Algorithms for Game Design Exercise #3

1) Maze Algorithms

I have given you **`mazes.py`**, a general purpose maze module. But, many of the methods are not implemented. Find any method with a pass statement and write the correct code for that algorithm.

I have also given you **`text_maze.py`** and **`show_maze.py`**, both of which use **`mazes.py`** in order to create mazes that they then display (sort of like a Model and View/Controller).

You should be able to execute **`show_maze.py`** and then type one of the following keys to get the effect:

Q or Escape ➞ Quit

N ➞ New maze, load a grid with no walls carved

F ➞ Save an image of your maze to a file named maze.png

B ➞ Binary Tree algorithm

S ➞ Sidewinder algorithm

A ➞ Aldous-Broder algorithm (will report number of steps with print statements)

W ➞ Wilson's algorithm (will report loops removed)

R ➞ Recursive Backtracker

L ➞ Will mark the longest path with brown circles

D ➞ Will mark all deadend cells with red, the rest white. Will report the number of deadends.

C ➞ Colorize. Will run Dijkstra's algorithm from the center of the maze. Will then color each cell with darker greens for cells further away from center

Make sure your code works for all these situations

---

2) Hybrid

Can you speed up the execution of the random walk algorithms by mixing them?

Aldous-Broder starts fast, but is very slow in the end, when it needs to random walk to a small number of cells

Wilson's starts slow, with lots of loop removals when it needs to find that first cell. But in the end it is quite fast.

Write a hybrid version of AB + Wilson's that is as fast as you can make it. It should start with Aldous-Broder, but after some number of runs (how many?) it should switch to Wilson's algorithm. Make sure to do some measurements to ensure you've made it run faster.

---

3) Textbook

If you haven't already, read Chapters 0 - 3.

Then, read Chapter 5: Navigation and Control.