

IEEE CITY
ROBOTICS SOCIETY

PROGRAMMING 2: FLOODFILL & WALL DETECTION

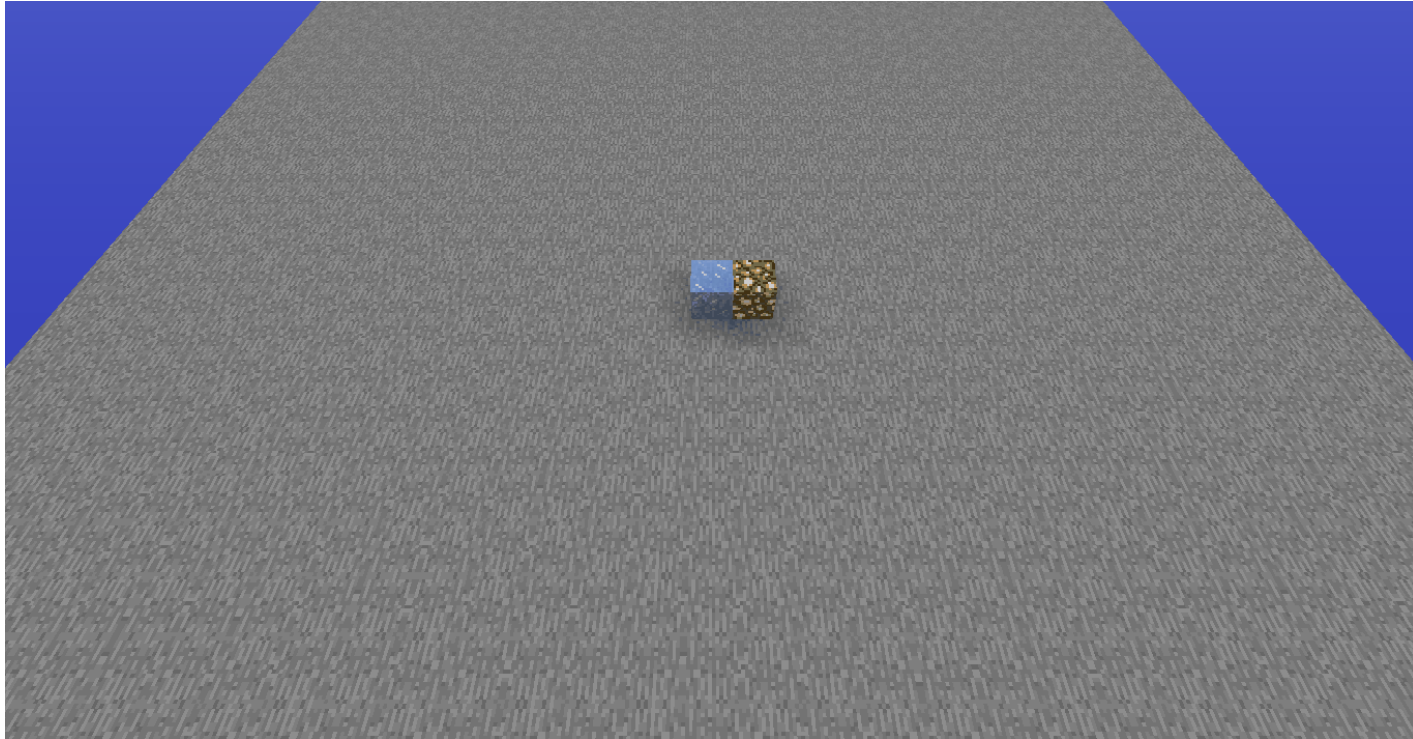


Floodfill: High Level Idea

Goal: Find the shortest path between start and end points

Intuitively, water flows in the shortest valid path from a start to end

Working principle: Water will flow around the wall and instead of through it

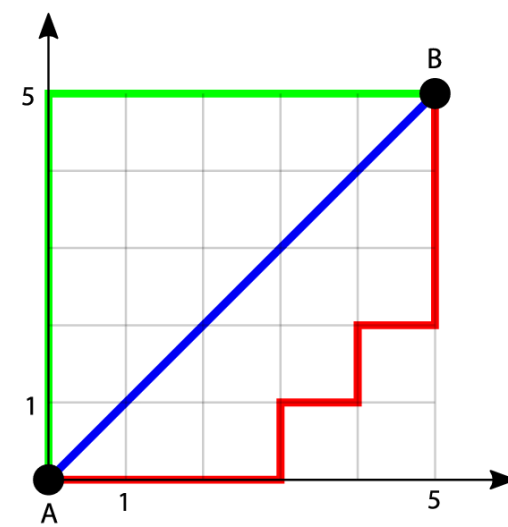


Floodfill: High Level Idea

Your mouse cannot travel through walls.

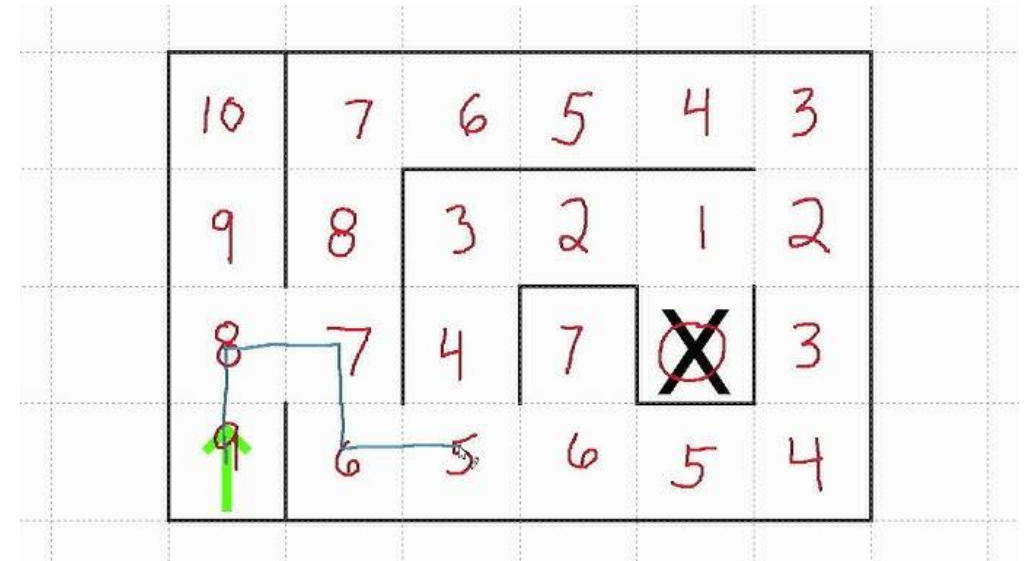
Split maze into cells, each cell has a Manhattan distance away from the goal

You want to move to the next smallest cell in the direction of the goal



— Euclidean distance

— Manhattan distance





Floodfill Initialisation

Assume the maze has no walls to start. Calculate Manhattan distance to the goal

Travel to a **smaller number** each move

If you get stuck (encounter a wall), recalculate the distance using floodfill

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

 Unknown Wall  Known Wall



Traversing the maze



 Unknown Wall  Known Wall



Traversing the maze



 Unknown Wall  Known Wall



Traversing the maze



 Unknown Wall  Known Wall

Traversing the maze




 Unknown Wall  Known Wall

Traversing the maze

Mouse is stuck



 Unknown Wall  Known Wall

Traversing the maze



Perform Floodfill →



Unknown Wall



Known Wall

We need to change the cell values to reflect the presence of the walls and how water would flow around it

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

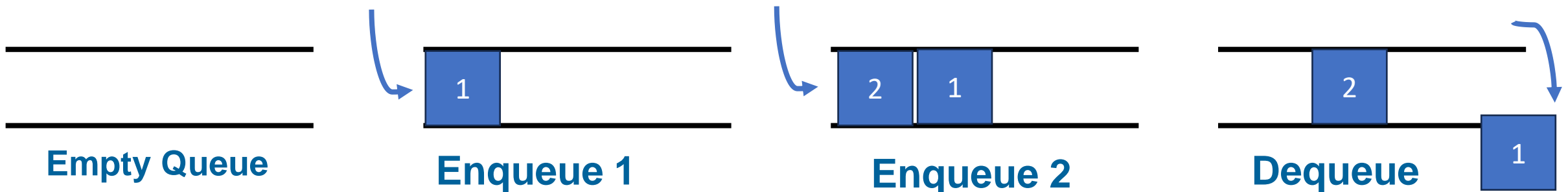
Perform Floodfill →

4	3	2	3	4
3	2	1	2	3
4	3	0	1	2
5	2	1	2	3
6	3	2	3	4

 Unknown Wall  Known Wall

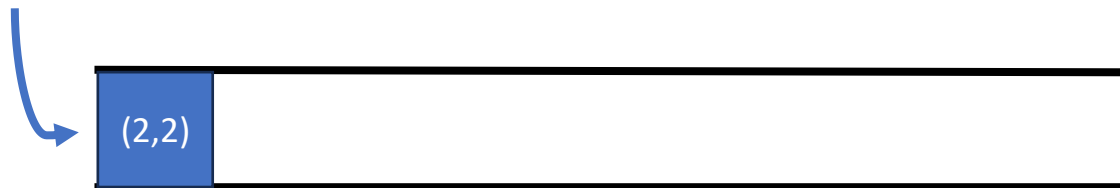
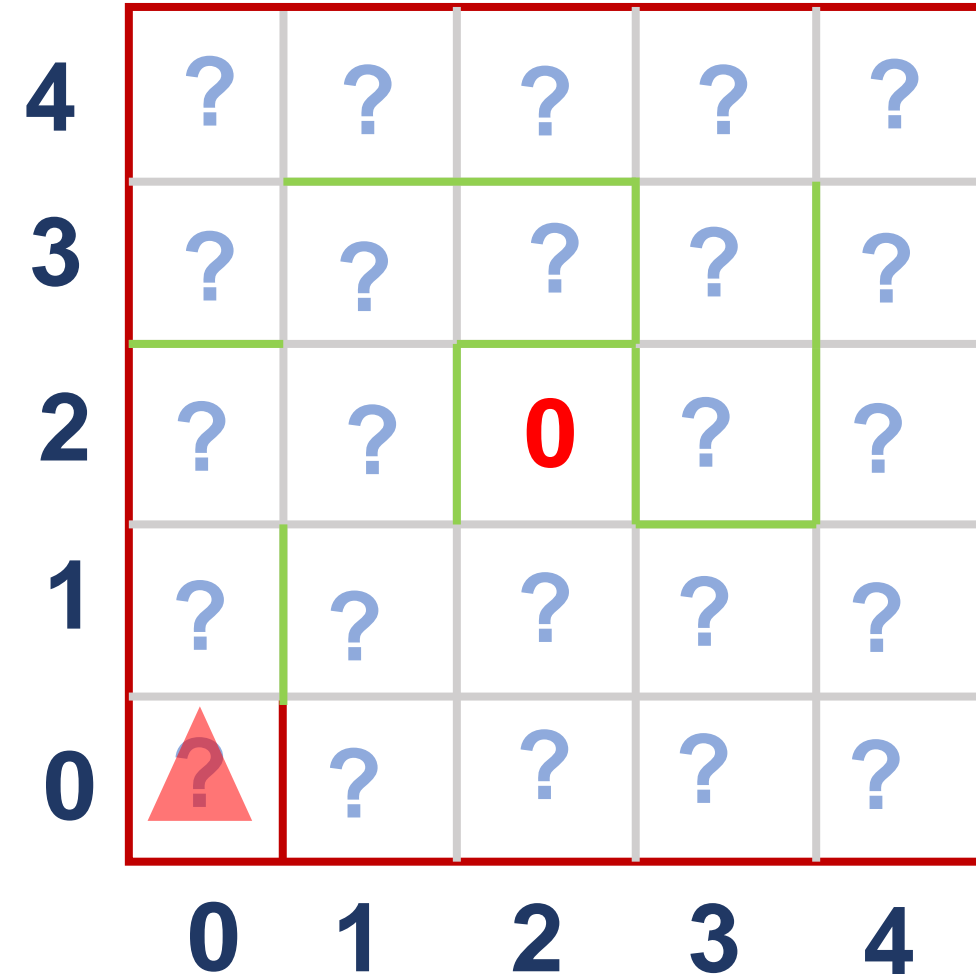
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



