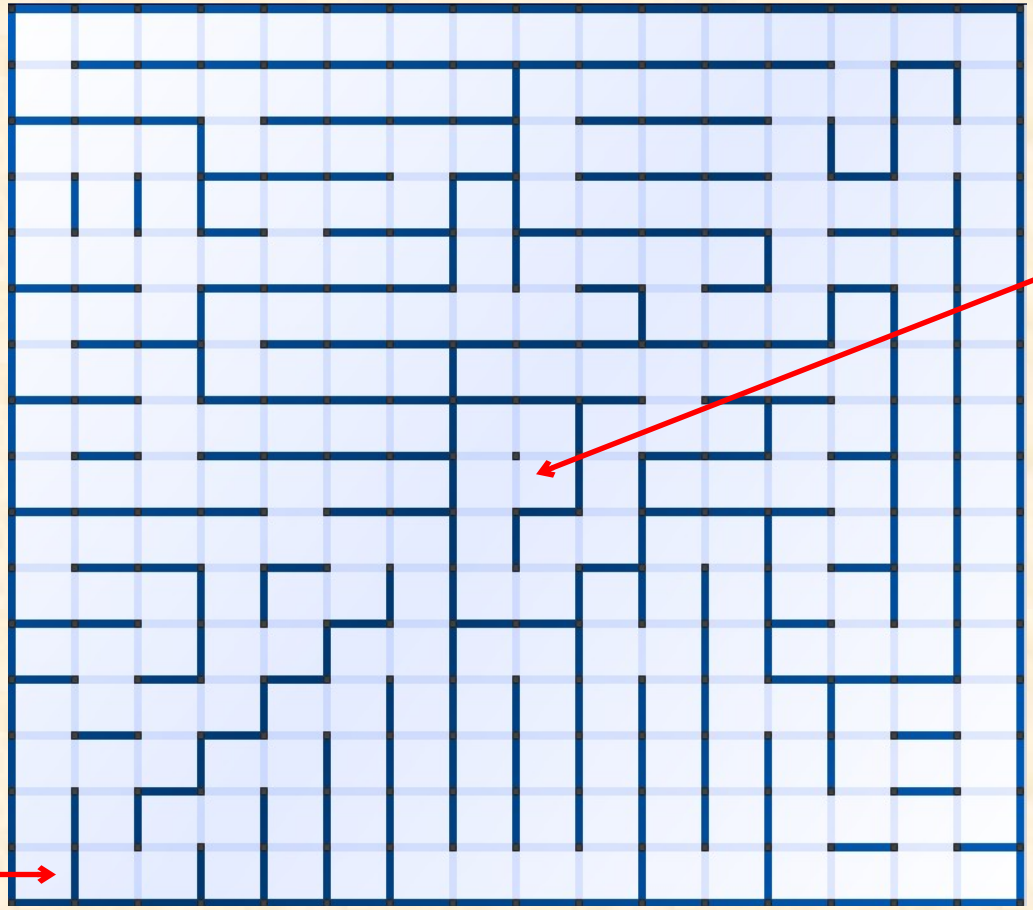
IEEE Competition Sample Maze

- **16-cells by 16-cells Maze**

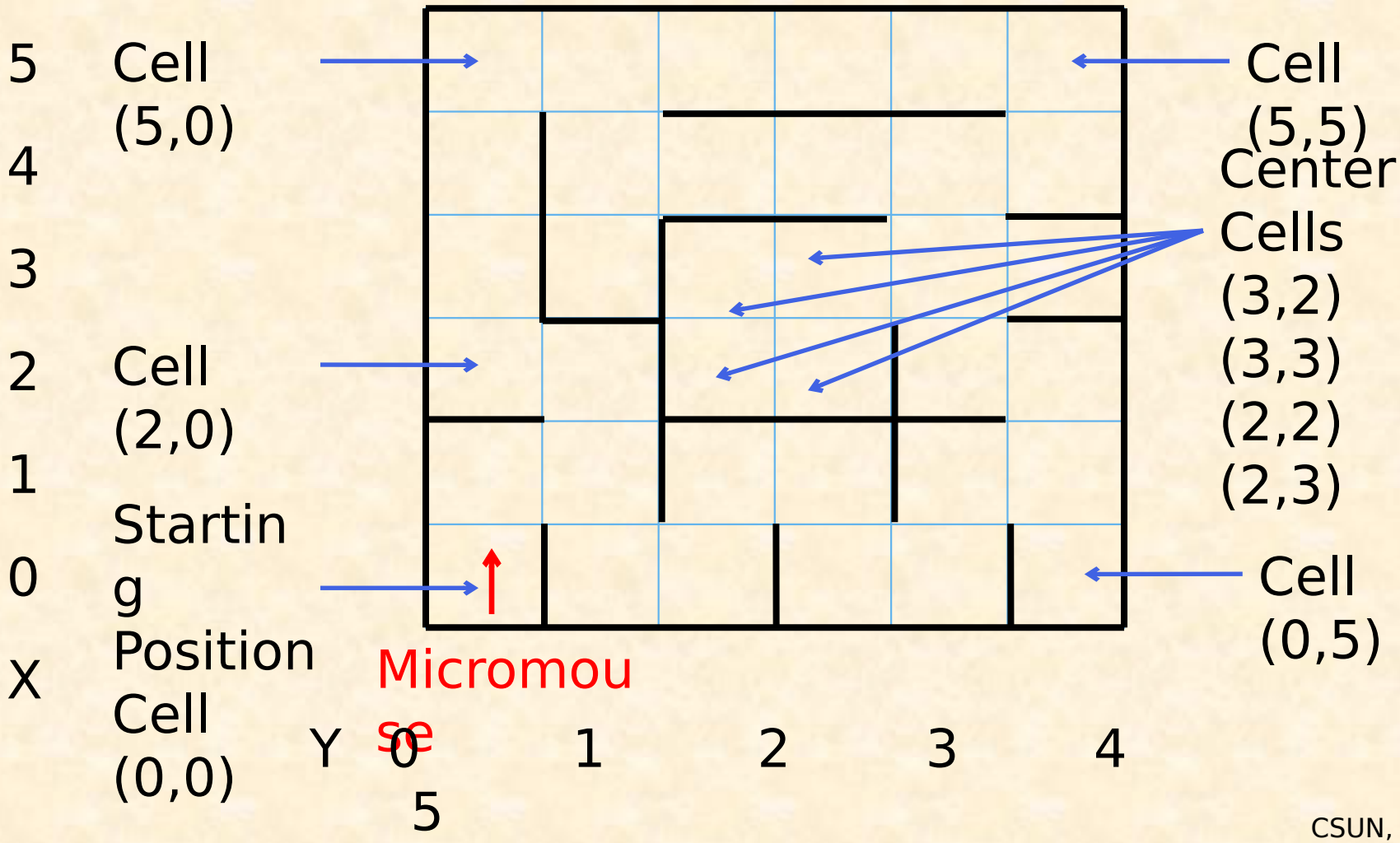


Floodfill Algorithm

- Applies concept of water always flowing from a higher elevation to a lower one
- Assigns each cell in the maze a value that represents how far the cell is from the center. The center cells value is zero; the farther from the center the higher the values.
- Higher values represent higher elevation. The center cells have lowest elevation.

