DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Duale Hochschule Baden-Württemberg Mannheim

**Seminararbeit**

# Analyse der Lösbarkeit von Instanzen des 15-Puzzzles mit Implementierung

**Studiengang Informatik**

**Studienrichtung angewandte Informatik**


| | |
|---|---|
| Verfasser(in): | Kai Fischer, Max Stubenbord |
| Matrikelnummer: | 3683691, 5379506 |
| Kurs: | TINF18AI1 |
| Studiengangsleiter: | Prof. Dr. Holger Hofmann |
| Wissenschaftliche(r) Betreuer(in): | Prof. Dr. Karl Stroetmann |
| Bearbeitungszeitraum: | 24.03.2021 – 11.06.2021 |

# Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Titel "*Analyse der Lösbarkeit von Instanzen des 15-Puzzzles mit Implementierung*" selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum Kai Fischer, Max Stubenbord

# Danksagung

Hier können Sie eine Danksagung schreiben.

# Kurzfassung (Abstract)

Hier können Sie die Kurzfassung (engl. Abstract) der Arbeit schreiben. Beachten Sie dabei die Hinweise zum Verfassen der Kurzfassung.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Quelltextverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **AD** | Archiv für Diplomatik, Schriftgeschichte, Siegel- und Wappenkunde |
| **BMBF** | Bundesministerium für Bildung und Forschung |
| **DHBW** | Duale Hochschule Baden-Württemberg |
| **ECU** | European Currency Unit |
| **EU** | Europäische Union |
| **RDBMS** | Relational Database Management System |

# 1 Einleitung

## 1.1 Geschichte

Die erste bekannte Erscheinung des heute so bekannten Puzzels ist in den 1870'er Jahren in Amerika aufgetreten. Bisher ist man davon ausgegangen, dass der Erfinder des Puzzels der Amerikaner Sam Loyd ist, jedoch ist nach einer Untersuchung von Jerry Slocum and Dieter Gebhardt Sam Loyd gar nicht der echte Erfinder des 15-Puzzels. [4, 9] Demnach hat Sam Loyd die Idee nur gestohlen und als seine weiterverkauft.
`zitat? -> woher die information` Nichtsdestotrotz hat Sam Loyd es geschafft die Aufmerksamkeit der Öffentlichkeit auf das Puzzel zu lenken und somit das Interesse vieler geweckt.
Das Ziel des Puzzels ist es 15 Puzzelsteine numeriert von 1-15 auf einer 4 x 4 Ebene durch Verschiebung in seine Ursprüngliche sortierte Form zurückzubringen (vgl. Abb.1.1).

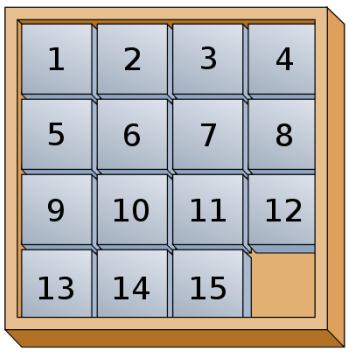


Abbildung 1.1: Zielzustand des 15 Puzzels

Der schnelle Ruhm des Puzzels ergab sich nun daraus, dass Sam Loyd ein Preisgeld von 1000 USD für denjenigen ausschrieb, der sein Puzzel lösen kann. Darauf folgte ein öffentlicher Ansturm auf das Puzzel, welches als „14-15 Puzzel" bekannt wurde, da lediglich die 14 und 15 vertauscht wurden. (Vgl. Abb.1.2)

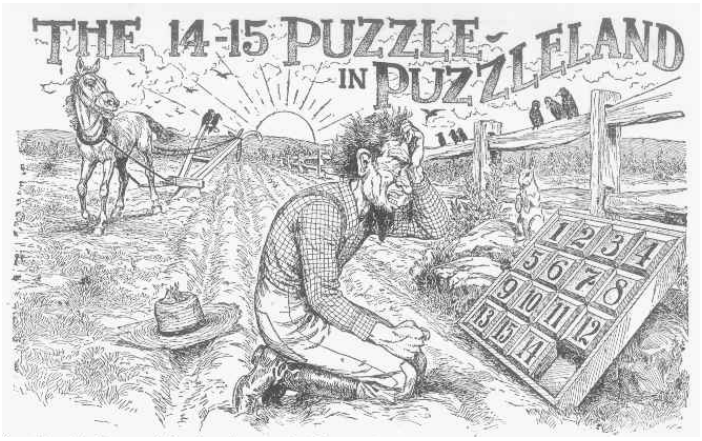


Abbildung 1.2: Illustration des 14-15 Puzzels von Sam Loyd
Quelle: [8, pp. 234-235]

