Zauberwürfellöseroboter

Koordinaten-Berechnung

Koordinaten-Berechnung

Alle Algorithmen, die ich nutze, funktionieren ähnlich: sie haben alle Pattern-Databases, in denen sie die benötigte Anzahl von Zügen zum Ziel nachschlagen.

Die PDBs (Pattern-Databases) bestehen aus Nibbles. Ein Nibble ist die Hälfte eines Bytes. Um die Position des benötigten Nibbles herauszufinden, wird eine Koordinate aus den relevanten Teilen des Würfels berechnet und dann in der Liste nachgeschaut. Diese Koordinatenberechnung unterscheidet sich für jede einzelne PDB.

Beispiel

Die PDB `corner.pdb` speichert die Orientierung und Permutation der Ecken. Um die Koordinate (auch Index genannt) herauszufinden, werden als Erstes die Daten aus dem Würfel extrahiert. Diese haben folgende Struktur:

- Die Orientierungen sind eine Liste von Zahlen je 0 bis 2. Sie hat die Länge 8.
- Die Permutationen sind eine Liste der Zahlen von 0 bis 11. Jede Zahl kommt einmal vor.

Die Orientierungen

Da die Orientierungen im sogenannten Ternärsystem (alle Zahlen zwischen 0 und 2) sind, kann man die Werte mit einem jeweiligen Stellenwert multiplizieren, um die Zahl im Dezimalsystem (0 bis 9, das *normale* System) zu erhalten.

O repräsentiert hier die Orientierungen. Die letzte Ecke wird ignoriert, weil die Summe modulo 3 immer 0 sein muss und sie sich so aus den Anderen ergibt.

$$o = \sum_{i=0}^{7} O_i \cdot 3^{6-i}$$

Oder einfacher:

$$o = O_0 \cdot 3^6 + O_1 \cdot 3^5 + O_2 \cdot 3^4 + O_3 \cdot 3^3 + O_4 \cdot 3^3 + O_5 \cdot 3^1 + O_6 \cdot 3^0$$

Die Permutationen

Da die Permutationen komplizierter sind als die Orientierungen siehe PermutationIndexer.

Der Gesamtwert

Um am Ende eine Zahl zu haben, müssen die Zahlen noch kombiniert werden. p wird mit 4^7 multipliziert, weil das die größte Zahl ist, die o haben kann und so garantiert ist, dass jeder Würfel einen anden Index hat.

$$c = p \cdot 3^7 + o$$

PermutationIndexer

Um aus den Permutationen einen Index zu errechnen, verwende ich den Lehmer-Code, um die Position der Permutation in der Liste aller Permutationen zu berechnen.

Angenommen, wie haben die Zahlen $\{0, 1, 2, 3\}$ und wollen wissen, an welcher Stelle die Permutation $(2, 3, 1, 0)$ steht. Wir könnten alle Permutationen durchgehen, bis wir sie gefunden haben:

- $0, 1, 2, 3$
- $0, 1, 3, 2$
- $0, 2, 1, 3$
- $1, 0, 2, 3$
- ...

Allerdings ist dieses Verfahren sehr ineffizient, weil extrem große Mengen an Permutationen berechnet werden müssen. An dieser Stelle kommt der Lehmer-Code ins Spiel:

Er berechnet für jede Zahl der Permutation, wie viele Zahlen es rechts davon gibt, die kleiner sind. Diese sind dann das Ergebnis, das danach noch in einen Index verwandelt wird.

In unserem Beispiel wäre das dann:

1. Rechts von der Zahl 2 sind drei Zahlen, von denen zwei kleiner sind.
2. Die 3 hat nur zwei Zahlen rechts von sich, beide kleiner.
3. Die 1 hat eine kleinere Zahl rechts von sich, die Null.
4. Da die 0 ganz rechts steht, kann sie keine weiteren Zahlen rechts neben sich haben.

Das Ergebnis ist also $(2, 2, 1, 0)$. Warum dieser Algorithmus funktioniert, habe ich nicht verstanden.

Um am Ende eine einzige Zahl zu erhalten, wird noch jede Zahl mit der größtmöglichen Zahl der nächsten Zahl (Anzahl der Zahlen rechts Fakultät) multipliziert und am Ende zusammengerechnet.

$$l = L_0 \cdot 3! + L_1 \cdot 2! + L_2 \cdot 1! + L_3 \cdot 0!$$

$$l = 2 \cdot 6 + 2 \cdot 2 + 1 \cdot 1 + 0 \cdot 1$$

$$l = 12 + 4 + 1 + 0$$

$$l = 17$$