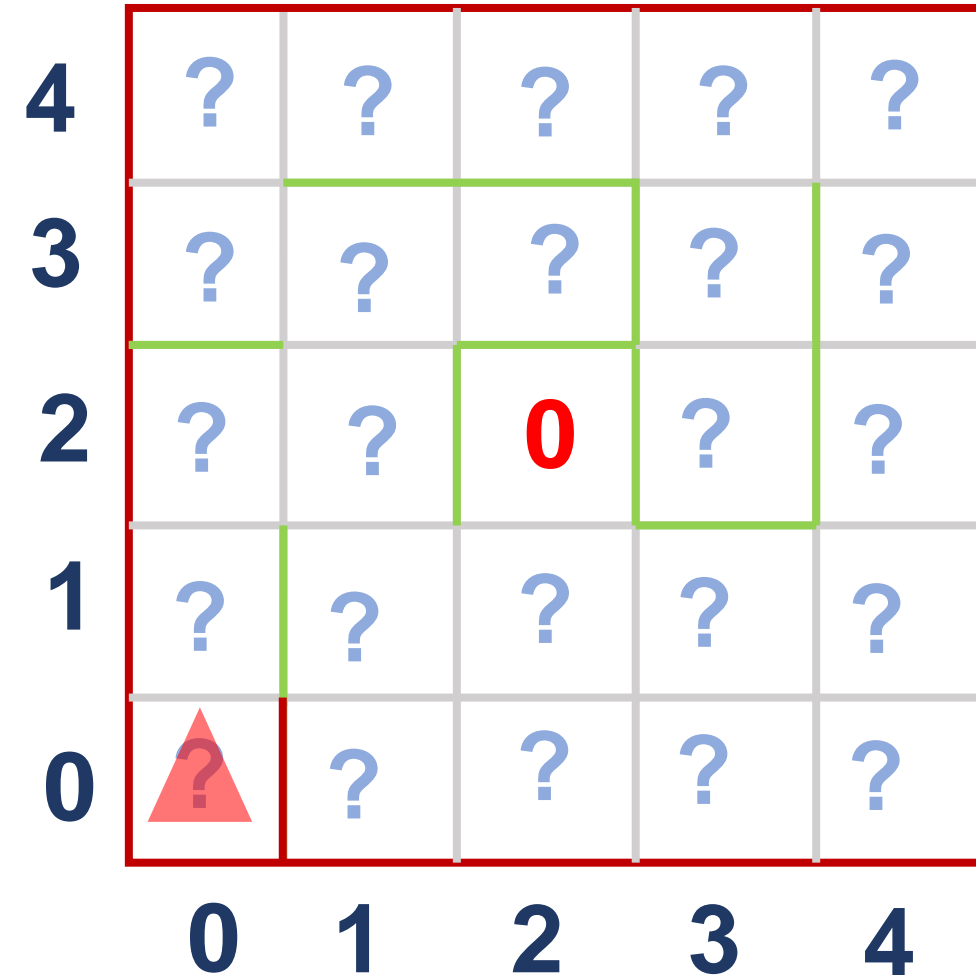
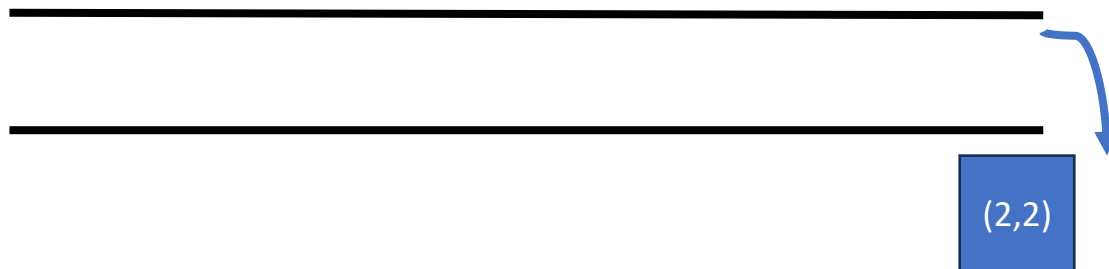
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



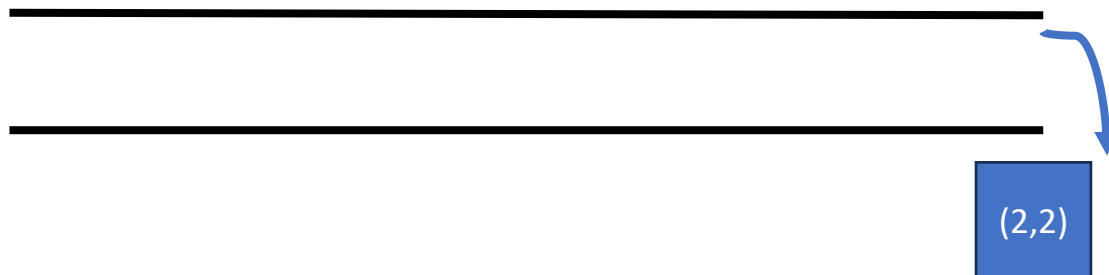
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Floodfill Pseudocode

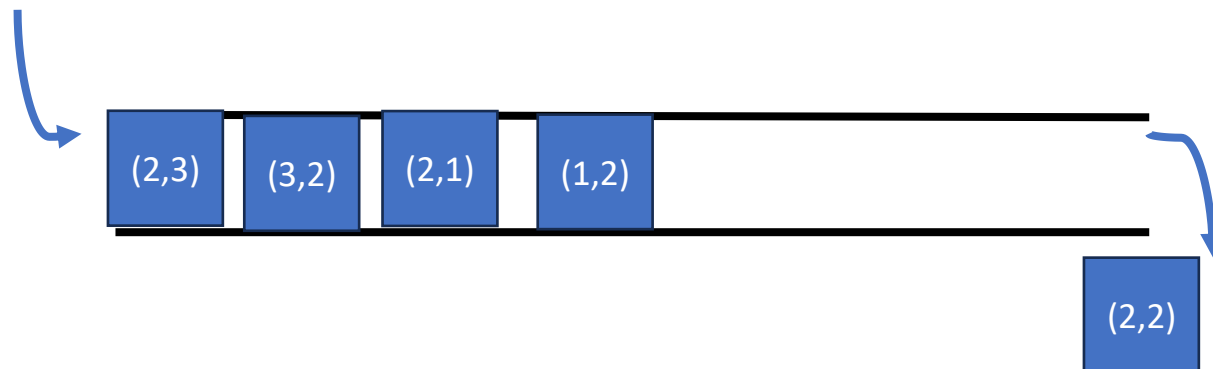
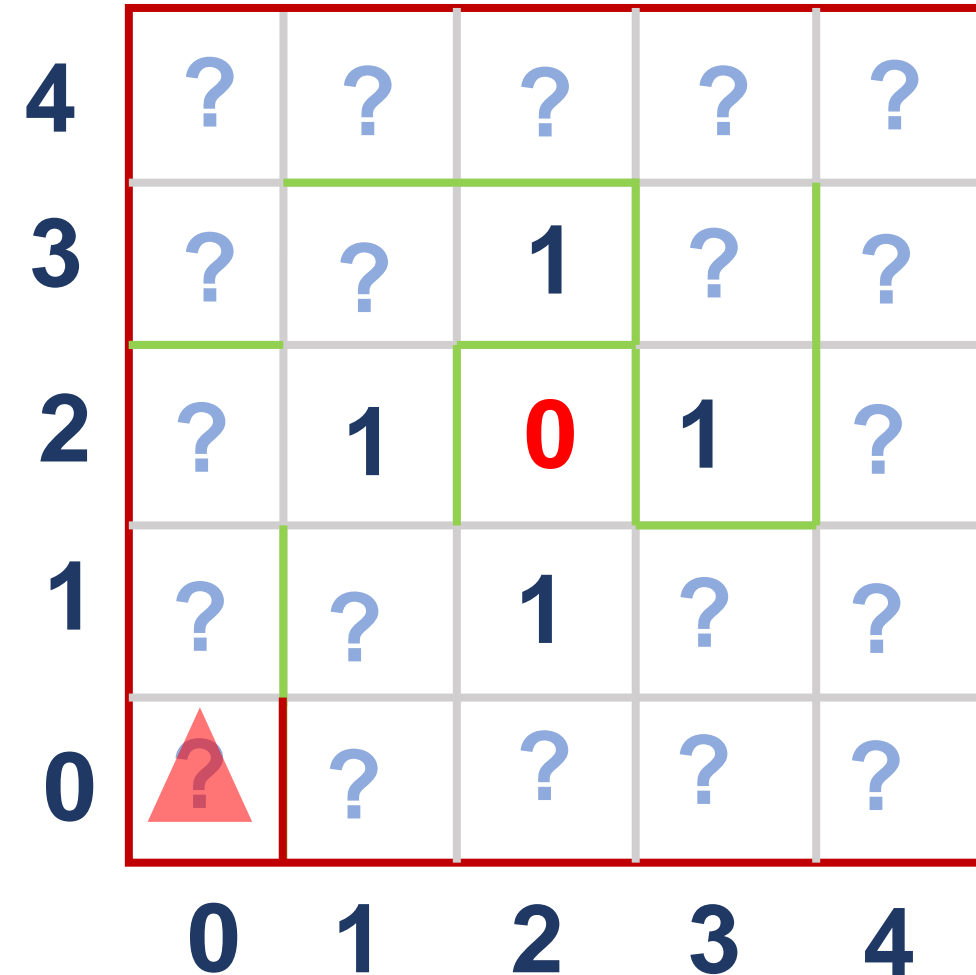
- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



4	?	?	?	?	?
3	?	?	1	?	?
2	?	1	0	1	?
1	?	?	1	?	?
0	?	?	?	?	?
	0	1	2	3	4

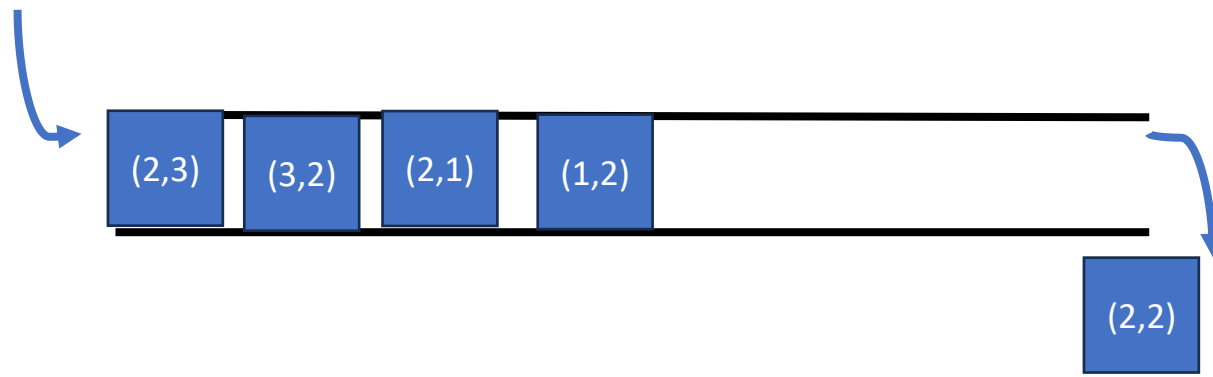
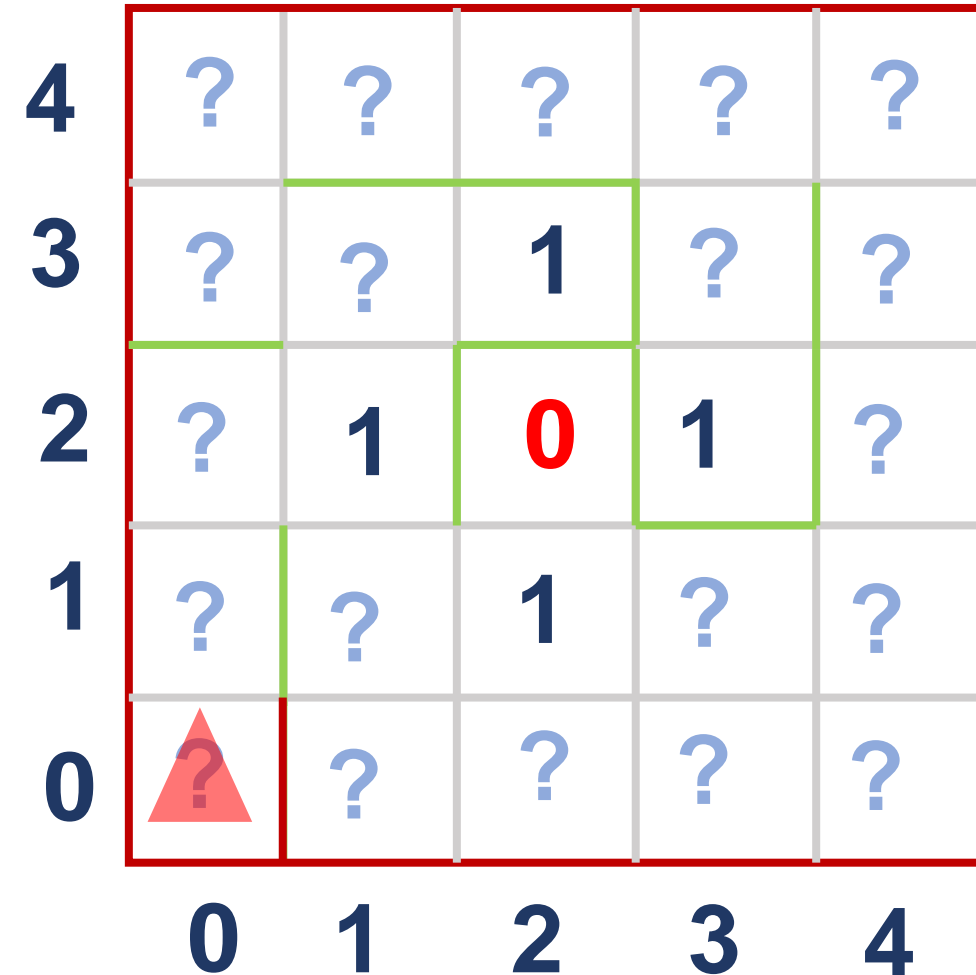
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



