CSE 250: Maze Exploration

Lecture 17

Oct 7, 2024

Reminders

- WA3 due Sun, Oct 13 at 11:59 PM

The Stack ADT

A stack of objects on top of one another.

- **Push**
  Put a new object on top of the stack.
- **Pop**
  Remove the object from the top of the stack.
- **Top**
  Peek at what's on top of the stack.

The Queue ADT

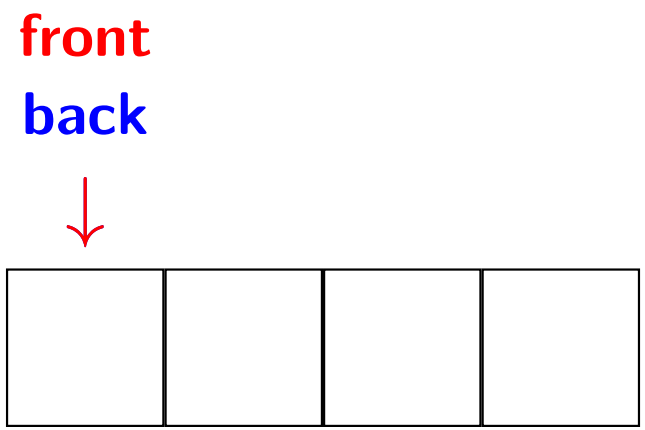
Outside of the US, "queueing" is lining up.

- **Enqueue** (`add(item)` or `offer(item)`)
  Put a new object at the end of the queue.
- **Dequeue** (`remove()` or `poll()`)
  Remove the object from the front of the queue.
- **Peek** (`element()` or `peek()`)
  Peek at what's at the front of the queue.

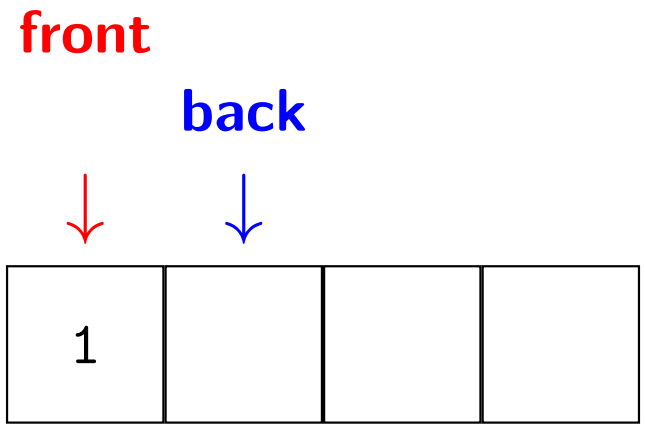
Queues vs Stacks

- **Queue**
  First in, First out (FIFO)
- **Stack**
  Last in, First out (LIFO, FILO)

Circular Buffer

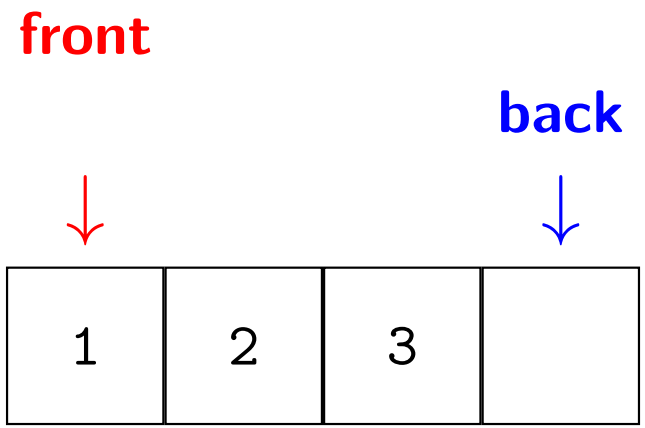


Circular Buffer



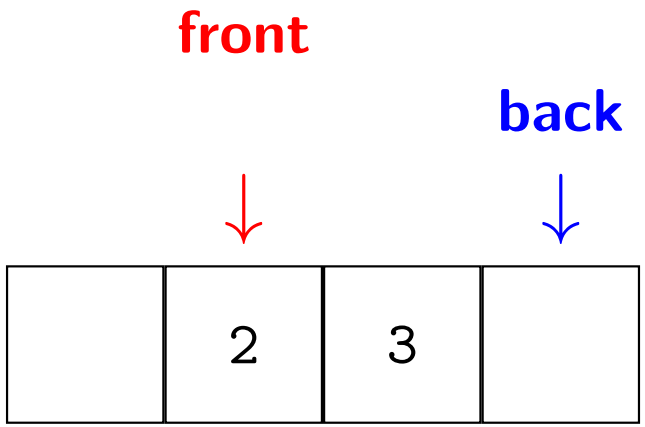

```
list.add(1)
```

Circular Buffer



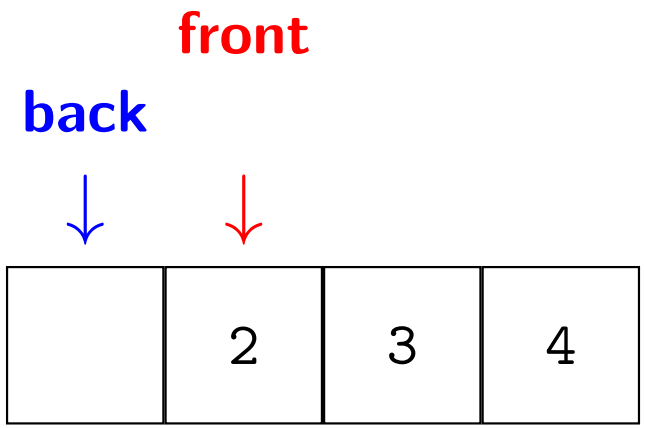

```
list.add(1)
list.add(2)
list.add(3)
```

Circular Buffer



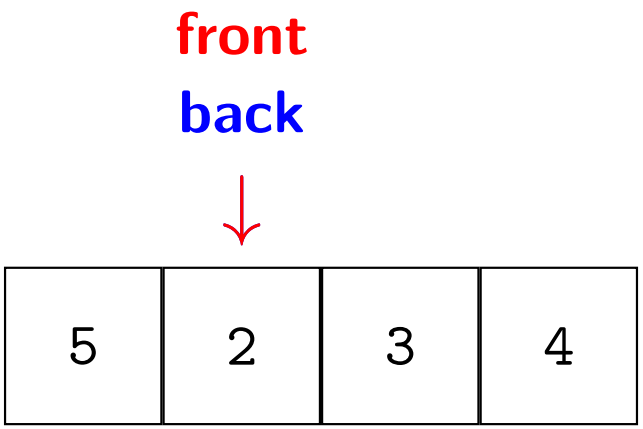

```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
```

Circular Buffer



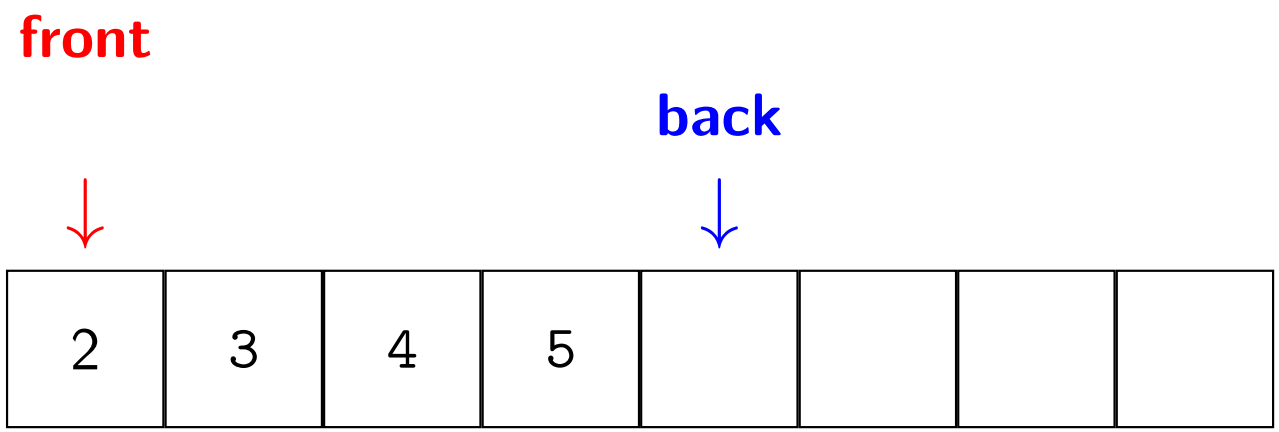

```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
```

Circular Buffer



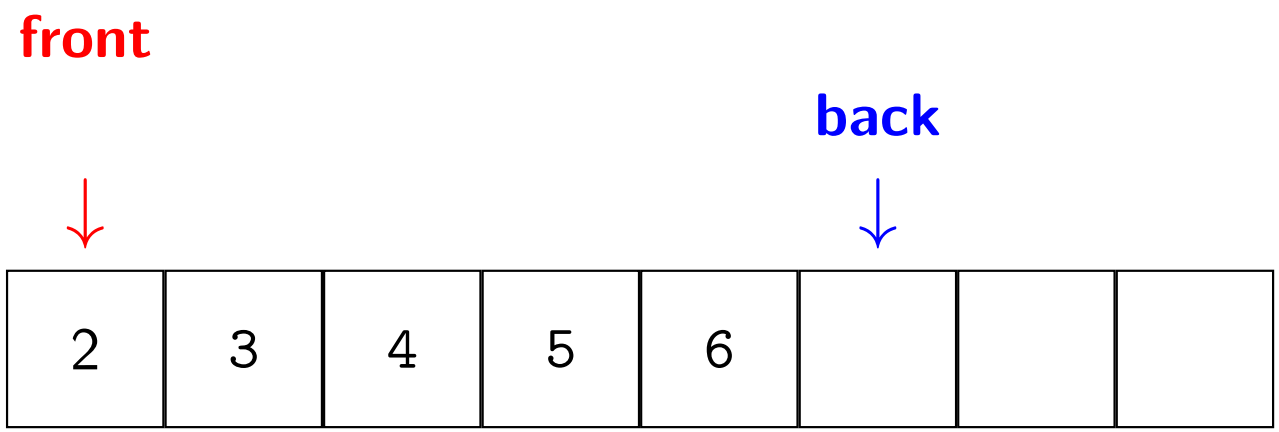

```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
list.add(5)
```

Circular Buffer



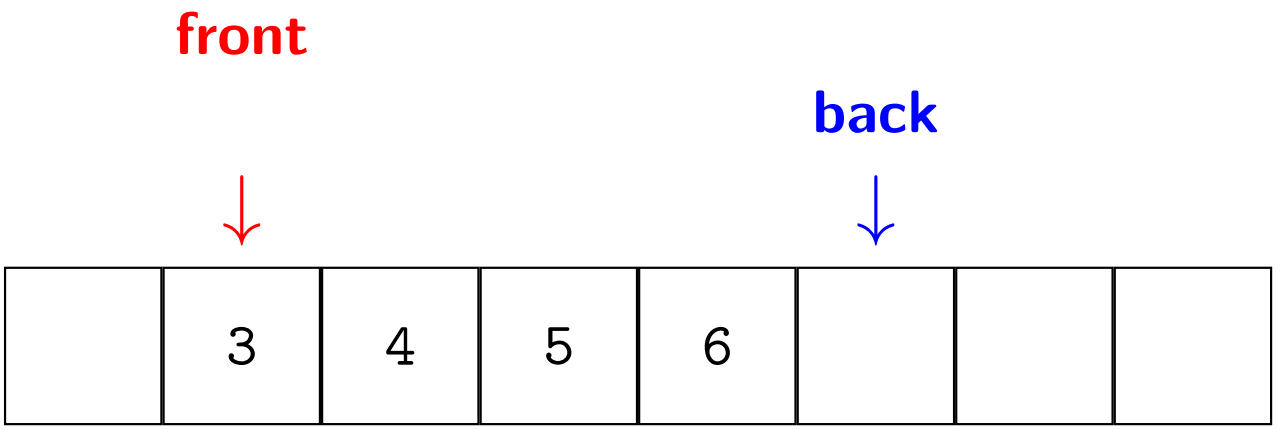

```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
list.add(5)
```

Circular Buffer



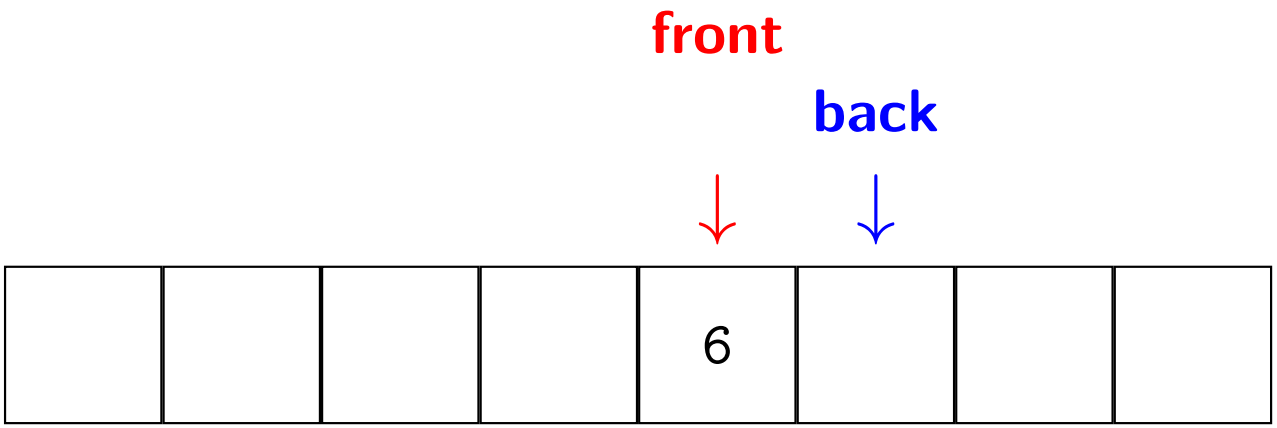

```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
list.add(6)
```

