As you walk in

Quiz will start at the beginning of lecture

- Have pencil/pen ready
- Don’t use your own scratch paper
  - We have some if you need it
- Silence phones

Quiz

Before we start

- Don’t open until we start
- Make sure your name and Andrew ID are on the front
- Read instruction page
- No questions (unless clarification on English)

Additional info

- 20 min

15-112
Lecture 2

Week 10 Tue

Backtracking & Object-Oriented Programming 2

Instructor: Pat Virtue

Announcements

Last quiz!

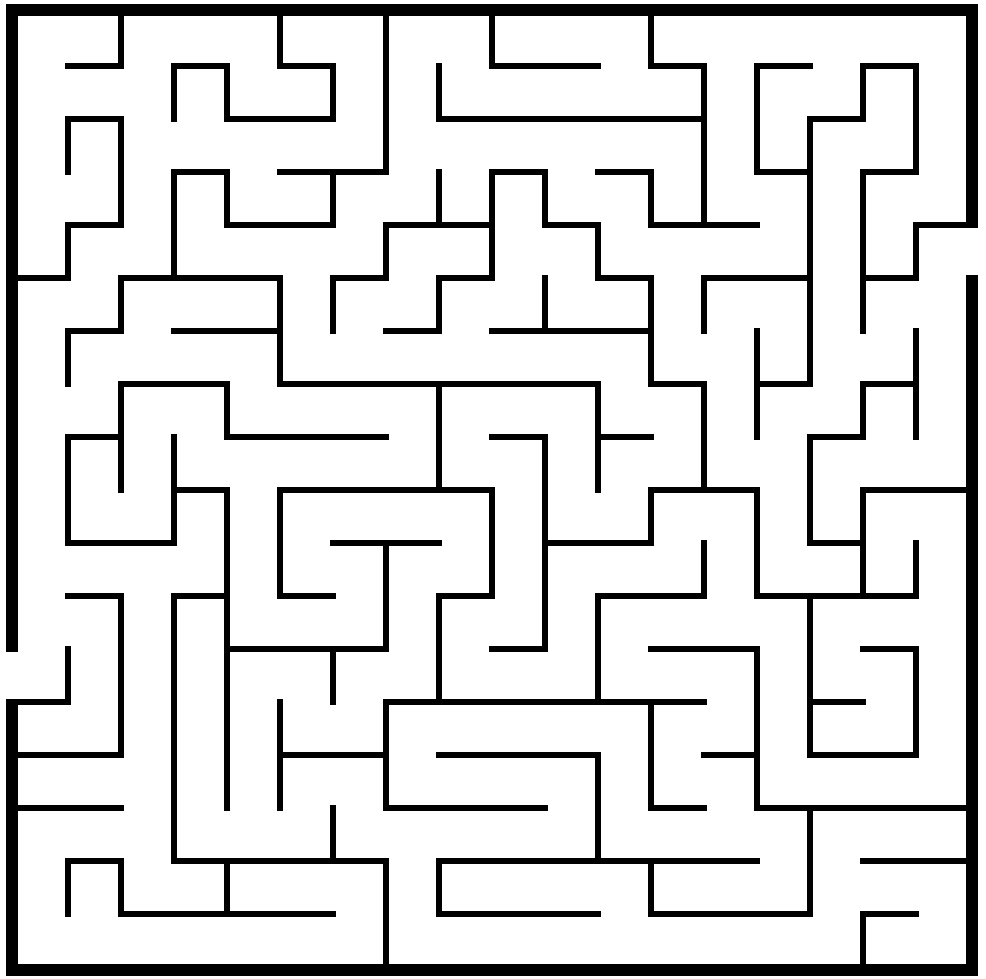
Hack 112!

TP

- Wed: decision form due
- Sat: last day for tech demos
- No grace days
- Extensions extremely rare
- Cite everything!

Backtracking

Incredibly generic problem-solving algorithm



Backtracking: N-Queens Example

N-by-N chessboard

Place exactly N queen pieces on the board, such that no queens are in positions to attack each other

- Queens can move any number of spaces:
  - Horizontally
  - Vertically
  - Diagonally

Backtracking: N-Queens Example

solve(board)

1. If all Qs placed

   Return board as solution!

2. For each valid action
   a) Apply action
   b) Recurse: result = solve(board)
   c) If result is success

      Return result

   Else

      Undo action

3. Return failure

Backtracking: N-Queens Example

Backtracking: N-Queens example

Code demo

https://www.cs.cmu.edu/~112/notes/notes-recursion-part2.html#nQueens

Backtracking: Solving maze example

Start: top-left

Goal: bottom-right

Strategy

- Path: Keep ordered list of locations representing the current path
- Visited: Avoid revisiting same locations by storing
- Try actions in order: N, S, E, W
- Recursively solve from next location

Backtracking: Solving maze example

solve(maze, path, visited)

1. If at goal

   Return path as solution!

2. For each valid action
   a) Apply action
   b) Recurse:

      result = solve(maze, path, visited)

   c) If result is success

      Return result

   Else

      Undo action

3. Return failure

Backtracking pattern

solve(maze, path, visited) — Maze

1. If at goal

   Return path as solution!

2. For each valid action
   a) Apply action
   b) Recurse:

      result = solve(maze, path, visited)

   c) If result is success

      Return result

      Else

      Undo action

3. Return failure

solve(board) — N-Queens

1. If all Qs placed

   Return board as solution!

2. For each valid action
   a) Apply action
   b) Recurse:

      result = solve(board)

   a) If result is success

      Return result

      Else

      Undo action

3. Return failure

OOP – next level

Special methods

__str__(self):

__repr__(self):

__eq__(self, other):

OOP: Inheritance