

Mod 5 Homework – Recursion

In this assignment, you will use recursion to solve a simple maze game.

This game contains a two-dimensional grid of squares that serves as the maze. Each square in the maze is either a Point square (containing an integer value), a Wall square (denoted by “*”), the Start square (“S”), or the Finish square (“F”). The goal is to find the path from Start to Finish that gathers the most “points.” The path cannot pass through a wall nor can it double back on itself. The path can only flow horizontally or vertically between two adjacent squares. The path cannot flow diagonally.


Here is an example of simple 5x5 maze. We will use the notation of (row, col) to reference a given square. Notice the Start is at (0,1) and the Finish is at (0,3) and the walls are marked by asterisks. The row and column labels are also provided to help you reference the correct square but are not considered part of the maze.

	0	1	2	3	4
0	*	S	*	F	1
1	2	5	*	*	2
2	3	*	*	*	8
3	9	*	4	7	3
4	1	3	1	*	2

Shown below is the best (and in this case, only) solution to this example maze. You can see the winning path marked by the @ character. If you count up all the point values originally occupied by the @ squares, the winning total is 49 (which is 5+2+3+9+1+3+1+4+7+3+8+2+1).

	0	1	2	3	4
0	*	S	*	F	@
1	@	@	*	*	@
2	@	*	*	*	@
3	@	*	@	@	@
4	@	@	@	*	2

	0	1	2	3	4
0	*	S	*	F	1
1	2	5	*	*	2
2	3	*	*	*	8
3	9	*	4	7	3
4	1	3	1	*	2



Here is another example, this time a 3x3 maze that has three possible paths from Start to Finish.

	0	1	2
0	6	5	S
1	5	F	3
2	8	*	*

	0	1	2
0	6	5	S
1	5	F	3
2	8	*	*

Path: (0,2), (1,2), (1,1)
Score: 3

	0	1	2
0	6	5	S
1	5	F	3
2	8	*	*

Path: (0,2), (0,1), (1,1)
Score: 5

	0	1	2
0	6	5	S
1	5	F	3
2	8	*	*

Path: (0,2), (0,1), (0,0),
(1,0), (1,1)
Score: 16 (5+6+5)

All three paths are valid, but the winning path is the one with the score of 16.

We will solve this puzzle using recursion, which allows the computer to try every permutation or possible path. The trick to writing a recursive function is to identify the “stop condition(s)” and then let your function try all options before reaching that stopping condition(s).

You have been given the code for the Maze class (maze.py), complete with many attributes and methods that will be useful to your solution. You have also been given the shell of the Game class (game.py), which you will complete. An example test case is contained in test_game.py but you must add more test cases to ensure your recursive algorithm is correct for various edge cases.

You can assume that the smallest maze will be 2x2, but mazes can be square or rectangular (e.g. 4x2, 5x5, 3x7). If the maze has no solution path from Start to Finish, your function should return a score of -1 and an empty List for the winning path. You must use recursion to find the best solution for any given maze.

Submitting

At a minimum, submit the following files:

- game.py (adding your recursive function)
- test_game.py (adding additional tests cases)
- maze.py (do not alter any code in this file)

Additionally, you must include any other files necessary for your code to run, such as modules containing data structures you wrote yourself.

Do not import any modules you did not write yourself. The functionality added by external modules is one of the great things about python, but we restrict them for two reasons:

- Some of them circumvent the purpose of these assignments, which is to develop basic algorithm design habits
- Some of them are not supported by our Python install on Gradescope, making it impossible for us to run your code there (and thus slowing down our grading significantly)

Students must submit **individually** by the due date (typically, Tuesday at 11:59 pm EST) to receive credit.