LECTURE 15

RECURSION WITH BACKTRACKING

MCS 275 Spring 2022

Emily Dumas

LECTURE 15: RECURSION WITH BACKTRACKING

Course bulletins:

- Project 2 description available.
- Project 2 due 6pm central Friday 25 February.
- Check out the recursion sample code.

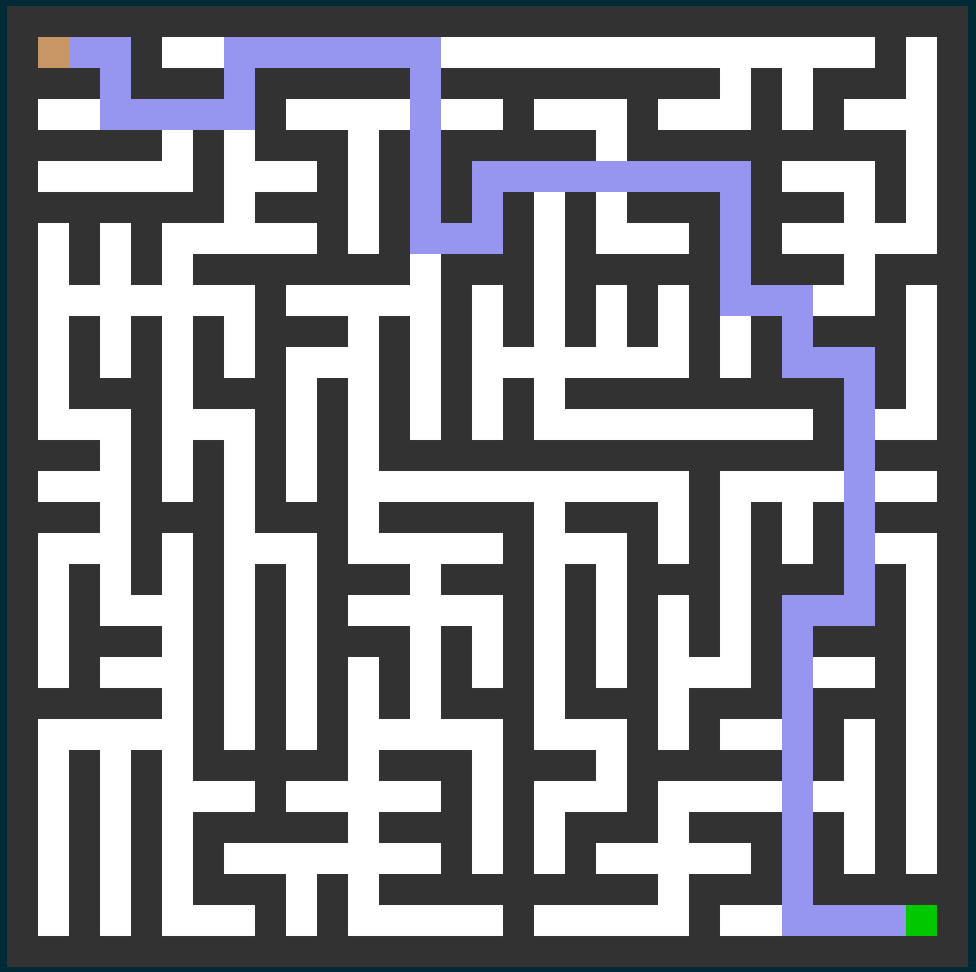
PLAN

- Review backtracking algorithm to solve a maze
- Present a module for working with mazes
- Implement the maze solver

RECURSION WITH BACKTRACKING

How do you solve a maze?

RECURSION WITH BACKTRACKING



How do you solve a maze?

We'll solve a maze like this using **recursion with backtracking**.

We make a function that takes:

- The maze
- The path so far

Its goal is to add one more step to the path, never backtracking, and call itself to finish the rest of the path.

But if it hits a dead end, it needs to notice that and **backtrack**.

BACKTRACKING

Backtracking is implemented through the return value of a recursive call.

Recursive call may return:

- A solution, or
- `None`, indicating that only dead ends were found.

