

Praktikumsaufgaben für GKA

SoSe 2015

9. April 2015

J. Padberg

Aufgaben

Praktikum 1	3
Praktikum 2	5

Allgemeines zu allen Aufgaben

Die Aufgabenstellung ist aus folgenden Gründen nicht ganz genau spezifiziert:

- Sie sollen einen Entwurfsspielraum haben. Das heißt, Sie können relativ frei entscheiden, wie Sie die Aufgabe lösen, allerdings sollten Sie Ihre Entscheidungen bewusst fällen und begründen können. Insofern gibt es auch keine Musterlösung.
- Durch die unterschiedlichen Lösungen können Sie von Ihren Kommilitonen noch lernen und das *Structured Walk-Through* bleibt spannend.

Darüber hinaus dürfen Sie gerne unterschiedliche Quellen nutzen, aber nur wenn Sie diese auch angeben. Sie sollen auch nicht unbedingt alles selber programmieren, nutzen Sie den vorgegeben Datentyp oder gerne auch andere Libraries.

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

- Beispielgraphen für das Praktikum
- Links zu verschiedenen Libraries

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

1. Implementierung 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 und
- hinreichende Testfälle in JUnit und ihre Kommentierung.

2. Schriftliche Erläuterung Ihrer Lösung (Lösungsdokumentation)

- Bitte geben Sie folgende Daten im Kopf Ihrer Lösungsdokumentation an:
 - **Team:** Teamnummer sowie die Namen der Teammitglieder

- **Aufgabenaufteilung:**
 - (a) Aufgaben, für die Teammitglied 1 verantwortlich ist;
Dateien, die komplett/zum Teil von Teammitglied 1 implementiert/bearbeitet wurden
 - (b) Aufgaben, für die Teammitglied 2 verantwortlich ist;
Dateien, die komplett/zum Teil von Teammitglied 2 implementiert/bearbeitet wurden
- **Quellenangaben:** Angabe von wesentlichen Quellen, z.B. Web-Seiten/Bücher, von denen Quellcode/Algorithmen übernommen wurden , Namentliche Nennung von Studierenden der HAW, von denen Quellcode übernommen wurde
- **Bearbeitungszeitraum:** Datum und Dauer der Bearbeitung an der Aufgabe von allen Teammitgliedern und Angabe der gemeinsamen Bearbeitungszeiten
- **Aktueller Stand:** Welche Teile der Software sind fertig inklusive Tests, welche sind fertig, aber noch nicht getestet, welche müssen noch implementiert werden
- kurze Beschreibung der Algorithmen und
- Datenstrukturen
- wesentliche Entwurfsentscheidungen ihrer Implementierung
- Umsetzung der Aspekte der Implementierung
- umfassende Dokumentation der Testfälle, wobei die Abdeckung der Testfälle diskutiert werden soll

3. Schriftliche Bearbeitung des jeweiligen Theorieteils

Es gibt vier Praktikumsaufgaben, die jeweils unterschiedliche Implementierungsaspekte haben.

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

Weitere Details werden in der jeweiligen Aufgabenstellung angegeben.

Eine Praktikumsaufgabe gilt als erledigt, wenn

1. Sie die Lösungsdokumentation und die Bearbeitung des Theorieteils am Praktikumsanfang abgegeben haben,
2. Sie im Praktikum Ihre Implementierung vorgestellt haben und diese von mir (mindestens) als ausreichend anerkannt wurde oder
3. wenn die verbesserte Lösungsdokumentation mitsamt Bearbeitung der gestellten Aufgaben spätestens nach 6 Tagen bei mir als PDF vorliegt.