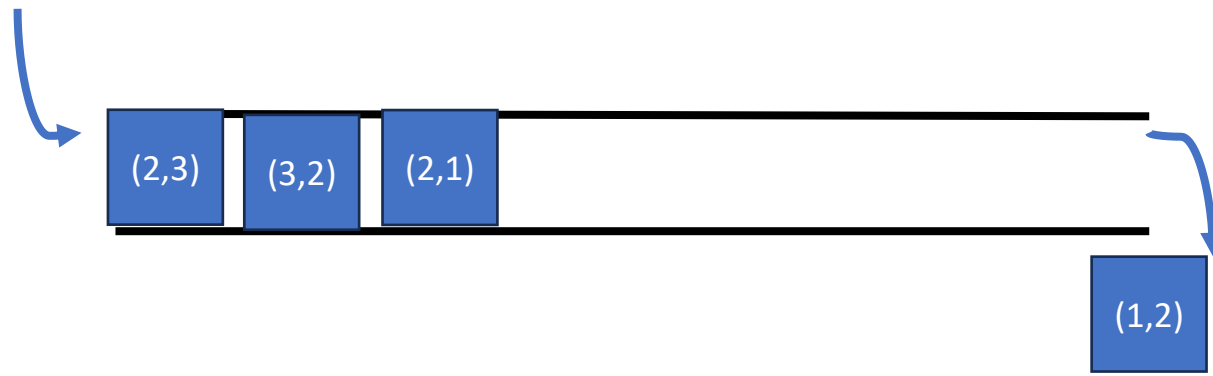
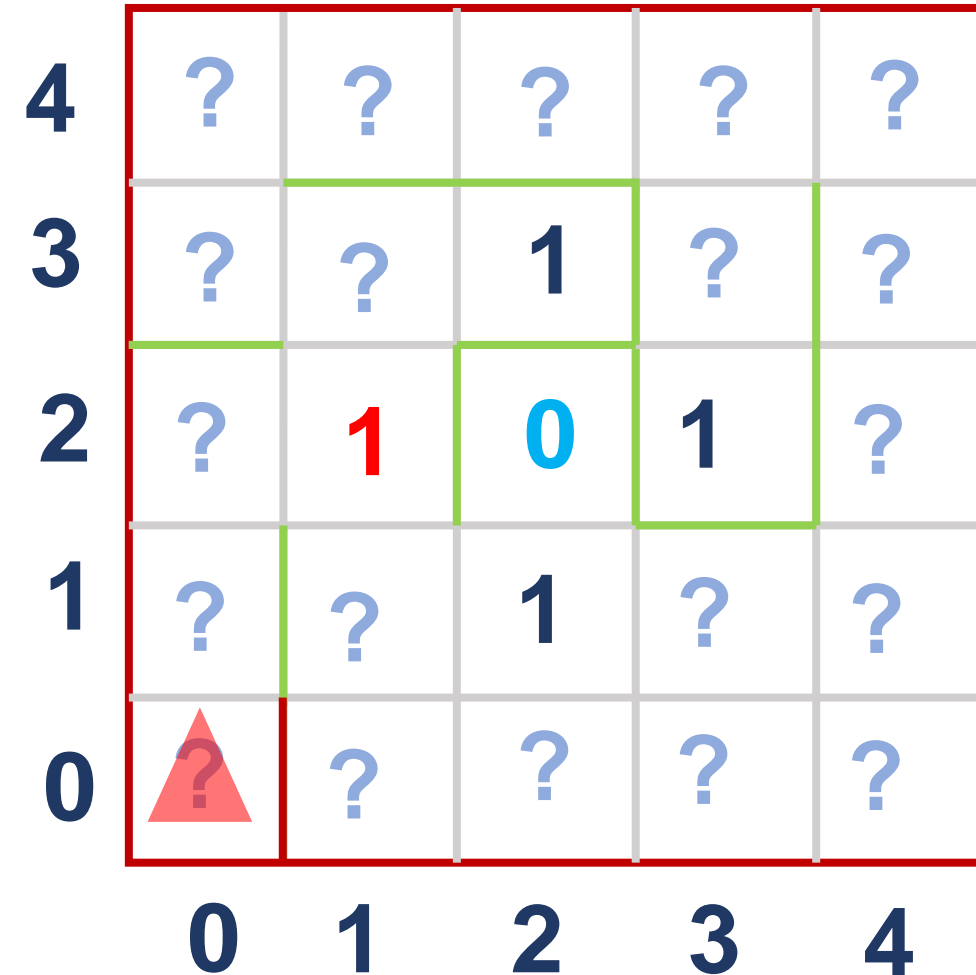
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) **While queue is not empty:**
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Floodfill Pseudocode

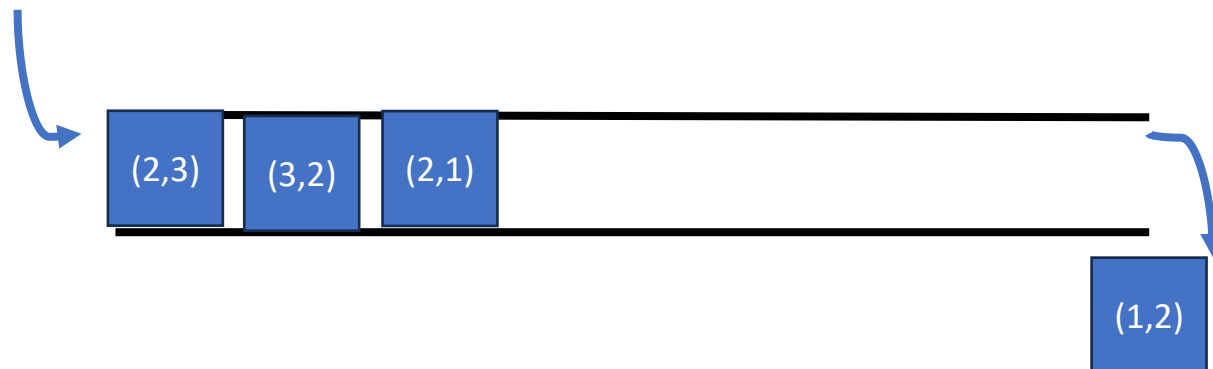
- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

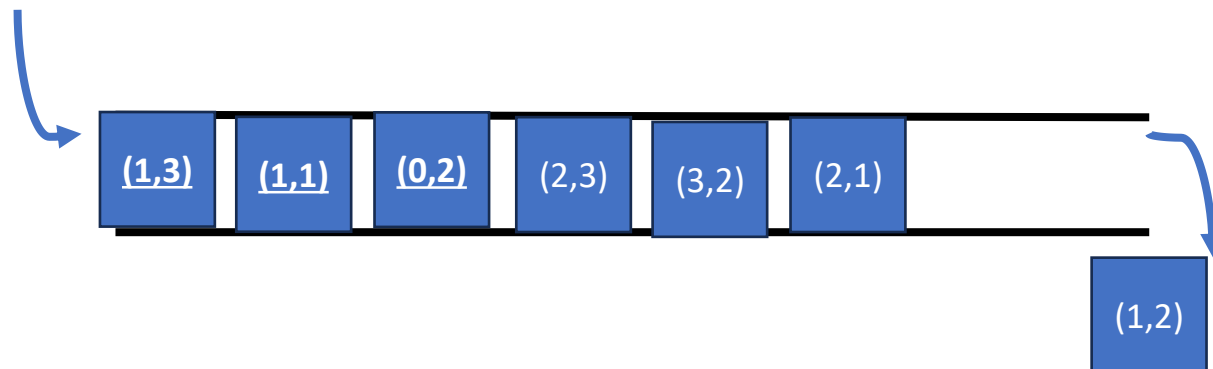
4	?	?	?	?	?
3	?	2	1	?	?
2	2	1	0	1	?
1	?	2	1	?	?
0	?	?	?	?	?
	0	1	2	3	4



Floodfill Pseudocode

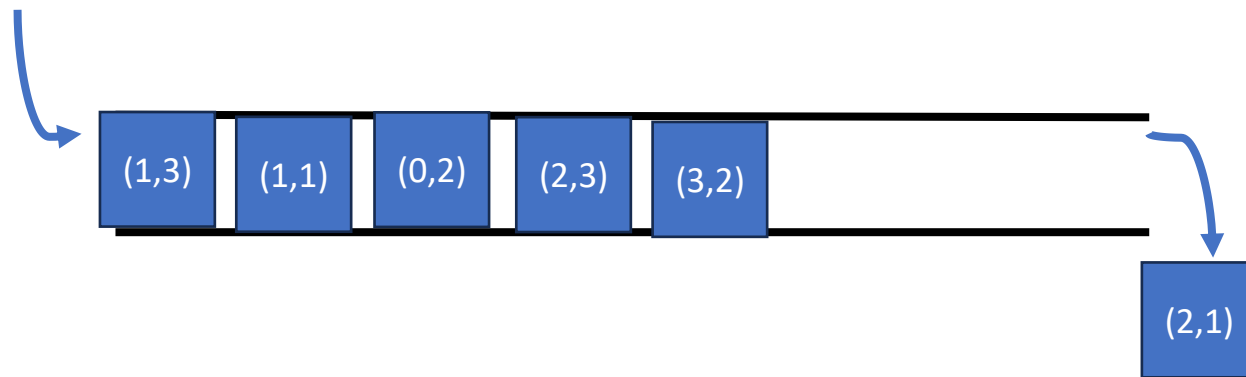
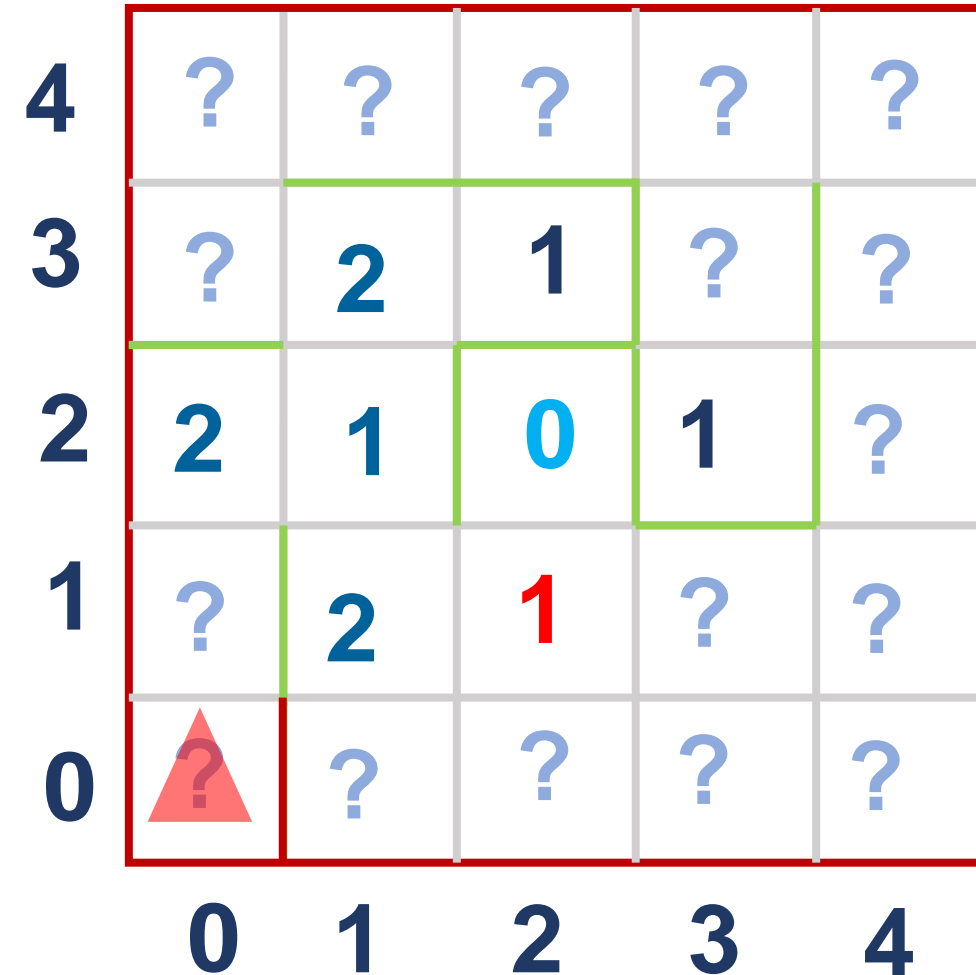
- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

