Praktikumsaufgaben für GKA

WiSe 2016/17

13. September 2016

J. Padberg

Aufgaben

Allgemeines zu allen Aufgaben

Für die Bearbeitung der Praktikumsaufgaben erhalten Sie auf der Homepage der LV

- Beispielgraphen für das Praktikum
- Links zu verschiedenen Libraries:
  - ADT für Graphen: JGraphT
  - Graph Visualization: Graphviz oder JGraph
  - Java Universal Network/Graph Framework: JUNG
  - Framework zum Testen von Programmen JUnit

Das Ergebnis der Bearbeitung einer Aufgabe wird von jeder Gruppe im Praktikum vorgestellt. Das heißt, die Aufgaben muss *zum Praktikumstermin fertig bearbeitet* sein. Über die Vorstellung hinaus wird für jede Aufgabe erwartet,

1. Implementierung (bzw. Modellierung) der gestellten Aufgabe, also
   - eine korrekte und möglichst effiziente Implementierung in Java die der vorgegeben Beschreibung entspricht,
   - die Kommentierung der zentralen Eigenschaften/Ereignisse etc. im Code (bzw. in der Modellierung) und
   - hinreichende Testfälle in JUnit und ihre Kommentierung.
2. Schriftliche Erläuterung Ihrer Lösung (Lösungsdokumentation)
   - Bitte geben Sie Ihr Team, die Aufgabenaufteilung, die Bearbeitugsdauer, Ihre Quellen und den aktuellen Stand an.
   - Algorithmus

- Datenstrukturen
- Implementierung
  Umsetzung der Aspekte der Implementierung
- umfassende Dokumentation der Testfälle

Anforderungen an die Bearbeitung sind:

- Alle Lösungen aller Aufgaben sollen als eine Lösung laufen können.
- Alle Algorithmen sollen zum Vergleich über Möglichkeiten zur Messung der Anzahl der Zugriffe auf den Graphen verfügen.
- Alle Algorithmen sollen durch JUnit-Tests getestet werden, wobei die Abdeckung der Testfälle diskutiert werden soll.

Es gibt drei Praktikumsaufgaben, die jeweils unterschiedliche Implementierungsaspekte haben, wobei immer auf die Qualität der Modellierung zu achten ist.

- Schönheit/Benutzbarkeit der Visualisierung
- Qualität und Umfang der Tests
- Schnelligkeit

Weitere Details werden in der jeweiligen Aufgabenstellung angegeben.
Eine Aufgabe gilt als erledigt, wenn sie

1. die Dokumentation Ihrer Arbeit am Praktikumsanfang abgegeben und
2. im Praktikum vorgestellt und von mir (mindestens) als ausreichend anerkannt wurde und
3. die (ggf verbesserte) Lösungsdokumentation der Aufgabe mitsamt der beantworteten Fragen bei mir als PDF vorliegt.

1 Visualisierung, Speicherung und Traversierung von Graphen

Die Graphen, mit denen Sie arbeiten, sollen in diesem Format (siehe auch VL-Folien) gespeichert und gelesen werden:

gerichtet

```
<name node1>[ -> <name node2>][(edge name)][: <edgeweight>];
```

ungerichtet

```
<name node1>[ -- <name node2>][(edge name)][: <edgeweight>];
```

Die Aufgabe umfasst:

- die Einarbeitung in die gewählte library
- das Einlesen/ Speichern von ungerichteten sowie gerichteten Graphen und die Visualisierung der Graphen,
- das Implementieren eines Algorithmus zur Traversierung eines Graphen (Breadth-First Search (BFS)),
- dabei soll als Ergebnis der kürzeste Weg und die Anzahl der benötigten Kanten angegeben werden.
- einfache JUnit-Tests, die Ihre Methoden überprüfen und
- JUnit-Tests, die die Algorithmen überprüfen unter Benutzung der gegeben *.gka*-Dateien
- die Beantwortung der folgenden Fragen:
  1. Was passiert, wenn Knotennamen mehrfach auftreten?
  2. Wie unterscheidet sich der BFS für gerichtete und ungerichtete Graphen?
  3. Wie können Sie testen, dass Ihre Implementierung auch für sehr große Graphen funktioniert?

