Grundlagen der Künstlichen Intelligenz

Prof. Dr. M. Helmert
Dr. M. Wehrle, S. Sievers
Frühjahrssemester 2015

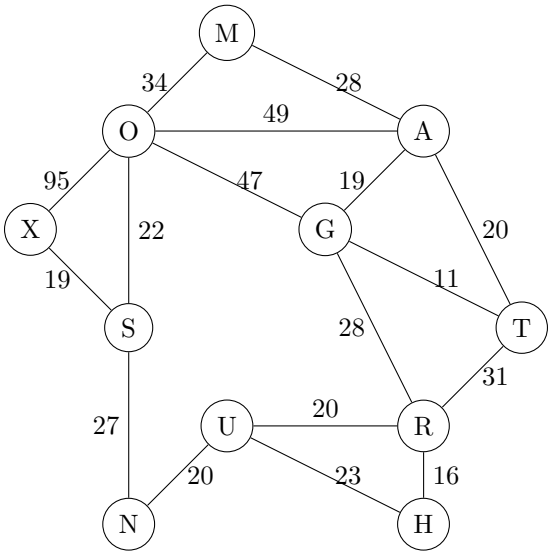
Universität Basel
Fachbereich Informatik

Übungsblatt 5

Abgabe: 10. April 2015

Aufgabe 5.1 (2+2 Punkte)

Betrachten Sie die folgende Strassenkarte:



Die Luftliniendistanzen zwischen Atlantis (A) und den anderen Städten sind in der folgenden Tabelle gegeben:

Stadt	**Distanz**
G	14
H	50
M	25
N	36
O	42
R	38
S	42
T	13
U	51
X	47

Betrachten Sie die Heuristik, die jedem Zustand s die Luftliniendistanz der aktuellen Stadt in s nach Atlantis (A) zuordnet.

(a) Zeichnen Sie den Suchbaum, der vom A*-Algorithmus (ohne Reopening) für die Suche eines kürzesten Weges von Hel (H) nach Atlantis (A) erzeugt wird. Geben Sie an, in welcher Reihenfolge die Knoten expandiert werden, und annotieren Sie jeden Knoten mit seinen f-, g-, und h-Werten.

(b) Zeichnen Sie den Suchbaum, der von gieriger Bestensuche (ohne Reopening) für die Suche eines Weges von Hel (H) nach Atlantis (A) erzeugt wird. Geben Sie an, in welcher Reihenfolge die Knoten expandiert werden. Vergleichen Sie das Ergebnis mit (a).

Aufgabe 5.2 (3+3+2 Punkte)

In dieser Aufgabe soll A* implementiert werden. Wir erwarten, dass Sie Ihre Implementierung selbständig, das heisst ohne Anwendung von fremdem Code z.B. aus dem Internet erstellen. Uns ist bewusst, dass Programmieraufgaben aufwendiger sind als die üblichen theoretischen Aufgaben und helfen bei technischen Schwierigkeiten und Verständnisproblemen gerne weiter. Bitte wenden Sie sich dazu *mit genügend zeitlichem Abstand zum Abgabetermin* an Cedric Geissmann, Silvan Sievers oder Martin Wehrle.
Diese Aufgabe baut auf Aufgabe 3.2 auf. Sie können hierbei als Grundlage Ihre eigene Lösung für Aufgabe 3.2, eine Lösung Ihrer Kommilitonen oder die Musterlösung von der Webseite verwenden.

(a) Erweitern Sie das Interface `StateSpace` um eine Methode `public int h(State s)`, die für einen Zustand einen heuristischen Wert zurückliefern soll.

Implementieren Sie die *Gap-Heuristik* für das Pfannkuchenproblem. Die Gap-Heuristik eines Zustands ist definiert als die Anzahl *Lücken* in dem Zustand. Wir definieren hierbei, dass zwischen zwei benachbarten Positionen im Pfannkuchenstapel eine Lücke existiert, wenn die beiden Pfannkuchen an diesen Positionen "nicht benachbarte Grössen" haben, also die Differenz zwischen den beiden Grössen mindestens 2 beträgt. (Beispiel: Pfannkuchen der Grösse 4 auf Pfannkuchen der Grösse 2 ist eine Lücke; Pfannkuchen der Grösse 4 auf Pfannkuchen der Grösse 3 oder 5 ist keine Lücke.) Als Lücke zählt auch, wenn der unterste Pfannkuchen im Stapel nicht der grösste ist.

Beispielzustand s Zielzustand

Beispiel: Von unten nach oben gelesen lässt sich der Beispielzustand s als $\langle 4, 6, 1, 5, 2, 3\rangle$ repräsentieren. Er weist 5 Lücken auf: unterhalb von 4 (weil nicht der grösste Pfannkuchen zuunterst liegt), zwischen 4 und 6, zwischen 6 und 1, zwischen 1 und 5, und zwischen 5 und 2. Daher ist $h(s) = 5$.

(b) Implementieren Sie A* ohne Reopening. Leiten Sie für die Implementierung von der abstrakten Klasse `SearchAlgorithmBase` ab, die Sie auf der Webseite finden, und implementieren Sie die Methode `run()`. Sie dürfen hierzu Datenstrukturen (wie z.B. eine Prioritätswarteschlange) aus der Standardbibliothek Ihrer gewählten Programmiersprache verwenden. Lassen Sie sich die Anzahl der generierten Suchknoten ausgeben.

(c) Wenden Sie die Implementierung aus (b) an. Generieren Sie sich hierzu mindestens 5 Beispiel-Inputs und testen Sie Ihre Implementierung auf diesen Problemen. Versuchen Sie, Beispiel-Inputs von unterschiedlicher Schwierigkeit für Ihre Implementierung zu finden. Verwenden Sie ein Zeitlimit von 10 Minuten pro Suche (z.B. unter Linux können Sie solch ein Zeitlimit mit `ulimit -t 600` für die aktuelle Shell-Sitzung setzen).

Implementieren Sie die Null-Heuristik, d.h., die Heuristik, die 0 für jeden Zustand zurückliefert. Vergleichen Sie die Ergebnisse von A* mit der Gap-Heuristik und der Null-Heuristik. Geben Sie jeweils Laufzeit und die Anzahl der generierten Suchknoten aus, wenn der Algorithmus innerhalb der Zeitgrenze terminiert.

Reichen Sie bitte den Code inklusive Hinweisen zu Kompilierung und Aufruf bei courses ein (sofern nicht selbsterklärend).

Die Übungsblätter dürfen in Gruppen von zwei Studierenden bearbeitet werden. Bitte schreiben Sie beide Namen auf Ihre Lösung.