4	?	?	?	?	?
3	?	2	1	?	?
2	2	1	0	1	?
1	?	2	1	?	?
0	?	?	?	?	?
	0	1	2	3	4



Floodfill Pseudocode

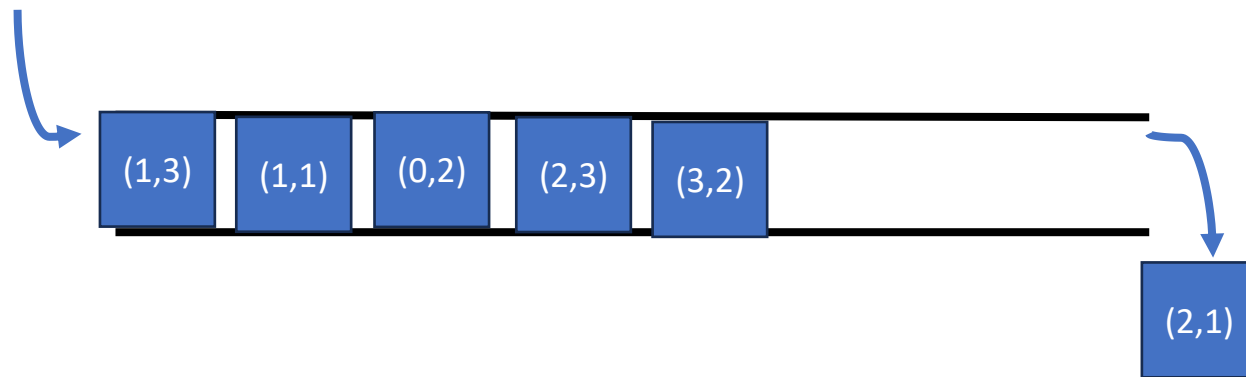
- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

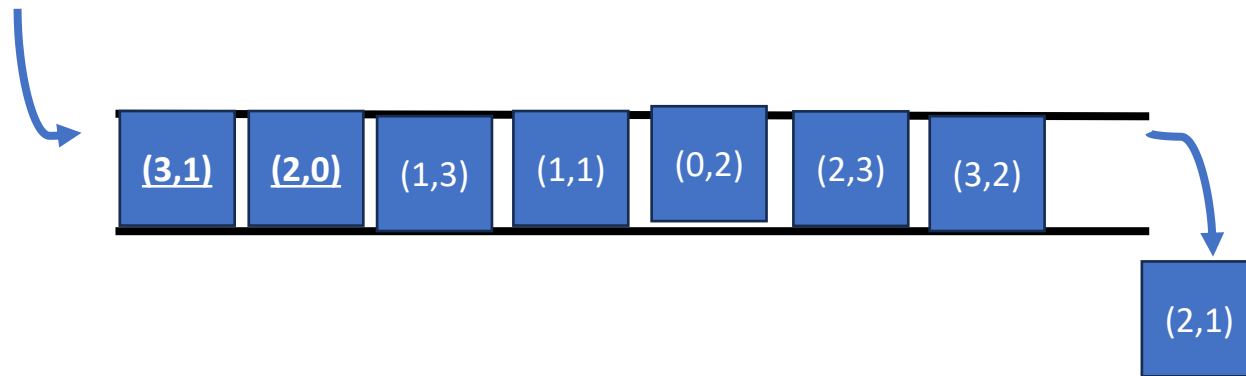
4	?	?	?	?	?
3	?	2	1	?	?
2	2	1	0	1	?
1	?	2	1	2	?
0	?	2	?	?	?
	0	1	2	3	4



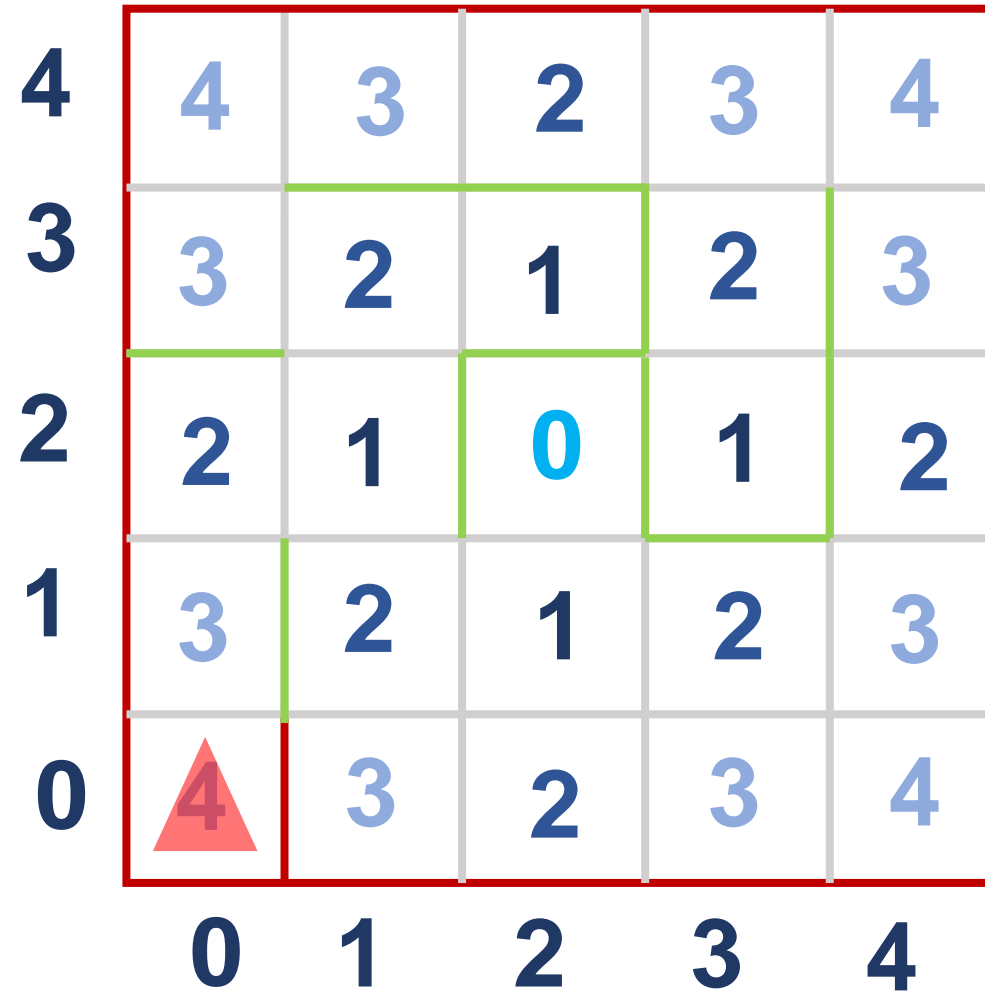
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