Sample 16-cell by 16-Cell maze

- **Cells labeling (X,Y)**



Floodfill Algorithm Flooding

- **Initially, no walls have been detected.**
- **Flooding of Maze with distances from center**

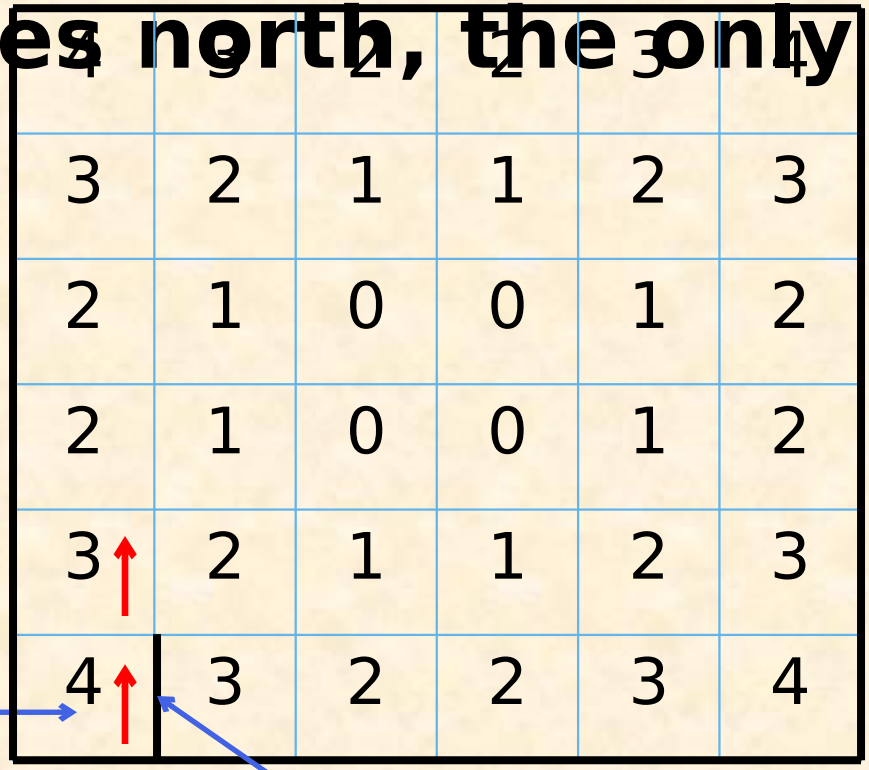
3	2	1	1	2	3
2	1	0	0	1	2
2	1	0	0	1	2
3	2	1	1	2	3
4	3	2	2	3	4

Starting Position Cell (0,0)

Micromouse

Floodfill Algorithm

- **Micromouse is placed at the starting position**
- **It moves north, the only opening**



Starting Position Cell (0,0)

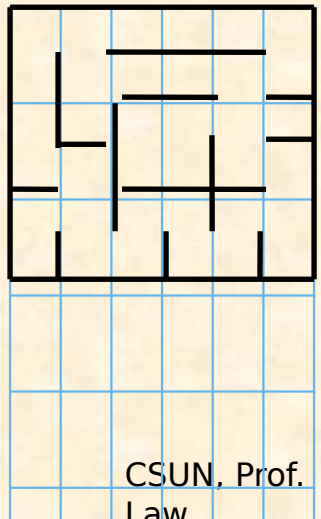
East wall always exists

Floodfill Algorithm

- **Micromouse sees north wall**
- **Each time micromouse reaches a new cell, flooding occurs**

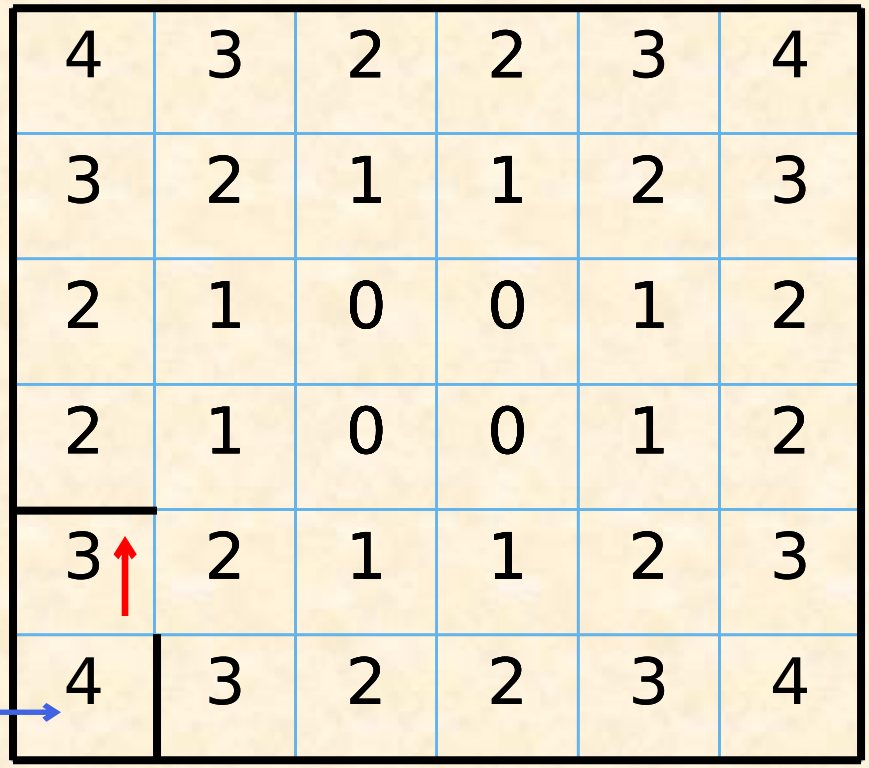
4	3	2	2	3	4
3	2	1	1	2	3
2	1	0	0	1	2
2	1	0	0	1	2
3	2	1	1	2	3
4	3	2	2	3	4

Starting Position Cell (0,0)



Floodfill Algorithm

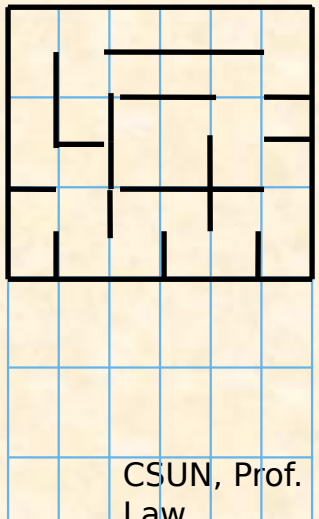
- **Micromouse sees north wall**
- **Each time micromouse reaches a new cell, flooding occurs**



Floodfill Algorithm

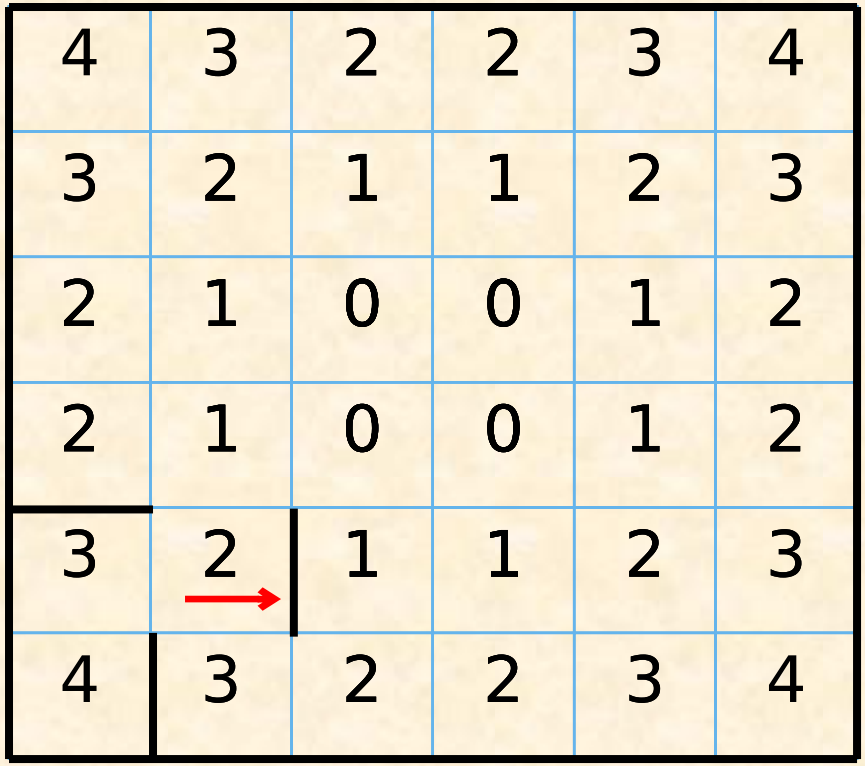
- **After flooding, micromouse moves eastwards, the only opening & lower distance.**
- **Micromouse sees the east wall.**

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



Floodfill Algorithm

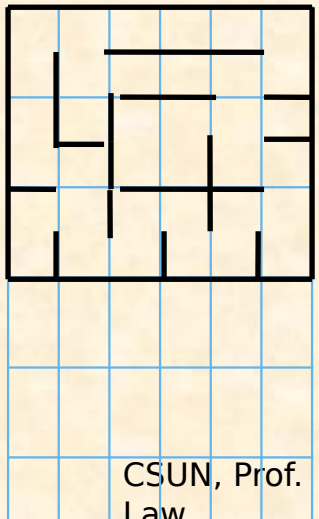
- **Micromouse sees the east wall.**
- **Each time micromouse reaches a new cell, flooding occurs**



Floodfill Algorithm

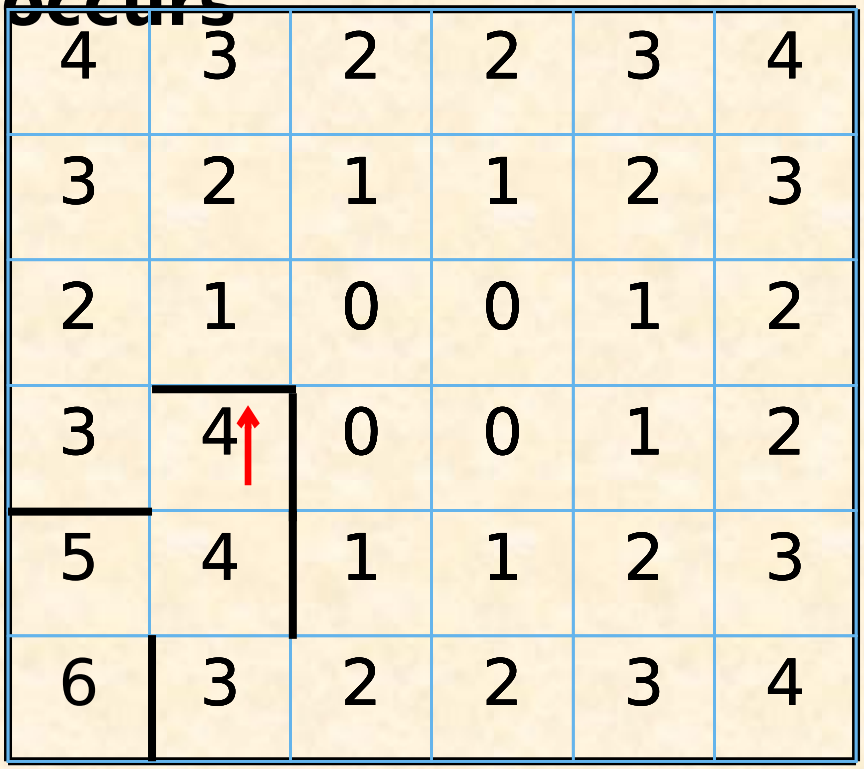
- **After flooding, micromouse moves northwards to lower distance cell.**
- **Micromouse sees the east wall and the north wall.**

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



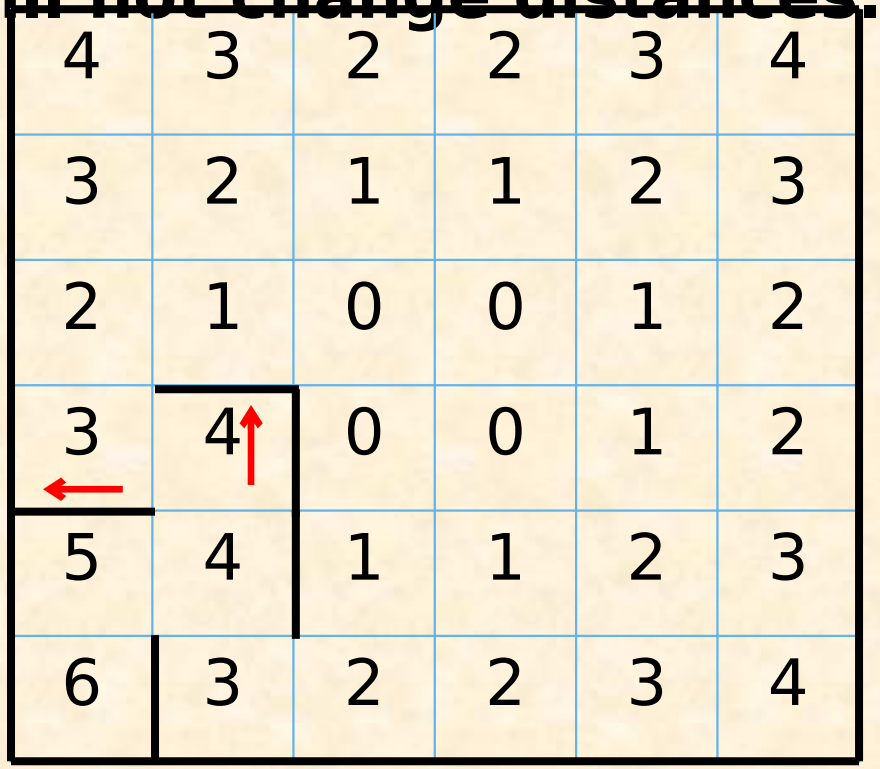
Floodfill Algorithm

- **Micromouse sees the east wall and the north wall.**
- **Each time micromouse reaches a new cell, flooding occurs**



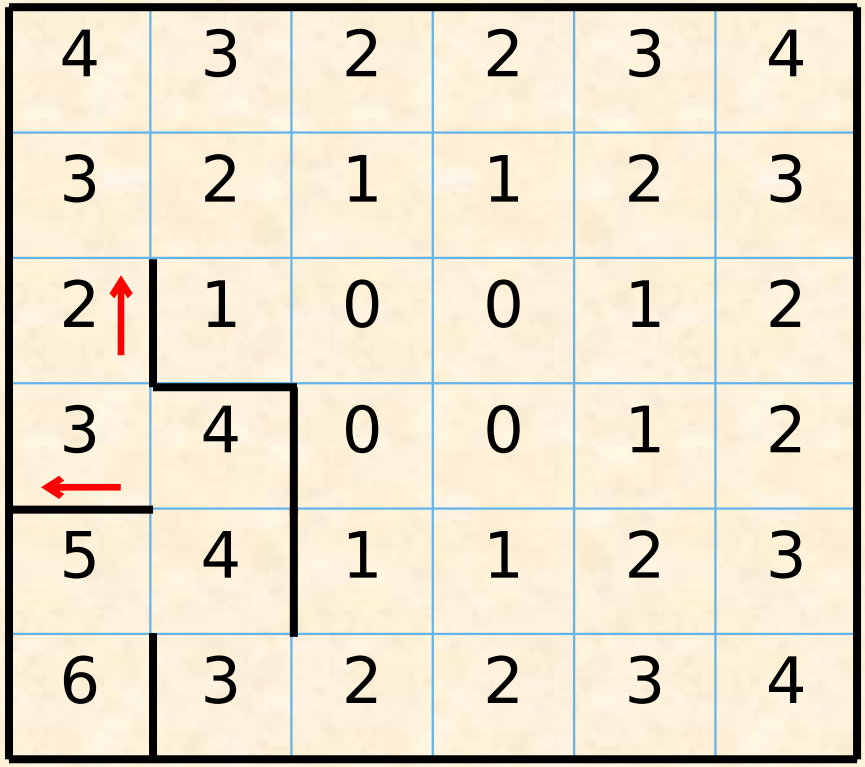
Floodfill Algorithm

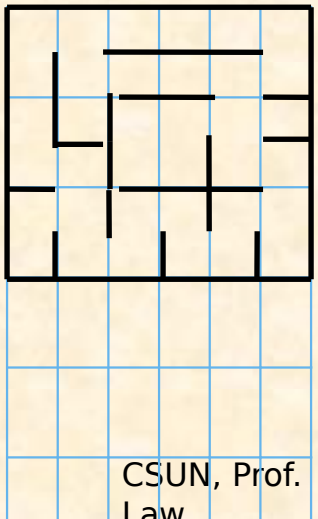
- **After flooding, micromouse moves westwards to lower distance cell.**
- **Each time micromouse reaches a new cell, flooding occurs. Since no new wall is found, flooding will not change distances.**



