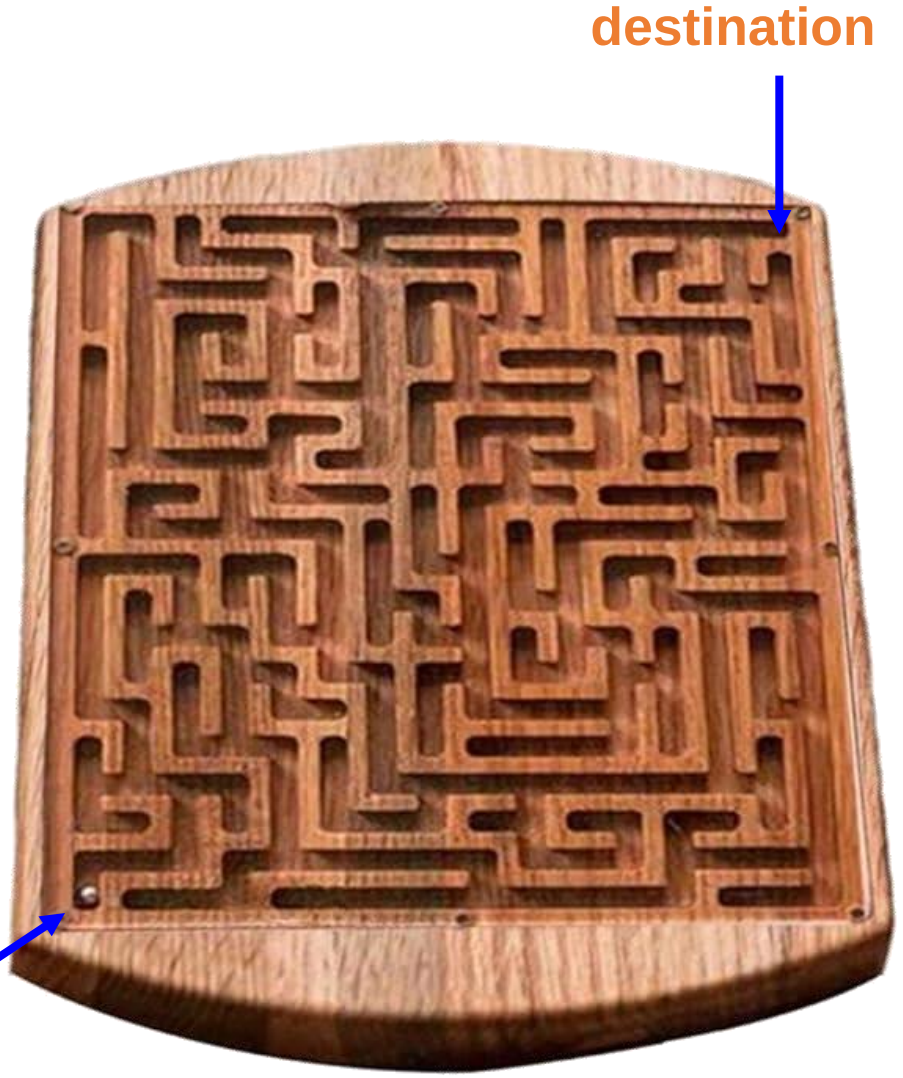
Data Structures
Programming Project #1

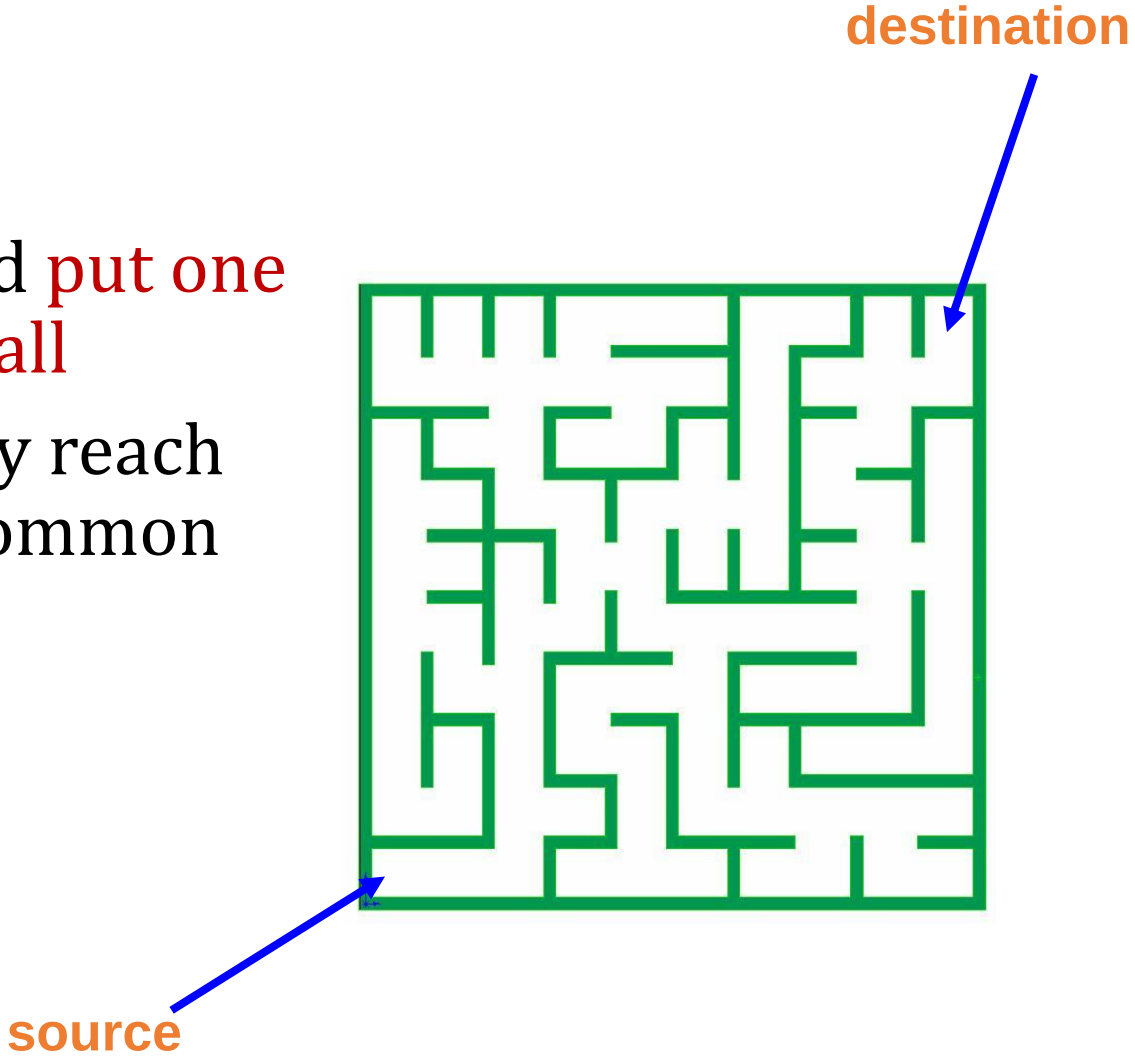
Rolling Ball Maze

- A maze:
- Navigate the ball through the maze
- Try and get the minimum number of steps to arrive the destination