4	?	?	?	?	?
3	?	2	1	?	?
2	2	1	0	1	?
1	?	2	1	2	?
0	?	2	?	?	?
	0	1	2	3	4



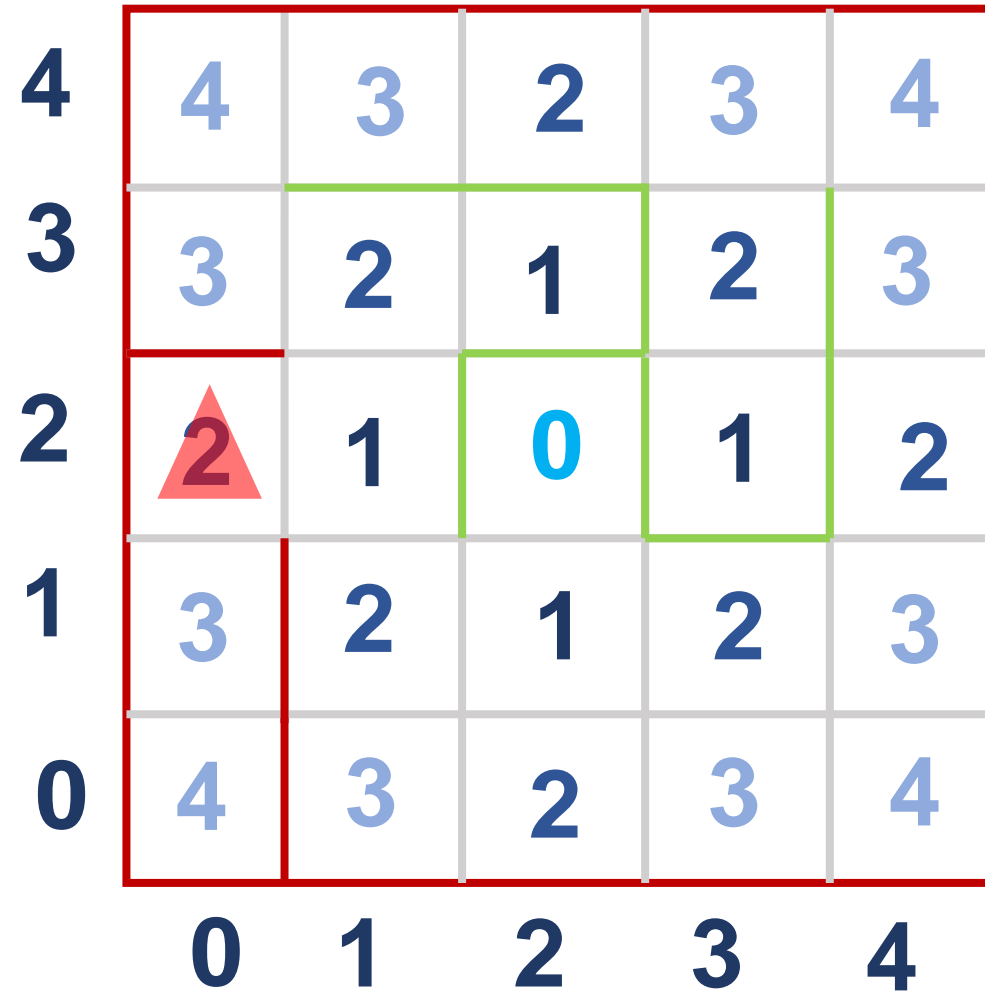
Floodfill Pseudocode



Floodfill Pseudocode

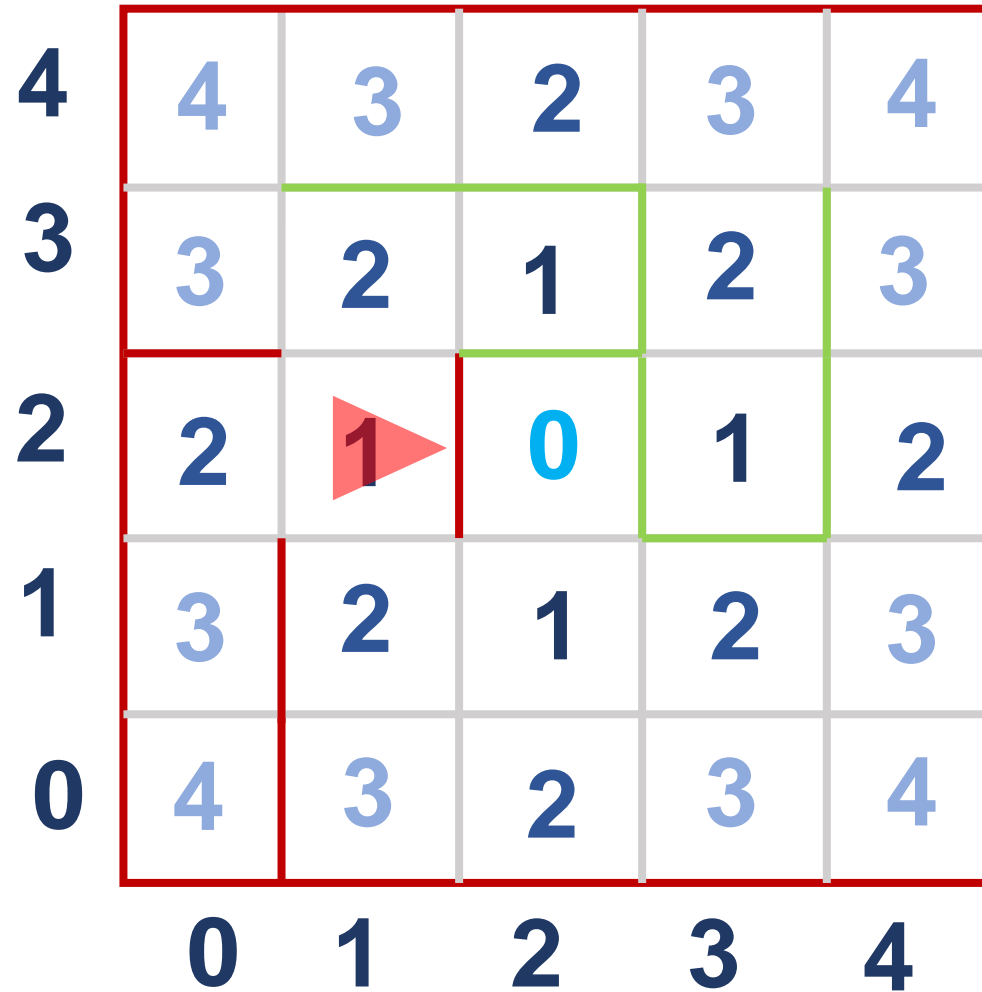


Floodfill Pseudocode



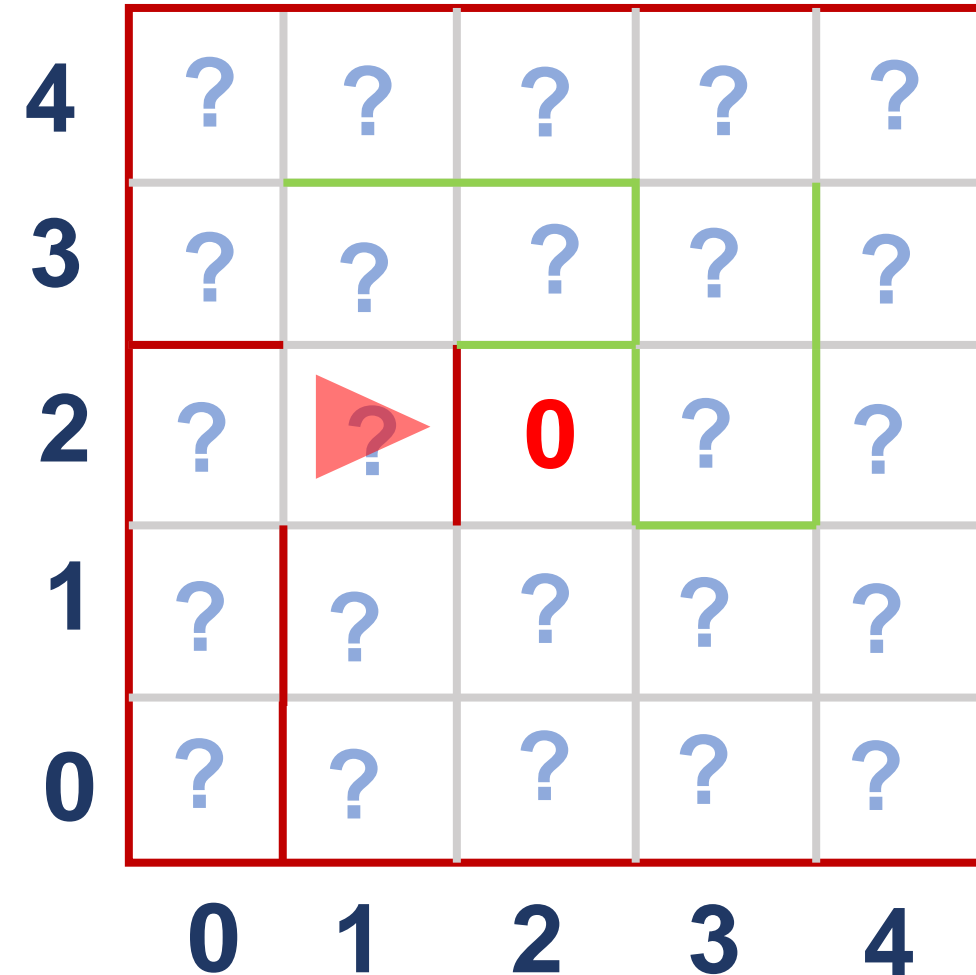
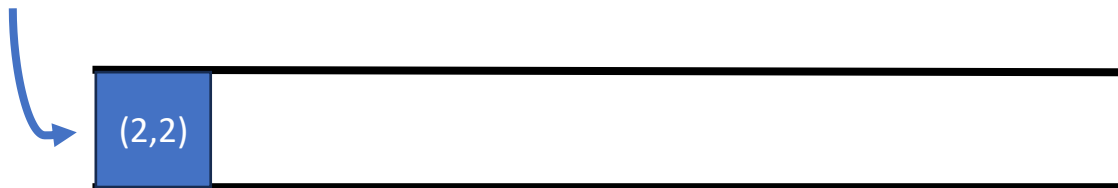
Floodfill Pseudocode

Mouse is stuck!



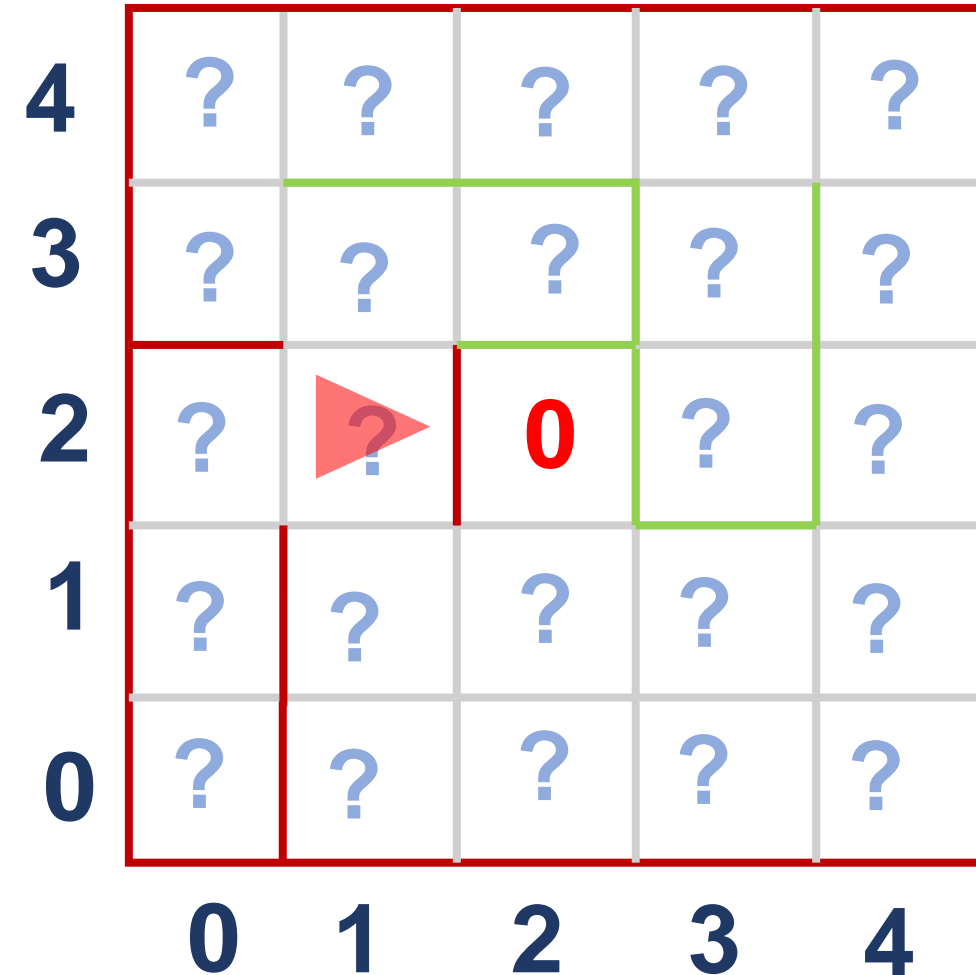
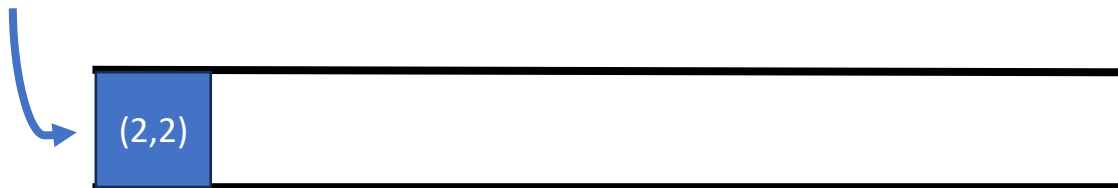
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



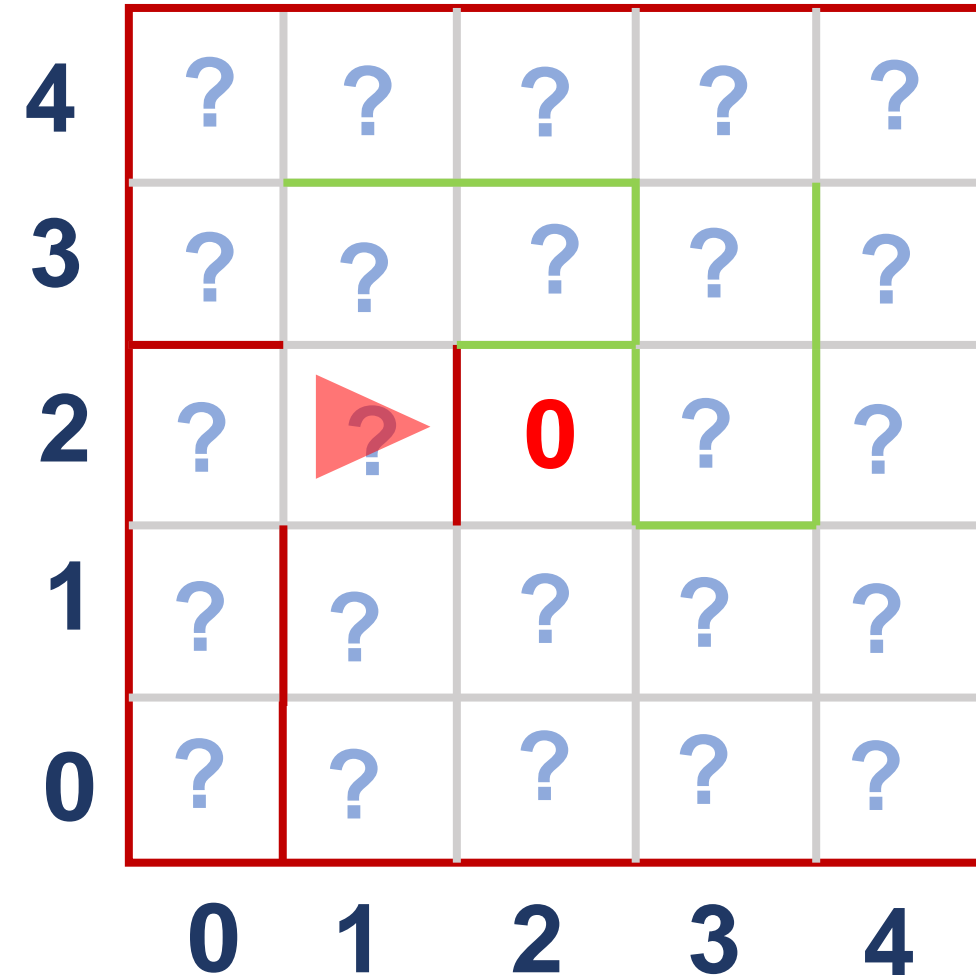
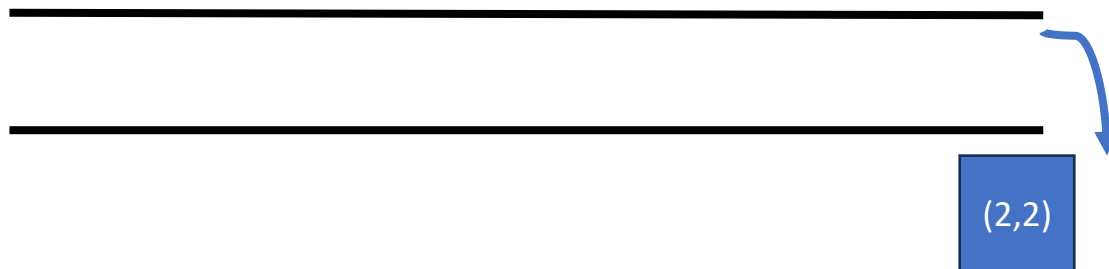
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