Floodfill Algorithm

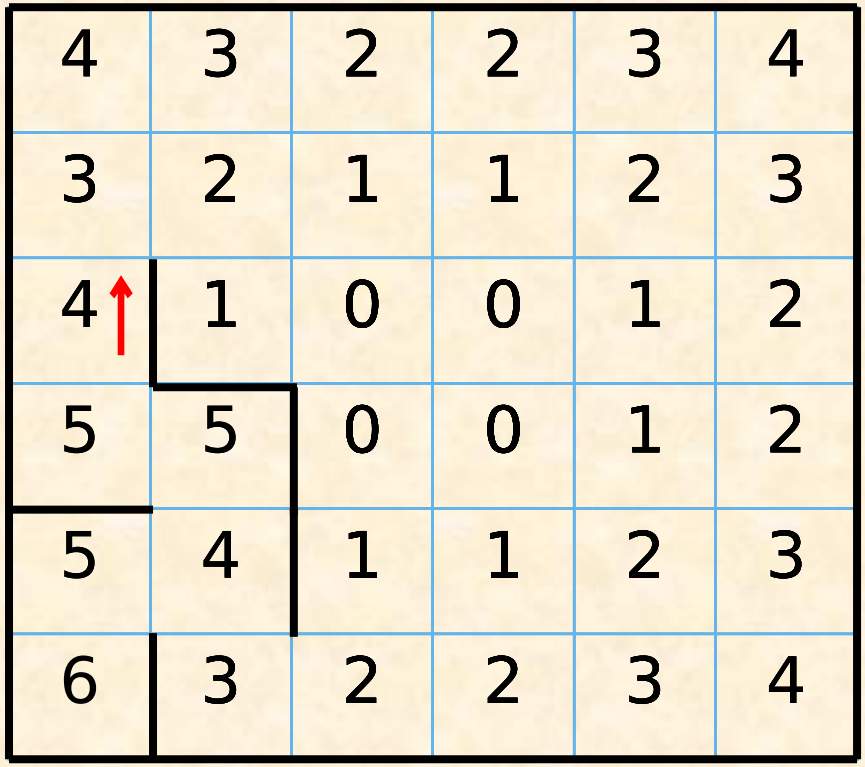
- **After flooding, micromouse moves northwards to lower distance cell.**
- **Micromouse sees east wall.**





Floodfill Algorithm

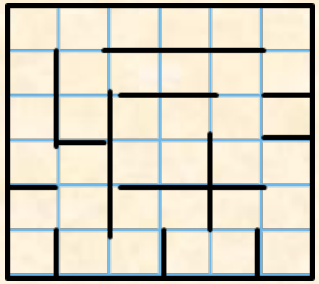
- **Micromouse sees east wall.**
- **Each time micromouse reaches a new cell, flooding occurs.**



Floodfill Algorithm

- **After flooding, micromouse moves northwards to lower distance cell.**
- **Micromouse sees east wall.**

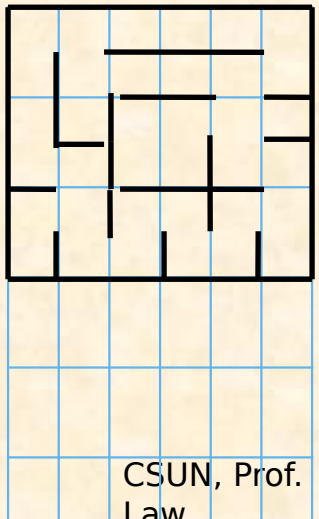
4	3	2	2	3	4
3 ↑	2	1	1	2	3
4 ↑	1	0	0	1	2
5	5	0	0	1	2
5	4	1	1	2	3
6	3	2	2	3	4



Floodfill Algorithm

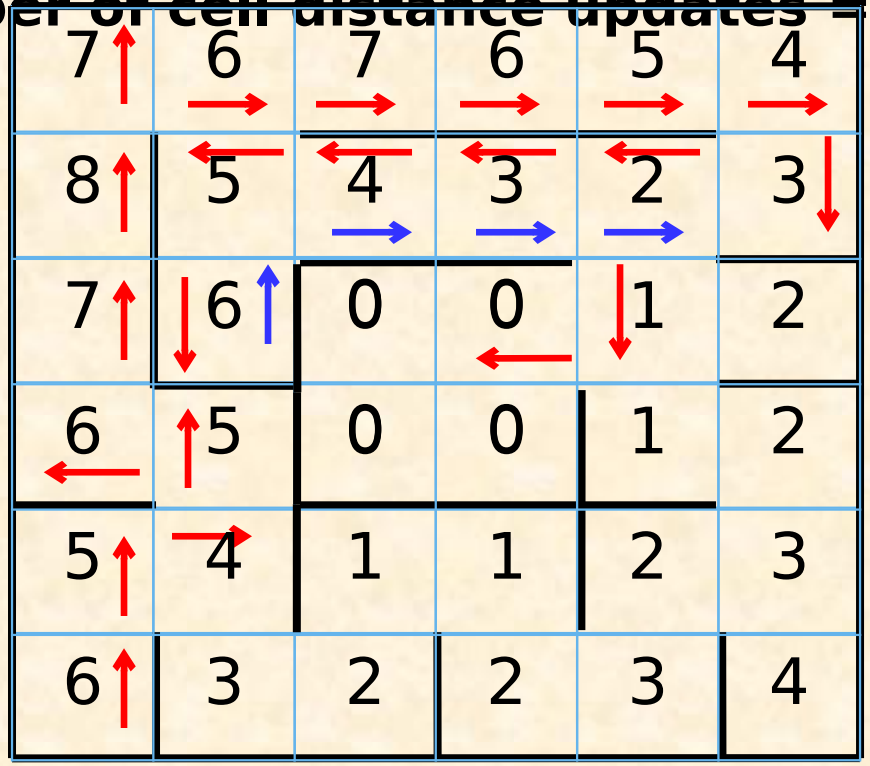
- **Micromouse sees east wall.**
- **Each time micromouse reaches a new cell, flooding occurs.**

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



Floodfill Algorithm

- **The same process continues until micromouse reaches the center.**
- **Final distances and paths to center are shown.**
- **Total squares traversed (not counting starting square) = 23**
- **Total number of cell distance updates = 23 * 36 = 828**



Modified Floodfill Algorithm

- **The modified flood fill algorithm differs from Floodfill algorithm in that it does not flood the maze each time a new cell is reached. Instead it updates only the relevant neighboring cells using the following revised recursive steps:**
  - Push the current cell location (x,y) onto the stack.
  - Repeat the following steps while the stack is not empty.
    - a. Pull the cell location (x,y) from the stack.

      b. If the minimum distance of the neighboring open cells, md, is not equal to the present cell's distance - 1, replace the present cell's distance with md + 1, and push **all neighbor locations** onto the stack.