Vorstellung und Abgabe der Lösungen	der Teams in	am
	GKAP/01	24.10.
	GKAP/02	31.10.
	GKAP/03	7.11.
	GKAP/04	8.11.

Der Zusatzaspekt betrifft die Visualisierung der Graphen und der Traversierung

*

2 Optimale Wege

In dieser Teilaufgabe soll wird den Dijkstra-Algorithmus aus der VL ein (etwas) effizienterer gegenübergestellt werden und die beiden experimentell untersucht werden. Dabei sollen größere Graphen (größer hinsichtlich Kanten- und Knotenanzahl) zum Vergleich genutzt werden.

Vergleich Dijkstra mit Floyd-Warshall

Für gerichtete Graphen mit beliebigen Kantenbewertungen liefert der Algorithmus von Floyd und Warshall kürzeste Wege zwischen allen Knotenpaaren, falls der Graph keine Kreise mit negativer Kantenbewertungssumme besitzt. Andernfalls findet er einen solchen Kreis.

Der Algorithmus arbeitet mit zwei $|V| \times |V|$-Matrizen, der Distanzmatrix $D = (d_{ij})$ und der Transitmatrix $T = (t_{ij})$. Am Ende des Algorithmus gibt d_{ij} die Länge des kürzesten Weges von v_i nach v_j an, und t_{ij} benennt den Knoten mit der höchsten Nummer auf diesem Weg (hierdurch läßt sich der Weg rekonstruieren).

Algorithmus

1. Setze $d_{ij} := \begin{cases} l_{ij} & \text{für } v_i v_j \in E \text{ und } i \neq j \\ 0 & \text{für } i = j \\ \infty & \text{sonst} \end{cases}$ und $t_{ij} := 0$
2. Für $j = 1, \ldots, |V|$:
   - Für $i = 1, \ldots, |V|;\ i \neq j$:
     - Für $k = 1, \ldots, |V|;\ k \neq j$:
       - Setze $d_{ik} := \min\{d_{ik}, d_{ij} + d_{jk}\}$.
       - Falls d_{ik} verändert wurde, setze $t_{ik} := j$.
     - Falls $d_{ii} < 0$ ist, bricht den Algorithmus vorzeitig ab.
       (Es wurde ein Kreis negativer Länge gefunden.)

Die Aufgabe umfasst:

1. die Implementierung des Dijkstra- und des Floyd-Warshall Algorithmus'
2. einfache JUnit-Tests, die die Methoden überprüfen und
3. JUnit-Tests, die die Algorithmen überprüfen:
   - Testen Sie für *graph3* in graph3.gka dabei Floyd-Warshall gegen Dijkstra und geben Sie den kürzesten Weg, sowie die Anzahl der Zugriffe auf den Graphen an
   - eine allgemeine Konstruktion eines gerichteten Graphen für eine vorgegebene Anzahl von Knoten und Kanten mit beliebigen, aber unterschiedlichen, nicht-negativen Kantenbewertungen
   - Implementierung eines gerichteten Graphen *BIG* mit 100 Knoten und etwa 2500 Kanten. Beschreiben Sie die Konstruktion von *BIG* und weisen Sie für zwei nicht-triviale Knotenpaare die kürzesten Wege nach.
   - Lassen Sie bitte beide Algorithmen auf dem Graphen *BIG*, die kürzesten Wege berechnen und vergleichen diese.
4. die Beantwortung der folgenden Fragen:
   a) Bekommen Sie für einen Graphen immer den gleichen kürzesten Weg? Warum?
   b) Was passiert, wenn der Eingabegraph negative Kantengewichte hat?
   c) Wie allgemein ist Ihre Konstruktion von *BIG*, kann jeder beliebige, gerichtete Graph erzeugt werden?
   d) Wie testen Sie für *BIG*, ob Ihre Implementierung den kürzesten Weg gefunden hat?
   e) Wie müssten Sie Ihre Lösung erweitern, um die Menge der kürzesten Wege zu bekommen?
   f) Wie müssten Sie Ihre Lösung erweitern, damit die Suche nicht-deterministisch ist?