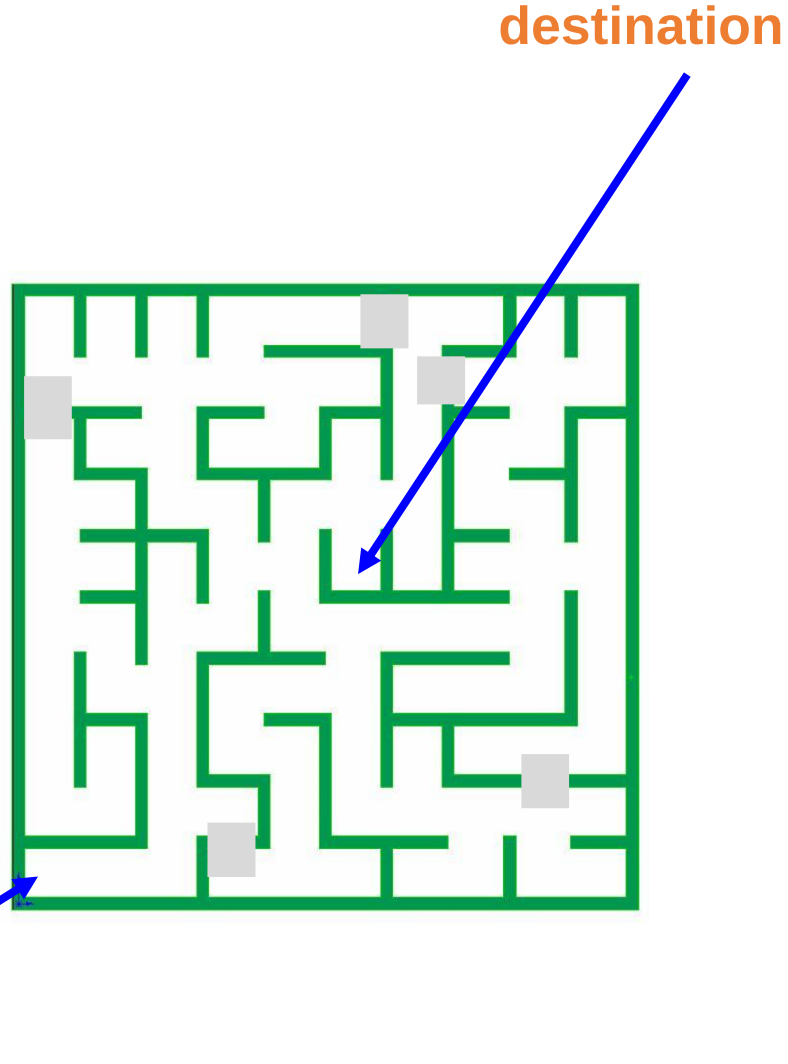
Rolling Ball Maze – The First Strategy

- A simple strategy:
- Close your eye and put one hand along one wall
- You will eventually reach the end of most common mazes



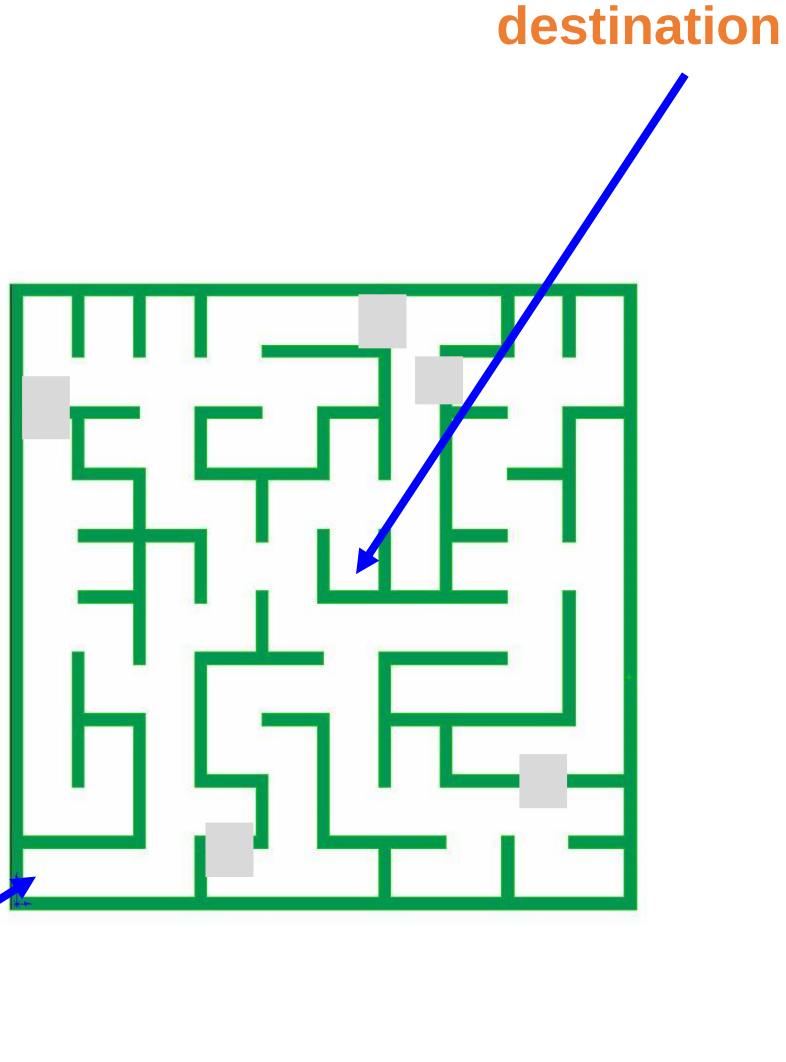
Rolling Ball Maze – The First Strategy

- A simple strategy:
- Close your eye and put one hand along one wall
- You will eventually reach the end of most common mazes
- But the goal may be away from the edges while some walls are removed