Circular Buffer



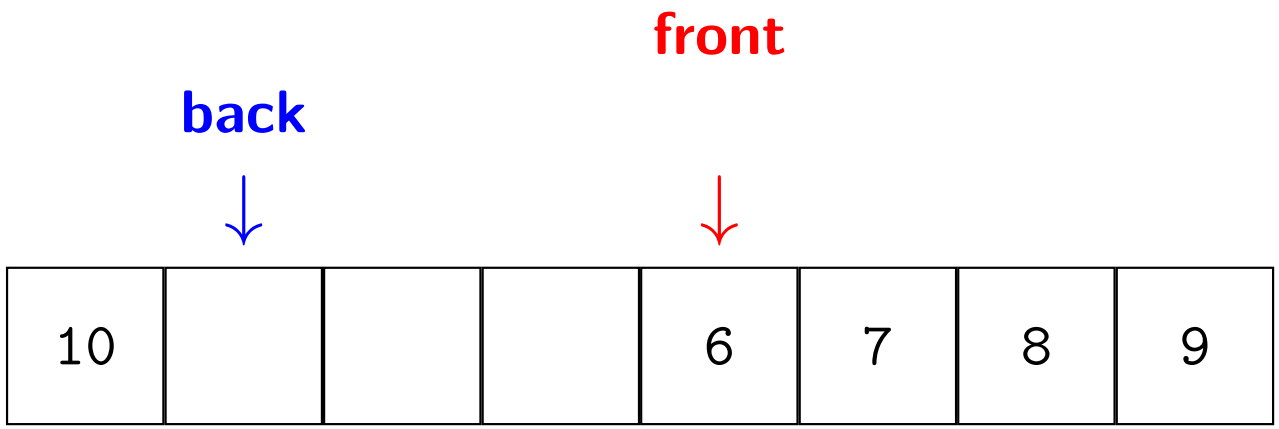

```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
list.add(6)
list.remove() → 2
```

Circular Buffer




```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
list.add(6)
list.remove() → 2
list.remove() → 3
list.remove() → 4
list.remove() → 5
list.add(7)
list.add(8)
list.add(9)
list.add(10)
```

Circular Buffer

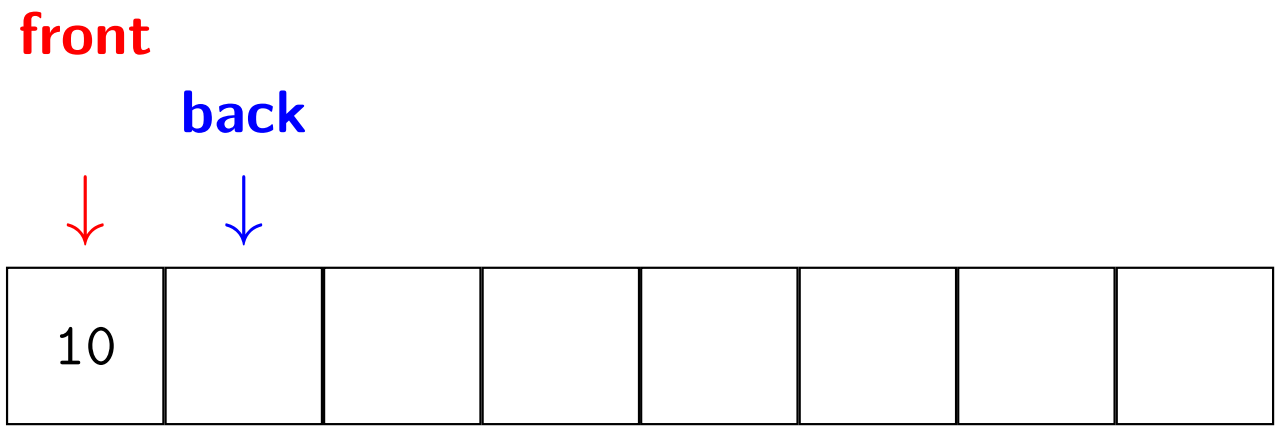



```
list.add(1)
list.add(2)
list.add(3)
list.remove() → 1
list.add(4)
list.add(6)
```

```
list.remove() → 2
list.remove() → 3
list.remove() → 4
list.remove() → 5
list.add(7)
list.add(8)
```

```
list.add(9)
list.add(10)
list.remove() → 6
list.remove() → 7
list.remove() → 8
list.remove() → 9
```

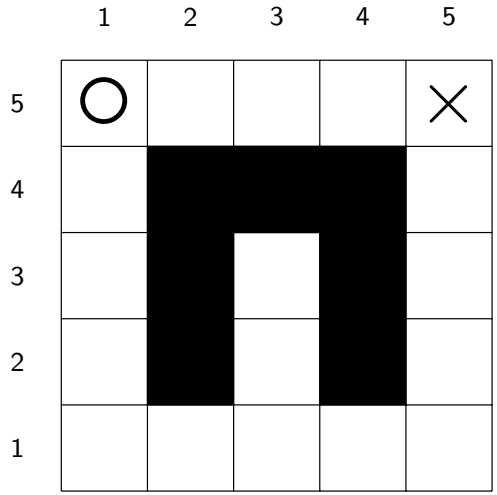
Circular Buffer




```
list.add(1)              list.remove() → 2   list.add(9)
list.add(2)              list.remove() → 3   list.add(10)
list.add(3)              list.remove() → 4   list.remove() → 6
list.remove() → 1        list.remove() → 5   list.remove() → 7
list.add(4)              list.add(7)         list.remove() → 8
list.add(6)              list.add(8)         list.remove() → 9
```

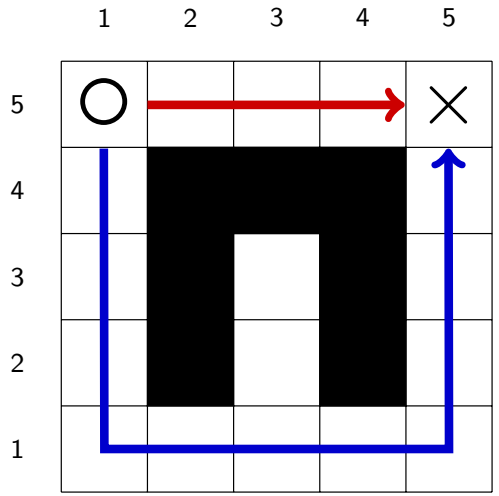
Mazes

- ○ is the start, × is the objective.
  - There may be multiple paths.
  - Generally, we want the shortest
- **Approach 1**: Take the first available route in one direction.
  - **Right, Down, Left, or Up**
  - **Down, Right, Up, or Left**

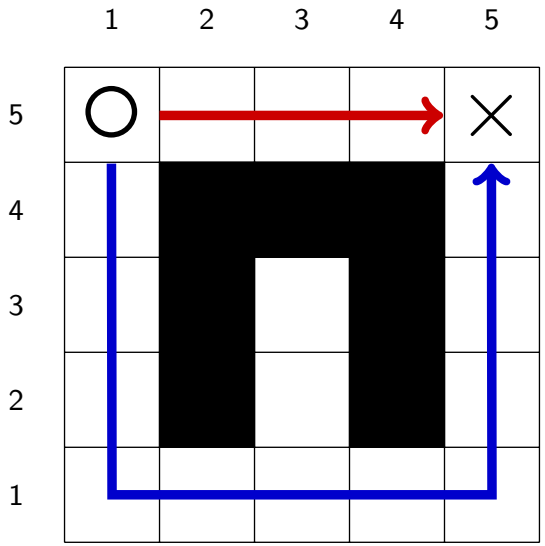


Mazes

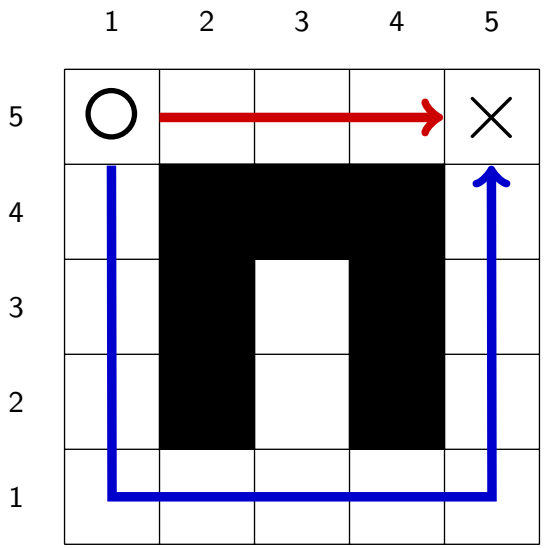
- ○ is the start, × is the objective.
  - There may be multiple paths.
  - Generally, we want the shortest
- **Approach 1**: Take the first available route in one direction.
  - **Right, Down, Left, or Up**
  - **Down, Right, Up, or Left**



Mazes

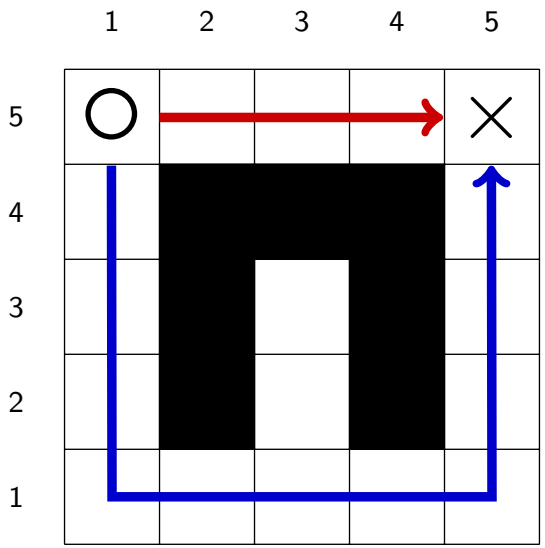


Mazes



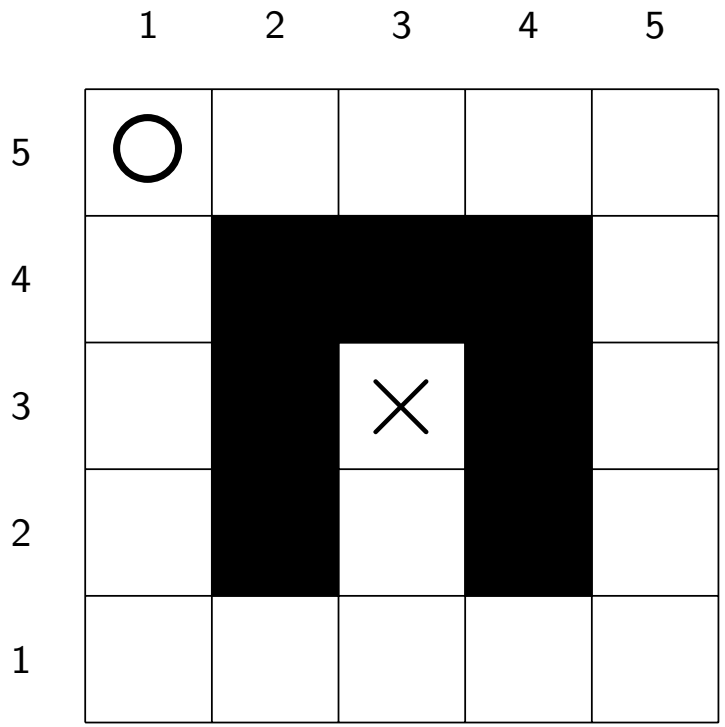
- How do we know which one is best?

Mazes

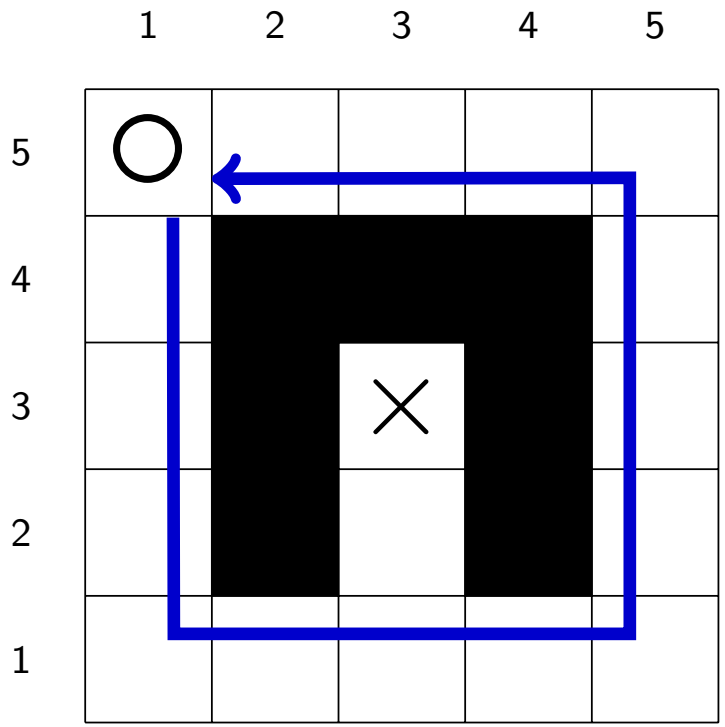


- How do we know which one is best?
- Are there any other problems?