Modified Floodfill Algorithm Flooding

- **Initially, no walls have been detected.**
- **Flooding of Maze with distances from center**

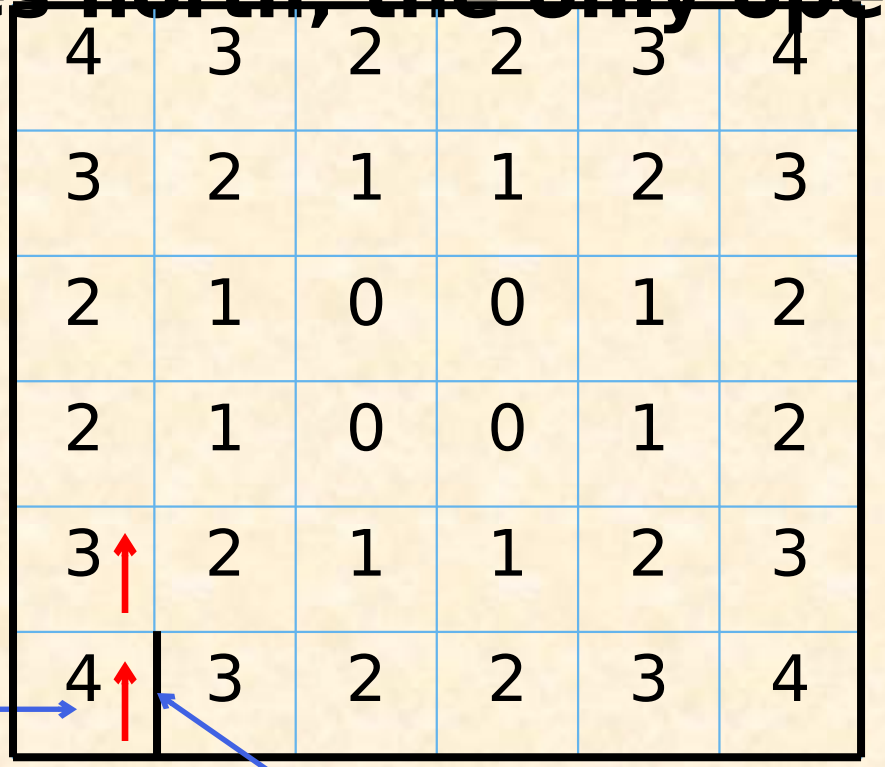
4	3	2	2	3	4
3	2	1	1	2	3
2	1	0	0	1	2
2	1	0	0	1	2
3	2	1	1	2	3
4	3	2	2	3	4

Starting Position Cell (0,0)

Micromouse

Modified Floodfill Algorithm

- **Micromouse is placed at the starting position**
- **It moves north, the only opening**



Starting Position Cell (0,0)

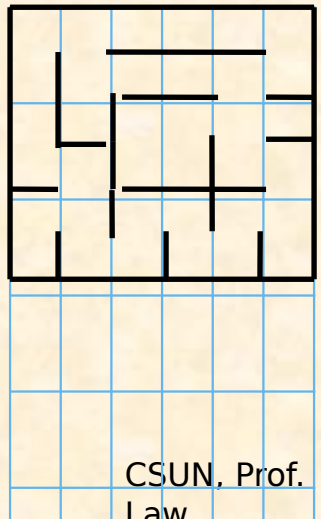
East wall always exists

Modified Floodfill Algorithm

- **Micromouse sees north wall**
- **Each time micromouse reaches a new cell, recursive updating steps are evaluated. The minimum distance of the neighboring open cells, md is 2 = 3 - 1 ; hence no update is necessary.**

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

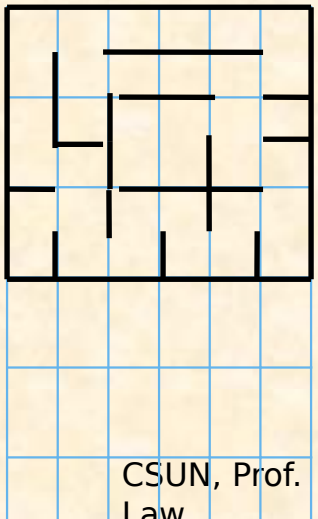
Starting Position Cell



Modified Floodfill Algorithm

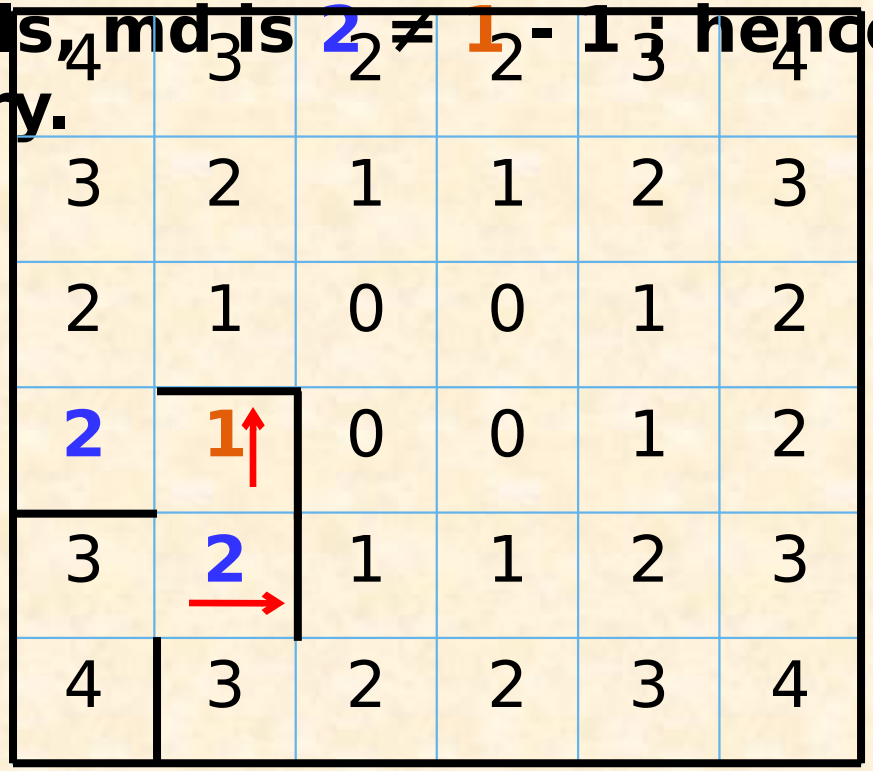
- **Micromouse moves eastwards, the only opening & lower distance. It sees the east wall.**
- **Each time micromouse reaches a new cell, recursive updating steps are evaluated. The minimum distance of the neighboring open cells, md is 1 = 2 - 1; hence no update is necessary.**

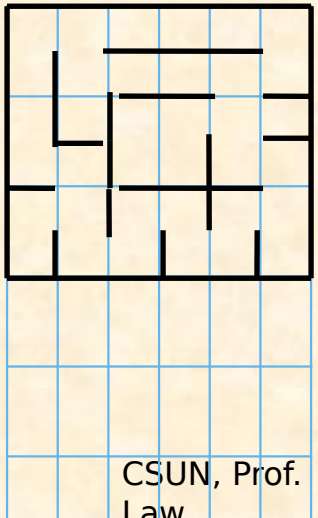
4	3	2	2	3	4
3	2	1	1	2	3
2	1	0	0	1	2
2	1	0	0	1	2
3	2	1	1	2	3
4	3	2	2	3	4



Modified Floodfill Algorithm

- **Micromouse moves northwards to lower distance cell and sees the east wall and the north wall.**
- **The minimum distance of the neighboring open cells, md is 2 ≠ 1 - 1 ; hence update is necessary.**





Modified Floodfill Algorithm

- **Recursive distance update steps**
- **1. Push the current cell location (2,1) onto the stack.**
- **2a. Pull the cell location (2,1) from the stack.**
  2b. Since the distance at (2,1) – 1 = 0 is not equal to md = 2, the minimum of its open neighbors (2,0) and (1,1), update the distance at (2,1) to md + 1 = 2 + 1 = 3. Push all neighbor locations (3,1), (2,0) and (1,1), except the center location (2,1), onto the stack.