Aufgabe 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:

Dabei ist folgendes Format (in etwa die EBNF) zu verwenden:

```
[ "#directed" ], [ " #attributed" ], [ " #weighted" ];  
node1, [ ":" attribute1 ], [ ", " node2, [ ":" attribute2 ], [ "::" weight ] ] ;
```

node1 und node2 sind Zeichenketten

attribute1, attribute2 und weight sind Zahlen

Die Aufgabe umfasst:

- die Einarbeitung in das Paket JGraphT,
- das Einlesen und 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 gegebenen *.graph*-Dateien
- die Bearbeitung von Theorieteil 1

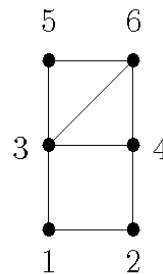
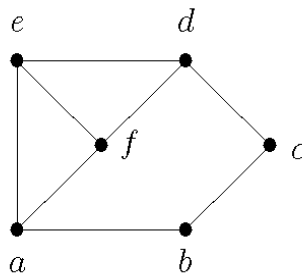
Vorstellung und Abgabe der Lösungen	der Teams in	am	letztendlich
	GKAP/01	27.4.	3.5.
	GKAP/02	16.4.	22.4.
	GKAP/03	23.4.	29.4.
	GKAP/04	9.4.	15.4

Wenn Sie sich wegen dieser Aufgabe per email an mich wenden, bitte geben Sie im Betreff die Teamnummer und die Aufgabennummer an.

Theorieteil 1

Aufgabe I :

Sind die beiden folgenden Graphen isomorph? Geben Sie entweder einen Isomorphismus an, oder begründen Sie, warum keiner existiert.



Aufgabe II :

Existiert ein schlichter Graph mit fünf Knoten und den folgenden Knotengraden?

Wenn ja, wie groß ist die Anzahl der Kanten?

Falls möglich, zeichnen Sie einen Graphen mit den gegebenen Eigenschaften.

1. 3, 3, 3, 3, 2
2. 1, 2, 3, 4, 4
3. 0, 1, 2, 2, 3
4. 1, 2, 3, 4, 5

Aufgabe III :

Ein vollständiger, bipartiter Graph $K_{n,m}$ hat eine Partitionierung X und Y mit $|X| = n$ und $|Y| = m$.

1. Geben Sie $K_{1,1}$, $K_{2,2}$ und $K_{3,3}$ und die Anzahl der Kanten an.
2. Bitte bestimmen Sie die Anzahl von Kanten in vollständigen, bipartiten Graphen $K_{n,n}$.
3. Beweisen Sie bitte diesen Zusammenhang.

Aufgabe 2: Optimale Wege

In dieser Teilaufgabe sollen der Dijkstra- und der A*-Algorithmus implementiert werden. Dabei sollen größere Graphen (größer hinsichtlich Kanten- und Knotenanzahl) zum Vergleich genutzt werden.

Die Aufgabe umfasst:

1. aufbauend auf Aufgabe (1) zwei Algorithmen für optimale Wege Dijkstra und A*, die beide auf ungerichteten Graphen arbeiten, zu implementieren. Dijkstra ignoriert gegebenenfalls die Attribute (i.e. Heuristik-Werte), aber A* benötigt diese. Als Ergebnis soll einer der kürzesten Wege und dessen Länge, sowie die Anzahl der Zugriffe auf den Graphen ausgegeben werden.
2. einfache JUnit-Tests, die die Methoden überprüfen und
3. JUnit-Tests, die die Algorithmen überprüfen:
 - Testen Sie für die gegebenen Graphen Dijkstra und geben Sie den kürzesten Weg, sowie die Anzahl der Zugriffe auf den Graphen an
 - Testen Sie mit Graph3 (erstellt mit Hilfe von <http://www.luftlinie.org/>), indem der kürzeste Weg – von Husum, – von Minden und – von Münster nach Hamburg gesucht werden soll. Testen sie dabei A* gegen Dijkstra und geben Sie jeweils den kürzesten Weg, sowie die Anzahl der Zugriffe auf den Graphen an.
 - Überlegen Sie sich eine allgemeine Konstruktion eines ungerichteten Graphen für eine vorgegebene Anzahl von Knoten und Kanten mit beliebigen, aber unterschiedlichen, nicht-negativen Kantengewichten.
 - Darauf aufbauend konstruieren Sie bitte Graphen, die eine monotone und nicht überschätzende Heuristik haben.
Beispielsweise können den Knoten zufällig natürliche Zahlen zugeordnet werden. Die Distanz zum Zielknoten liefert dann die Heuristik. Die Kanten bekommen auch zufällige Werte, die größer als die Distanz zwischen den beiden Knoten sind.
 - Implementieren Sie diese Konstruktion von Graphen und speichern Sie das Ergebnis. Lassen Sie sich mehrere kleine Graphen erzeugen und testen Sie damit ihre beiden Algorithmen, indem Sie bitte mit Dijkstra und A* jeweils den kürzesten Weg von einem Startknoten zu einem Zielknoten berechnen.
 - Erzeugen Sie bitte einen ungerichteten Graphen *BIG* mit 100 Knoten und etwa 6000 Kanten und lassen Sie bitte beide Algorithmen auf dem Graphen *BIG*, die kürzesten Wege zwischen verschiedenen Knoten berechnen und vergleichen Sie deren Länge sowie die Anzahl der Zugriffe auf den Graphen.