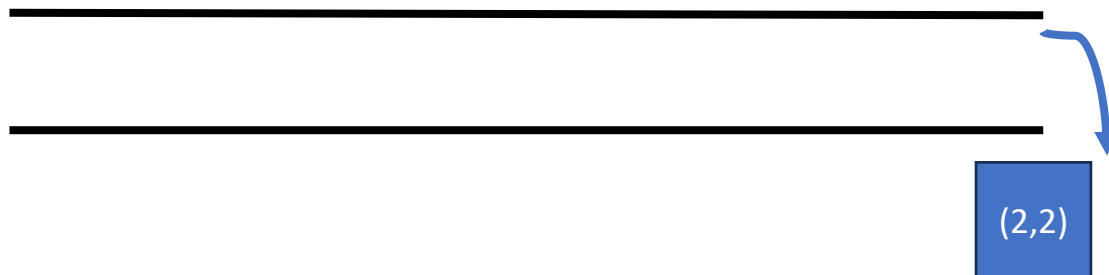
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Floodfill Pseudocode

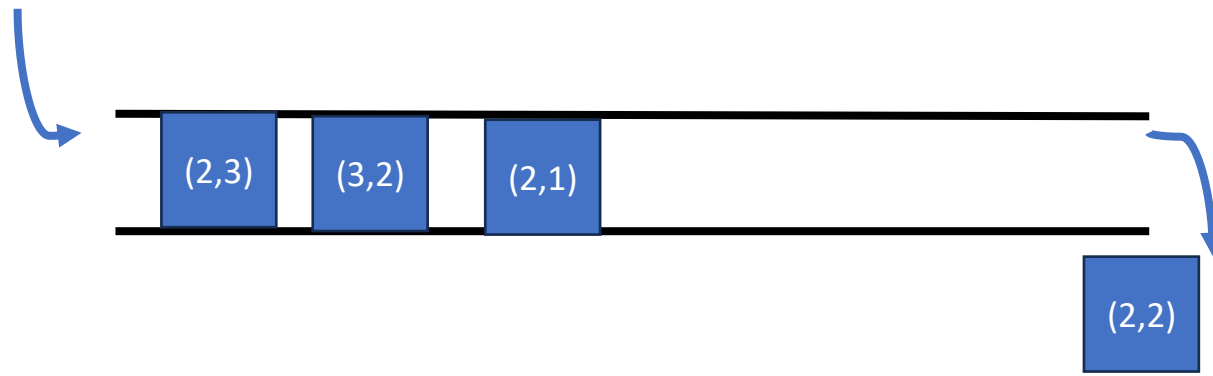
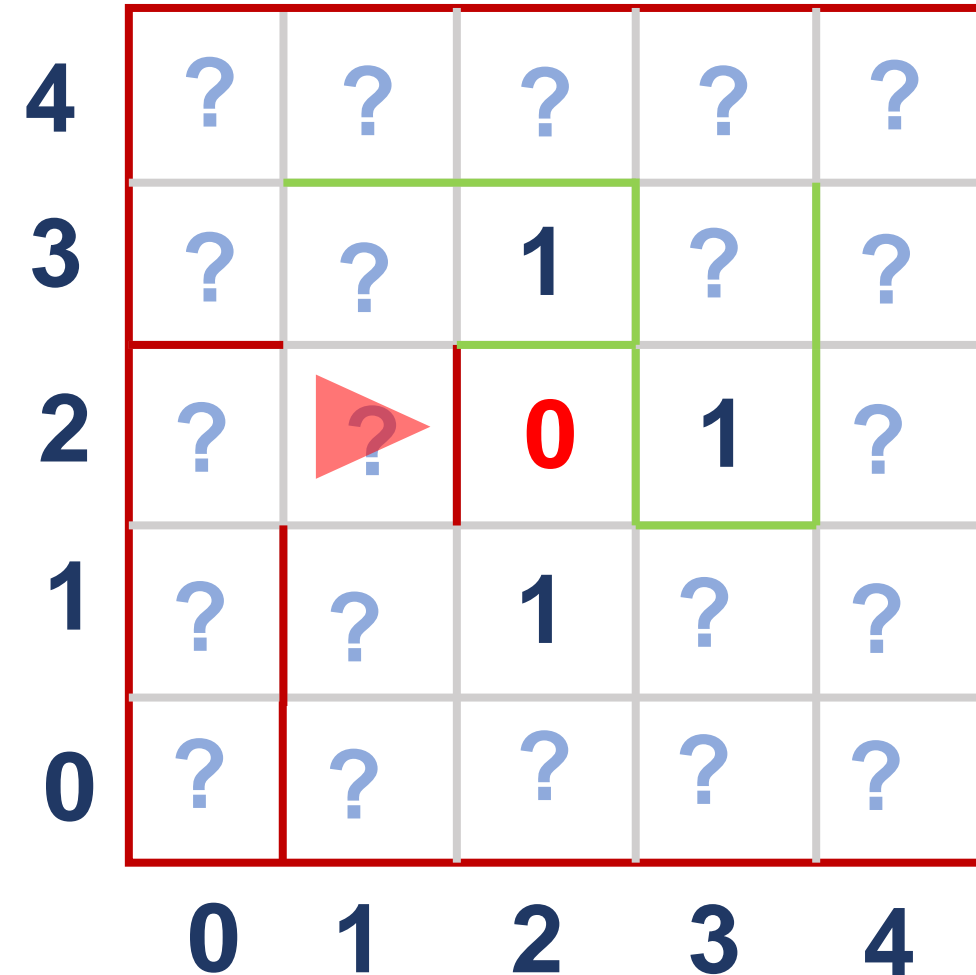
- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



4	?	?	?	?	?
3	?	?	1	?	?
2	?	?	0	1	?
1	?	?	1	?	?
0	?	?	?	?	?
	0	1	2	3	4

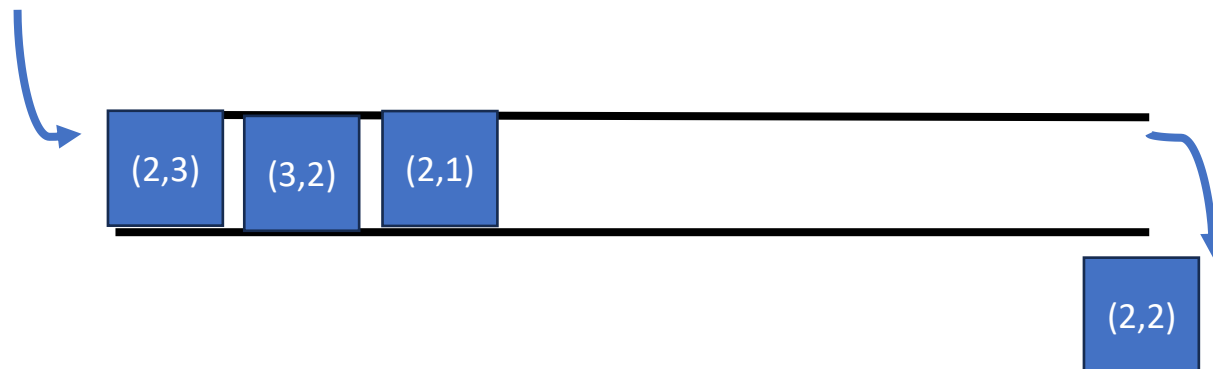
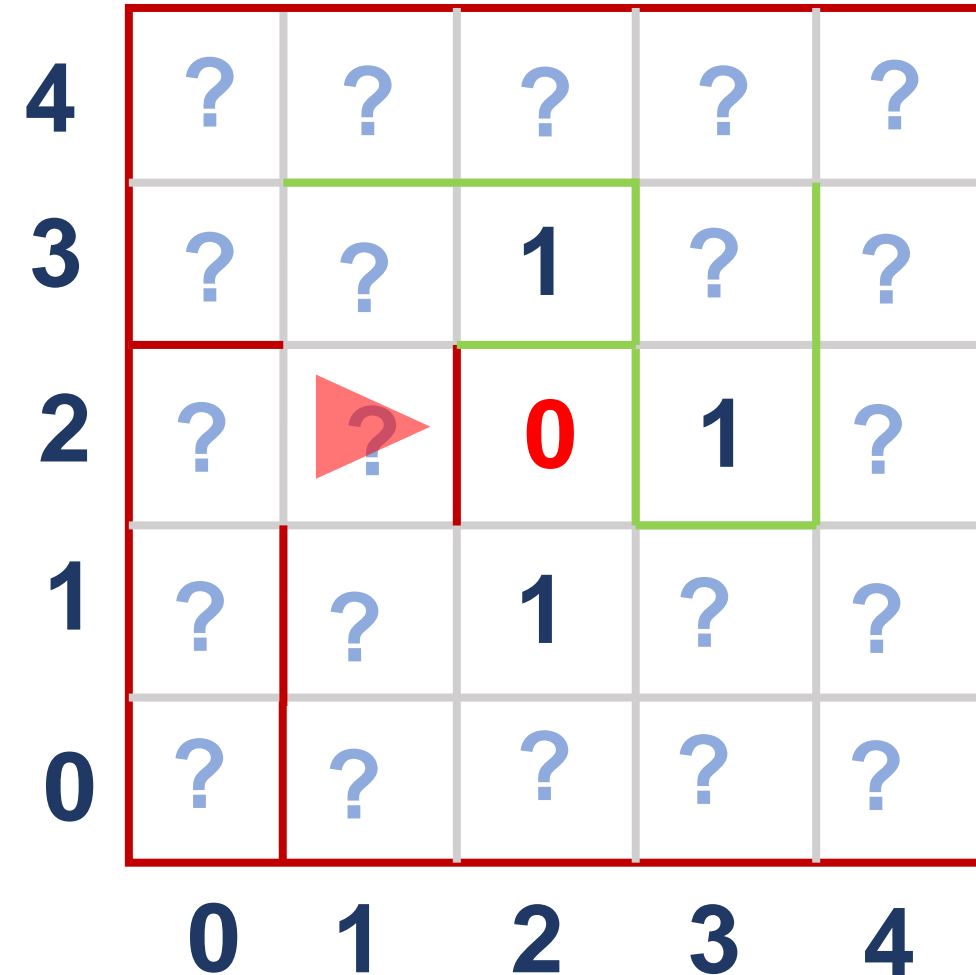
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



