Zauberwürfellöseroboter

Korf

Korfs Algorithmus

Aus anderem Projekt übertragen

Diesen Algorithmus habe ich von benbotto's Rubiks-Cube-Cracker übernommen. Die PDBs habe ich unverändert genutzt, den Algorithmus nachprogrammiert (war in einer anderen Programmiersprache).

Korfs Algorithmus ist ein Algorithmus, der die optimale Lösung (maximal 20 Züge) für einen Zauberwürfel findet, aber dafür relativ lange braucht, da sehr viele Züge durchsucht werden müssen.

Der Algorithmus basiert auf IDA* und nutzt Pattern-Databases, um zu schätzen, wie viele Züge es noch braucht, um den Würfel zu lösen.

Um die Anzahl der Unterknoten zu verringern, verwende ich eine einfache Move-Pruning Strategie, die sich aufhebende Züge nicht weiterverfolgt.

Die Heuristik

Die Heuristik, die für Korfs Algorithmus verwendet wird, ist der Maximalwert aus den Lookups der PDBs.

Die Summe ist deshalb nicht möglich, da es eventuell Züge in den verschiedenen PDBs gibt, die sich gleich sind, was dazu führen würde, dass es einen Zug weniger braucht, um das Ziel zu erreichen, wodurch die Heuristik unzulässig würde und die optimale Lösung nicht garantiert wäre.

Die Pattern-Databases (PDBs)

Die PDBs sind Datenbanken, die gespeichert haben, wie viele Züge es aus der aktuellen Position braucht, um einen Teil des Würfels zu lösen.

Es gibt drei PDBs, um die verschiedenen Teile des Würfels zu repräsentieren:

- `corner.pdb`, die Orientierung und Permutation der Ecken.
- `edgeG1.pdb`, die Orientierung und Permutation der ersten sieben Kanten.
- `edgeG2.pdb`, die Orientierung und Permutation der letzten sieben Kanten.

In den Edge-PDBs werden sieben Ecken genutzt, da es sonst keinen Anhaltspunkt geben würde und die Kanten beliebig gedreht werden könnten.

Man braucht diese verschiedenen PDSs, weil es etwa 43 Trillionen Kombinationen gibt und eine passende Datenbank 23 Exabyte groß sein würde. Das wären 24 696 061 952 Gigabyte.