Twentyseven 1.0.0

Posted on August 1, 2025

Twelve years of Haskell

Twentyseven is a Rubik's cube solver and one of my earliest projects in Haskell. The first commit dates from January 2014, and version 0.0.0 was uploaded on Hackage in March 2016.

I first heard of Haskell in a course on lambda calculus in 2013. A programming language with lazy evaluation sounded like a crazy idea, so I gave it a try. Since then, I have kept writing in Haskell as my favorite language. For me it is the ideal blend of programming and math. And a Rubik's cube solver is a great excuse for doing group theory.

Twentyseven 1.0.0 is more of a commemorative release for myself, with the goal of making it compile with the current version of GHC (9.12). There was surprisingly little breakage:

1. `Semigroup` has become a superclass of `Monoid`
2. A breaking change in the Template Haskell AST

Aside from that, the code is basically just as it was 9 years ago, including design decisions that I would find questionable today. For example, I use `unsafePerformIO` to read precomputed tables into top-level constants, but the location of the files to read from can be configured by command-line arguments, so I better make sure that the tables are not forced before the location is set…

How Twentyseven works

The input of the program is a string enumerating the 54 facelets of a Rubik's cube, each character represents one color.

```
DDDFUDLRB FUFDLLLRR UBLBFDFUD ULBFRULLB RRRLBBRUB UBFFDFDRU
```

The facelets follow the order pictured below. They are grouped by faces (up, left, front, right, back, top), and in each face they are listed in top-down, left-right order.

```
                00 01 02
                03 04 05
                06 07 08

      10 11 12  20 21 22  30 31 32  40 41 42
      13 14 15  23 24 25  33 34 35  43 44 45
      16 17 18  26 27 28  36 37 38  46 47 48

                50 51 52
                53 54 55
                56 57 58
```

The output is a sequence of moves to solve that cube.

```
U L B' L R2 D R U2 F U2 L2 B2 U B2 D' B2 U' R2 U L2 R2 U
```

The implementation of Twentyseven is based on Herbert Kociemba's notes about *Cube Explorer*, a program written in Pascal!

The search algorithm is iterative deepening A*, or IDA*. Like A*, IDA* finds the shortest path between two vertices in a graph. A conventional A* is not feasible because the state space of a Rubik's cube is massive (43 252 003 274 489 856 000 states, literally billions of billions). Instead, we run a series of depth-first searches with a maximum allowed number of moves that increases for each search. As it is based on depth-first search, IDA* only needs memory for the current path, which is super cheap.

IDA* relies on an estimate of the number of moves remaining to reach the solved state. We obtain such an estimate by projecting the Rubik's cube state into a simpler puzzle. For example, we can consider only the permutation of corners, ignoring their orientation. We can pre-compute a table mapping each corner permutation (there are 8! = 40320) to the minimum number of moves to put the corners back to their location. This is a lower bound on the number of moves to actually solve a Rubik's cube. Different projections yield different lower bounds (for example, by looking at the permutation of edges instead, or their orientation), and we can combine lower bounds into their maximum, yielding a more precise lower bound, and thus a faster IDA*.

Putting all that together, we obtain an optimal solver for Rubik's cubes. But even with these heuristics, Twentyseven can take hours to solve a random cube optimally. Kociemba's Cube Explorer is apparently much faster (I've never tried it myself). My guess is that the difference is due to a better selection of projections, yielding better heuristics. But I haven't gotten around to figure out whether I've misinterpreted his notes or those improvements can only be found in the code.

A faster alternative is Kociemba's two phase algorithm. It is suboptimal, but it solves Rubik's cubes in a fraction of a second (1000 cubes per minute). The first phase puts cubies into a "common orientation" and "separates" the edges into two groups. In other words, we reach a state where the permutation of 12 edges can be decomposed into two disjoint permutations of 4 and 8 edges respectively. In the second phase, we restrict the possible moves: quarter- and half-turns on the top and bottom faces, half-turns only on the other faces. These restricted moves preserve the "common orientation" of edges and corners from phase 1, and the edges in the middle slice stay in their slice. Each phase thus performs an IDA* search in a much smaller space than the full Rubik's cube state space (2 217 093 120 and 19 508 428 800 states respectively).