Cell (3,1)

Cell (2,0)

Cell (1,1)

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

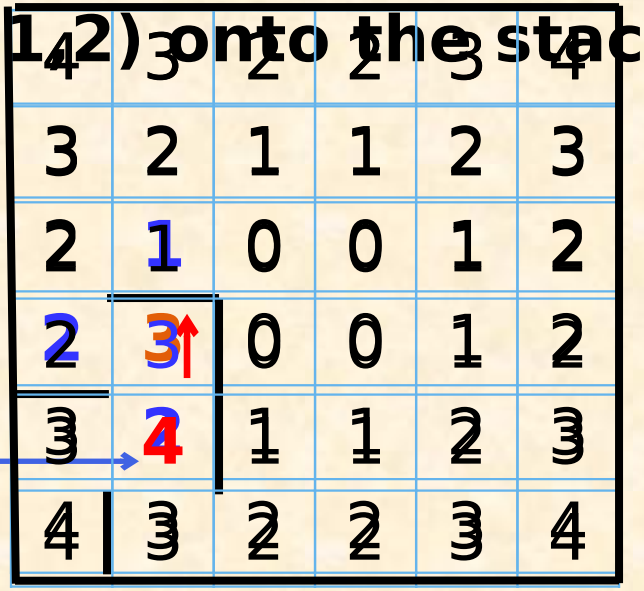
Stack

(1,1)
(2,0)
(3,1)

Top

Modified Floodfill Algorithm

- **Recursive distance update steps**
- **Since the stack is not empty,**
- **2a. Pull the cell location (1,1) from the stack.**
  2b. Since the distance at (1,1) – 1 = 1 is not equal to md = 3, the minimum of its open neighbors (2,1), (0,1) and (1,0), update the distance at (1,1) to md + 1 = 3 + 1 = 4. Push all neighbor locations (2,1), (0,1), (1,0), and (1,2) onto the stack.



Cell (1,1)

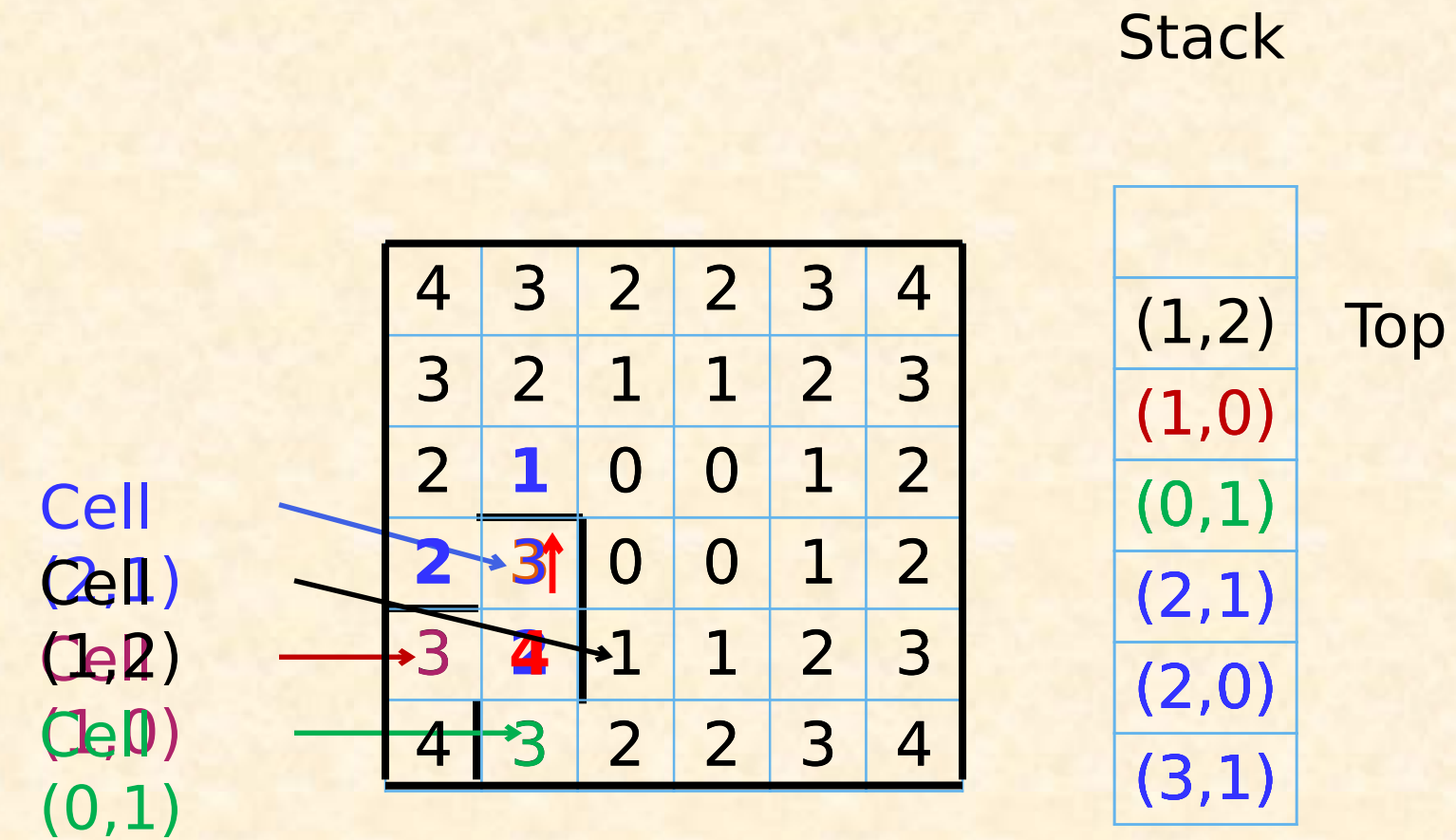
Stack

(1,1)
(2,0)
(3,1)

Top

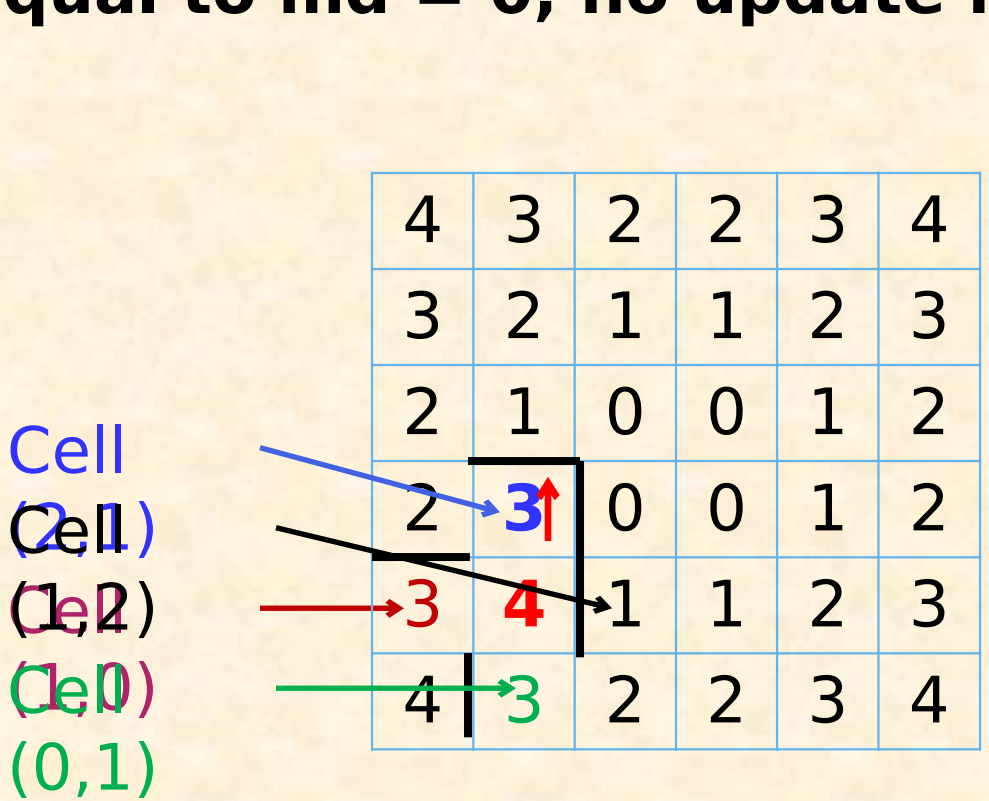
Modified Floodfill Algorithm

- **After the distance update, push all neighbor locations (2,1), (0,1), (1,0), and (1,2) onto the stack.**



