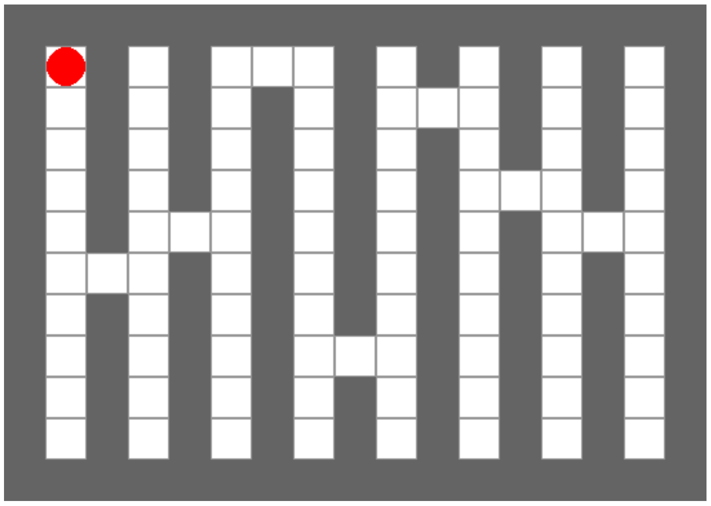
Solving Edlund's Maze



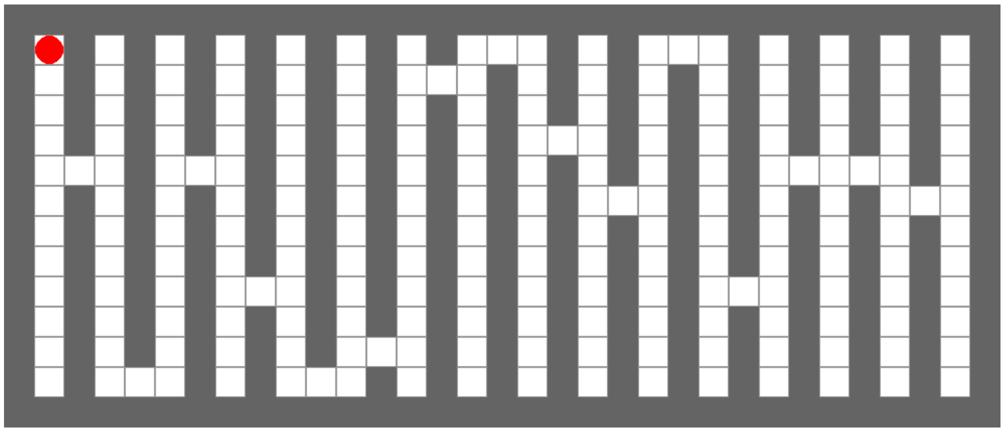
Edlund's Maze is a simple navigation task used to test AI agents. The agent is placed on the left side of the maze with the goal of reaching the right side of the maze.

An example maze if given to the right.

Warm up: Can you think through how to solve this maze by moving one step at a time? Do not write out the whole thing, though!

Here's the trick: We want your code (without modification) to be able to solve **ANY** maze! This means that we can't just hard-code the movements, we need to look around and sense where the openings are.

Here is another example maze:



First, I want you to think about the *algorithm* to solve an arbitrary maze. Don't worry about putting it into code yet. Just describe the logic as a set of steps in English. You can assume that you can move one step at a time, and also can call a function to sense the tile next to you in any direction.

Once you're feeling confident in your algorithm, try implementing it in code! I have provided some code in the "Class Activities" folder on Blackboard. The file contains comments to explain what the code does and what you should be changing. **You can tweak the parameters at the top of the code and write your algorithm at the bottom, but you should not change the middle portion.**

You will have access to several functions:

`move_up()`, `move_down()`, `move_right()`, `move_left()`

Move the agent one step in that direction

`sense_up()`, `sense_down()`, `sense_right()`, `sense_left()`, `sense_current()`

Sense the tile in that direction, relative to the agent.
Returns a string (" " for an open tile, "#" for a wall).

`draw()`

Prints the current state of the board. Floors are spaces, walls are #, the agent is `@`

`check_finished()`

Returns true if the agent has reached the right side of the maze.

Good luck! As always, I'll be around to answer questions.
Note that I have initially set the random seed (at the top) to a fixed value. If you want to start testing your algorithm on random mazes, set the random seed to 0.

Bonus task: if you finish the code, feel free to turn on `hard_mode` at the top of the file.
This adds two new types of tiles:

- `"^"`, which indicates a hole in the wall and that the next hole will be above this one.
- `"v"`, which indicates a hole in the wall and that the next hole will be below this one

If the next hole is straight across, a space will still be used.
We can actually solve the maze *faster* in hard mode, because we have more information.
However this means we have to write more code.

Here is an example hard_mode maze: