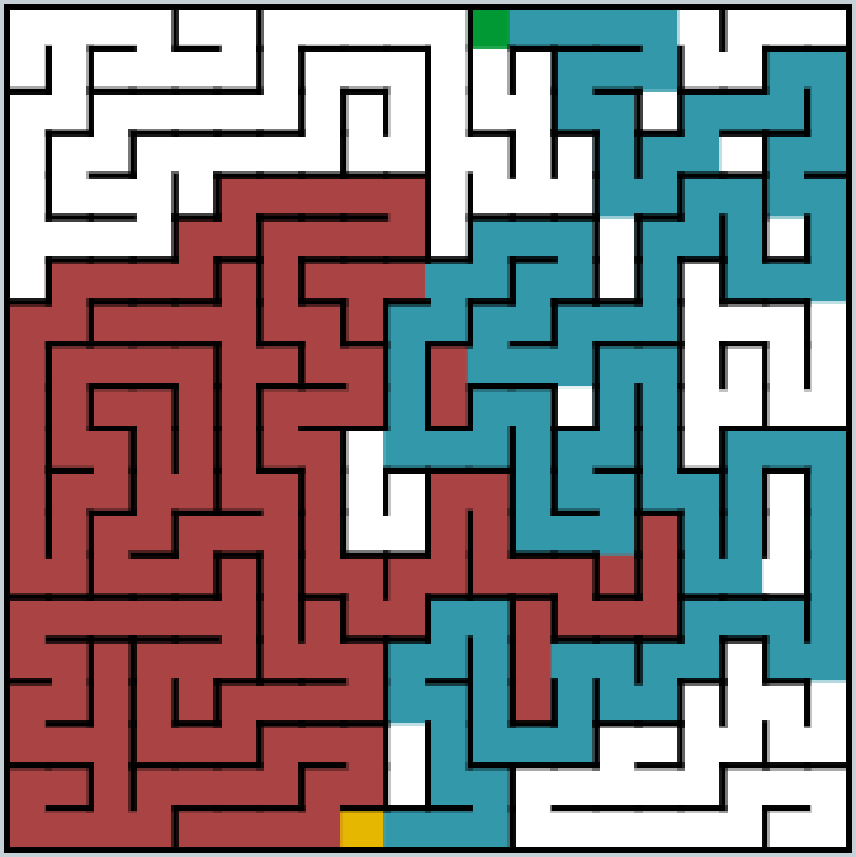
Generating and Displaying Mazes in Two and Three Dimensions