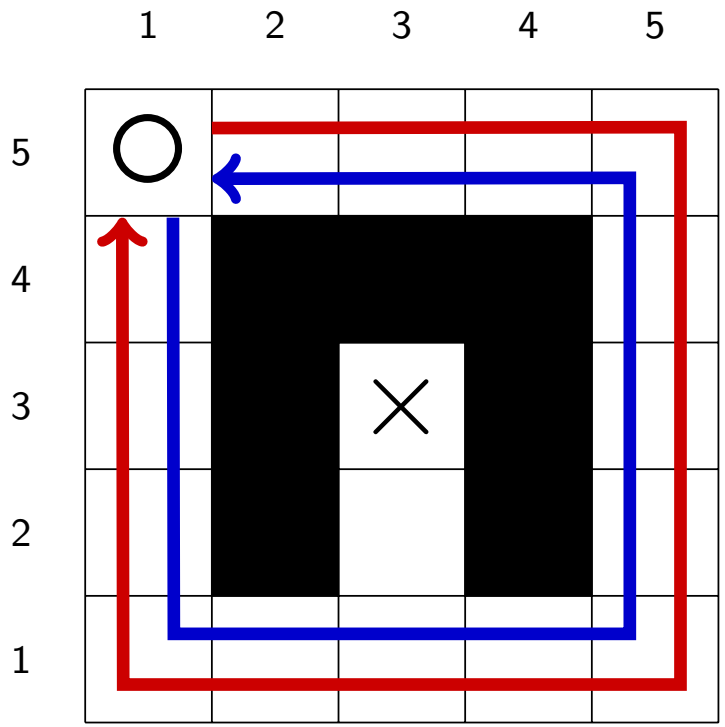
Mazes



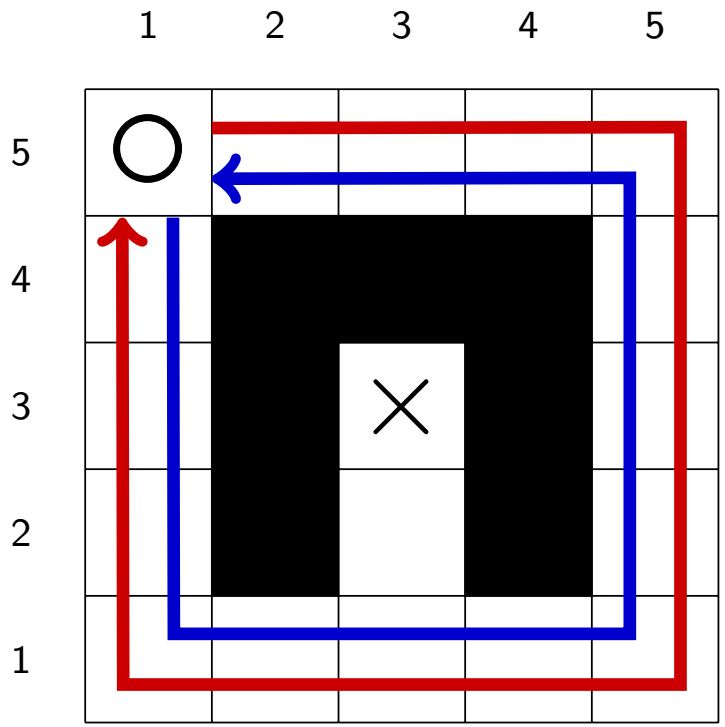
Mazes



Mazes



Mazes



- Priority order doesn't guarantee exploring the entire maze

Formalizing Maze Solving

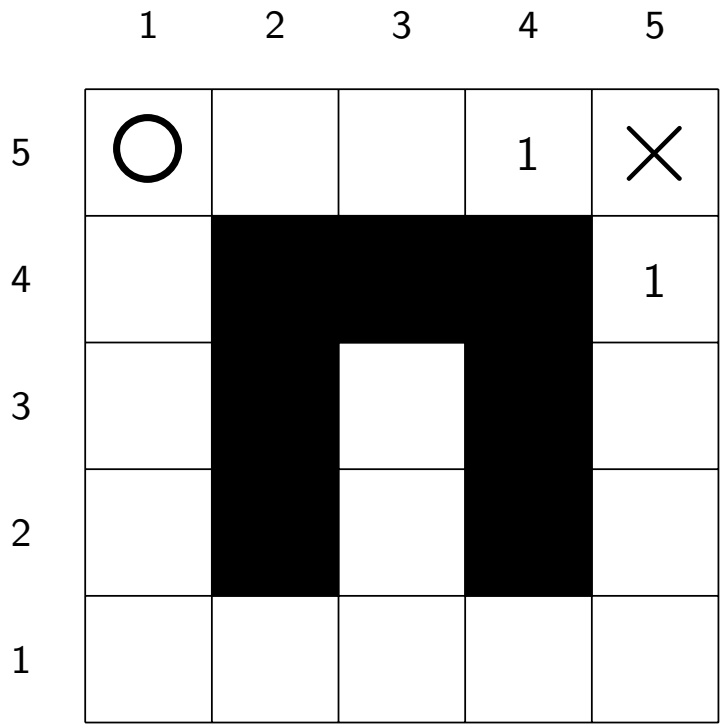
- **Inputs**
  - The map: An $n \times m$ grid of filled/empty squares.
  - The $\circ$ is at position `start`
  - The $\times$ is at position `dest`
- **Goal**
  - Compute `steps`(`start`, `dest`), the minimum steps from start to end.

Formalizing Maze Solving

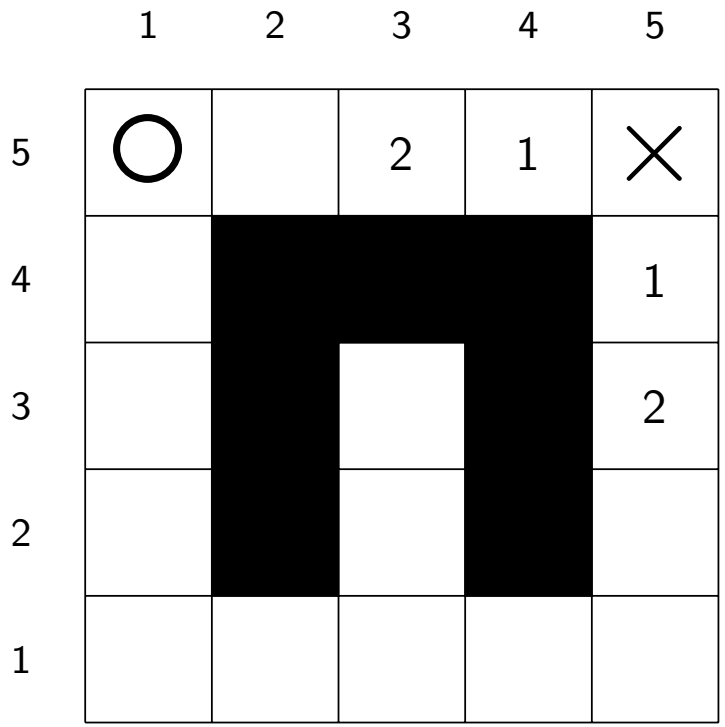
- **Inputs**
  - The map: An $n \times m$ grid of filled/empty squares.
  - The $\circ$ is at position `start`
  - The $\times$ is at position `dest`
- **Goal**
  - Compute `steps`(`start`, `dest`), the minimum steps from start to end.

How do we define `steps`?

