Intro to Generative Art
Problem Set 5

Harvard University
Taught by Sudhan Chitgopkar

This problem set is optional, but *highly encouraged.* You may work in groups and/or solicit outside help including but not limited to Generative AI tools. All outside sources/code used *must* be properly credited in the code documentation submitted.

1 Lost & Found

During lecture, we used the iterative version of a randomized depth-first-search in order to generate a maze. It turns out we can use a very similar algorithm to solve our maze. *Consider how this might be possible and develop an animated maze-solver.*

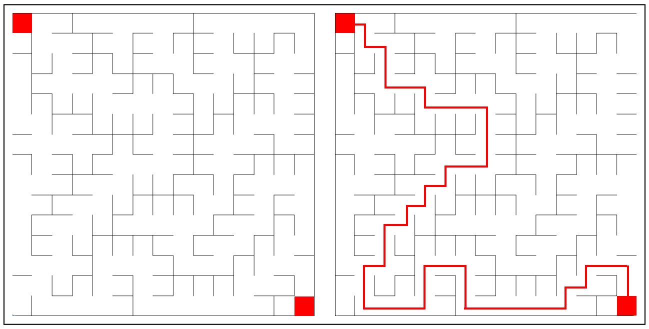

Figure 1: A 15 × 15 DFS-generated maze and its solution

Note that in the maze-generation algorithm we wrote during lecture, there *is* a defined start point but there is *not* a defined exit point. To fix this issue, you may arbitrarily designate the exit as the first cell on the opposite side of the start position reached using the maze-generation algorithm developed. You can find the maze-generation code written during lecture here.

2 Do It Yourself!

Rather than solve these mazes algorithmically, let's gamify the maze-solving process. *Modify the starter code linked above so that the user can now control a rat that must solve the maze.* Consider how to detect collision between the rat and the walls of the maze.

Note, the math here may get a little tricky. It may be prudent to use a small maze as a proof of concept and consider the differences between using mouse and keyboard input. If you're stuck with the collision detection math, consider instead how the `get()` and `strokeWidth()` functions can be used for collision detection with minimal math.