BY ROBERT KAUFMAN
FOR
MATH 198 FALL, 2015

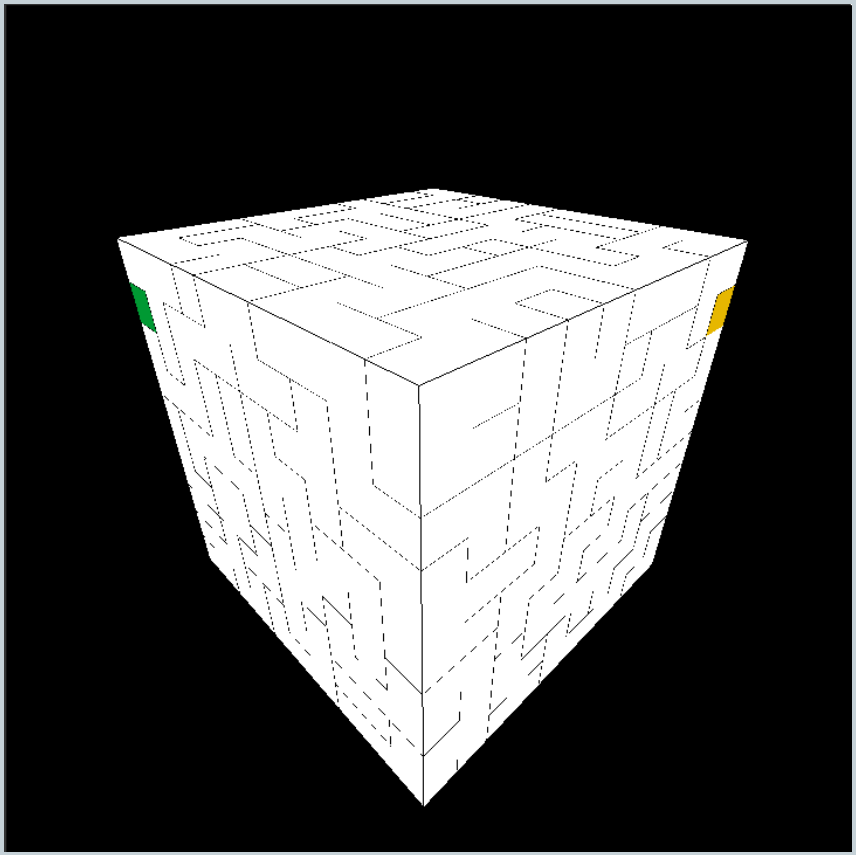
Background

- A maze is a simple puzzle in which a set of walls divide an area
- The goal is to get from one point to another



Project Purpose

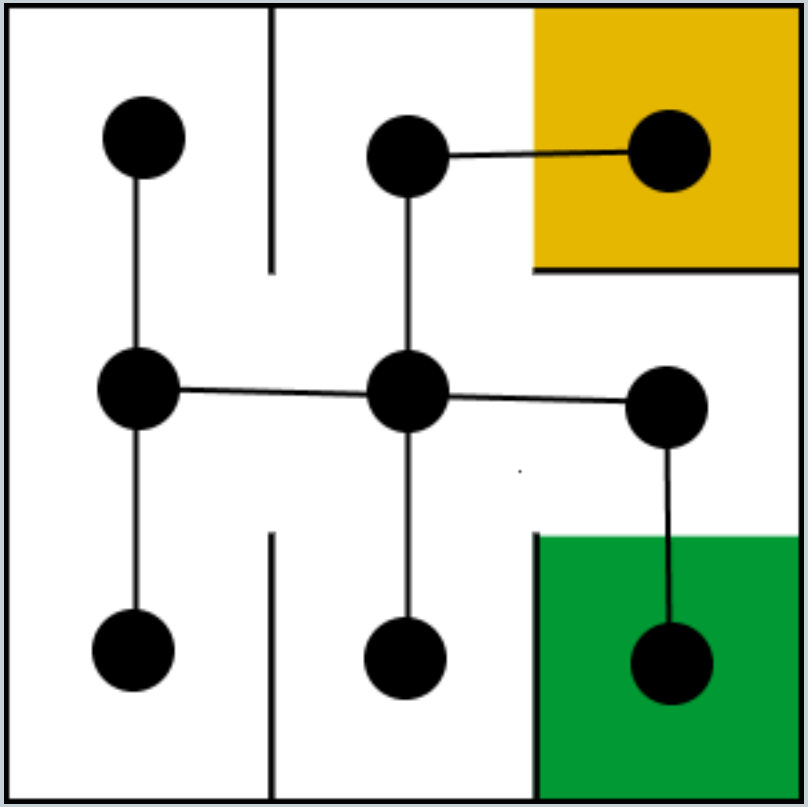
- Generate and display 2D mazes
- Generate and display 3D mazes
- Automatically solve both 2D and 3D mazes
- Manually solve 2D and 3D mazes