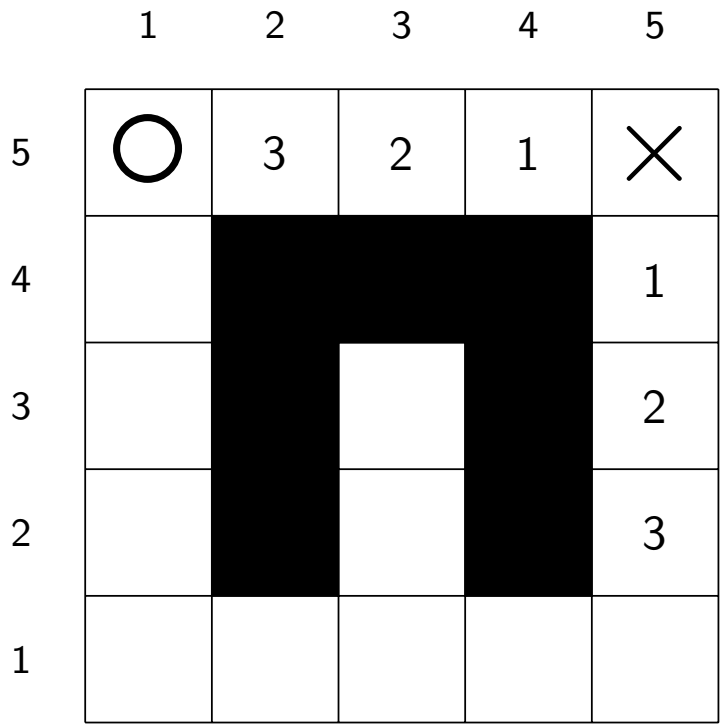
Formalizing Maze Solving



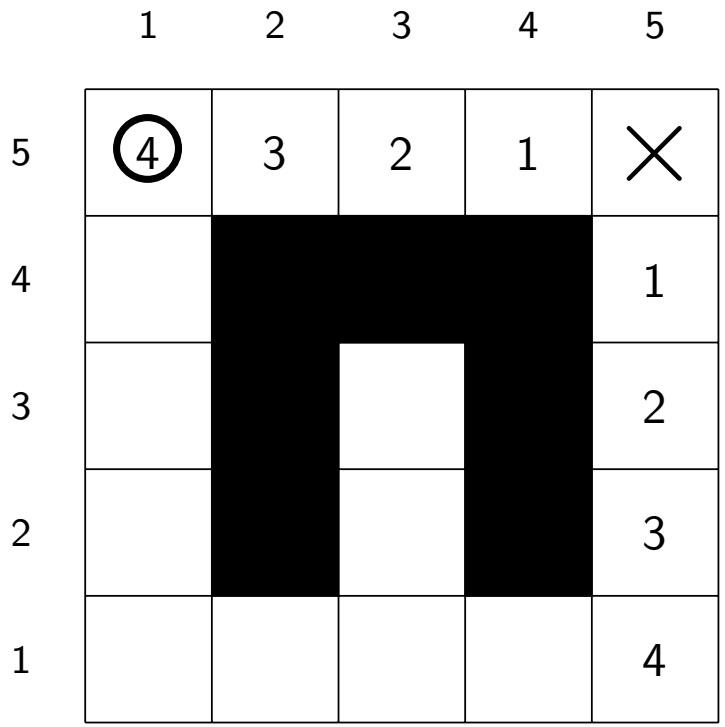
Formalizing Maze Solving



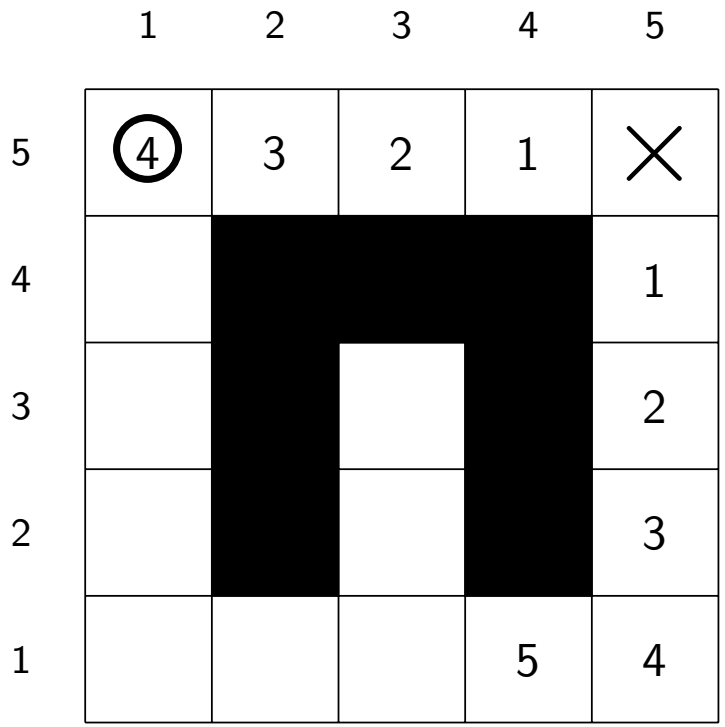
Formalizing Maze Solving



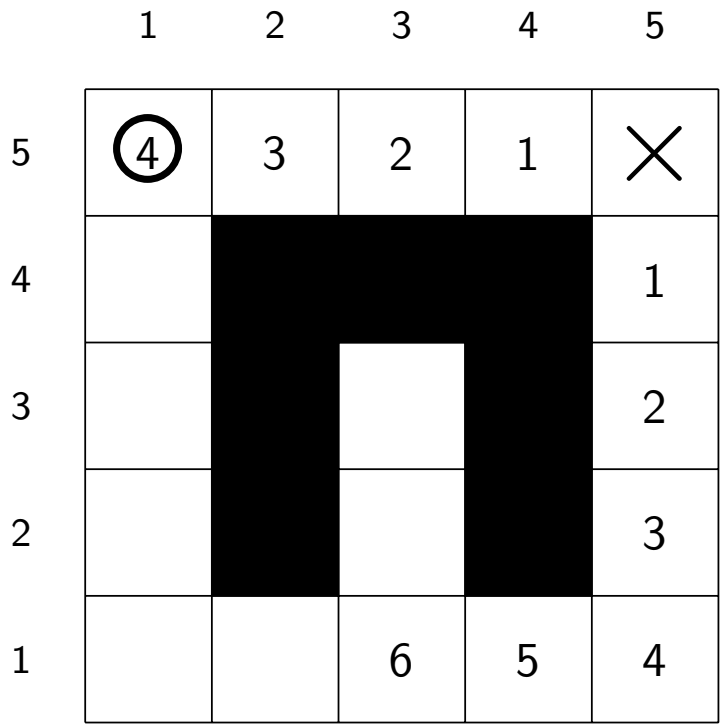
Formalizing Maze Solving



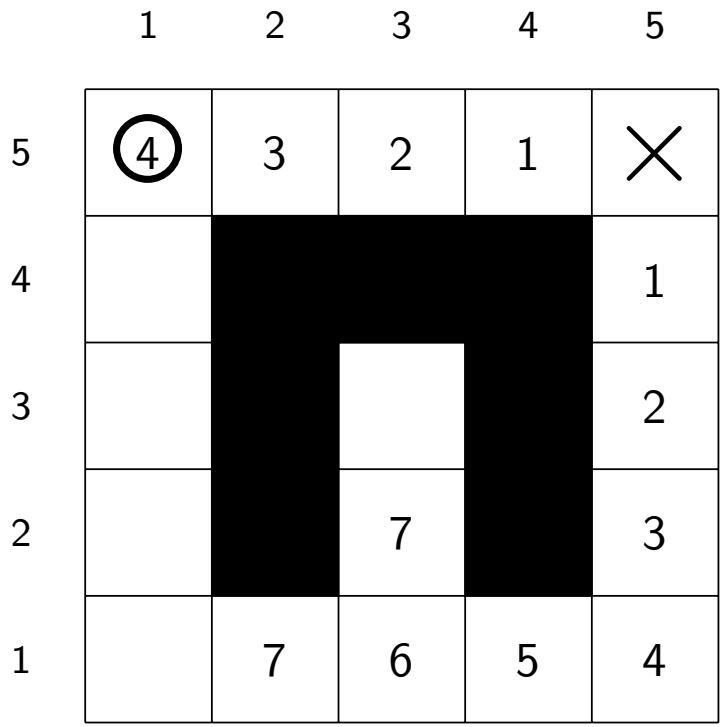
Formalizing Maze Solving



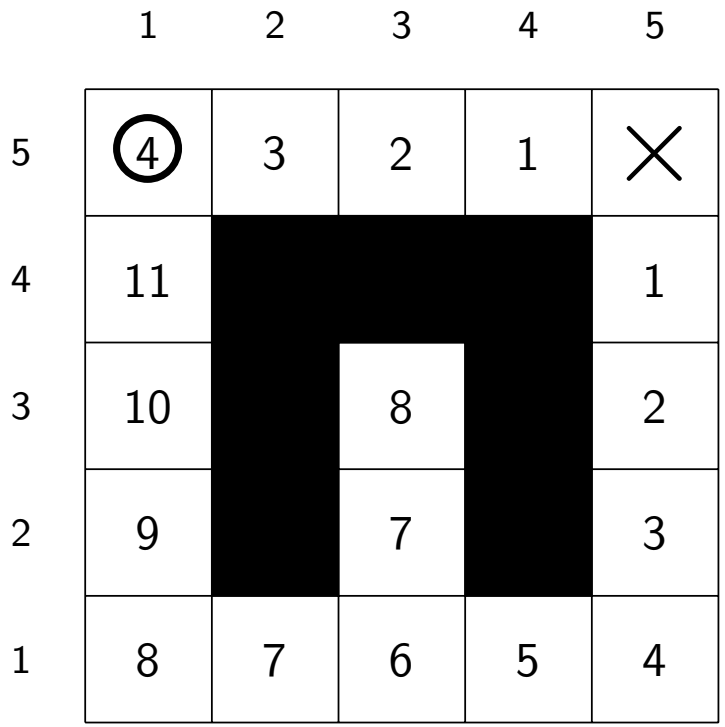
Formalizing Maze Solving



Formalizing Maze Solving



Formalizing Maze Solving



Formalizing Maze Solving

	1	2	3	4	5
5	(4)	3	2	1	✕
4	11	∞	∞	∞	1
3	10	∞	8	∞	2
2	9	∞	7	∞	3
1	8	7	6	5	4

Formalizing Maze Solving

$$\texttt{steps}(\texttt{pos}, \texttt{dest}) = \begin{cases} 0 & \textbf{if } \texttt{pos} = \texttt{dest} \end{cases}$$

Formalizing Maze Solving

$$\texttt{steps}(\texttt{pos}, \texttt{dest}) = \begin{cases} 0 & \textbf{if } \texttt{pos} = \texttt{dest} \\ \infty & \textbf{if } \texttt{pos} \textbf{ is filled} \\ \end{cases}$$

Formalizing Maze Solving

$$\texttt{steps}(\texttt{pos}, \texttt{dest}) = \begin{cases} 0 & \textbf{if } \texttt{pos} = \texttt{dest} \\ \infty & \textbf{if } \texttt{pos} \textbf{ is filled} \\ 1 + \texttt{min\_nearby}(\texttt{pos}, \texttt{dest}) & \textbf{otherwise} \end{cases}$$

Formalizing Maze Solving

$$\texttt{steps}(\texttt{pos}, \texttt{dest}) = \begin{cases} 0 & \textbf{if } \texttt{pos} = \texttt{dest} \\ \infty & \textbf{if } \texttt{pos} \textbf{ is filled} \\ 1 + \texttt{min\_nearby}(\texttt{pos}, \texttt{dest}) & \textbf{otherwise} \end{cases}$$

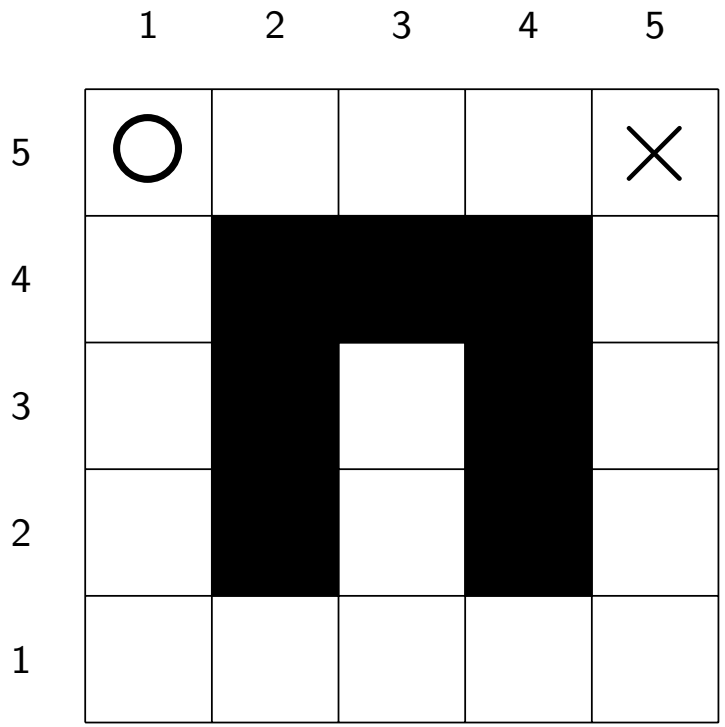
Where...

$$\texttt{min\_nearby}(\texttt{pos}, \texttt{dest}) = \min \begin{cases} \texttt{steps}(\texttt{moveDown}(\texttt{pos}), \texttt{dest}) \\ \texttt{steps}(\texttt{moveLeft}(\texttt{pos}), \texttt{dest}) \\ \texttt{steps}(\texttt{moveRight}(\texttt{pos}), \texttt{dest}) \\ \texttt{steps}(\texttt{moveUp}(\texttt{pos}), \texttt{dest}) \end{cases}$$

Formalizing Maze Solving

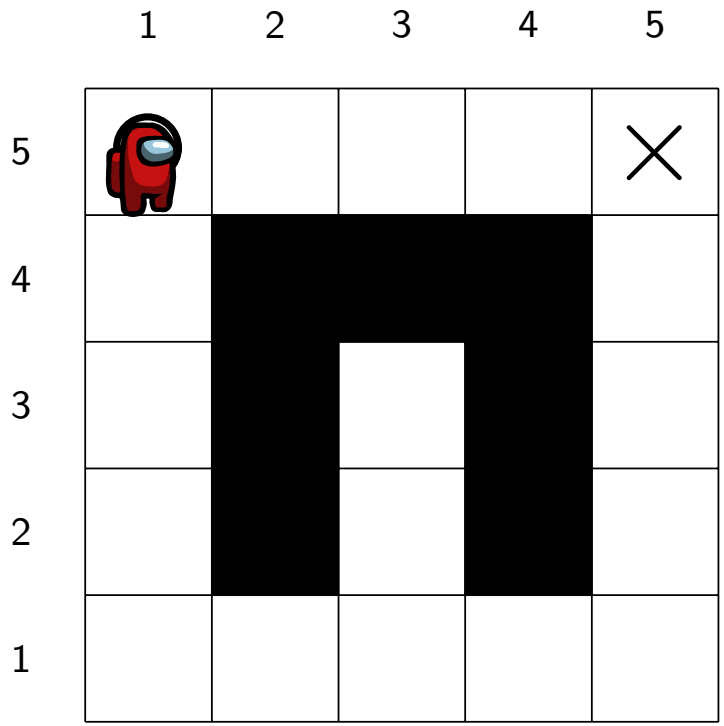
```
1   public int steps(Point pos, Point dest)
2   {
3     if(pos == dest){ return 0; }
4     else if(is_filled(pos){ return ∞; }
5     else {
6       return 1 + Math.min(
7         steps(pos.moveDown, dest),
8         steps(pos.moveLeft, dest),
9         steps(pos.moveRight, dest),
10        steps(pos.moveUp, dest)
11      );
12    }
13  }
```

Formalizing Maze Solving



Among Us Beans ©Innersloth LLC

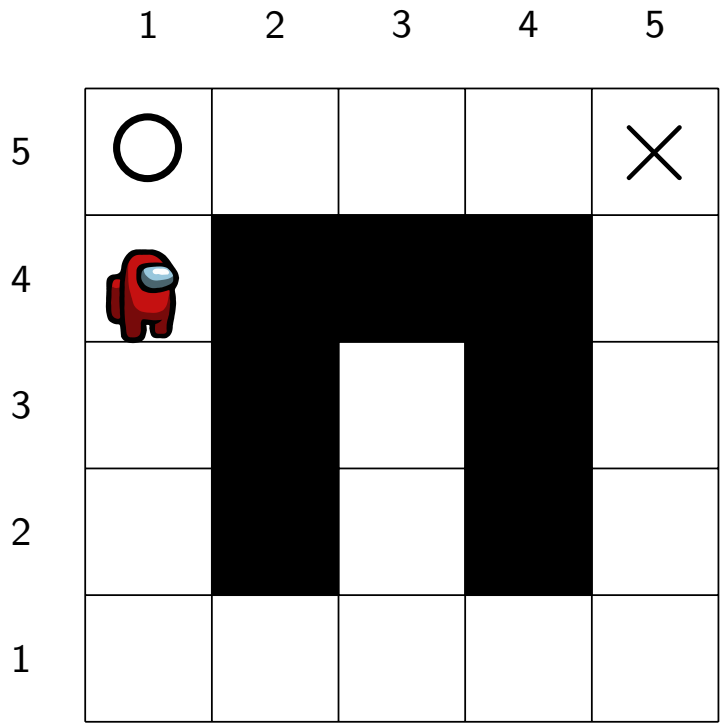
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

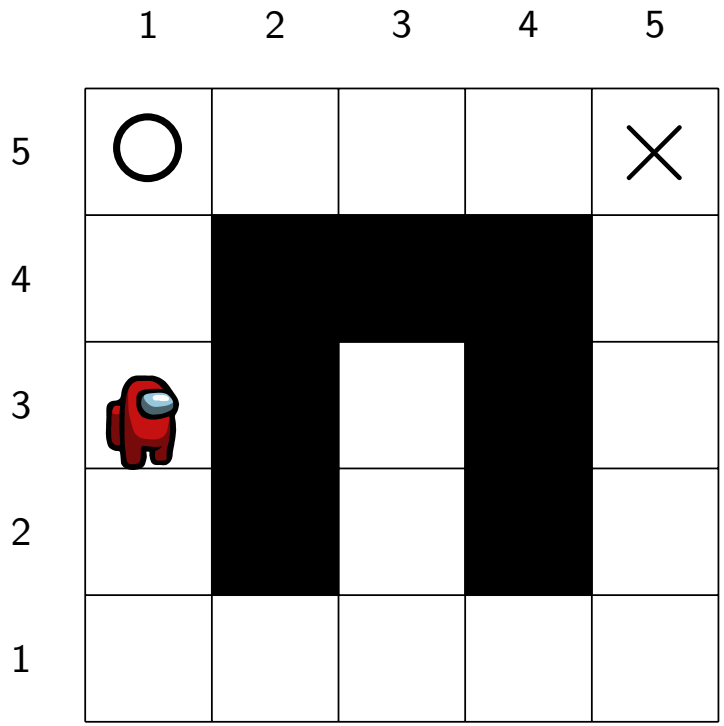
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