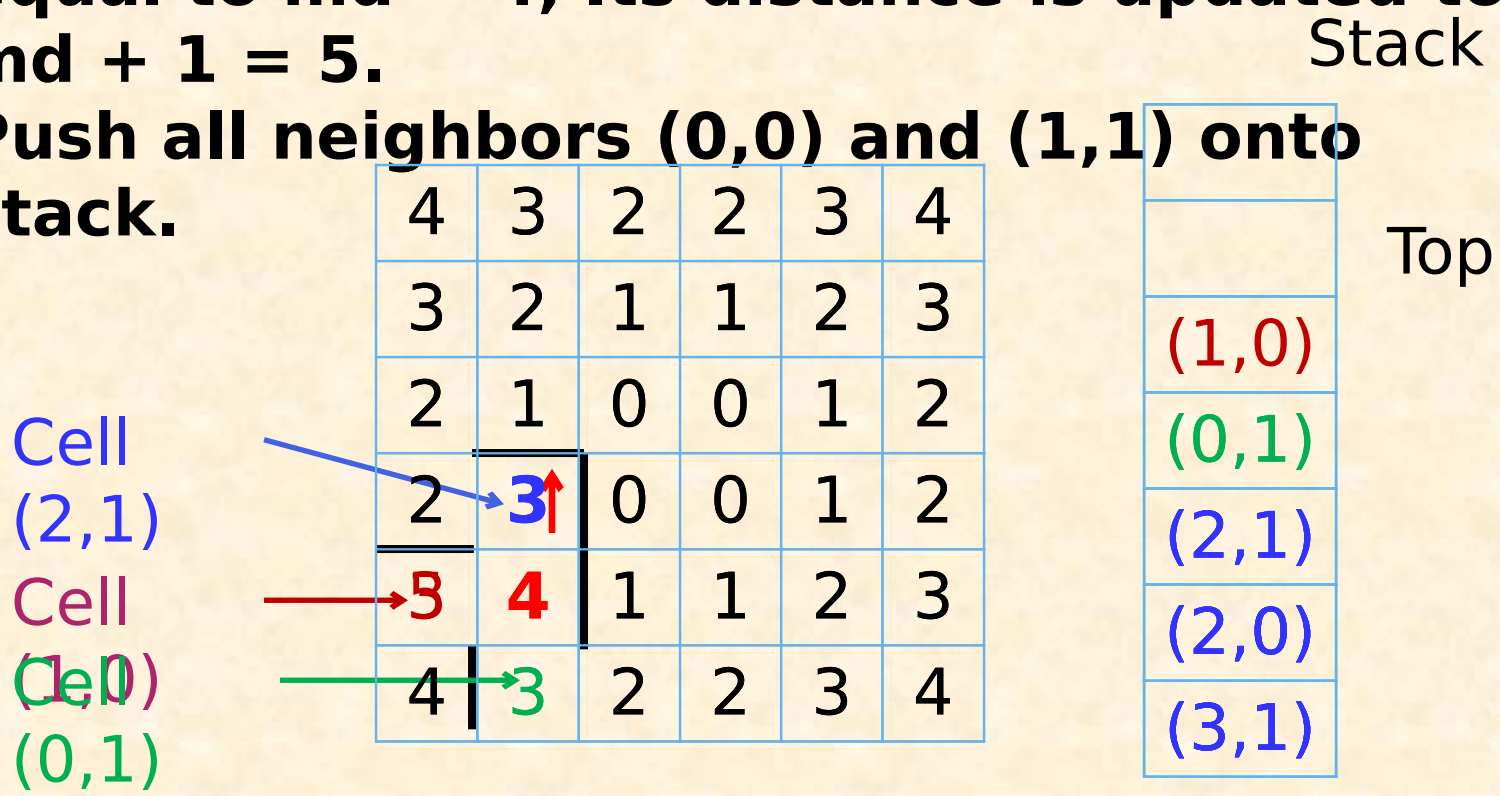
Modified Floodfill Algorithm

- **Recursive distance update steps**
- **Since the stack is not empty,**
- **2a. Pull the cell location (1,2) from the stack.**
  2b. Since the distance at (1,2) – 1 = 0 is equal to md = 0, no update is necessary.



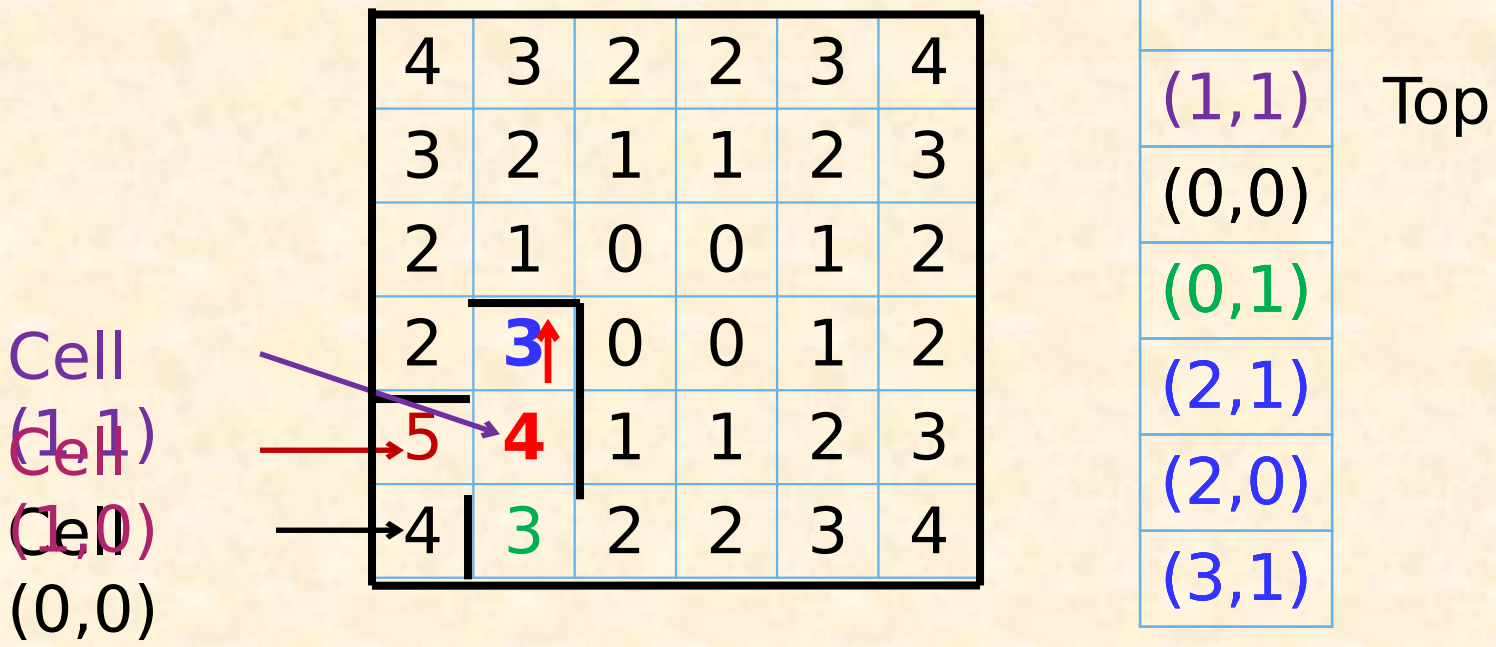
Modified Floodfill Algorithm

- **Recursive distance update steps**
- **Since the stack is not empty,**
- **2a. Pull the cell location (1,0) from the stack.**
  2b. Since the distance at (1,0) – 1 = 2 is not equal to md = 4, Its distance is updated to md + 1 = 5.
  Push all neighbors (0,0) and (1,1) onto stack.



