Übung 9, Aufgabe 1

Sebastian Schiener

1. Krisensituation an der FH Konstantinopel (8 + 3 + 3 + 6 Punkte)

Bis jetzt hatte Prof. Mayar sozusagen das Exklusivrecht auf alle Lehrinhalte im Bereich der Algorithmik. Die Betonung liegt auf „bis jetzt". Im Zuge des Ausbaus der Fachhochschule Konstantinopel wurden in allen Bereichen neue Leute eingestellt, so auch in der Informatik.

Mit Anfang des Wintersemesters 2012 wurde ein neuer Hochschullehrer, Prof. Oktobar, eingestellt, der sich selbst gerne als den ungekrönten Optimierungskaiser bezeichnet. Im Gegensatz zu Prof. Mayar ist er davon überzeugt, dass Heuristiken die Laufzeiten von verschiedensten Algorithmen entscheidend verbessern können. Mittlerweile ist ein regelrechter „Religionskrieg" entbrannt. Um seine Ansicht gegenüber Prof. Mayar eindrucksvoll zu untermauern, verwendet Prof. Oktobar das „8-Puzzle" wie in der Vorlesung bzw. im Seminar vorgestellt. Trotz der äußerst beeindruckenden Unterschiede der Suche mit und ohne Heuristiken ist Prof. Mayar nicht überzeugt bzw. er vertraut den Ergebnissen von Prof. Oktobar nicht. Er hat daher beschlossen, das „8-Puzzle" Problem selber zu implementieren und die Ergebnisse der Suche ohne, mit Heuristik 1 und mit Heuristik 2 zu bestimmen.

Leider ist Prof. Mayar im Zuge der baldigen Weihnachtsnachbereitungen (Geschenke auspacken, Baum abschmücken) ziemlich im Stress. Helfen Sie also Prof. Mayar, indem Sie das „8-Puzzle"-Problem implementieren und die Daten von Prof. Oktobar verifizieren.

Folgende Teilaufgaben sind zu lösen:

a) „8-Puzzle" ohne Heuristik
b) „8-Puzzle" mit Heuristik *h1*: Anzahl der Kästchen in falscher Position
c) „8-Puzzle" mit Heuristik *h2*: Summe der Distanzen zu ihrer Zielposition
d) Tabellarische Ausgabe der Anzahl der untersuchten Knoten und des effektiven Branching-Faktors für die Varianten a, b, c

Vergessen Sie nicht auf die Lösungsidee!

Hinweis:

- Testen Sie die Klassen ausführlich und vollkommen automatisch
- Achten Sie bei der Implementierung auf die Wiederverwendbarkeit Ihrer Klassen
- Halten Sie sich an die Programmierkonventionen!

Lösungsidee:

Ich habe die Klasse EightPuzzle die aus 9 PuzzlePieces besteht.
Jedes Puzzlepiece hat 2, 3 oder 4 Nachbarn und einen Value.
Das EightPuzzle ist eine PuzzlePiece-Matrix. Die Matrix kann ausgegeben werden und gibt in tabellarischer Form zusätzlich noch die Anzahl der PuzzlePieces in korrekter Position, die Tiefe, die ausgeführten Züge (auch die, die nicht zum Ergebnis führen) und den effektiven Branchingfaktor aus. Der Solver testet dann rekursiv alle Möglichenkeiten per iterative deepening aus, in dem er PuzzlePieces swappen und deswappen kann, dies probiert er immer für bis zu allen 4 Richtungen aus. Nach dem swappen prüft er immer, wieviele PuzzlePieces in korrekter Position stehen, wenn alle passen, returnt die Methode.