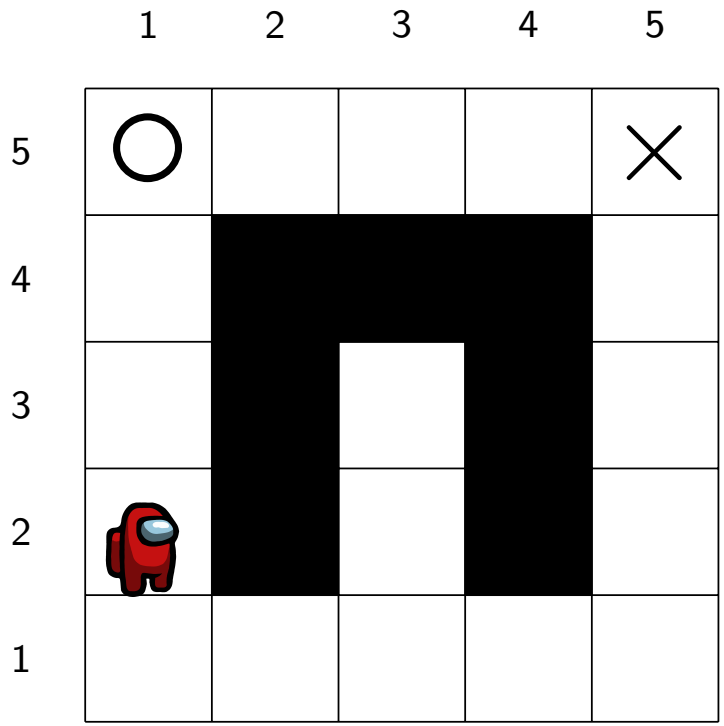
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

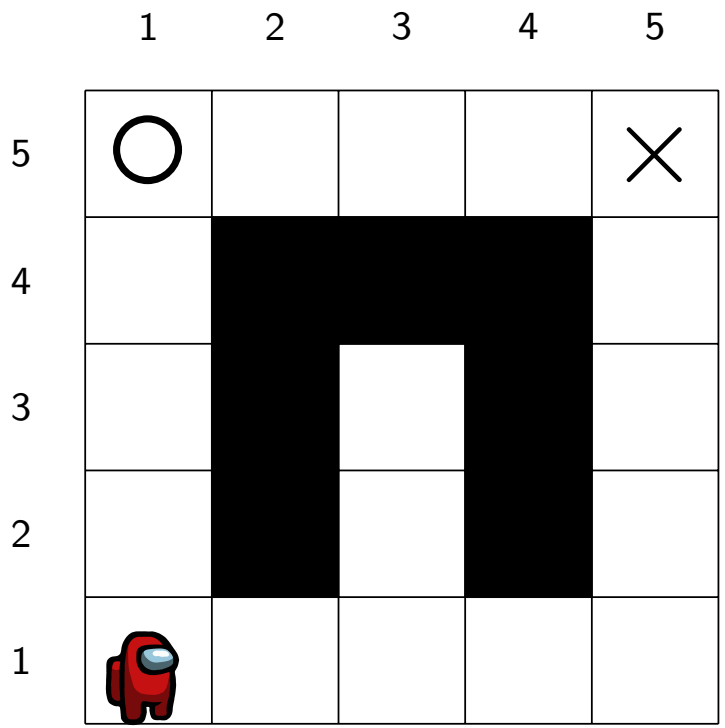
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

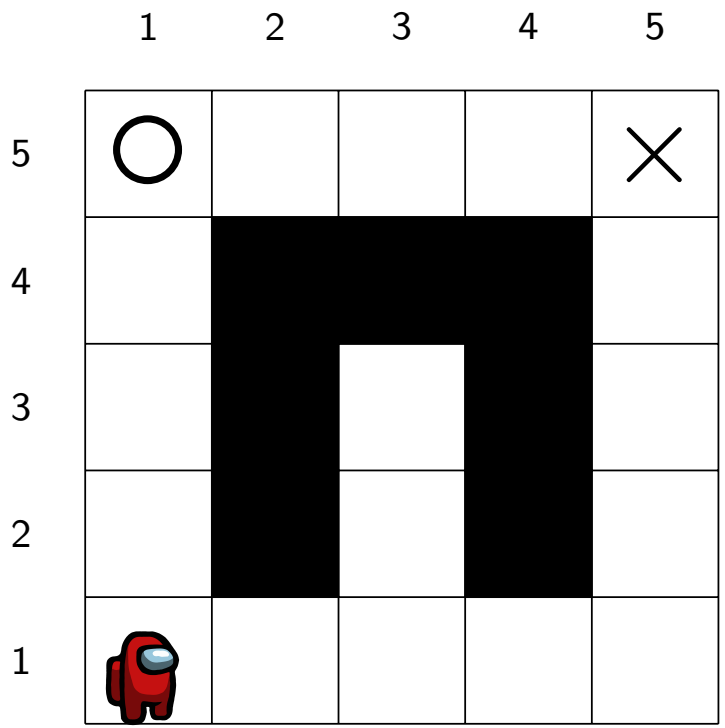
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

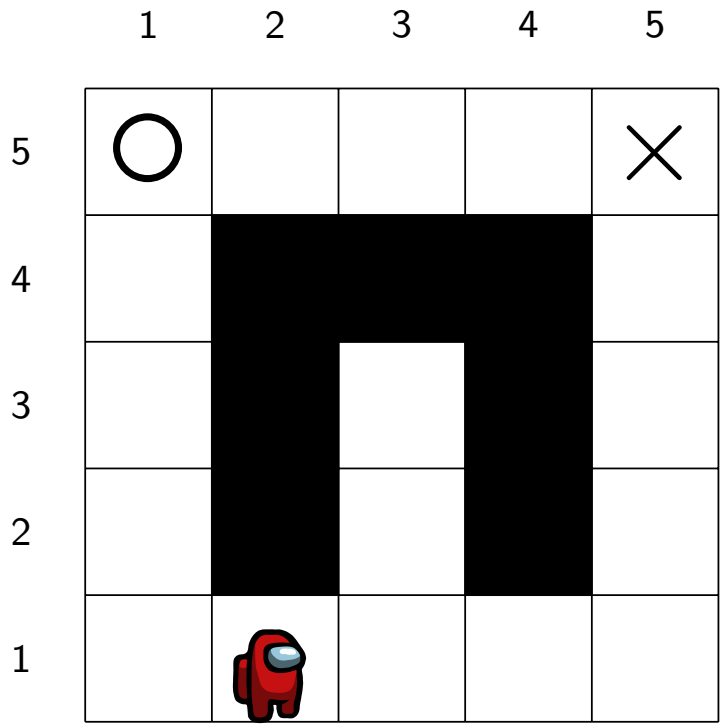
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

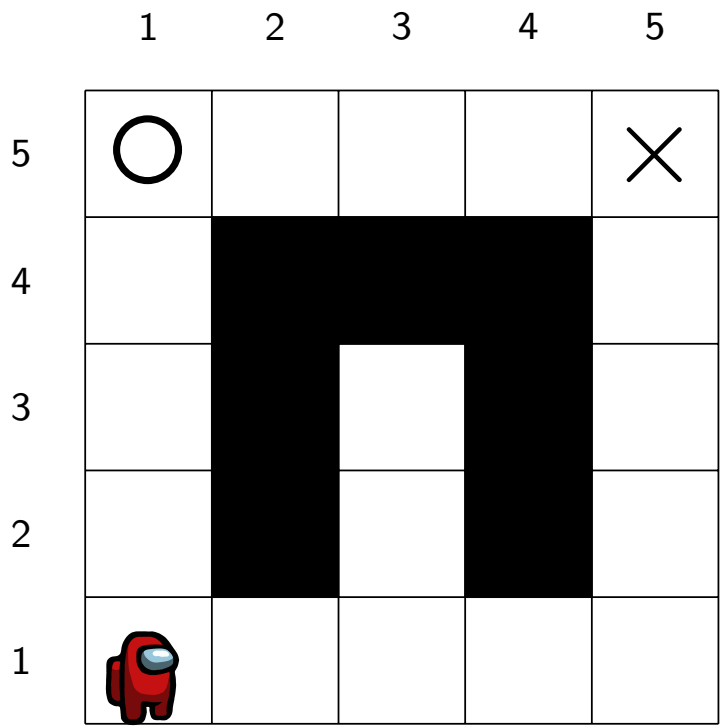
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

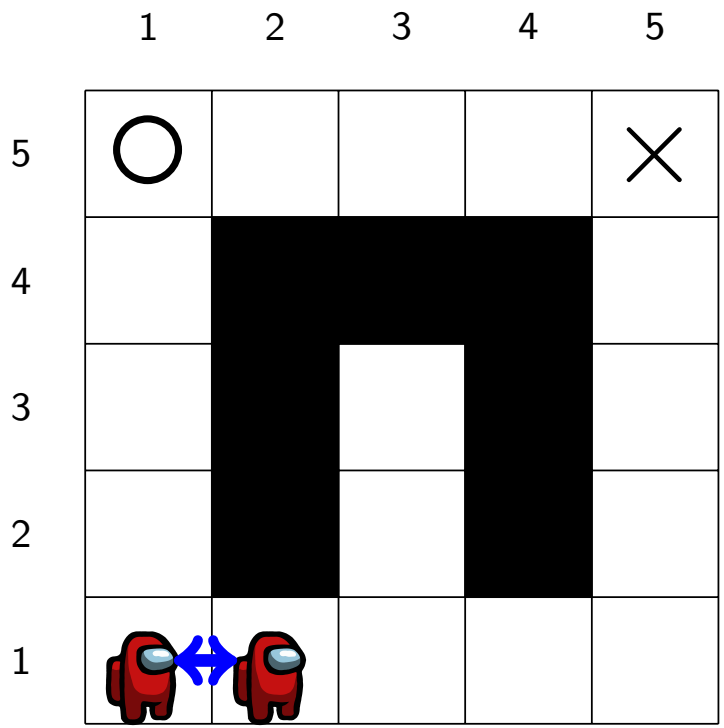
Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving



Shortest Path: ∞

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

Problem: Infinite Loop

Formalizing Maze Solving

Problem: Infinite Loop

Insight: A path with a loop in it can't be shorter than one without the loop.

Formalizing Maze Solving

Problem: Infinite Loop

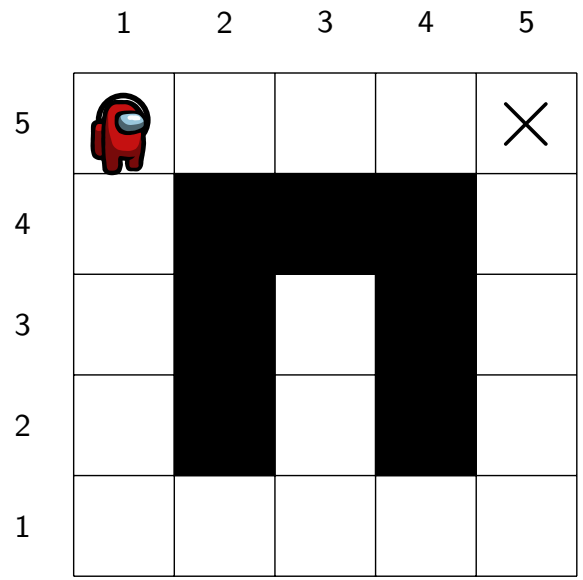
Insight: A path with a loop in it can't be shorter than one without the loop.

Mark nodes as visited so you don't visit them twice.

Formalizing Maze Solving

```
1   public int steps(Point pos, Point dest)
2   {
3     if(pos == dest){ return 0; }
4     else if(is_filled(pos){ return ∞; }
5     else if(is_visited(pos){ return ∞; }
6     else {
7       mark_visited(pos);
8       return 1 + Math.min(
9         steps(pos.moveDown, dest),
10        steps(pos.moveLeft, dest),
11        steps(pos.moveRight, dest),
12        steps(pos.moveUp, dest)
13      );
14    }
15  }
```