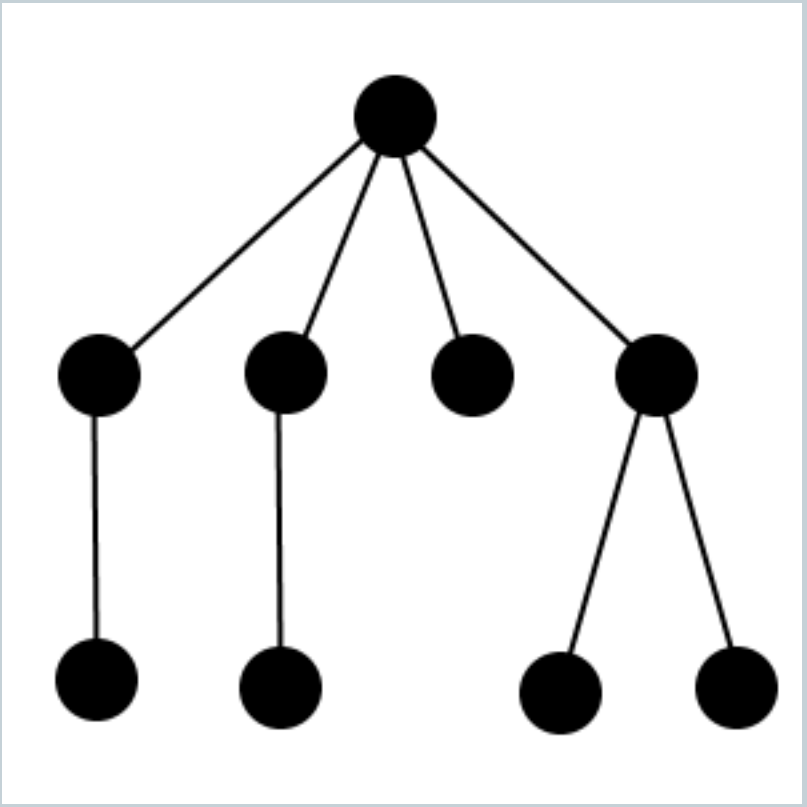


Mazes as Trees

- Mazes can be viewed as a spanning tree of a graph with a grid of vertices each connected to all their neighbors with equally weighted edges.
- Means that any spanning tree algorithm can create a maze
- The dimension of the maze does not matter as it can just be represented by a 2n-ary spanning tree for a nD maze (ex: 2D has a 4-ary spanning tree, 3D – 6-ary, 4D – 8-ary)