Allerdings zeigte sich im Nachhinein, dass das Puzzel unlösbar ist, da es eine Transformation von einer geraden zu einer ungeraden Permutation erfordert.[1] Die Unlösbarkeit des Puzzels wurde erstmals von Wm. Woolsey Johnson und William E. Story im Jahre 1879 bei einer Veröffentlichung des American Journal of Mathematics bewiesen. [6] Somit gewann keiner das Preisgeld von 1000 USD. Das 15-Puzzel wurde daraufhin auch bei einer Illustration mit dem Titel „The Great Presidential Puzzle“ der US-Präsidentschaftswahl 1880 referenziert. Zu finden ist dies in der United States Library of Congress's Prints and Photographs division unter der Digitalen ID ppmsca. 15782. [2]

## 1.2 Aufgabenstellung

Hätte man damals bewiesen, dass das Puzzel aus der Einleitung von Sam Loyd nicht lösbar ist, wäre wohl vielen Menschen nächtelange Verzweiflung erspart geblieben.
Nun stellt sich Frage wann eine Instanz des Puzzels lösbar ist. Allgemein gesprochen ist eine Instanz eines 15 Puzzels dann lösbar, falls es eine Sequenz von zulässigen Zügen gibt, welche vom Startzustand zum Zielzustand führen.
Diese Annahme gilt als Zutreffend für genau die hälfte aller möglichen $16! \approx 2 \cdot 10^{13}$ Puzzel Kombinationen. [5, 3] Das Puzzel Problem ist ein klassischer Anwendungsfall wenn es um Modellierung von Algorithmen mit Heuristiken geht. Üblicherweise werden Algorithmen wie *A** oder *IDA** zur Lösung solcher Heurisiken genutzt.[1, 3, 7] Im Rahmen dieser Arbeit geht es nun darum bei gegebener Puzzelinstanz vorherzusagen, ob diese als Lösbar oder Unlösbar gilt.

Abgrenzung + Vorauswahl (was genau machen wir) => verweis auf stroeti: Wir beschaeftigen uns nicht, referenzierung auf skriptstelle, an der xy zu finden ist.

## 1.3 Vorgehen

Das Vorgehen der Arbeit ist dabei wiefolgt:

# 2 Grundlagen

## 2.1 Permutationen

## 2.2 Kontext Vorlesung + Abgrenzung

## 2.3 Sortieralgorithmen

Mit Ordnung?!

# 3 Implementation

## 3.1 Umsetzung

Zur Umsetzung wird zu erst eine Datenstruktur definiert, in welcher die Instanzen des 15-Puzzels ausgewertet werden. Hierfür werden Tupel verwendet.

## 3.2 Testing

# 4 Zusammenfassung

# Literaturverzeichnis


[1] *15 puzzle - Wikipedia*. https://en.wikipedia.org/wiki15_puzzle. (Accessed on 05/30/2021).

[2] *15–14–13. The great presidential puzzle*. http://loc.gov/pictures/resource/ppmsca.15782/. (Accessed on 05/30/2021).

[3] B. Bischoff et al. *Solving the 15-Puzzle Game Using Local Value-Iteration*. 2013. URL: https://mediatum.ub.tum.de/doc/1283911/1283911.pdf.

[4] Jerry Slocum & Dieter Gebhardt. *THE ANCHOR PUZZLE BOOK*. Art of Play, 2012.

[5] Edward Hordern. *Sliding Piece Puzzles*. New York: Oxford University Press, 1986. ISBN: 978-0-198-53204-0.

[6] Wm. Woolsey Johnson und William E. Story. „Notes on the "15"Puzzle“. In: *American Journal of Mathematics* 2.4 (1879), S. 397–404. ISSN: 00029327, 10806377. URL: http://www.jstor.org/stable/2369492.

[7] Richard E. Korf. „Depth-first iterative-deepening: An optimal admissible tree search“. In: *Artificial Intelligence* 27.1 (1985), S. 97–109. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(85)90084-0. URL: https://www.sciencedirect.com/science/article/pii/0004370285900840.

[8] S. Loyd und S. Sloan. *Sam Loyd's Cyclopedia of 5000 Puzzles Tricks and Conundrums with Answers*. Ishi Press International, 2007. ISBN: 9780923891787. URL: https://books.google.de/books?id=noWnPQAACAAJ.

[9] *Review of The 15 Puzzle*. https://www.cut-the-knot.org/books/Reviews/The15Puzzle.shtml. (Accessed on 05/30/2021).