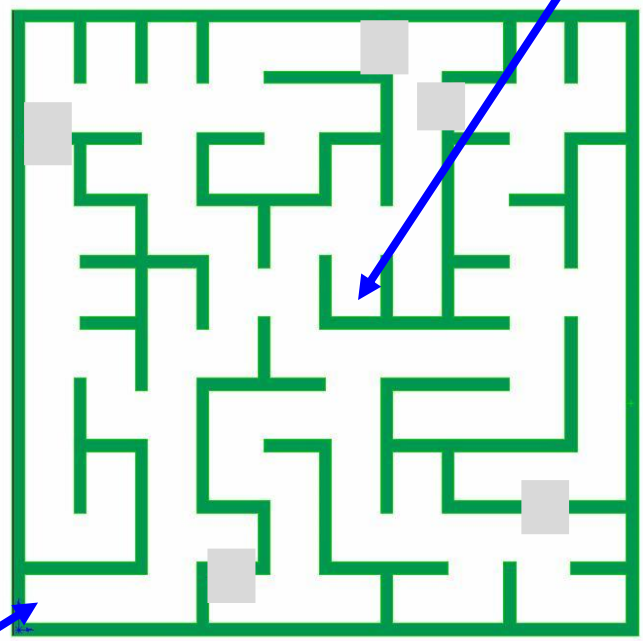
Rolling Ball Maze – The Second Strategy

- Depth-First Search:
- Run through the maze
- Take note of every fork in the road
- Whenever you reach a dead end or a loop, go back to the last intersection and try a different path
- But it might not be the shortest route



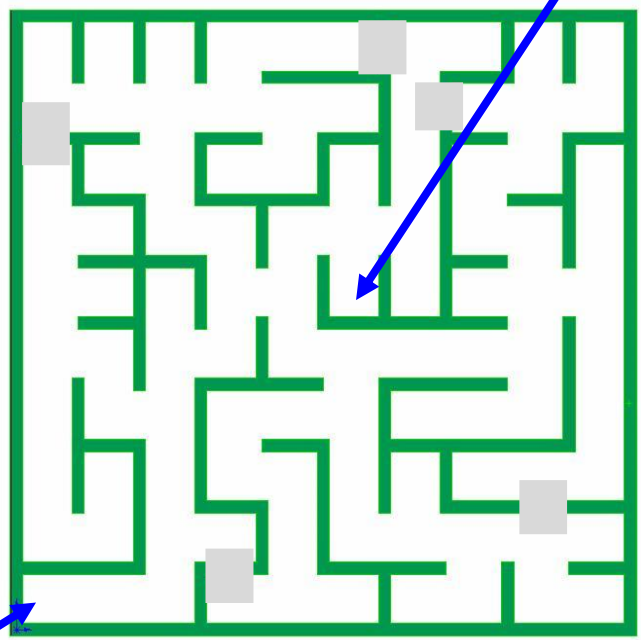
Rolling Ball Maze – The Third Strategy

- Breadth-First Search:
- Run through the maze
- Number every grid from the source until reaching the destination
- It can find the shortest route