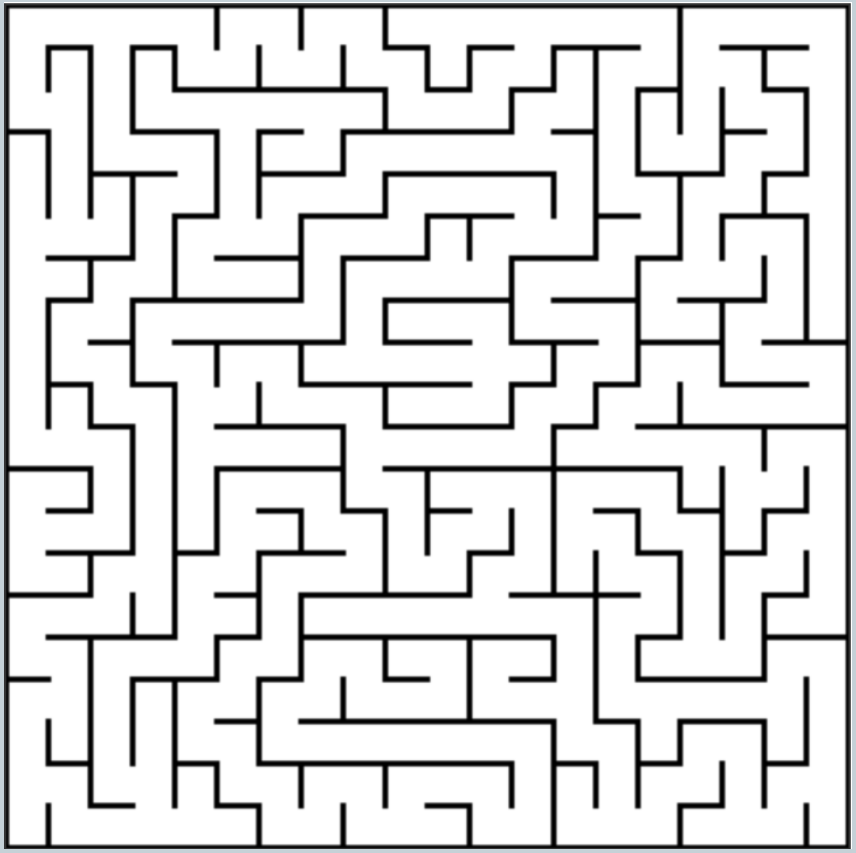
Mazes as Trees





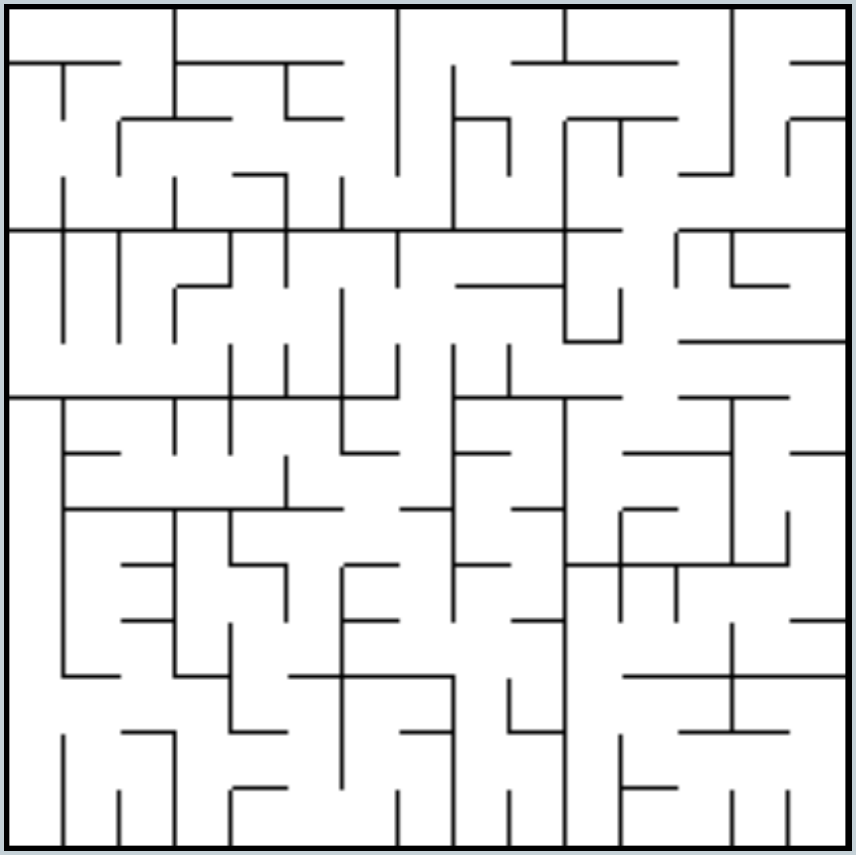
Recursive Back-tracker

- Pattern:
  - Pleasing, random appearance
  - Relatively long sections before a branch
- Algorithm:
  - Works by randomly selecting a path until there are no more valid moves (there are no unvisited adjacent cells).
  - Next moves back until another move is possible.
  - Continues until the whole maze has been visited.



