Grundlagen der Künstlichen Intelligenz (CS 205)

Prof. Dr. M. Helmert
Dr. M. Wehrle
Frühjahrssemester 2014

Universität Basel
Fachbereich Informatik

Übungsblatt 3

Abgabe: 28. März 2014

Aufgabe 3.1 (2+1+1 Punkte)

Betrachten Sie das 8-Puzzle (analog zum 15-Puzzle aus der Vorlesung) im folgenden Anfangszustand `init`.

1	2	3
4	5	
7	8	6

Der einzige Zielzustand sei folgendermassen gegeben.

1	2	3
4	5	6
7	8	

Wie beim 15-Puzzle seien die möglichen Aktionen folgendermassen modelliert: Das leere Feld kann nach oben (*up*), links (*left*), rechts (*right*) und unten (*down*) verschoben werden, wenn es sich nicht bereits am entsprechenden Rand befindet.

(a) Wenden Sie tiefenbeschränkte Suche mit einer maximalen Tiefe von 3 auf den Anfangszustand an (d.h., simulieren Sie den Aufruf `depth_limited_search(init(), 3)`). Geben Sie hierzu den Suchbaum und die Expansionsreihenfolge der Knoten an. Die Reihenfolge, in der Nachfolgezustände berechnet werden, sei durch $up < left < right < down$ gegeben (d.h., für einen gegebenen Zustand wird stets zuerst dessen Nachfolger bzgl. *up* und als letztes dessen Nachfolger bzgl. *down* von `succ` berechnet).

(b) Was passiert, wenn keine Tiefenbeschränkung gegeben ist? Begründen Sie Ihre Antwort.

(c) Nehmen Sie an, die Reihenfolge der Aktionen sei nun im Vergleich zu a) invertiert, d.h. $down < right < left < up$. Simulieren Sie für diese Reihenfolge noch einmal den Aufruf `depth_limited_search(init(), 3)`. Diskutieren Sie das Ergebnis im Vergleich zu a).

Aufgabe 3.2 (6+2 Punkte)

Bei dieser Aufgabe handelt es sich um eine Programmieraufgabe. 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 Lukas Beck oder Martin Wehrle.

(a) Implementieren Sie iterative Tiefensuche für den Puzzle- oder den Pancake-Zustandsraum. Sie können hierzu Ihre Zustandsraum-Implementierung des letzten Übungszettels verwenden. Alternativ können Sie die Beispiellösung (in Java) für den Puzzle-Zustandsraum von der Webseite verwenden. Leiten Sie für die Implementierung der iterativen Tiefensuche von der abstrakten Klasse `SearchAlgorithmBase` ab, die Sie ebenfalls auf der Webseite finden, und implementieren Sie die Methode `run()`. Lassen Sie sich die Anzahl der generierten Suchknoten in jeder Suchebene sowie die Gesamtzahl der generierten Suchknoten ausgeben.

(b) Generieren Sie sich mindestens 3 Beispiel-Inputs für das (Puzzle- oder Pancake-) Problem, 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). Geben Sie jeweils die Anzahl der generierten Suchknoten für jede Suchebene sowie die Gesamtzahl 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.