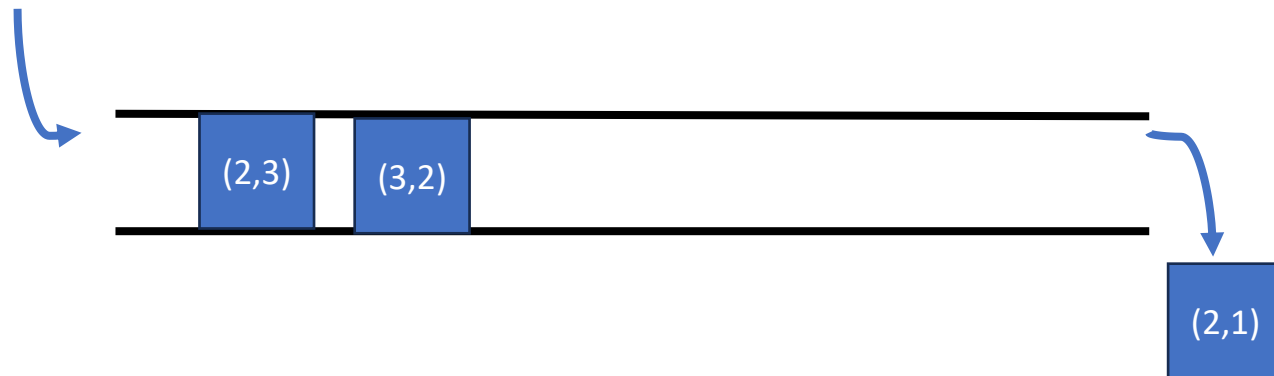
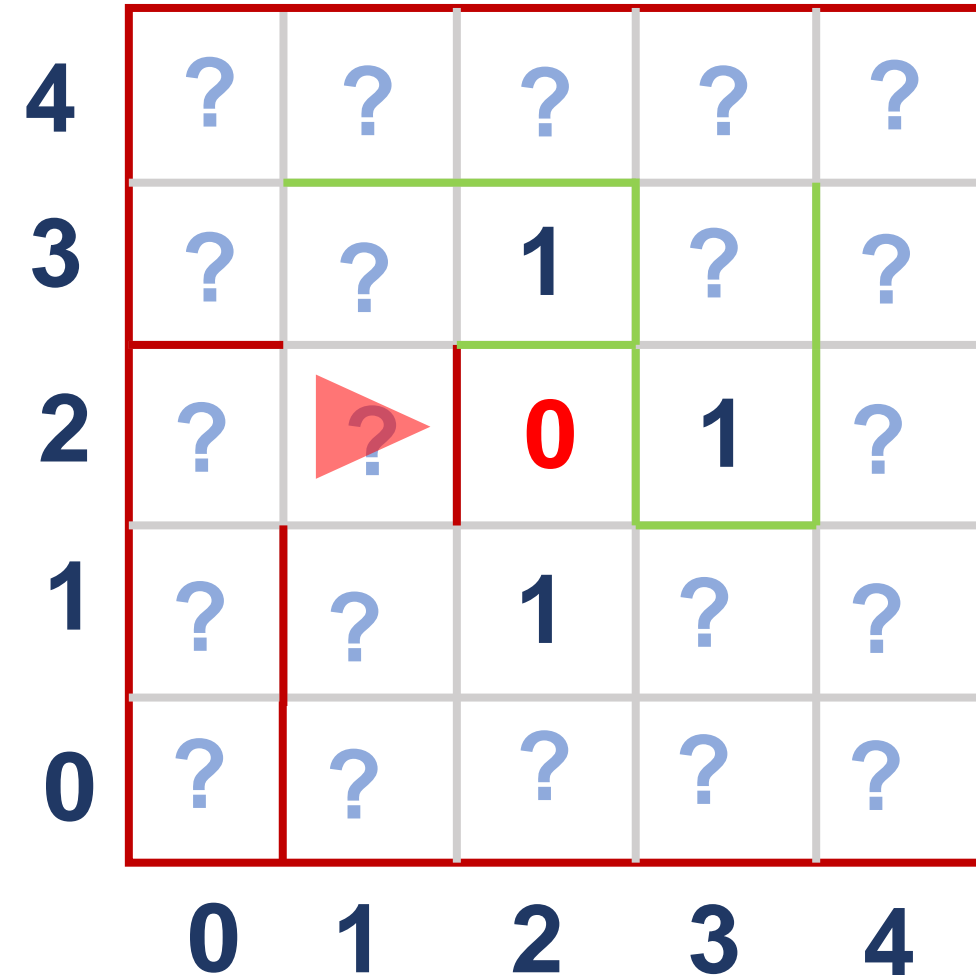
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) **While queue is not empty:**
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



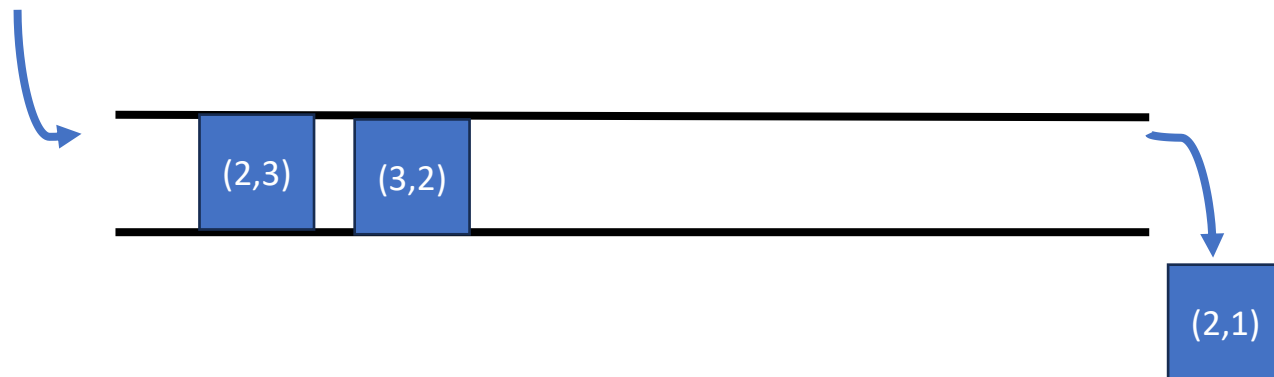
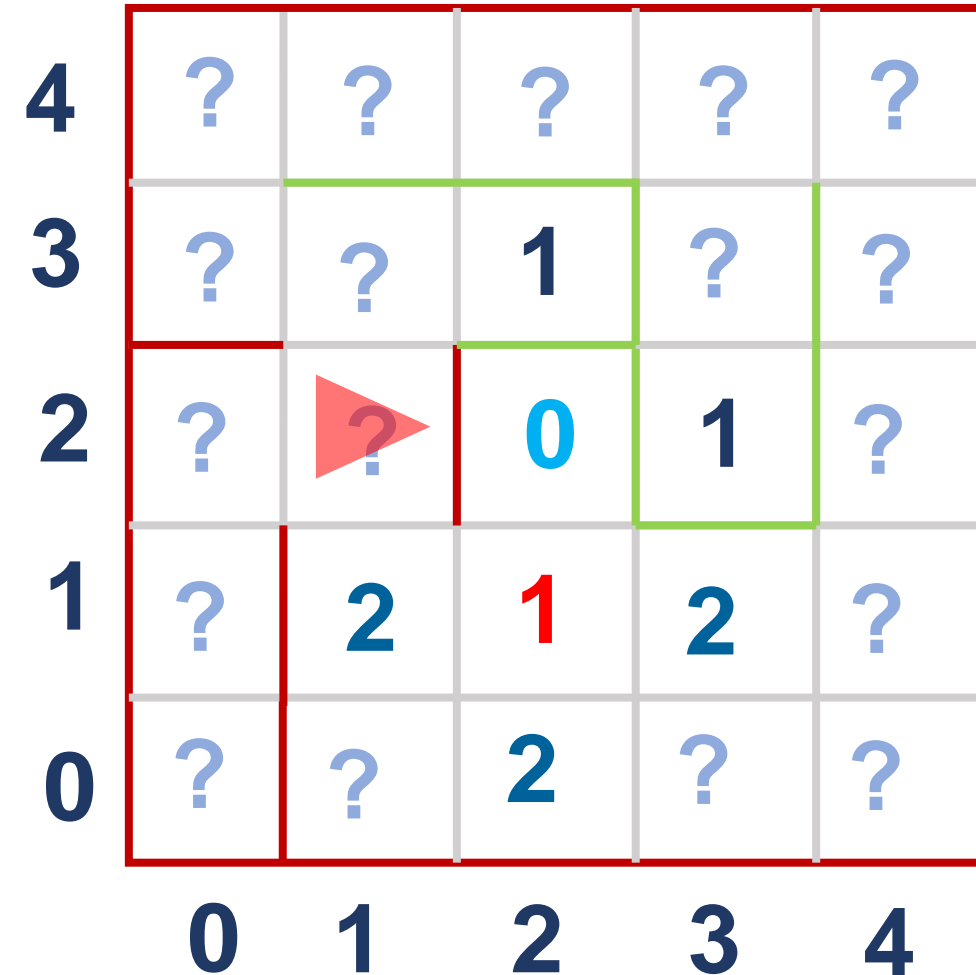
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



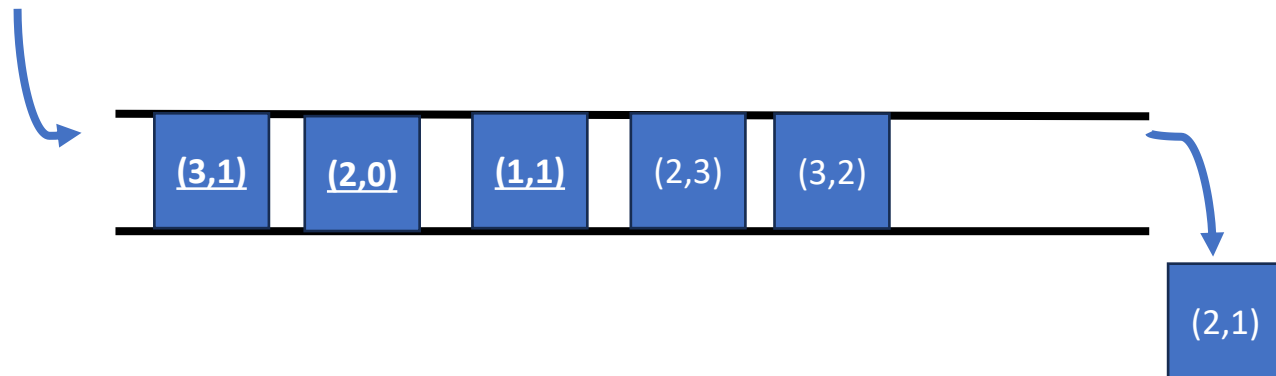
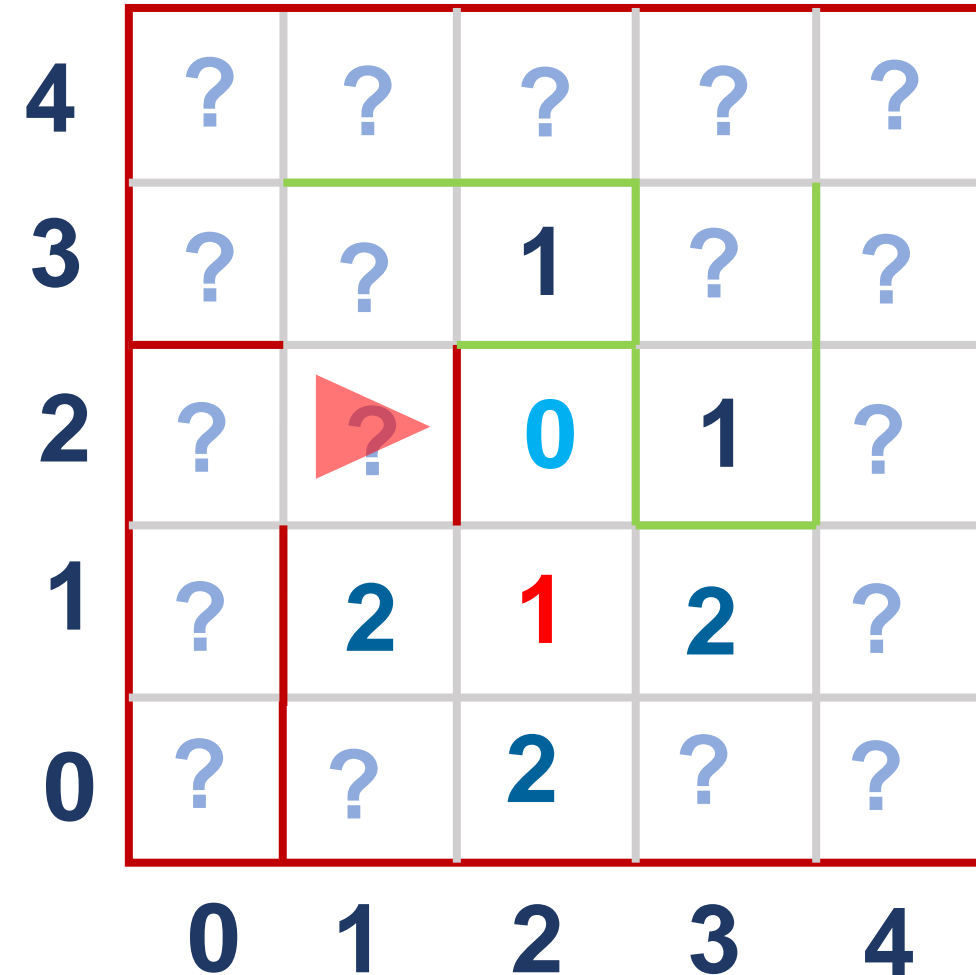
Floodfill Pseudocode