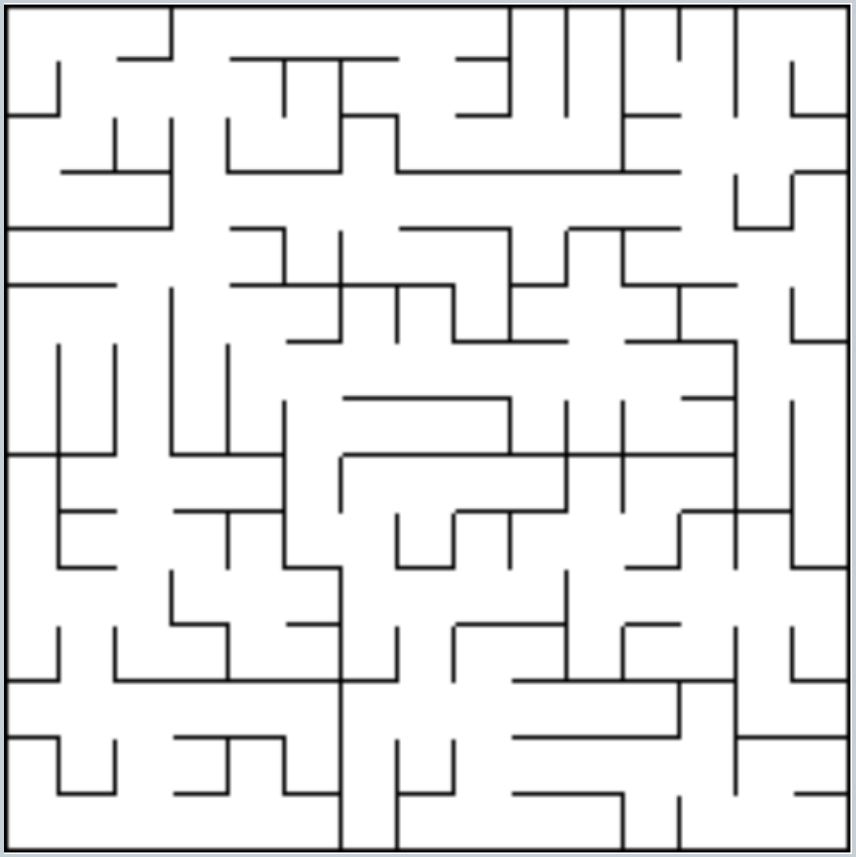
Recursive Division

- Pattern:
  - Lots of intersections between paths
  - Long straight lines of walls
- Algorithm:
  - Randomly places a wall dividing the maze in two and picks a random opening
  - Then does that same division with the two new sections



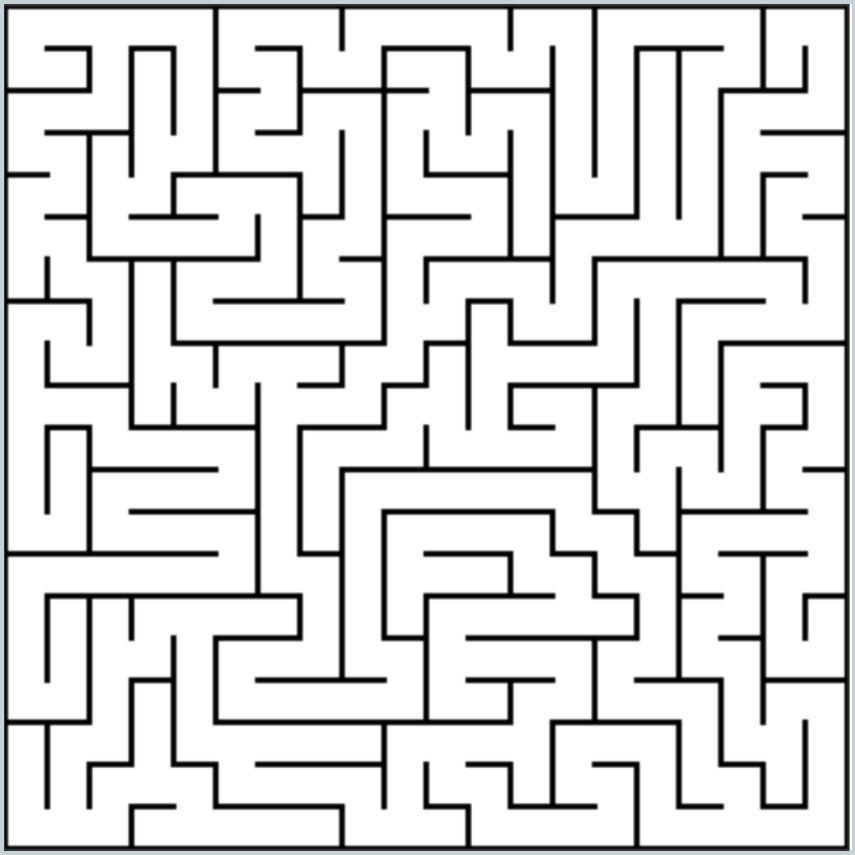
Prim's Algorithm

- Pattern:
  - Looks like a mix between previous two
  - Many intersections between paths
  - But shorter and more random wall segments
- Algorithm:
  - Picks a random starting point
  - Continuously picks a random cell from the unvisited neighbors of the current visited cells