Rolling Ball Maze – The Third Strategy

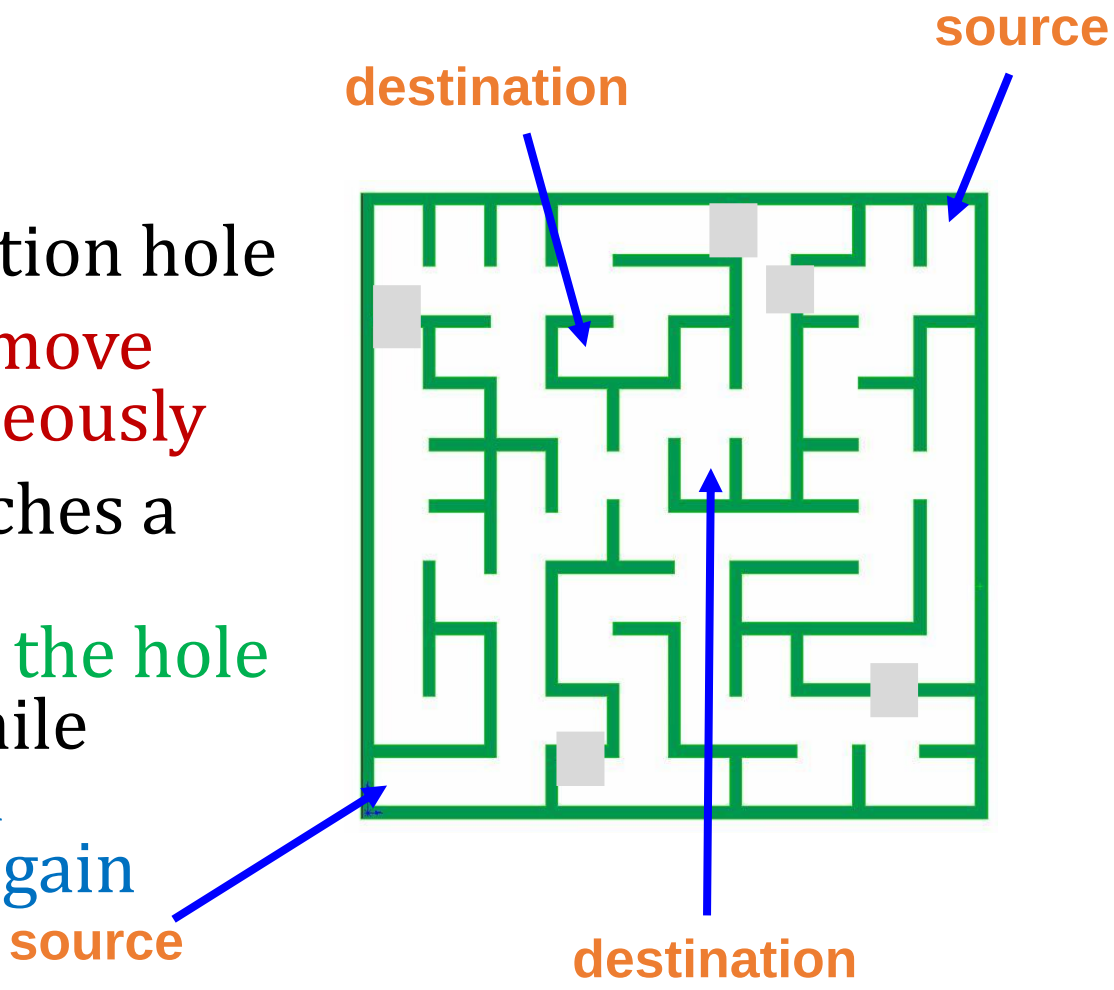
- Breadth-First Search:
- Run through the maze
- Number every grid from the source until reaching the destination
- It can find the shortest route
- But it's not fun



One Day I Saw…

Rolling Ball Maze

- How about this?
- One more ball
- One more destination hole
- Your control will move the balls simultaneously
- Once one ball reaches a destination, then
  1) the ball will fill the hole and disappear, while
  2) the destination will not be filled again



Problem Formulation

- Given:
- A $n \times n$ maze with two sources and two destinations

- Goal: minimize the completion time
- Constraints:
- Each action moves a ball to its nearby grid (i.e., up, down, left, right)
- Each action moves the two balls simultaneously
- Balls cannot walk through any wall
- Balls can overlap (for simplicity)
- Once a ball reaches a destination, the ball and destination will disappear

Programming Project #1: Rolling Ball Maze with Two Balls

- Input:
  - A maze represented by a 0-1 matrix
  - Two sources and two destinations
- Procedure:
  - Computer the sequence of actions (i.e., up, down, left, right) to move the two balls toward the destinations
- Output:
  - The sequence of actions
- The grade is inversely proportional to the number of actions
- We have a competition (see next page)

The Competition

- The grade is inversely proportional to # actions
- Basic: 75 (deadline)
  - Any solution that can complete the mazes
- Performance ranking (decided after the deadline)
  - [0%, 50%) (bottom): +0
  - [50%, 75%): + 5
  - [75%, 90%): + 9
  - [90%, 95%): + 12
  - [95%, 100%] (top): + 15
- Homework assistant (superb deadline)
  - +10