Vorstellung und Abgabe der Lösungen der Teams in am Bei der Abgabe des Protokolls bitte im Subjekt der email folgendes angeben: GKA-Praktikumsgruppe-TeamNr.

GKAP/01	14.11.
GKAP/02	21.11.
GKAP/03	28.11.
GKAP/04	29.11.

Der Zusatzaspekt betrifft die Qualität der Tests; also Ausagekraft, Abdeckung und Dokumentation der Tests

3 Flußprobleme

Zwei Algorithmen für Flußprobleme sollen implementiert werden:
Ford und Fulkerson sowie Edmonds und Karp, die beide auf den gleichen Graphen arbeiten. Eine direkte Beschreibung von Ford und Fulkerson finden Sie in den Folien. Recherchieren Sie eine geeignete Beschreibung des Algorithmus' von Edmonds und Karp.
Beide Algorithmen brauchen für die GKA-Competition Möglichkeiten zur Laufzeitmessung, nämlich die Zeitdifferenz vor dem Start und nach dem Ende des Algorithmus.

Die Aufgabe umfasst:

1. die Implementierung der beiden Algorithmen
2. eine (kurze) Darstellung des Algorithmus' von Edmonds und Karp und die Unterschiede zu Ford-Fulkerson
3. einfache JUnit-Tests, die die Methoden überprüfen und
4. JUnit-Tests, die die Algorithmen überprüfen:
   - Testen Sie für *Graph*4 dabei Ford und Fulkerson sowie Edmonds und Karp und geben Sie den maximalen Fluss, sowie die Laufzeit der beiden Algorithmen an.
   - Implementierung eines Netzwerks (gerichteter, gewichteter Graph mit Quelle und Senke) *BigNet_Gruppe_TeamNr* mit 50 Knoten und etwa 800 Kanten. Beschreiben Sie die Konstruktion von *BigNet_Gruppe_TeamNr* und erläutern sie, inwiefern die Konstruktion ein Netzwerk liefert.
   - Lassen Sie bitte beide Algorithmen auf dem Netzwerk *BigNet_Gruppe_TeamNr*, den maximalen Fluss berechnen und vergleichen diese.
   - Benutzen Sie bitte unterschiedlich große Big-Net-Graphen mit 800 Knoten und 300.000 Kanten und mit 2.500 Knoten und 2.000.000 Kanten. Und lassen Sie bitte Ihre die Algorithmen jeweils 100 durchlaufen.
   - Machen Sie bitte Ihre Testgraphen für die anderen Teams verfügbar.
5. die Beantwortung der folgenden Fragen:
   a) Welcher Algorithmus/welche Implementierung ist schneller? Wie schnell für die Netzwerke Ihrer Praktikumsgruppe?
   b) Was haben Sie unternommen, um eine bessere Laufzeit zu erreichen?
   c) Lässt sich die Laufzeit Ihrer Implementierung durch andere Datenstrukturen verbessern?
   d) Was passiert, wenn Sie nicht-ganzzahlige Kantengewichte wählen?
   e) Was passiert bei negativen Kantengewichten?

Vorstellung und Abgabe der Lösungen	der Teams in	am
	GKAP/01	5.12.
	GKAP/02	12.12.
	GKAP/03	19.12.
	GKAP/04	20.12.

Der Zusatzaspekt betrifft die Schnelligkeit, also die Implementierung mit der kürzesten Laufzeit bei gegeben Graphen.

*