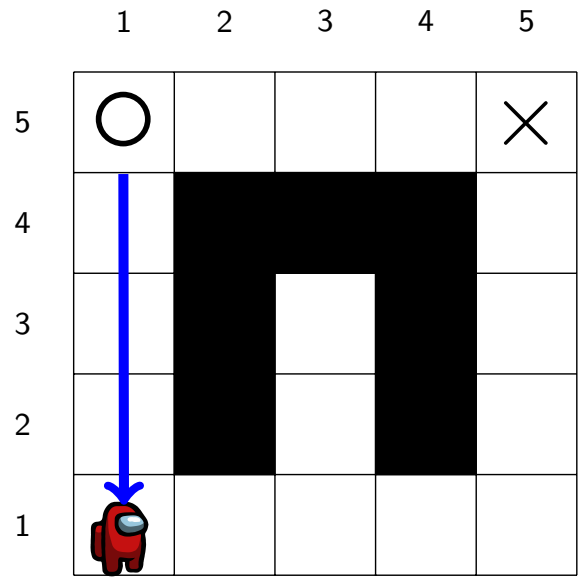
Formalizing Maze Solving



Shortest: $1 + \min(?, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

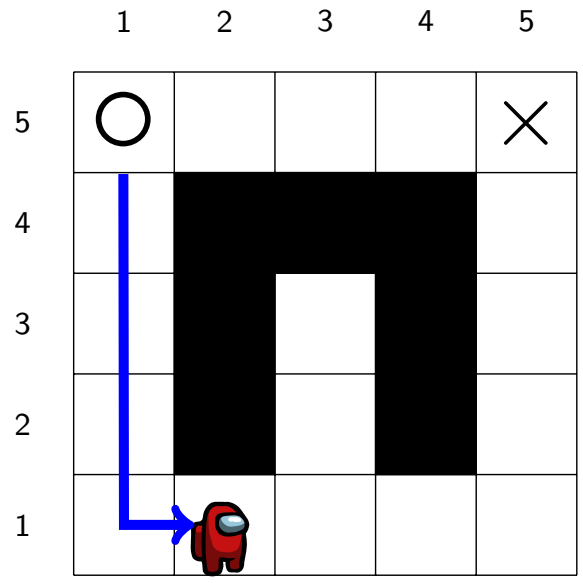
Formalizing Maze Solving



Shortest: $1 + \min(\infty, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

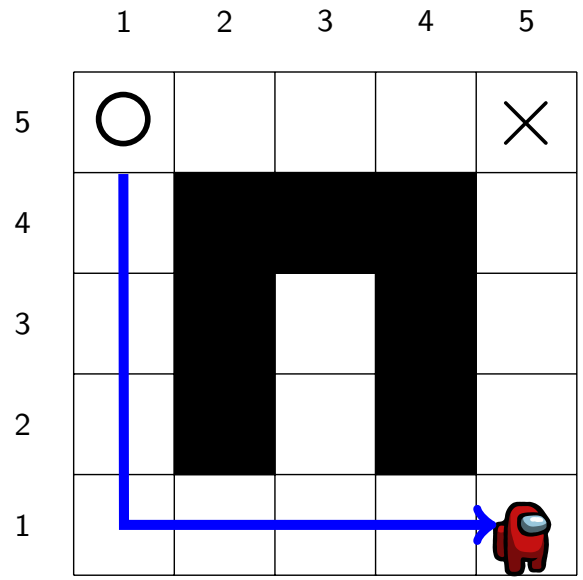
Formalizing Maze Solving



Shortest: $1 + \min(\infty, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

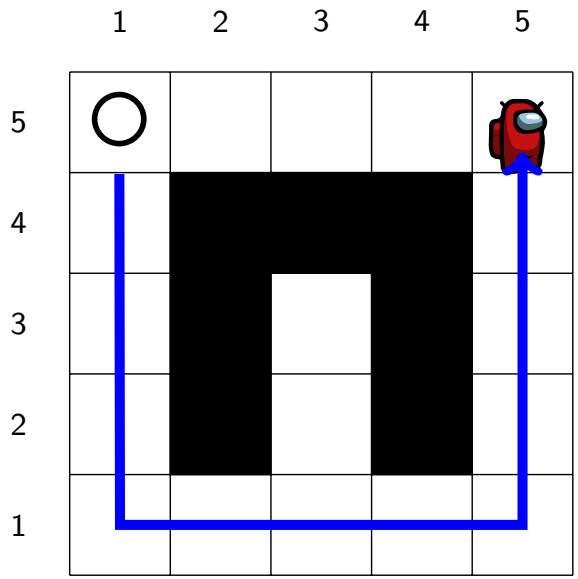
Formalizing Maze Solving



Shortest: $1 + \min(\infty, \infty, \infty, ?) = ?$

Among Us Beans ©Innersloth LLC

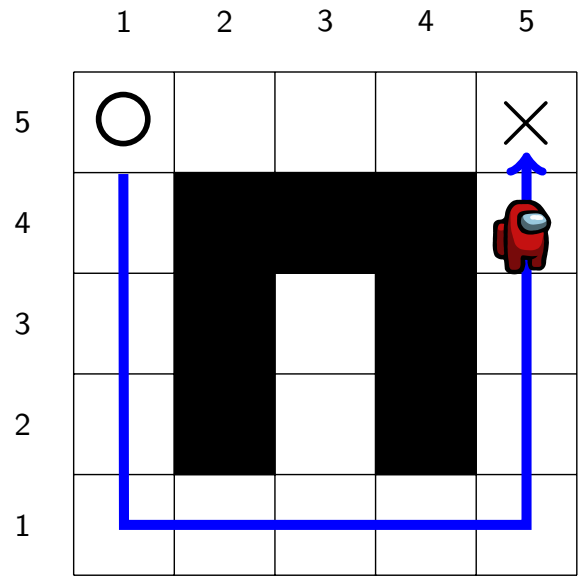
Formalizing Maze Solving



Shortest: 0

Among Us Beans ©Innersloth LLC

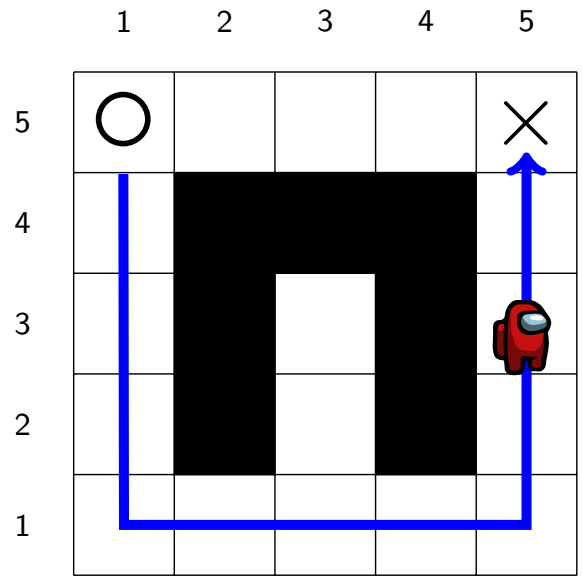
Formalizing Maze Solving



Shortest: $1 + \min(\infty, \infty, \infty, 0) = 1$

Among Us Beans ©Innersloth LLC

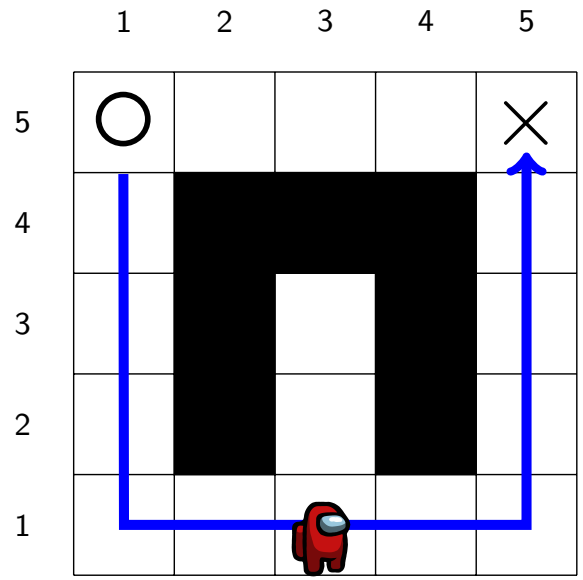
Formalizing Maze Solving



Shortest: $1 + \min(\infty, \infty, \infty, 1) = 2$

Among Us Beans ©Innersloth LLC

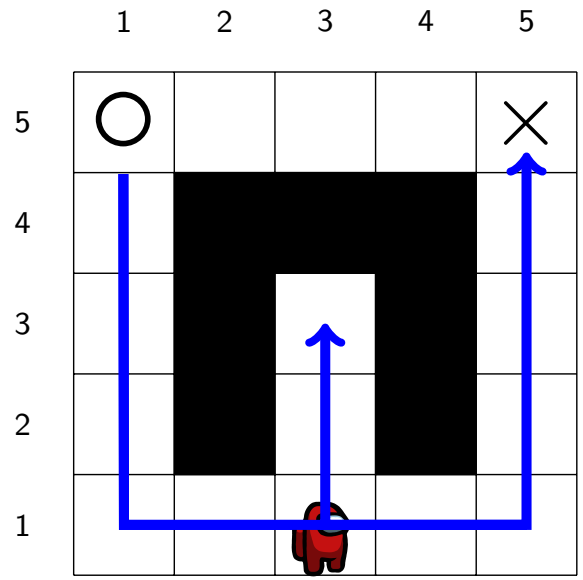
Formalizing Maze Solving



Shortest: $1 + \min(\infty, 5, \infty, ?) = ?$

Among Us Beans ©Innersloth LLC

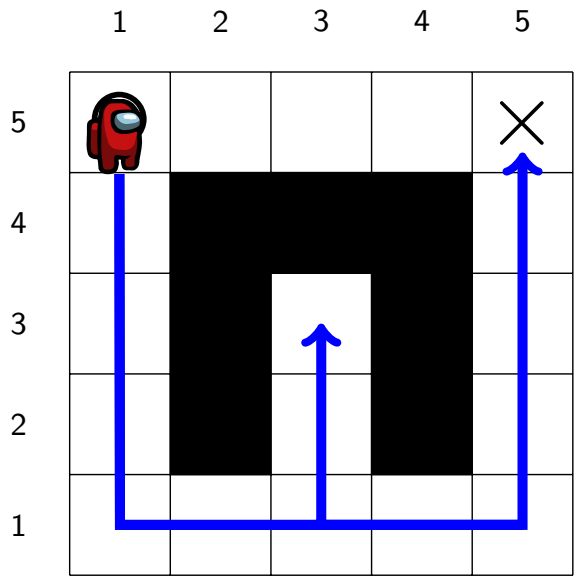
Formalizing Maze Solving



Shortest: $1 + \min(\infty, 5, \infty, \infty) = 6$

Among Us Beans ©Innersloth LLC

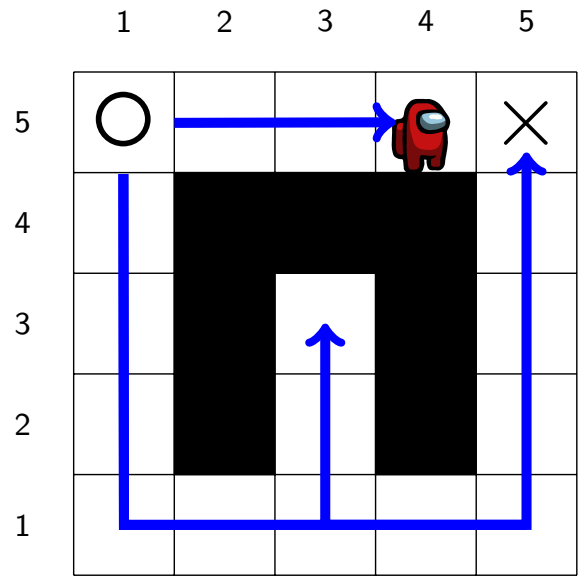
Formalizing Maze Solving



Shortest: $1 + \min(11, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

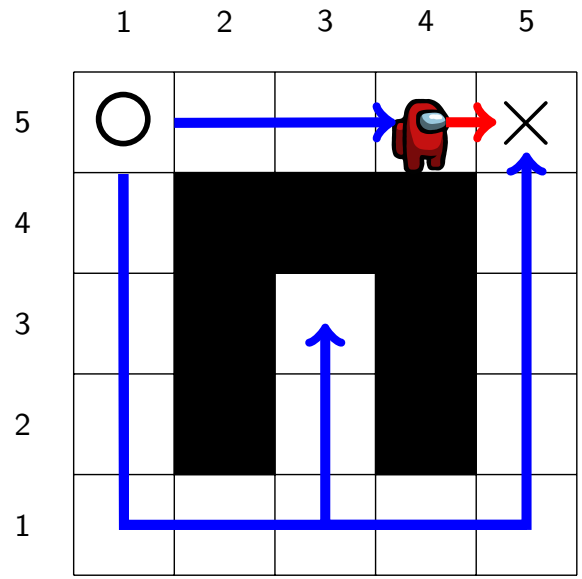
Formalizing Maze Solving



Shortest: $1 + \min(\infty, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving



Shortest: $1 + \min(\infty, \infty, \infty, \infty) = \infty$

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

Problem: The first time you visit a node it may be via a longer path!

Formalizing Maze Solving

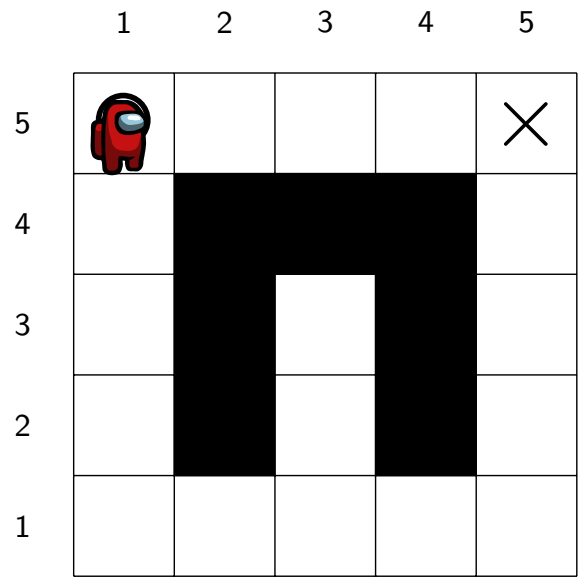
Problem: The first time you visit a node it may be via a longer path!

Unmark nodes as you leave them.

