Modeling Challenge I: Cube

Eckel, TJHSST AI1, Fall 2022

Background & Explanation

Cube is a particularly challenging spatial brainteaser about rolling a cube around on a grid; see it here: https://www.chiark.greenend.org.uk/~sgtatham/puzzles/js/cube.html

The rules couldn't be simpler.

- Each move is the cube rolling one space in any direction (it just can't roll off the board).
- Each time the cube lands, the bottom face of the cube and the space on the board where it has landed switch surfaces. (ie, if that side of the cube was previously painted blue, and the space on the grid was empty, the space becomes painted and the cube side becomes empty, or vice versa.) If they're both painted or both empty, there is no change.

For this modeling challenge, you may work in partners if you wish. **You may not assist or get assistance from another student other than your partner for any reason**. If you discover that you can't quite manage it, that's ok on this assignment. You can get full BLUE credit for succeeding or full BLUE credit for failing – see the two specifications below.

Required Task

You're provided with a puzzle file on the course website. It contains a puzzle of each size from 3x3 to 10x10. For each puzzle you are given the size, the board (with "." representing empty spaces and "@" representing blue ones), and the index of the cube on the board. (Note that you can't just use a character on the board to represent the cube since the empty/painted status of the space below the cube also needs to be remembered.) Next to each puzzle is a URL you can use to pull up that particular puzzle on Simon Tatham's site, so you can test your code's solution.

Your goal is to write code that solves from size 3x3 to size 8x8 in a total time of less than 2 minutes. You can consider the 9x9 and 10x10 boards optional extra challenges. (If you're curious, my best solution so far gets all of the puzzles, including the two optional ones, in just over 20 seconds total. I'm absolutely sure there are better solutions out there.)

So, to be specific: write code that reads in the given file and, for each puzzle from 3x3 to 8x8 inclusive, finds a solution and outputs the length of that solution **as well as the complete step-by-step path** for each puzzle (that is, in some way it must give a step by step solution in a format that's easy to read – you get to pick the format, but I need to be able to understand it). If you can do that in less than 2 minutes, you are successful! If not, that's also ok – see the next page.

Correct Answers

Unless there's an error in my code (which is always possible), the shortest possible solutions for each puzzle are as follows:

- The given size 3 puzzle – 12 moves
- The given size 4 puzzle – 13 moves
- The given size 5 puzzle – 13 moves
- The given size 6 puzzle – 18 moves
- The given size 7 puzzle – 18 moves
- The given size 8 puzzle – 22 moves
- The given size 9 puzzle – 25 moves (optional challenge)
- The given size 10 puzzle – 27 moves (optional challenge)

Use this to verify that your code is working properly. If you get a shorter solution **and you try it on Simon Tatham's website and it works**, show me please! It's always fun when students write code better than mine.

Specification if you Declare Success

If you believe you have succeeded in the task, submit to the link on the course website.

This assignment is **complete** if:

- You follow the instructions on the submission form to format your submission properly.
- Your code accepts a **command line argument** for the name of a puzzle file (as usual) then solves each problem in the puzzle file successfully.
- For each puzzle, you output the path length & **a complete step-by-step solution** for each.
- If you think your solution path format might be unclear, feel free to add an extra line or two of printed output explaining how I should read it.
- Your total runtime is less than 2 minutes (comparable to solving puzzles from 3x3 to 8x8 in less than 2 minutes).

If you declare success and your code does not work, you are eligible to then declare failure, but only for 8/10 credit. (So, only declare success if you're confident and you have carefully checked your output!)

Specification if you Declare Failure

If you decide that your code can't be completed, I'll need a write-up turned in to me (submit to the link on the website.)

The assignment is **complete** if:

- You write a 500 – 1000 word analysis of your code including what you tried to do, what the code actually does, and why it isn't working.