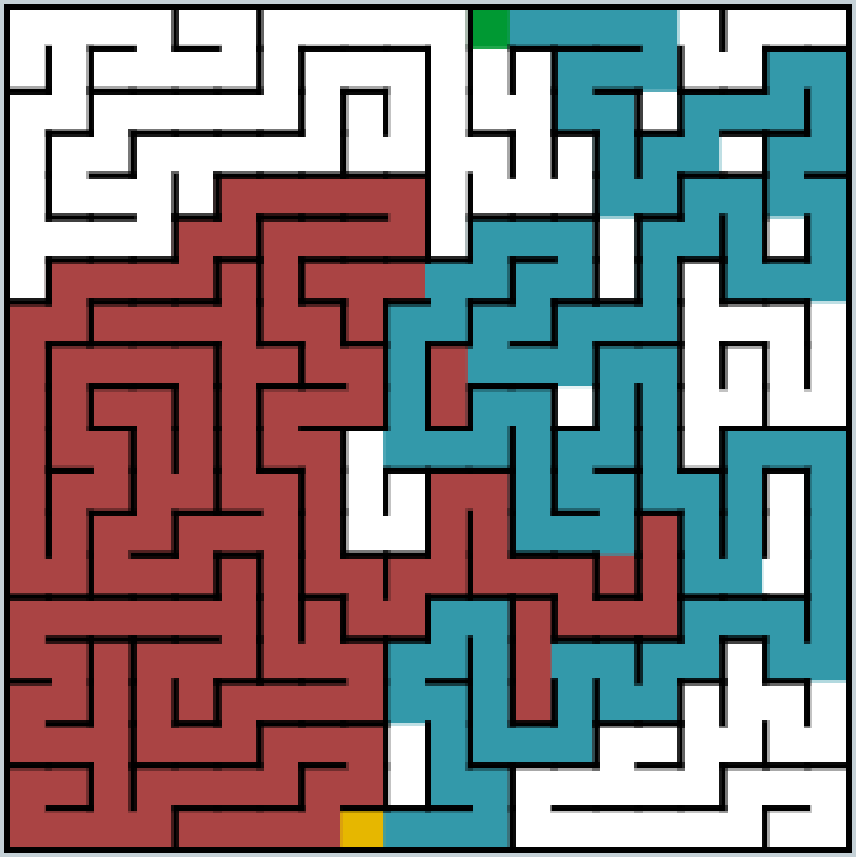
Growing Tree

- Pattern:
  - It depends on the parameters
  - Can look like Prim's, Recursive Back-tracker, both and more
- Algorithm:
  - Pick a random cell and store it in a list
  - Randomly pick more cells until no longer possible
  - Once a dead-end is hit, use some condition to pick the next cell to iterate from, ex:
    - Most recent added cell: performs like a back-tracker
    - Random cell: looks similar to Prim's algorithm