The Competition

- The grade
- Basic: 75
  - Any solu
- Performa
  - [0%, 50
  - [50%, 75
  - [75%, 90
  - [90%, 95
  - [95%, 10
- Homewor
  - +10

actions

deadline)

The Competition

- The grade
- Basic: 75
  - Any solu
- Performa
  - [0%, 50
  - [50%, 75
  - [75%, 90
  - [90%, 95
  - [95%, 10
- Homewor
  - +10

actions

相信你們在做完作業以後

The Baseline Algorithm

- Use BFS to find the path that moves one ball to the destination
- Use BFS to find the path that moves the other ball to the destination

Input Sample: use scanf

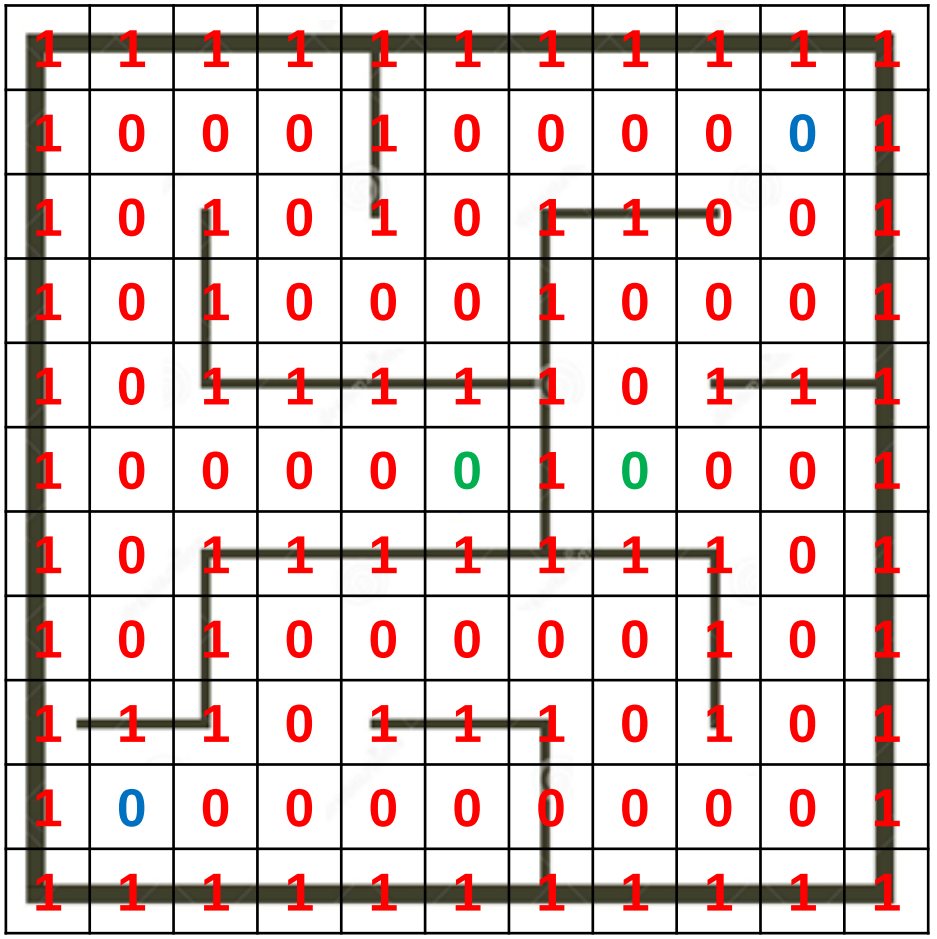
Format:

n

The map information

Src1 Src 2

Dst1 Dst2




```
Ex:
11
"11x11 matrix"
1 1 9 9
5 5 7 5
```

Output Sample: use printf

Format:

The sequence that completes the game
Ex: 0 1 2 3…

0: up
1: right
2: down
3: left

Note

- Superb deadline: 9/26 Tue (adjust?)
- Deadline: 10/3 Tue (adjust?)
- Pass the test of our online judge platform
- Submit your code to E-course2
- Demonstrate your code remotely with TA
- **C Source code (i.e., only .c)**
- Show a good programming style