenden Sie den Algorithmus dann mit dem Graphen in **Freiburg.txt** und den Start- und Zielorten aus **points.txt** an.

b) (4 Punkte) Wir nutzen nun Aufgabe 1 und die Information, dass der Graph eine geometrische Struktur hat, um den Dijkstra Algorithmus zu verbessern. Implementieren Sie die in Aufgabe 1 genannte Modifikation des Dijkstra Algorithmus, wobei h wie folgt gewählt ist.

$$h(v) := \text{Luftliniendistanz zwischen } v \text{ und } t. \tag{5}$$

Um die Funktion $h(u) = h(u, t)$ zu implementieren können Sie die Funktion

```
distance(self, lat1, lon1, lat2, lon2):
```

verwenden, welche die Luftliniendistanz zwischen zwei Knoten mit den GPS-Koordinaten $(lat1, lon1)$ und $(lat2, lon2)$ berechnet.

Wenden Sie den Algorithmus auf den Beispieldaten an.

c) (4 Punkte) Bestimmen Sie für beide beide Implementierungen die gefundenen Pfade von s nach t und speichern Sie das Resultat in den csv-Dateien *dijkstra_path.csv* und *a_star_path.csv*.

Speichern Sie außerdem für beide Algorithmen die von den Algorithmen besuchten Knoten in die Dateien *dijkstra_visited.csv* und *a_star_visited.csv*.

Laden Sie alle csv-Dateien in den SVN hoch. Laden Sie die Dateien (*dijkstra_visited.csv* und *a_start_visited.csv*) auf die Web-Seite

```
http://www.gpsvisualizer.com
```

hoch um die besuchten Knoten zu visualisieren. Fügen Sie die Visualisierung Ihrer Lösungen hinzu (ein Screenshot ist ausreichend). Um ein gutes Bild zu bekommen, wählen Sie auf der Webseite die Option "Plot data points".

d) (1 Punkt) Welcher Algorithmus hat weniger Knoten besucht und warum? Schreiben Sie Ihre Antwort in die Datei *erfahrungen.txt*

Dateiformat

Die Beschreibung der Dateiformate finden Sie auf dem Übungsblatt 07.

Hinweis: Falls Sie Probleme mit dem csv-Export oder dem Lesen von Textdateien haben, so können Sie die Musterlösung von Übungsblatt 7 zur Hilfe nehmen.