# A Beispiel-Anhang: Testanhang

### Python Code zur Validierung der Lösbarkeit von Instanzen des 15-Puzzels

Quelltext A.1: Python Code zur Validierung der Lösbarkeit von Instanzen des 15-Puzzels

```
#!/usr/bin/env python
# coding: utf-8
start1 = {
    'data': ((13, 2, 10, 3),
             (1, 12, 8, 4),
             (5, 0, 9, 6),
             (15, 14, 11, 7)),
    'name': 'start1'}

start2 = {
    'data': ((0,  1,  2,  3),
             (4,  5,  6,  8),
             (14,  7, 11, 10),
             (9, 15, 12, 13)
             ),
    'name': 'start2'
}

start3 = {
    'data': ((6, 13, 7, 10),
             (8, 9, 11, 0),
             (15, 2, 12, 5),
             (14, 3, 1, 4)),
    'name': 'start3'
}


```

```
28 start4 = {
29     'data': ((3, 9, 1, 15),
30              (14, 11, 4, 6),
31              (13, 0, 10, 12),
32              (2, 7, 8, 5)),
33     'name': 'start4'
34 }
35
36 start5 = {
37     'data': ((1,  2,  3, 4),
38              (5,  6,  7, 8),
39              (9, 10, 11, 12),
40              (13, 15, 14, 0)),
41     'name': 'start5'
42 }
43
44 upperLeft = {
45     'data': ((0, 1, 2, 3),
46              (4, 5, 6, 7),
47              (8, 9, 10, 11),
48              (12, 13, 14, 15)),
49     'name': 'solved - Blank upper Left'
50 }
51
52 downRight = {
53     'data': ((1, 2, 3, 4),
54              (5, 6, 7, 8),
55              (9, 10, 11, 12),
56              (13, 14, 15, 0)),
57     'name': 'solved - Blank down Right'
58 }
59 upperRight = {
60     'data': ((1, 2, 3, 0),
61              (4, 5, 6, 7),
62              (8, 9, 10, 11),
```

```
63                (12, 13, 14, 15)),
64      'name': 'solved - Blank upper Right'
65  }
66  downLeft = {
67      'data': ((1, 2, 3, 4),
68               (5, 6, 7, 8),
69               (9, 10, 11, 12),
70               (0, 13, 14, 15)),
71      'name': 'solved - Blank down Left'
72  }
73  spirale = {
74      'data': ((1, 2, 3, 4),
75               (12, 13, 14, 5),
76               (11, 0, 15, 6),
77               (10, 9, 8, 7)),
78      'name': 'spirale Goal'
79  }
80
81  Starts = [start1, start2, start3, start4, start5]
82  Goals = [upperLeft, upperRight, downLeft, downRight, spirale]
83
84
85  def to_1d(Puzzle: tuple) -> list:
86      return [elem for tupl in Puzzle for elem in tupl]
87
88
89  def swap(idxA, idxB, Puzzle_1d):
90      Puzzle_1d[idxA], Puzzle_1d[idxB] = Puzzle_1d[idxB],
        ↪  Puzzle_1d[idxA]
91
92
93  def find_tile_1d(tile, State_1d):
94      n = len(State_1d)
95      for it in range(n):
96          if State_1d[it] == tile:
```

```
97              return it
98
99
100 def get_inversion_count(Puzzle: tuple) -> int:
101     working_1d_puzzle = to_1d(Puzzle)
102     count = 0
103     old_count = -1
104     while old_count != count:
105         old_count = count
106         for i in range(len(working_1d_puzzle)):
107             if working_1d_puzzle[i] != i:
108                 count += 1
109                 swap(i, find_tile_1d(i, working_1d_puzzle),
                    ↪  working_1d_puzzle)
110     return count
111
112
113 def find_tile(tile, State):
114     n = len(State)
115     for row in range(n):
116         for col in range(n):
117             if State[row][col] == tile:
118                 return row, col
119
120
121 def manhattan(stateA, stateB):
122     tile = 0
123     result = 0
124     rowA, colA = find_tile(tile, stateA)
125     rowB, colB = find_tile(tile, stateB)
126     result += abs(rowA - rowB) + abs(colA - colB)
127     return result
128
129
130 def is_solvable(Start: tuple, verbose: bool = False) -> int:
```

```
131     Destination: tuple = upperLeft
132     if verbose:
133         print(f"Name: {Start['name']}")
134         print(f"Start is: {Start['data']}")
135         print(f"Inversion Count:
            ↪ {get_inversion_count(Start['data'])}")
136         print(
137             f"Manhattan Distance: {manhattan(Start['data'],
                ↪ Destination['data'])}")
138     return (get_inversion_count(Start['data']) +
        ↪ manhattan(Start['data'], Destination['data'])) % 2 == 0
139
140
141 for s in Starts:
142     print(f"is solvable: {is_solvable(s, verbose=True)}")
143     print('\n')
144
145 get_inversion_count(start4['data'])
```

# B Beispiel-Anhang: Noch ein Testanhang

nochmal: lipsum ...