**Fachbereich Informatik**
Dr. Michael Haupt
Visar Januzaj
Johannes Kinder
*Sommersemester 2010*

TECHNISCHE UNIVERSITÄT DARMSTADT

# Grundlagen der Informatik 2

## Praktikumsaufgabe 2

Abgabe: 08.06.2010, 12:00

**Puzzleproblem.** Implementieren Sie den A* Algorithmus, um das in den Übungen behandelte Puzzleproblem zu lösen. Verwenden Sie folgende Heuristik für die Berechnung der Entfernungskosten:

- Die Entfernung vom Startknoten ist die Anzahl der benötigten Schritte bis zum aktuellen Zustand, z.B. der unmittelbare Nachbarzustand hat die Entfernung 1, usw.
- Die Entfernung zum Zielknoten ist die Anzahl der Puzzleteile, die an den falschen Positionen sind, z.B.

```
 1 2 3   1 2 3
 4 5 6   4 5 6
 8 7 #   7 8 #
(Start)  (Ziel)
```

  The Entfernung zwischen Start und Ziel wäre 2, da 7 und 8 an falschen Positionen sind.(Achtung: Start ist unlösbar.)

*Hinweise*: Achten Sie beim Testen darauf, dass bestimmte Zustände möglicherweise nicht lösbar sind. Versuchen Sie die Vorgängerzustände nicht zu betrachten, da sie ja schon bearbeitet wurden.

**JGraphT.** Ziel dieser Aufgabe ist auch sich mit *JGraphT* vertraut zu machen.

- Laden Sie *JGraphT* herunter (`http://jgrapht.sourceforge.net/`).
- Setzen Sie den Pfad korrekt, z.B. in Eclipse: *Project, Properties, Java Build Path, Libraries, Add External JARs*.
- *JGraphT* Javadocs sind hier zu finden `http://jgrapht.sourceforge.net/javadoc/`.
- Ein hilfreiches Beispiel `jgrapht-0.8.1/src/org/jgrapht/demo/HelloJGraphT.java`

Verwenden Sie *DirectedGraph*$< V, E >$ bzw. *SimpleDirectedGraph*$< V, E >$. Wie auch den Quelltext-Vorgaben zu entnehmen ist, sind die Knoten ($V$) vom Typ `PuzzleMove` und die Kanten ($E$) vom Typ `Integer`.

**Hinweise.** Import der Quelltext-Vorgaben: Laden Sie die Quelltext-Vorgaben für Praktikumsaufgabe 2 aus dem moodle-Portal herunter (Project2.zip). Importieren Sie das Projekt in Eclipse (File, Import, General, Existing Projects Into Workspace, Select Archive File, Browse, Project1.zip auswählen, Finish).

**Zu implementieren.** Wie immer sind die zu bearbeitenden Stellen im Code mit **TODO:** markiert. Im einzelnen:

1. Implementieren Sie den Rumpf der Klasse `Puzzle` in `Puzzle.java`, den Sie benötigen werden, um die Daten vorzubereiten und anschließend zu bearbeiten.
2. `PuzzleAStar.java` - Implementieren Sie alle vorgegebenen Methoden als auch andere Hilfsmethoden. Die Klasse `PuzzleAStar` implementiert den A* Algorithmus, der für die Lösung des Puzzles verwendet wird. Die Kernmethode ist `findDestination()`.
3. `PuzzleMove.java` - Die Klasse `PuzzleMove` beschreibt den Typ der Knoten im Graphen und implementiert wichtige Methoden, die vom A* Algorithmus verwendet werden. Verwenden Sie `methodOne()` für die Implementierung der vorgegebenen Heuristik. Implementieren Sie alle vorgegebenen Methoden. Eigene Hilfsmethoden sind auch hier erlaubt.
4. `PuzzleInputOutput.java` - Die Klasse `PuzzleInputOutput` bietet die nötigen Methoden für das Lesen der Start- und Zielzuständen. Das Eingabeformat der Puzzlezuständen ist in den Testdateien beschrieben, z.B. `test_0.txt` in *Project2.zip*. Die erste Zeile gibt die Größe (NxN) der Puzzle an, z.B. N=3. Danach folgen die Puzzlezustände, welche mit einer Linie der Form "`---`" getrennt werden. Das leere Feld wird durch # definiert. Implementieren Sie alle vordefinierten Methoden.
5. Testen Sie Ihre Implementierungen mit den vorgegebenen Testfällen. Verwenden Sie auch selbst geschriebene Testfälle.*Hinweis:* Öffnen Sie `PuzzleTest.java` im Editor. Wählen Sie im Menü: *Run, Run As, JUnit Test*.
6. Beantworten Sie folgende Fragen im Kopf von `Puzzle.java` an den mit **TODO:** markierten Stellen:

a) Wählen und implementieren Sie zwei zusätzliche Heuristiken für die Berechnung der Entfernungskosten. Sie können z.B. die in der Übung vorgestellte Heuristik verwenden, oder Ähnliches. Verwenden Sie TPTP um `Puzzle.main()` zu profilen.

   i) Welche Heuristiken haben Sie gewählt? Wieso? Diskutieren Sie sie. Welche Heuristik war die beste?

   ii) Welche Größe (NxN) hatten die verwendeten Puzzles?

Tragen Sie mindestens 5 verschiedene, aussagekräftige Resultate in der Tabelle ein.

**Abgabe.** Reichen Sie alle Ihre fertig bearbeiteten Dateien `Puzzle.java`, `PuzzleAStar.java`, `PuzzleMove.java` und `PuzzleInputOutput.java` bis zum Abgabetermin in moodle ein. Die Abgabe ist in Gruppen von bis zu drei Personen möglich. **Ein** Mitglied der Gruppe reicht die Lösung zusammen mit einem Kommentar der folgenden Form ein: *Die Übung wurde bearbeitet von der Gruppe X, Y, Z; die Abgabe erfolgte durch X*. **Jedes andere** Mitglied der Gruppe reicht *anstelle* der Lösung nur diesen (identischen) Kommentar ein.

Bitte stellen Sie sicher, dass Sie und ggf. Ihre Gruppenmitglieder diese Regeln befolgen; wird weder eine Lösung noch der obenstehende Kommentar eingereicht, so vergibt der Tutor auch keine Punkte!

***Hinweis:*** *Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Mit der Abgabe einer Programmieraufgabe bestätigen Sie, dass Sie bzw. Ihre Lerngruppe die alleinigen Autoren sind.*