Shortest Path Finder for Rubik's Cube

Casey Fleck and Thomas Hoffman

Department of Computer Science and Mathematics
Coastal Carolina University
Honors Program

1. Introduction

A theory known as God's Number hypothesizes that there exists a single algorithm that can solve any Rubik's cube in the most minimal number of steps. Brute force is used to find this algorithm. If we eliminate all unnecessary steps we know to bring us back to the solved state, this will improve the efficiency of the brute force approach that solves for God's Number.

2. Methods

I. Simple Solution: Brute Force

A path is defined as the set of states needed to get back to the solved state such that P=$\{S_1, S_2 ... S_n\}$.

Each turn needed to achieve the next state is called a permutation.

Starting from any state in the sequence, we use brute force to visit every possible state for the Rubik's Cube to find unique paths that lead us back to the original state.

If any unique paths are found in a sequence of n paths, then we add them to a list of unique paths that will be used to find minimum number of moves from any permutation.

Given a graph G, and a State S, a path that leads back to the original state can be found by testing all possible turns until the correct path is found.

Example 1. Below, each vertex represents a different state of a Rubik's Cube. Starting at state 4, the algorithm finds a path through 4 states that leads back to its original state.

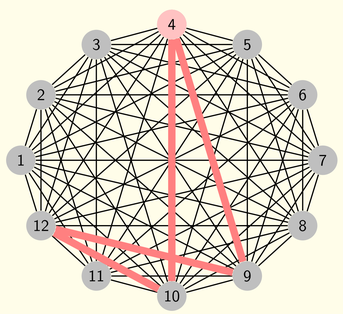


Figure 1: *Brute force using a complete graph.*

Each state the Rubik's Cube is in can make 12 possible moves - up, down, left, right, front, back - and all their inverses.

Therefore, there are 12^n possible moves that can be made. The efficiency of this algorithm can be written in Big O notation as $O(12^n)$.

II. An improvement on Brute Force

In order to make improvements on this algorithm, predicted branches of turns that are already known (such as FFFF) were removed from our list.

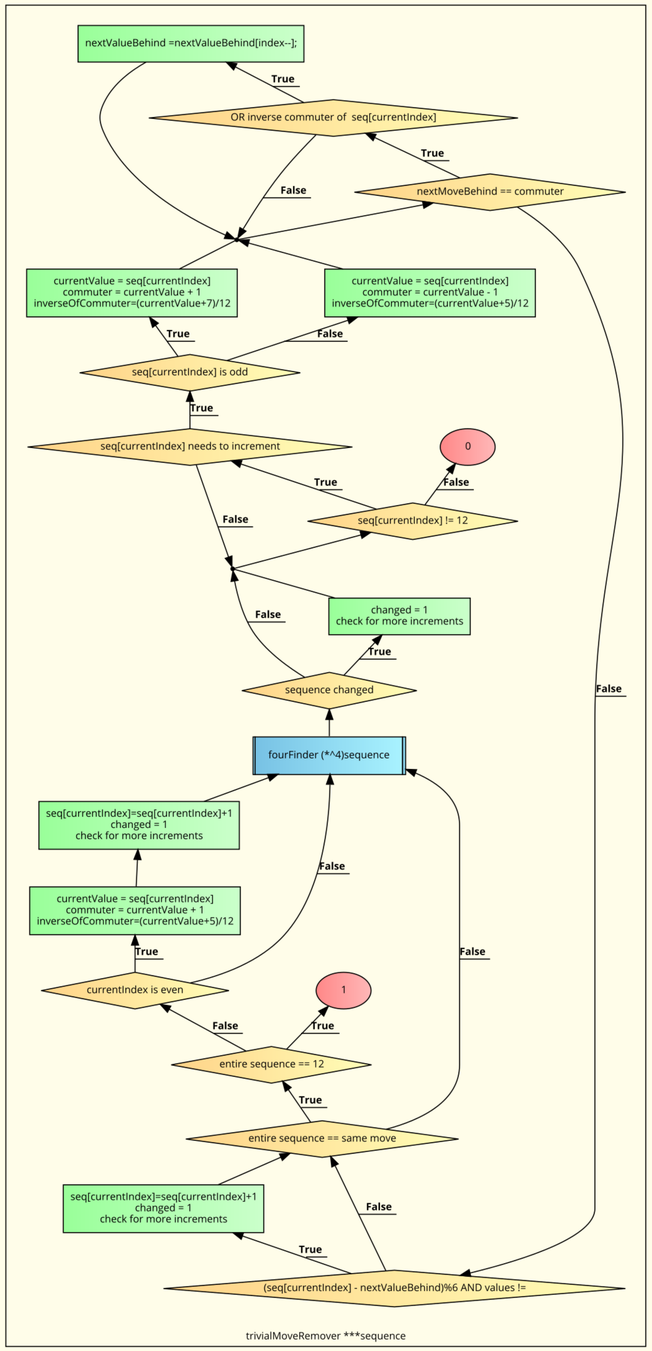


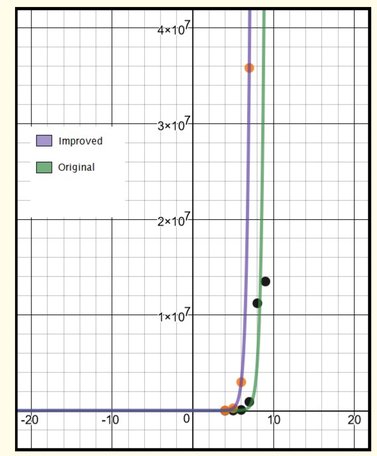
Figure 2: *Flow chart of improvements made to brute force.*

3. Results

Table 1: *Time efficiency for improved brute force.*

Permutation	Time	Time (ms)
4	00:00:00:43	43
5	00:00:05:76	5076
6	00:01:13:87	73087
7	00:15:18:20	918020
8	03:07:08:68	11228068
9	37:36:10:49	13500049

Figure 3: *Improved and original exponential regression of brute force.*



4. Conclusion

The exponential regression graph from Figure 3 shows that the improved brute force algorithm can be expressed as $O(12.68^n)$ and the original can be expressed as $O(12^n)$. Thus, the improved algorithm has a worse efficiency than the original algorithm. The improved algorithm brought back more unique algorithms that were not known than the original algorithm. These results could have been a consequence of an inability to test more data points due to timing constraints. In addition, threads could have been added to the program to further decrease the program's run time.

5. References

[1] Frey, Alexander H. and Singmaster, David. (2010). Handbook of Cubik Math. Cambridge: Lutterworth Press.

[2] The Mathematics of a Rubik's Cube. (2009). Retrieved December 2, 2018, from http://web.mit.edu/sp.268/www/rubik.pdf