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, kann jeder beliebige, ungerichtete 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?

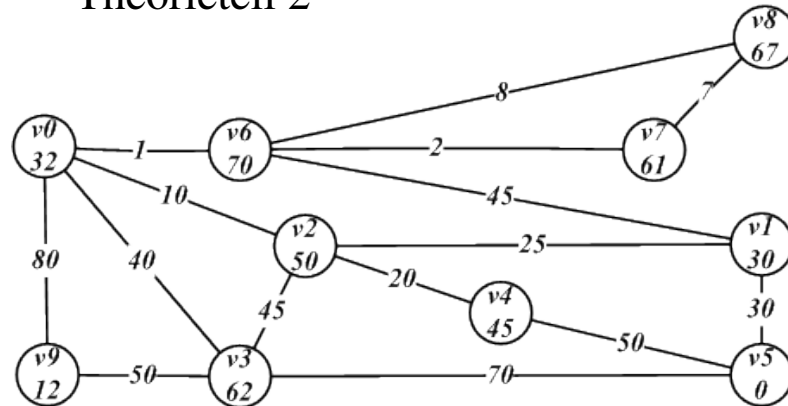
5. und die Bearbeitung von Theorieteil 2.

Vorstellung und Abgabe der Lösungen	der Teams in	am	letztendlich
	GKAP/01	22.5.	28.5.
	GKAP/02	7.5.	13.5.
	GKAP/03	21.5.	27.5.
	GKAP/04	30.4.	6.5.

Wenn Sie sich wegen dieser Aufgabe per email an mich wenden, bitte geben Sie im Betreff die Teamnummer und die Aufgabennummer an.

Theorieteil 2

Aufgabe IV :



1. Berechnen Sie bitte mit Hilfe des Dijkstra-Algorithmus den kürzesten Weg von v_0 nach v_5 . Die Heuristik ignorieren Sie bitte.
2. Berechnen Sie bitte mit Hilfe des A*-Algorithmus den kürzesten Weg von v_0 nach v_5 . Die Heuristik ist fest und im Knoten angegeben.
3. Was stellen Sie im Vergleich fest?

Aufgabe V :

In Nord-Amerika werden bestimmte Fernsehkanäle den Fernsehstationen so zugeteilt, dass niemals zwei Stationen, die weniger als 150 Meilen voneinander entfernt sind, denselben Kanal verwenden.

Wieviele verschiedene Kanäle werden dann für die sechs Stationen benötigt, deren Entfernungen voneinander in der folgenden Tabelle gegeben sind?

	1	2	3	4	5	6
1	-	85	175	200	50	100
2	85	-	125	175	100	160
3	175	125	-	100	200	250
4	200	175	100	-	210	220
5	50	100	200	210	-	100
6	100	160	250	220	100	-

Aufgabe VI :

Ein Graph mit $\chi(G) = k$ heißt kritisch k -chromatisch, wenn er sich durch Entfernen einer beliebigen Kante der chromatische Index von G $\chi(G) = k$ verringert, also wenn gilt:

$$\chi(G \setminus e) = k - 1$$

1. Geben Sie bitte für $k = 2, 3, \dots$ eine Familie von kritisch k -chromatischen Graphen an.
2. Geben Sie bitte für $n = 3, 5, \dots$ eine Familie von kritisch 3-chromatischen Graphen mit n Knoten an.