If we ever receive a solution from a recursive call, we return it immediately.

Algorithm `depth_first_maze_solution`:

Input: a *maze* and a *path* under consideration (partial progress toward solution).

1. If the path is a solution, just return it.
2. Otherwise, enumerate possible next steps that don't go backwards.
3. For each of the possible next steps:
    - Make a new path by adding this next step to the current one.
    - Make a recursive call to attempt to complete this path to a solution.
    - If recursive call returns a solution, we're **done**. Return it immediately.
    - (If recursive call returns `None`, continue the loop.)
4. If we get to this point, every continuation of the path is a dead end. Return `None`.

DEPTH FIRST

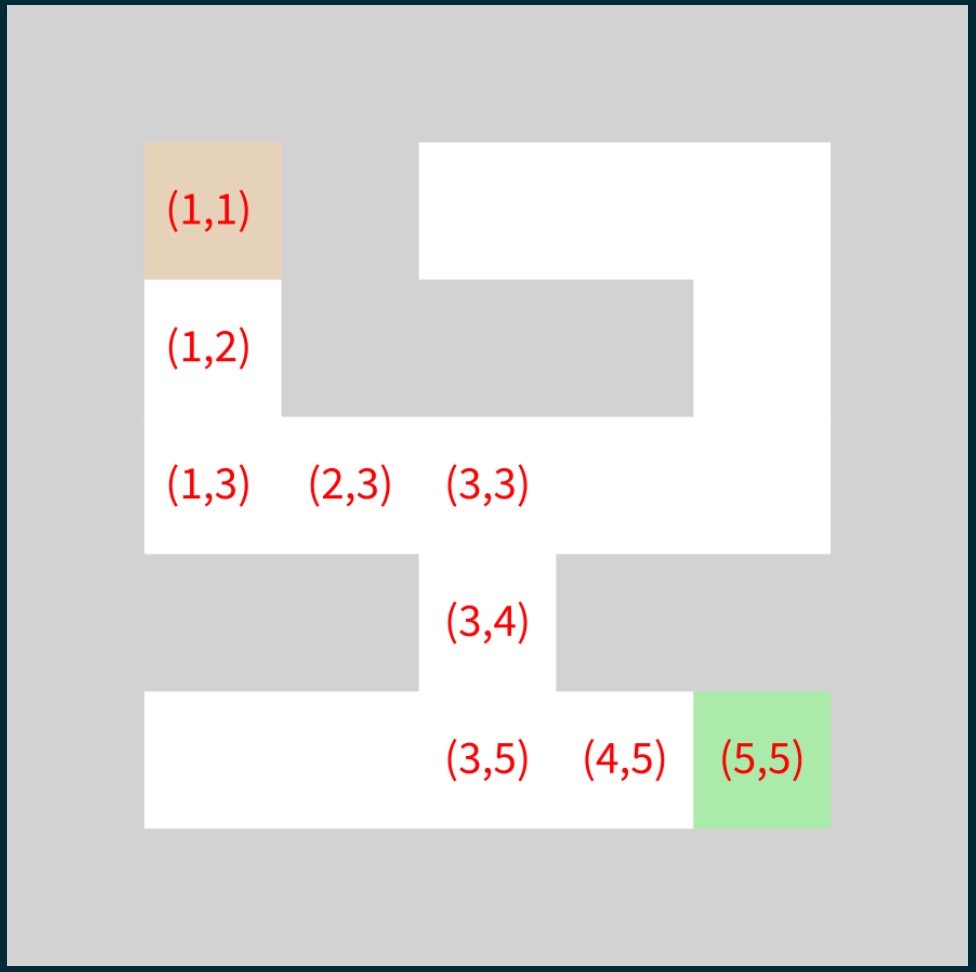
This method is also called a **depth first search** for a path through the maze.

Here, **depth first** means that we always add a new step to the path before considering any other changes (e.g. going back and modifying an earlier step).

MAZE COORDINATES

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)
(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)

MAZE COORDINATES



REFERENCES

Same suggested references as Lecture 13.

REVISION HISTORY

- 2022-02-14 Initial publication