Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Moritz Baum, Julian Dibbelt,
Ben Strasser

Übungsblatt 3

Praktikum Algorithm Engineering – Routenplanung (WS 13/14)

Ausgabe 6. November 2013
Abgabe 13. November 2013

Aufgabe 1: Code Update

Im Git-Repository liegt neuer Code. Bitte laden Sie diesen genauso wie die letzten beide Male herunter.

Achtung: Es gibt auch Änderungen an bereits bekannten Stellen.

Aufgabe 2: Contraction Hierarchies (CH)

Das Thema dieses Aufgabenblatts sind Contraction Hierarchies. Bei Bedarf/Interesse können Sie die Details in folgenden Quellen nachlesen. Der am letzten Poolraumtermin gegebene Überblick sollte aber für das Aufgabenblatt reichen.

- Diplomarbeit von Geisberger : http://algo2.iti.kit.edu/1094.php
- Geisberger, R., Sanders, P., Schultes, D., & Vetter, C. (2012). Exact routing in large road networks using contraction hierarchies. Transportation Science, 46(3), 388-404.
- Eine paar einfache und gute Ordnungsheuristikansätze werden im ersten Paragraph von Section 7 auf Seite 32 von folgendem Paper beschrieben:
  Abraham, I., Delling, D., Goldberg, A. V., & Werneck, R. F. (2012). Hierarchical hub labelings for shortest paths. In Algorithms–ESA 2012 (pp. 24-35). Springer Berlin Heidelberg.

Die Paper sind leider nicht Open Access und deswegen nur aus dem Uninetz verfügbar.

Aufgabe 3: CH-Erstellen

Im Verzeichnis `/amd.home/algoDaten/praktikum-ws-13-14/` finden Sie für jeden Graph Dateien mit der Endung `*.order`. Dies sind fertige Knotenordnungen wie sie zum Aufbau einer CH benötigt werden. Mit folgendem Aufruf soll aus der Knotenordnung und dem Graph eine CH erstellt werden:

```
./compute_contraction_hierarchy karlsruhe.gr karlsruhe.order karlsruhe.ch
```

Dazu müssen Sie aber zuerst die Zeugensuche in `contraction_hierarchy.h` vervollständigen. Wie gewohnt können Sie die Korrektheit der Anfrage mit folgendem Befehl überprüfen:

```
./check_contraction_hierarchy_against_distance_test_queries \
    karlsruhe.ch karlsruhe.100.q
```

Mit folgendem Befehl können Sie sich Details über aufgebaute CHs anzeigen lassen:

```
./examine_contraction_hierarchy karlsruhe.ch
```

Erstellen Sie für alle Graphen CHs und messen sie die benötigte Zeit. Was passiert, wenn Sie eine zufällige Knotenorndung verwenden? (Siehe `generate_random_order.cpp`)

Warnung: Die Vorberechnung auf Europa benötigt viel RAM. Sollten sie keine 8GB RAM haben, dann betrachten Sie nur die 3 restlichen Graphen.

Aufgabe 4: CH-Anfrage

Hinweis: Für diese Aufgabe genügt es, wenn Sie nur den Spielegraphen und den größten Straßengraphen betrachten, den Ihre Rechner verarbeiten kann.

In der Datei `check_contraction_hierarchy_against_distance_test_queries.cpp` wird eine elementare CH-Anfrage verwendet. Dies ist eine bidirektionale Suche mit unterschiedlichen Vorwärts- und Rückwärtssuchgraphen. Ferner unterscheidet sich das Abbruchkriterium. Warum war diese Modifikation nötig? Ferner werden hier `TimestampFlags` verwendet. Wie schnell ist die Anfrage mit `BoolFlags`?

In der Datei `evaluate_contraction_hierarchy_against_distance_test_queries.cpp` wird die CH-Anfrage stärker analysiert und optional eine weitere Optimierung namens Stall-on-Demand verwendet. Es gibt viele Varianten von Stall-on-Demand. Die hier verwendete ist die einfachste mir bekannte. Schauen Sie sich den Code an und versuchen Sie die Idee zu verstehen. Verwenden können Sie das Programm wie folgt:

```
./evaluate_contraction_hierarchy_against_distance_test_queries \
    europe.ch europe.q no_stall_on_demand > query.csv
```

Die Ausgabe ist eine CSV-Datei mit sehr vielen Kennzahlen. Berechnen Sie Mittelwerte für alle. Dies geht sehr einfach mit einem guten Statistikprogram. Eine Möglichkeit wäre R mit den Befehlen:

```
read.csv("query.csv")->q
summary(q)
```

Was stellen die Kennzahlen dar? Wie verändern sich die Kennzahlen, wenn Sie Stall-on-Demand verwenden?

Aufgabe 5: CH-Ordnung

Hinweis: Es reicht, wenn Sie für diese Aufgabe nur den Karlsruhe Graph betrachten.

Um die Knotenordnung zu bestimmen, werden alle noch nicht kontrahierten Knoten nach "Wichtigkeit" geordnet. Der Knoten mit der geringsten "Wichtigkeit" wird dann als erstes kontrahiert. Es gibt keine formale Definition von "wichtig". Es gibt nur heuristische Ansätze, die aber auf den untersuchten Graphen funktionieren. Auf diese Weise wurden auch die euch zur Verfügung gestellten Ordnungen erstellt. Man sagt, dass die Wahl des Knoten *online* geschieht und nicht *offline* wie bei einer vorgegebenen Ordnung. Die Implementierung ist in der Datei `compute_online_contraction_hierarchy.cpp`. Wir haben jedoch die Bewertungsfunktion gelöscht. Implementieren Sie einige Ansätze und versuchen Sie, eine ähnlich gute Knotenordnung wie die vorgegebenen zu berechnen. Sie dürfen auch gerne eigene Ideen ausprobieren. Alles, was zu einer guten Ordnung führt, ist erlaubt.