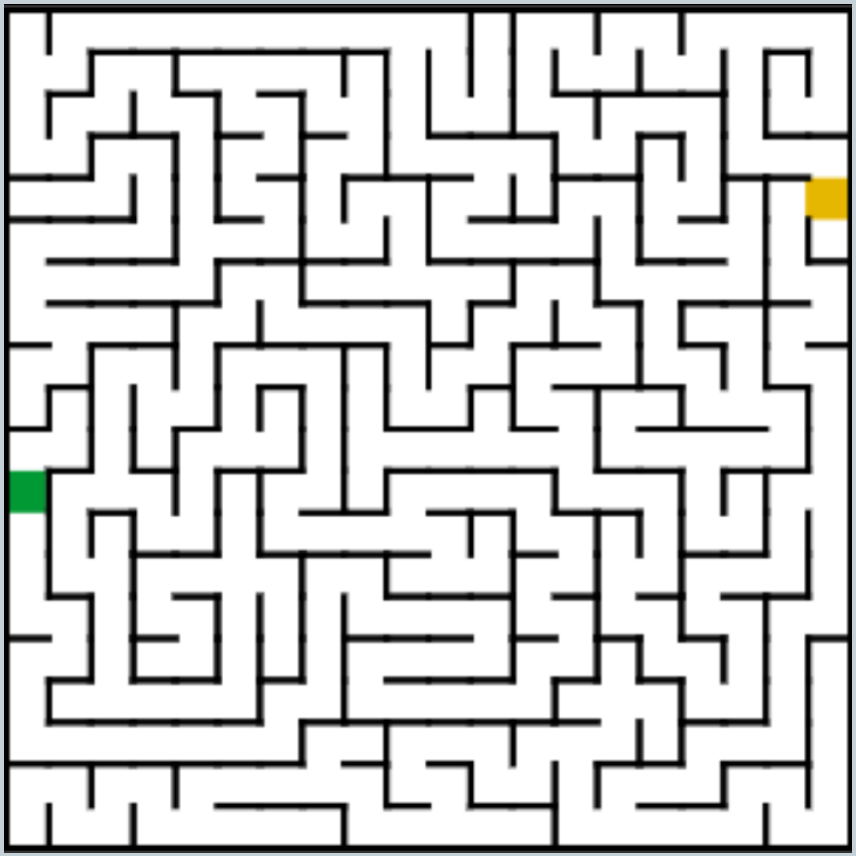
Solution Algorithm

- Implemented with a depth-first search
  - Starts at the start (yellow cell)
  - Recursively checks if each neighbor cell can lead to the path
  - Checks neighbors one at a time; the order of neighbors traversed may help efficiency
  - Stops once the end (green cell) is reached, marks solution path



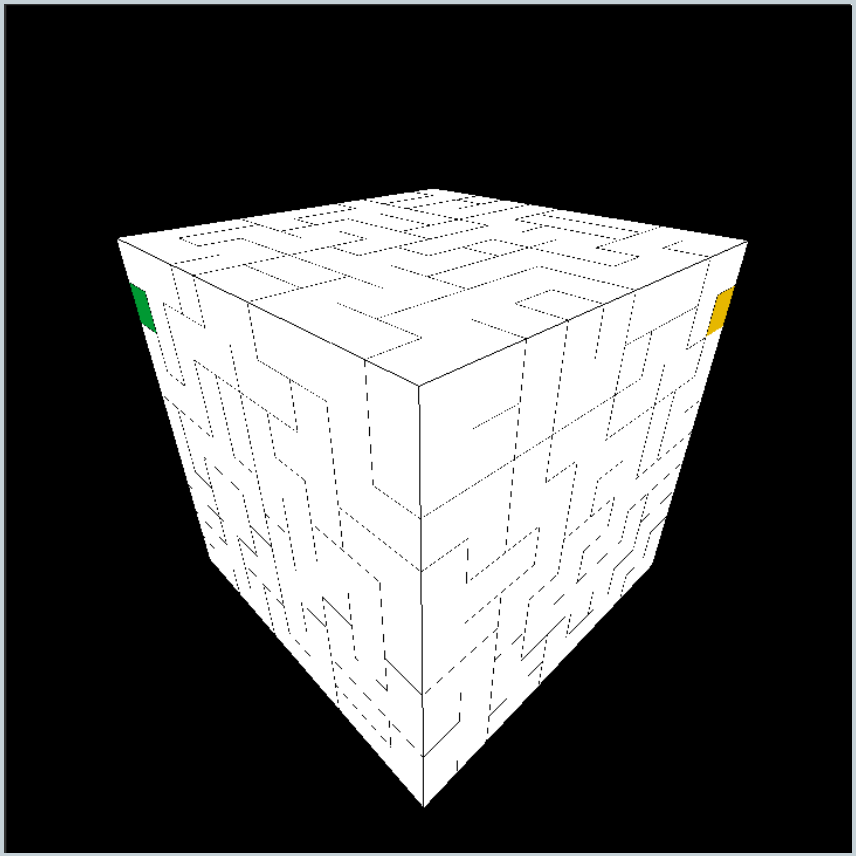
2D Mazes

- Displayed using canvas in HTML5
- Interactivity:
  - Change size
  - Change generation algorithm
  - Solve manually/automatically
  - Reset
  - Zoom in/out, pan over maze



3D Mazes

- Displayed using WebGL and ThreeJS
- Interactivity:
  - Change size
  - Change generation algorithm
  - Solve manually/automatically
  - Rotate maze, zoom camera
  - Switch to layer view



Questions/Comments

References

- Maze Algorithms:
  - https://en.wikipedia.org/wiki/Maze generation algorithm
  - http://weblog.jamisbuck.org/2011/2/7/maze-generation-algorithm-recap
- Images:
  - Recursive Division Completed Example
    - http://weblog.jamisbuck.org/2011/1/12/maze-generation-recursive-division-algorithm
  - Prim's Algorithm Completed Example
    - http://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm
  - Other Images: Robert Kaufman