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Floodfill Pseudocode

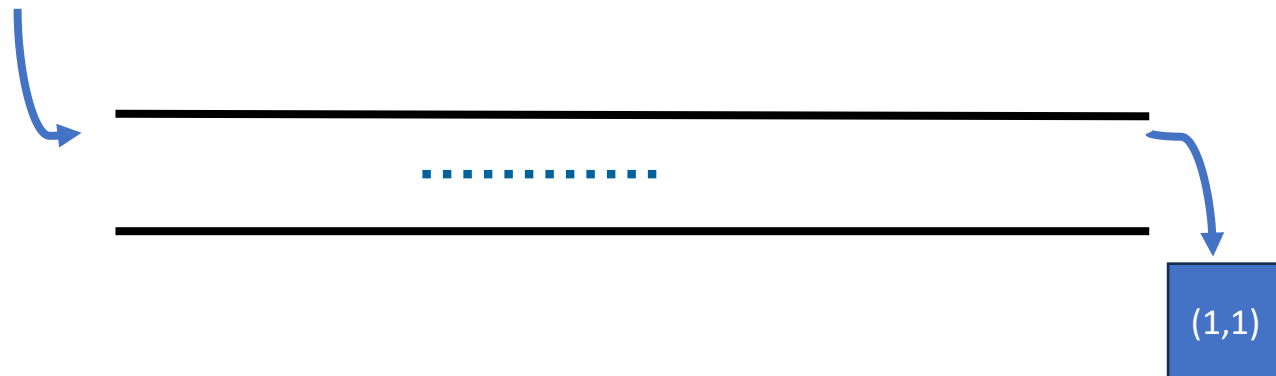
- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue



Skip some steps

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

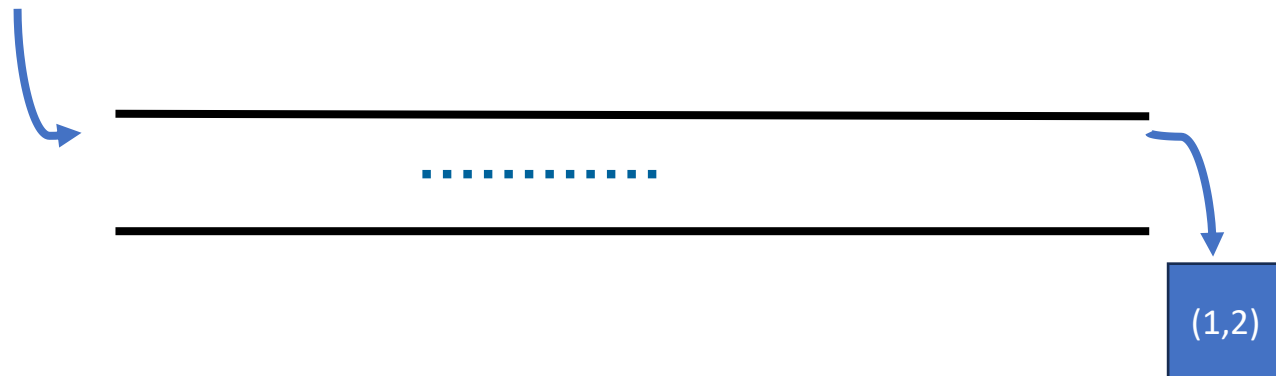
4	?	?	2	?	?
3	?	2	1	2	?
2	?	3	0	1	2
1	?	2	1	2	?
0	?	3	2	?	?
	0	1	2	3	4



Skip some more steps

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

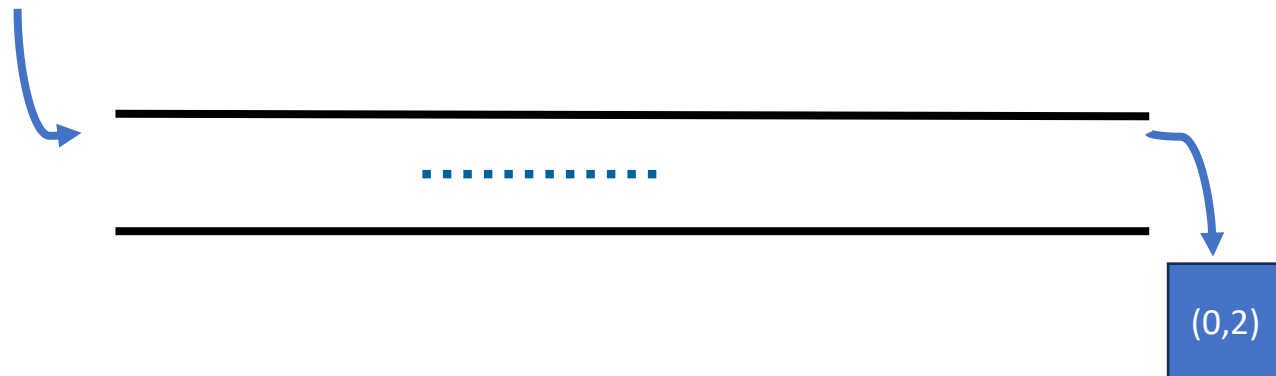
4	?	3	2	3	?
3	3	2	1	2	3
2	4	3	0	1	2
1	?	2	1	2	3
0	?	3	2	3	?
	0	1	2	3	4



Skip some more steps

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

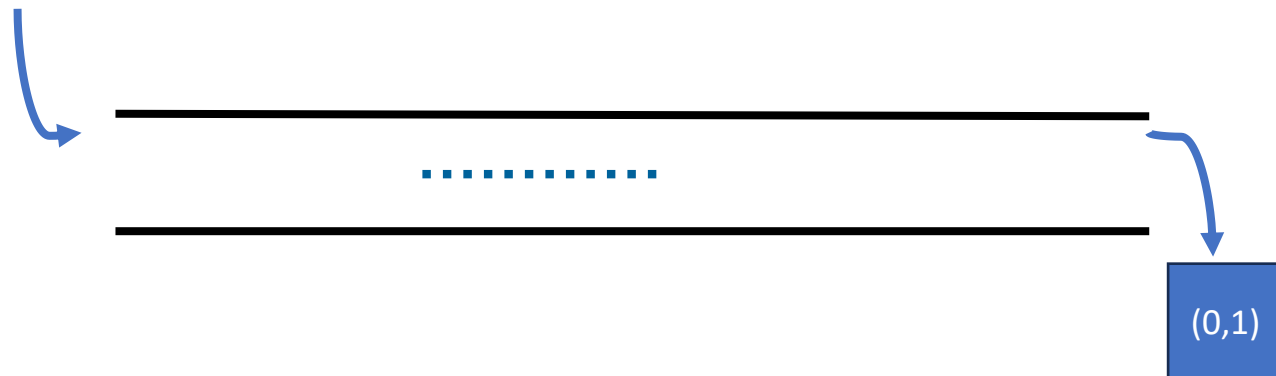
4	4	3	2	3	4
3	3	2	1	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	?	3	2	3	4
	0	1	2	3	4



Skip some more steps

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) While queue is not empty:
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

4	4	3	2	3	4
3	3	2	1	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4



Skip some more steps

- 1) Set all cells except goal to “blank state”
- 2) Set goal cell(s) value to 0 and add to queue
- 3) **While queue is not empty:**
 - a) Take front cell in queue “out of line” for consideration
 - b) Set all blank and accessible neighbors to front cell’s value + 1
 - c) Add cells we just processed to the queue

4	4	3	2	3	4
3	3	2	1	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

4	4	3	2	3	4
3	3	2	1	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

The image shows a 5x5 grid with numbers in each cell. The grid is labeled with row indices 0-4 on the left and column indices 0-4 on the bottom. A red path is highlighted, starting at row 2, column 1 and moving to row 2, column 2. A green path is highlighted, starting at row 2, column 2 and moving to row 2, column 3. A blue path is highlighted, starting at row 2, column 3 and moving to row 2, column 4. A red triangle points to the cell at row 2, column 2, which contains the number 3.

4	4	3	2	3	4
3	3	2	1	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

A 5x5 grid of numbers with a red border and a red triangle pointing right from the cell (3,2). A green vertical line is at column 4, and a green horizontal line is at row 2.

4	6	5	4	3	4
3	5	4	5	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

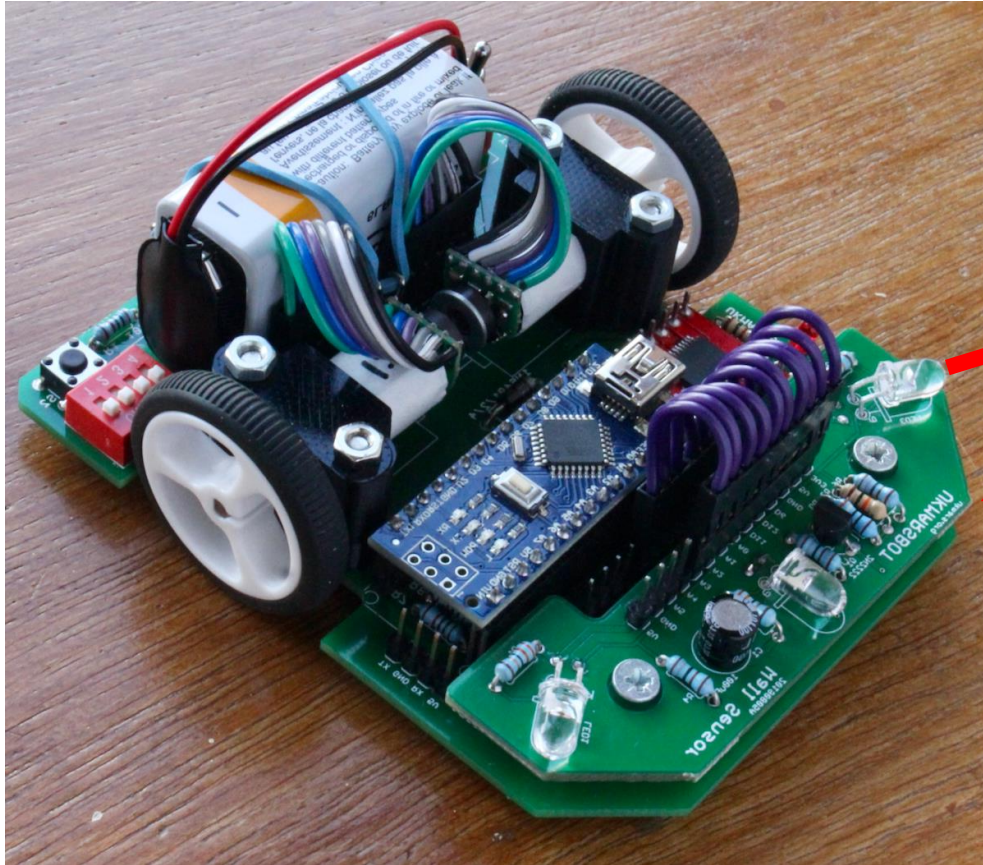
4	6	5	4	3	4
3	5	4	5	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

4	6	5	4	3	4
3	5	4	5	2	3
2	4	3	0	1	2
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

The image shows a 5x5 grid with numbers in each cell. The grid is labeled with row indices (0-4) on the left and column indices (0-4) at the bottom. A red path is highlighted, starting at (0,1) and ending at (4,4). A green path is highlighted, starting at (1,3) and ending at (4,4). A red triangle points to the cell (1,1) containing the number 2.

4	6	5	6	5	6
3	5	4	5	4	5
2	4	3	0	3	4
1	5	2	1	2	3
0	6	3	2	3	4
	0	1	2	3	4

Wall Detection



Right Receiver A0
Middle Receiver A1
Left Receiver A2

`analogRead(A1);` - > value
between 0-1023

Emitters: Pin 12
`digitalWrite(12, HIGH);`

Wall Detection

```
const int EMITTERS = 12; // EMITTERS
const int RED_LED = 13; // RED LED AT H BRIDGE

const int INDICATOR_LED_R = 6; // INDICATOR LED RIGHT
const int INDICATOR_LED_L = 11; // INDICATOR LED LEFT
// Phototransistors
const int RIGHT_SENSOR = A0;
const int LEFT_SENSOR = A2;
const int MIDDLE_SENSOR = A1;
void setup() {
  Serial.begin(9600);

  pinMode(EMITTERS, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(INDICATOR_LED_R, OUTPUT);
  pinMode(INDICATOR_LED_L, OUTPUT);
```

Wall Detection

```
void loop() {  
  
    digitalWrite(EMITTERS, HIGH);  
    Serial.println(analogRead(RIGHT_SENSOR));  
}
```

STARTER CODE ON GITHUB:

<https://github.com/ieecity/micromouse2024/>