Formalizing Maze Solving

```
1   public int steps(Point pos, Point dest)
2   {
3     if(pos == dest){ return 0; }
4     else if(is_filled(pos){ return ∞; }
5     else if(is_visited(pos){ return ∞; }
6     else {
7       mark_visited(pos);
8       int stepCount = 1 + Math.min(
9         steps(pos.moveRight, dest),
10        /* ... */
11      );
12      unmark_visited(pos);
13      return stepCount;
14    }
15  }
```

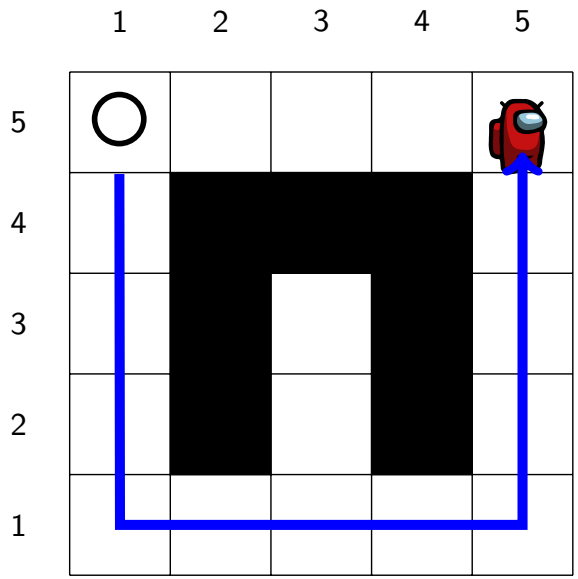
Formalizing Maze Solving



Shortest: $1 + \min(?, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

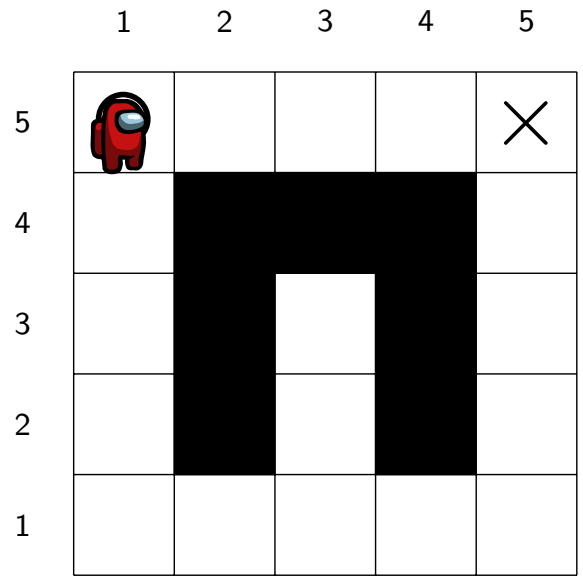
Formalizing Maze Solving



Shortest: 0

Among Us Beans ©Innersloth LLC

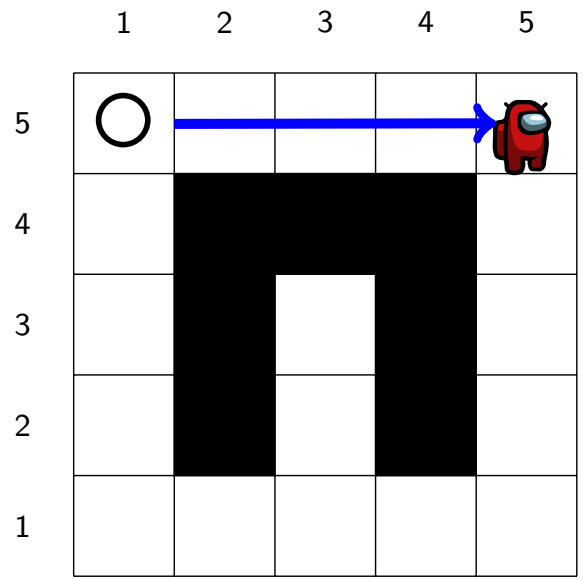
Formalizing Maze Solving



Shortest: $1 + \min(11, ?, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

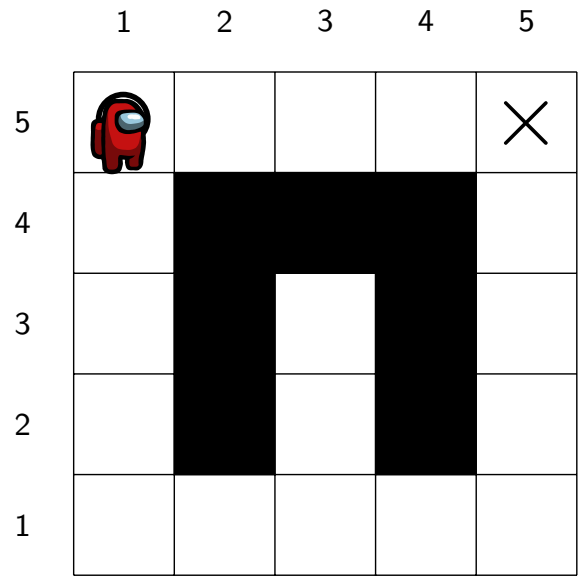
Formalizing Maze Solving



Shortest: 0

Among Us Beans ©Innersloth LLC

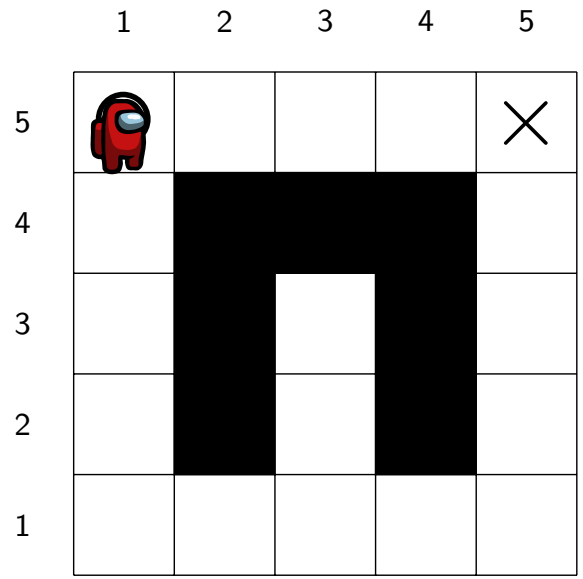
Formalizing Maze Solving



Shortest: $1 + \min(11, 3, \infty, \infty) = ?$

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving



Shortest: $1 + \min(11, 3, \infty, \infty) = 4$

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

Question: What path did we take?

Formalizing Maze Solving

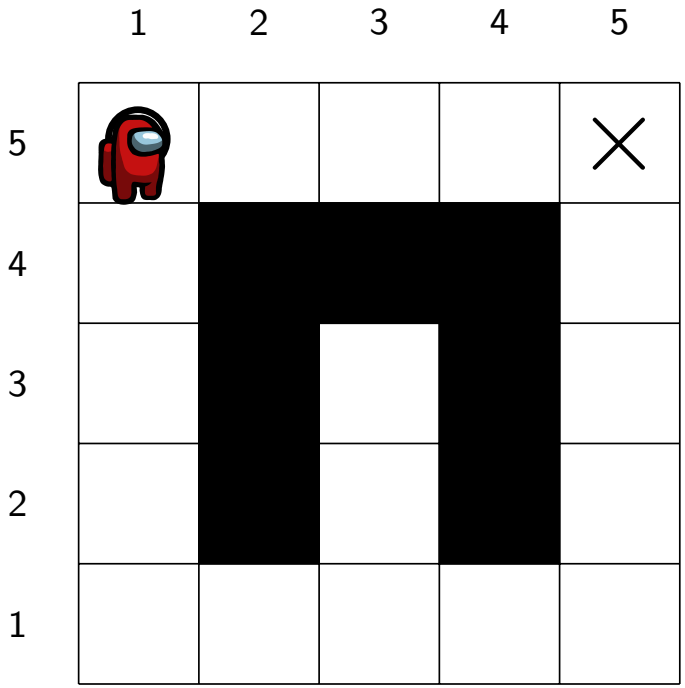
Question: What path did we take?

Track the current path in a `Stack`

Formalizing Maze Solving

```
1   public Array[Point] steps(Point pos, Point dest, Stack visited)
2   {
3     if(pos == dest){ return visited.toArray(); }          ←
4     else if(is_filled(pos){ return null; }
5     else if(visited.contains(pos){ return null; }         ←
6     else {
7       visited.push(pos);                                  ←
8       int steps = shortest_array(
9         steps(pos.moveRight, dest),
10        /* ... */
11      );
12      visited.pop(pos);                                   ←
13    }
14  }
```

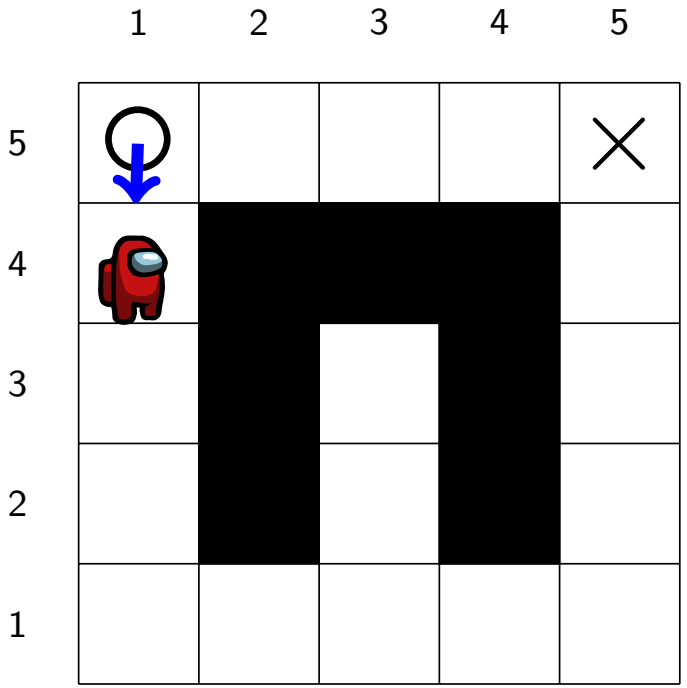
Formalizing Maze Solving



(1,5)

Among Us Beans ©Innersloth LLC

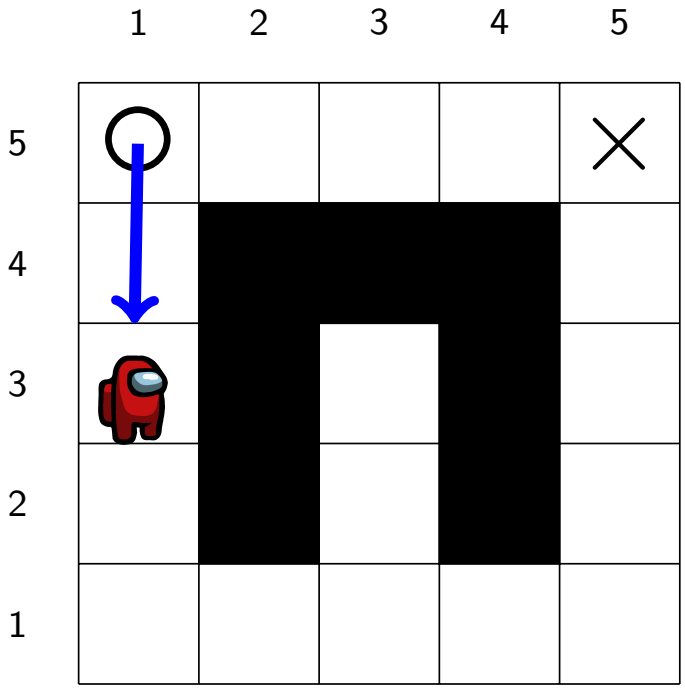
Formalizing Maze Solving



(1,5)
(1,4)

Among Us Beans ©Innersloth LLC

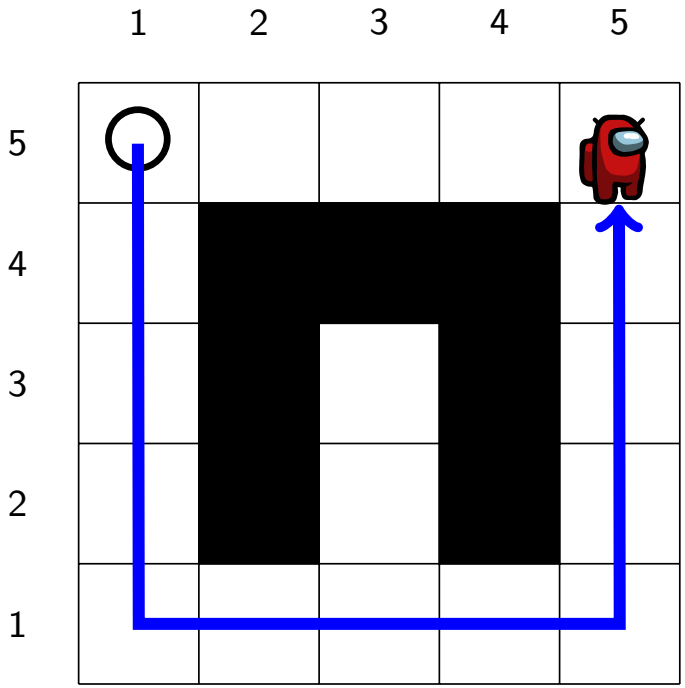
Formalizing Maze Solving



(1,5)
(1,4)
(1,3)

