Aufgaben-Blatt: Das 3×3-Schiebe-Puzzle

Ein 3×3-Schiebe-Puzzle wird auf einem quadratischen Spielfeld der Seitenlänge 3 gespielt. Dieses Spielfeld ist in $3 \times 3 = 9$ quadratische Felder unterteilt. Auf 8 der Felder liegen quadratischen Kacheln der Seitenlänge 1, während ein Feld frei bleibt. Die 8 Kacheln sind mit den Zahlen von 1 bis 8 durchnummeriert. Abbildung 1 zeigt zwei verschiedene Stellungen des Schiebe-Puzzles.

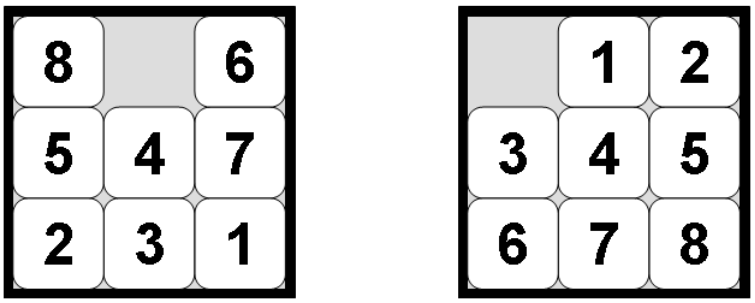

Abbildung 1: Das 3×3-Schiebe-Puzzle.

Die Aufgabe bei diesem Schiebe-Puzzle besteht darin, die auf der linken Seite der Abbildung gezeigte Konfiguration in die auf der rechten Seite der Abbildung gezeigte Konfiguration zu überführen. Dabei sind die folgenden Operationen erlaubt:

(a) Falls eine Kachel links neben dem freien Feld liegt, kann diese nach rechts auf das freie Feld geschoben werden.

(b) Falls eine Kachel rechts neben dem freien Feld liegt, kann diese nach links auf das freie Feld geschoben werden.

(c) Falls eine Kachel unter dem freien Feld liegt, kann diese nach oben auf das freie Feld geschoben werden.

(d) Falls eine Kachel über dem freien Feld liegt, kann diese nach unten auf das freie Feld geschoben werden.

Um einen Eindruck davon zu bekommen, wie sich dieses Puzzle lösen lässt, können Sie es auf der Seite

http://mypuzzle.org/sliding

ausprobieren. Ziel dieser Aufgabe ist die Entwicklung eines SetlX-Programms, das eine Lösung des oben gezeigte Puzzles berechnen kann. Laden Sie dazu von meiner Seite das Programm

https://github.com/karlstroetmann/Logik/blob/master/Aufgaben/Blatt-6/sliding-frame.stlx

herunter. Bei meinem Programm gehe ich davon aus, dass ein Zustand des Schiebe-Puzzles als eine Liste dargestellt wird, die 3 Listen der Länge 3 enthält. Jede dieser Listen stellt dabei eine Zeile des Schiebe-Puzzles dar. Das leere Feld wird dabei durch die Zahl 0 dargestellt. Die in der obigen Abbildung angegebenen Instanzen des Puzzles lassen sich daher durch die beiden Listen

```
start := [ [8, 0, 6],
           [5, 4, 7],
           [2, 3, 1]
         ];
```

beziehungsweise

```
goal  := [ [0, 1, 2],
           [3, 4, 5],
           [6, 7, 8]
         ];
```

darstellen. Sie können sich überlegen, dass es insgesamt $9! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot 8 \cdot 9$ verschiedene Konfigurationen des Schiebe-Puzzles gibt. Da diese Zahl bereits recht groß ist, verbietet es sich, die Relation, welche die verschiedenen Zustandsübergänge beschreibt, explizit als Menge darzustellen. Stattdessen können Sie diese Relation durch eine Funktion

`nextStates`(s)

darstellen, die für eine gegebene Konfiguration s des Schiebe-Puzzles die Menge aller der Konfigurationen berechnet, die in einem Schritt berechnet werden können. Die Funktion `findPath`, die in Zeile 30 der Datei `sliding-frame.stlx` definiert wird, wird nun in der Form

`findPath`(*start*, *goal*, *nextStates*)

aufgerufen. Hier ist *start* die Ausgangs-Konfiguration, *goal* ist die Ziel-Konfiguration und *nextStates* ist die Funktion, die für eine gegebene Konfiguration die Menge aller Konfigurationen berechnet, die in einem Schritt von dieser Konfiguration erreicht werden können. Ihre Aufgabe beschränkt sich darauf, die Definition der Funktion `nextStates` in Zeile 56 zu vervollständigen. Es ist sinnvoll, wenn Sie sich geeignete Hilfs-Prozeduren überlegen, mit deren Hilfe Sie diese Aufgabe lösen können. Sobald Sie die Prozedur `nextStates` korrekt implementiert haben, berechnet das Programm eine mögliche Lösung.

Hinweis: Auf meinem Rechner[1] benötigt das Programm knapp 22 Sekunden und belegt in der Spitze etwas mehr als 1 Gigabyte Speicher. Bei der berechneten Lösung wird 31 mal eine Kachel verschoben. Bei der Berechnung der Lösung werden insgesamt $181440 = 9!/2$ verschieden Konfigurationen untersucht.

[1] Dabei handelt es sich um eine iMac mit einem 2,7 GHz Intel Core i5 aus dem Jahr 2011.