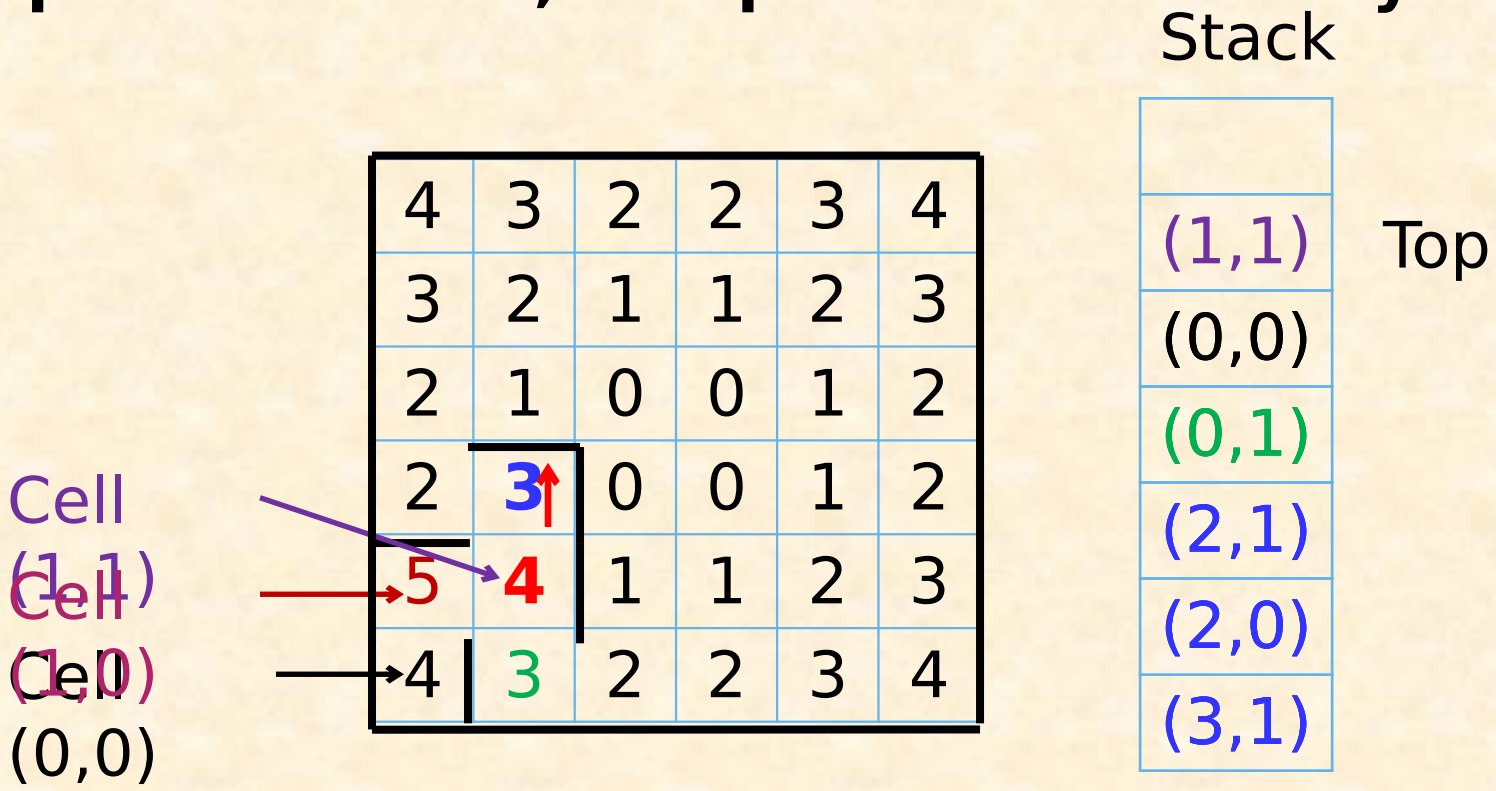
Modified Floodfill Algorithm

- **After the distance update at (1,0)**
  Push all neighbors (0,0) and (1,1) onto stack.



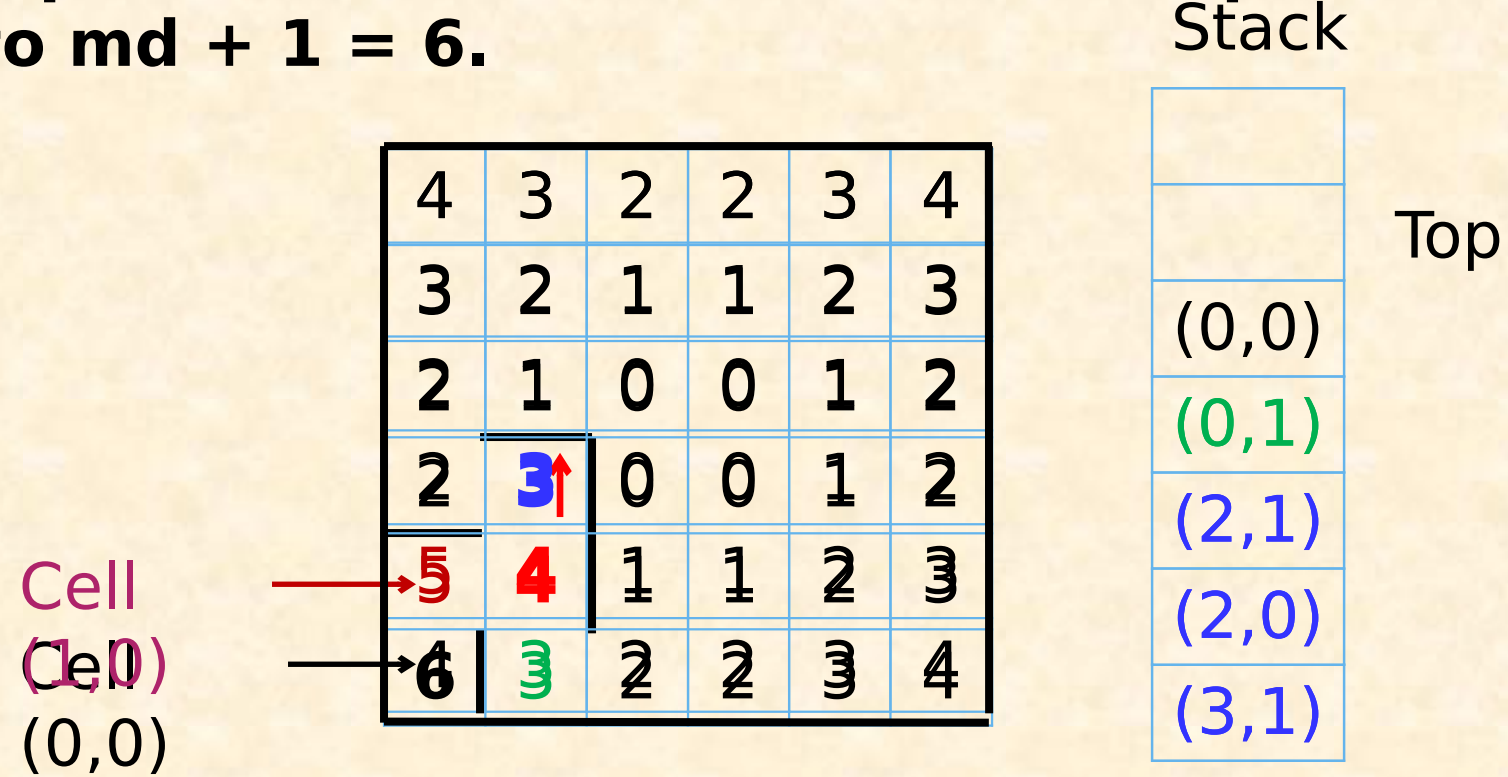
Modified Floodfill Algorithm

- **Recursive distance update steps**
- **Since the stack is not empty,**
- **2a. Pull the cell location (1,1) from the stack.**
  2b. Since the distance at (1,1) – 1 = 3 is equal to md = 3, no update is necessary.



Modified Floodfill Algorithm

- **Recursive distance update steps**
- **Since the stack is not empty,**
- **2a. Pull the cell location (0,0) from the stack.**
  2b. Since the distance at (0,0) – 1 = 4 is equal not to md = 5, its distance updated to md + 1 = 6.



Modified Floodfill Algorithm

- **The same process continues and it will not be shown here.**
- **When the stack is empty, the distances are as shown.**