# Aufgabenblatt 2 - Maximilian Schmidt - 301666

## Aufgabe 1 : Parität

### Welche Parität hat der Zustand S in Abb. 1?

Die Parität beträgt 18 und ist damit gerade, also in unserem Programm "True".

## Aufgabe 2 : Heuristiken

### Für einen Zustand z ist h1(z) die Anzahl der falsch platzierten Steine in Bezug auf den Zielzustand. In Abb. 1 ist h1(S) = 8. Wieso ist h1 eine monotone Heuristik?

Jeder falsch platzierte Stein muss immer mindestens einmal bewegt werden, um auf seine korrekte Position kommen zu können. Daher müssen für X falsch platzierte Steine immer mindestens X Züge durchgeführt werden, um eine Lösung erreichen zu können. Deshalb ist die Heuristik h1 immer kleiner oder gleich der Anzahl der zum Lösen notwendigen Zügen.

### Für einen Zustand z ist h2(z) die Summe der Manhattan-Distanzen der Steine von ihren Zielpositionen. In Abb. 1 ist h2(S) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18. Wieso ist h2 eine monotone Heuristik?

Da ich keine zwei Zahlen (ohne die 0, bzw. dem leeren Feld) miteinander tauschen darf, verringert sich die Manhattan-Distanz nach einem Zug immer maximal um 1 (sie kann sich auch um 1 erhöhen). Es existiert also kein Zug, der zwei Steine gleichzeitig näher an ihre Zielposition bringt. Die Manhattan-Distanz ist damit für jede Position immer kleiner oder gleich der bis zur Lösung notwendigen Züge.

### Wieso ist h1(n) ≤ h2(n)? Welche Heuristik ist also besser?

Ein Stein, der richtig platziert ist, bekommt in beiden Heuristiken den Wert 0. Jeder falsch platzierte Stein bekommt in h1 den Wert 1 und in h2 mindestens den Wert 1 oder einen größeren zugewiesen. Damit ist h1 immer kleiner oder gleich h2. Für eine gegebene Position sind die tatsächlichen Kosten einer optimalen Lösung konstant und immer größer oder gleich der Werte der Heuristiken h1 und h2. Zieht man von dieser Konstanten jeweils die Werte der beiden Heuristiken ab, so ist das Ergebnis für die Heuristik h2 immer kleiner oder gleich des Ergebnis für Heuristik h1. Die Heuristik h2 ist also schärfer, sie liegt also näher oder maximal gleich weit entfernt von den tatsächlichen Kosten im Vergleich zu h1.

## Aufgabe 3 : Suchverfahren IDS und A*

### Bestimmen Sie die Anzahl der vom Suchverfahren generierten Zustände und die Länge der Lösungsfolge für verschiedene Startzustände.

Die Suchkosten werden jeweils beim Aufruf des Suchalgorithmus mit 0 initialisiert. Bei IDFS werden die Suchkosten immer dann erhöht, wenn die rekursive Funktion aufgerufen wird. Bei A* werden die Suchkosten immer dann erhöht, wenn ein neuer Kandidat aus der openList betrachtet wird.

Startzustand 1: [2, 7, 6, 5, 1, 3, 8, 0, 4]
Anz. Züge zur Lösung: 30

Suchkosten A* mit h2: 9347
Suchkosten IDFS: 49919866

Startzustand 2: [0, 4, 3, 8, 7, 2, 1, 6, 5]
Anz. Züge zur Lösung: 19
Suchkosten A* mit h2: 96
Suchkosten IDFS: 97743

## Sind Ihre Zugfolgen optimal? Wenn ja, warum?

Die Zugfolgen sind optimal. DFS ist innerhalb seines Limits vollständig und kann damit auch die optimale Lösung innerhalb des Limits ausgeben, sofern es eine Lösung gibt. Wird innerhalb eines Limits keine Lösung gefunden, aber es existiert eine Lösung innerhalb eines größeren Limits, so muss diese Lösung auch immer größere Kosten haben als das alte Limit (wenn Kosten uniform und die Gesamtkosten tiefenabhängig sind). Damit eine optimale Lösung bei IDFS garantiert werden kann, muss das Limit bei 1 starten und mit jeder Iteration, wenn keine Lösung gefunden wurde, auch maximal um 1 steigen. Je nachdem, was für Kosten man für ein Problem im Schnitt erwartet, kann man das Limit auch zu Beginn auf eine größere Zahl als 1 setzen. Sobald dieser Wert aber den kleinsten Wert aller optimalen Lösungen aller Probleme überschreitet, kann es dazu kommen, dass die optimale Lösung nicht gefunden wird.

A* liefert mit den Heuristiken h1 und h2 optimale Lösungen zurück, da beide Heuristiken zulässig und monoton sind.

## Welches Problem könnte entstehen (nicht ausprobieren!), falls A* für die Lösung des 15-Puzzle eingesetzt werden würde?

IDFS speichert nur seinen aktuellen Pfad und wenn er keine Lösung gefunden hat, löscht er beim rekursiven Aufstieg die Elemente im Pfad, bis er wieder expandieren kann. A* dagegen hat eine Vielzahl von möglichen Kandidaten gleichzeitig im Speicher. Während A* in optimaler Zeit bei einer monotonen Heuristik eine optimale Lösung findet, kann der Speicherbedarf des Algorithmus allerdings enorm anwachsen. A* ist damit durch den Speicherplatzverbrauch beschränkt, vor allem dann, wenn der Lösungsweg besonders lang ist.