Among Us Beans ©Innersloth LLC

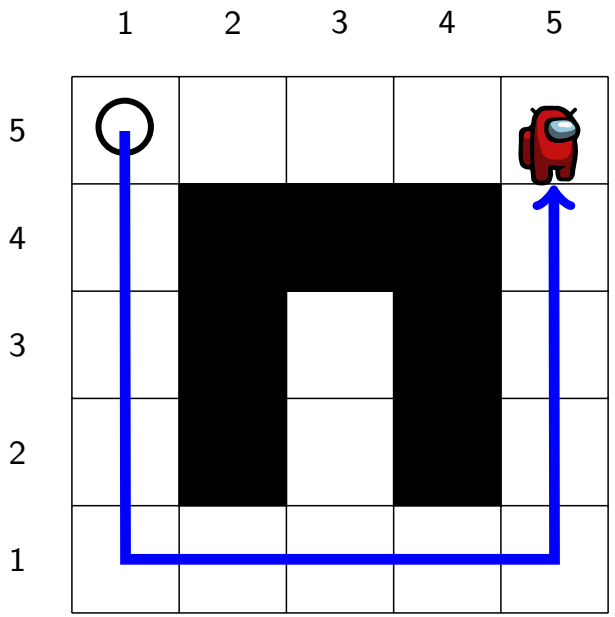
Formalizing Maze Solving



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

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

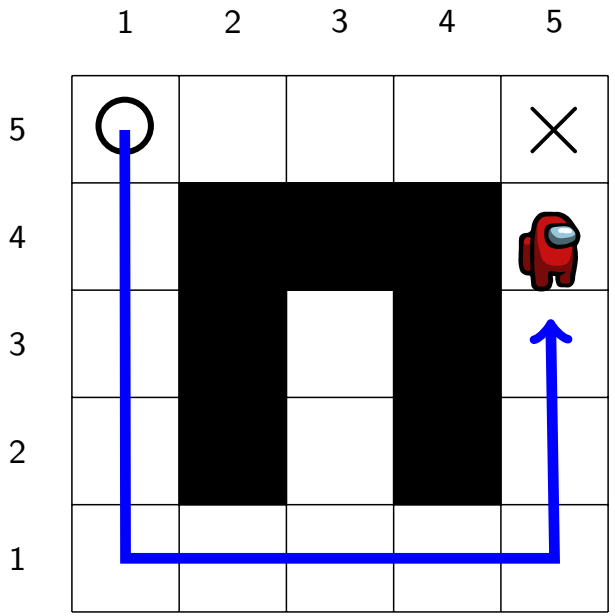


(1,5) (1,5)
(1,4) (1,4)
(1,3) (1,3)
(1,2) (1,2)
(1,1) (1,1)
(2,1) (2,1)
(3,1) (3,1)
(4,1) (4,1)
(5,1) (5,1)
(5,2) (5,2)
(5,3) (5,3)
(5,4) (5,4)
(5,5) (5,5)

✓

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

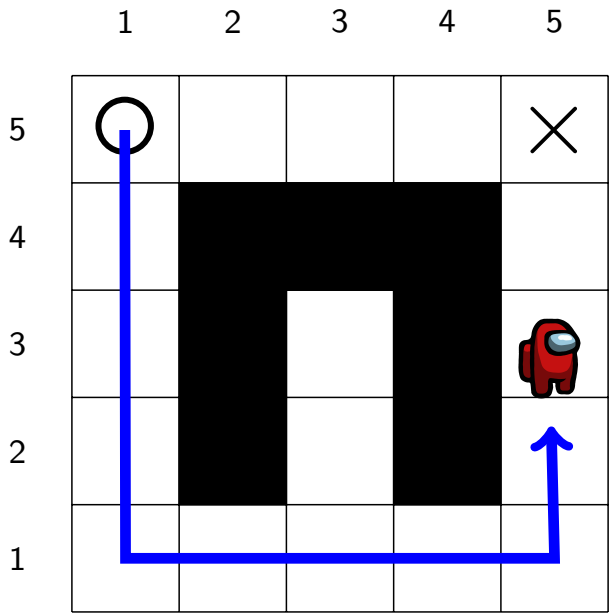


(1,5) (1,5)
(1,4) (1,4)
(1,3) (1,3)
(1,2) (1,2)
(1,1) (1,1)
(2,1) (2,1)
(3,1) (3,1)
(4,1) (4,1)
(5,1) (5,1)
(5,2) (5,2)
(5,3) (5,3)
(5,4) (5,4)
(5,5)

✓

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

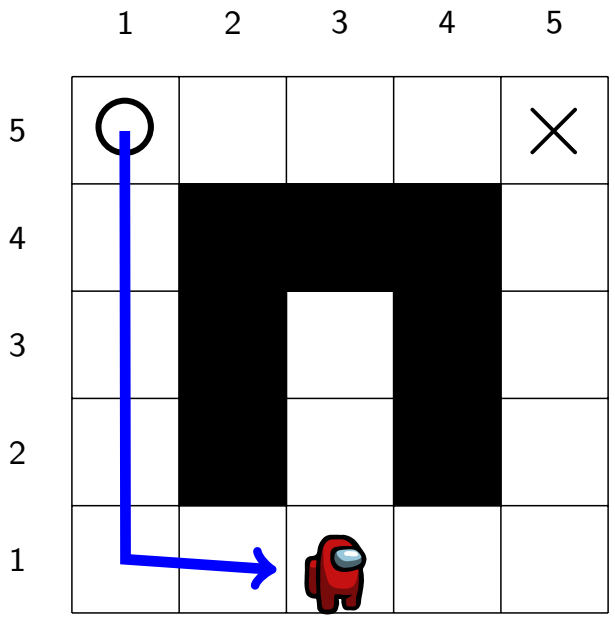


(1,5) (1,5)
(1,4) (1,4)
(1,3) (1,3)
(1,2) (1,2)
(1,1) (1,1)
(2,1) (2,1)
(3,1) (3,1)
(4,1) (4,1)
(5,1) (5,1)
(5,2) (5,2)
(5,3) (5,3)
(5,4)
(5,5)

✓

Among Us Beans ©Innersloth LLC

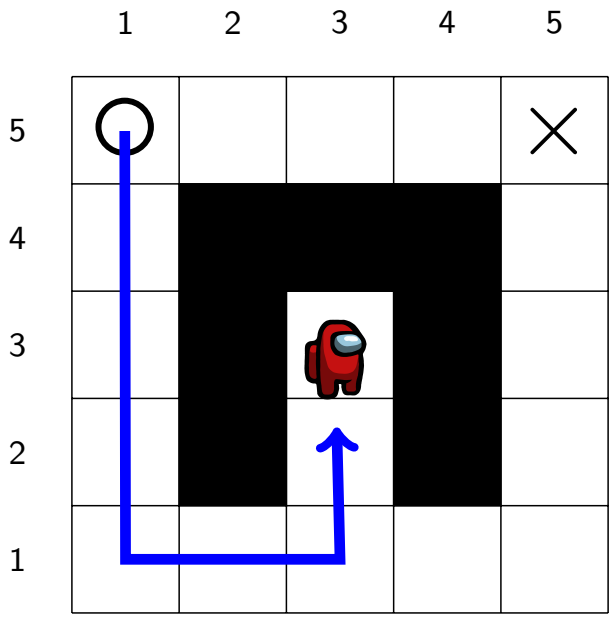
Formalizing Maze Solving



(1,5) (1,5)
(1,4) (1,4)
(1,3) (1,3)
(1,2) (1,2)
(1,1) (1,1)
(2,1) (2,1)
(3,1) (3,1)
(4,1)
(5,1)
(5,2)
(5,3)
(5,4)
(5,5)
✓

Among Us Beans ©Innersloth LLC

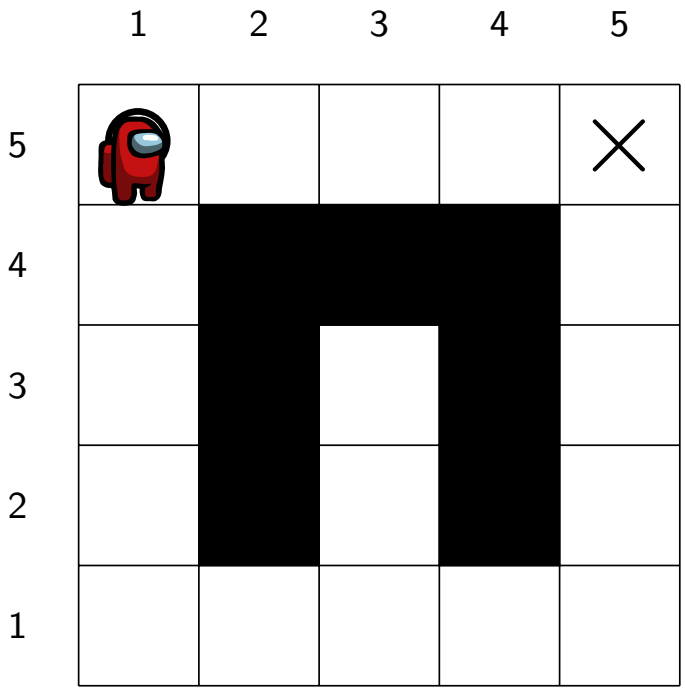
Formalizing Maze Solving



(1,5)	(1,5)
(1,4)	(1,4)
(1,3)	(1,3)
(1,2)	(1,2)
(1,1)	(1,1)
(2,1)	(2,1)
(3,1)	(3,1)
(4,1)	(3,2)
(5,1)	(3,3)
(5,2)	✗
(5,3)	
(5,4)	
(5,5)	
✓	

Among Us Beans ©Innersloth LLC

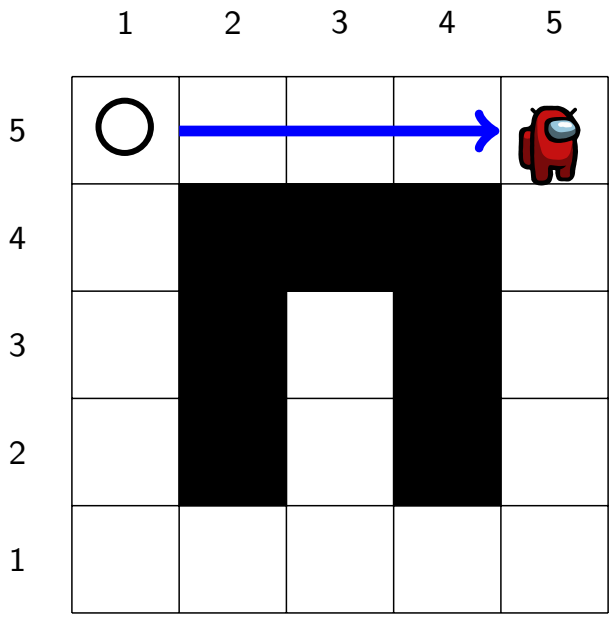
Formalizing Maze Solving



(1,5)
(1,4)
(1,3)
(1,2)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(5,2)
(5,3)
(5,4)
(5,5)
✓

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

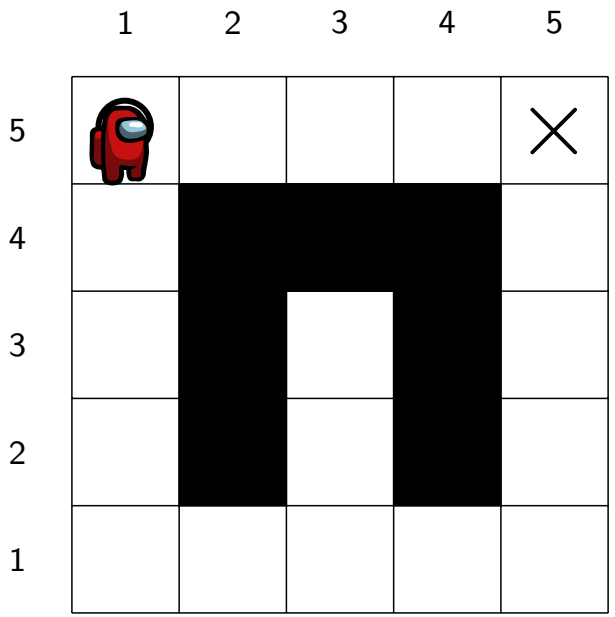


(1,5)
(1,4)
(1,3)
(1,2)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(5,2)
(5,3)
(5,4)
(5,5)
✓

(1,5)
(2,5)
(3,5)
(4,5)
(5,5)
✓

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving

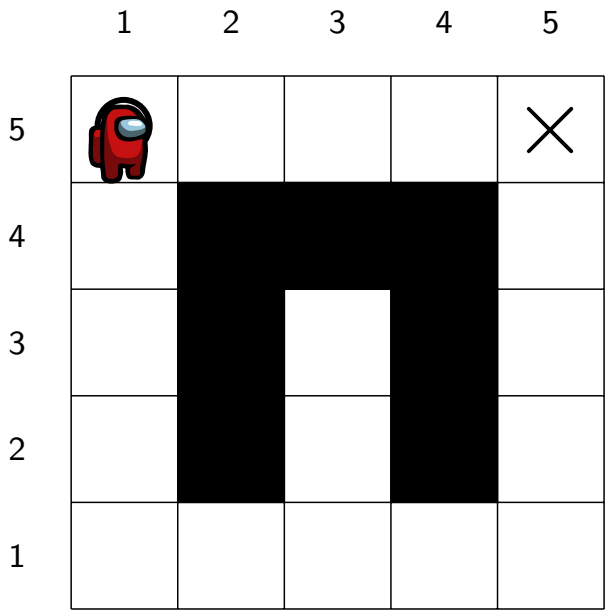


(1,5)
(1,4)
(1,3)
(1,2)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(5,2)
(5,3)
(5,4)
(5,5)
✓

(1,5)
(2,5)
(3,5)
(4,5)
(5,5)
✓

Among Us Beans ©Innersloth LLC

Formalizing Maze Solving



(1,5)
(1,4)
(1,3)
(1,2)
(1,1)
(2,1)
(3,1)
(4,1)
(5,1)
(5,2)
(5,3)
(5,4)
(5,5)
✓

(1,5)
(2,5)
(3,5)
(4,5)
(5,5)
✓

Among Us Beans ©Innersloth LLC

Queues

Thought Question: How would we need to change the algorithm to use a queue instead of a stack?