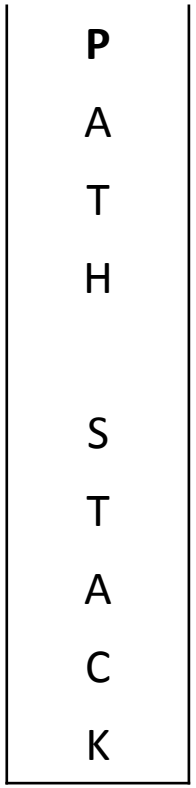
CS 211 - Project 3

Maze Solving

5x5 Maze

- Adding 2 rows and columns to put a border around the maze.
- Green cell is Start position
- Blue cell is End position
- Yellow cells are from input file

*	*	*	*	*	*	*
*	S					*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

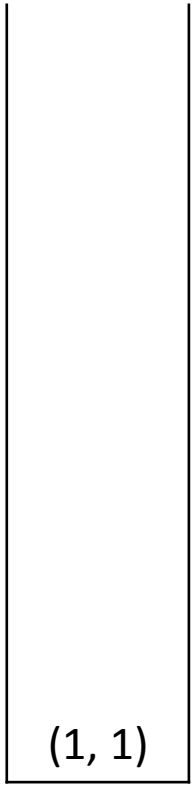


Depth First Search Algorithm

- First: Push start position on the Stack
- Mark start position as Visited

- Assume Search Order: Right, Down, Left Down

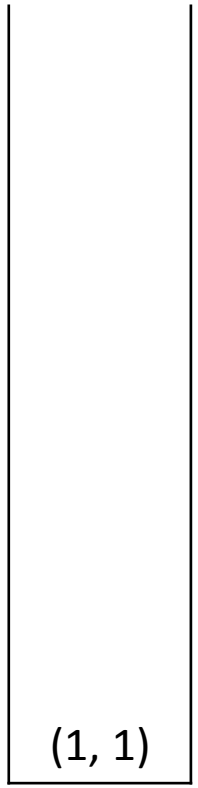
*	*	*	*	*	*	*
*	V					*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

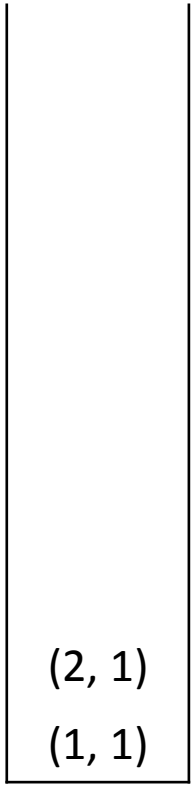
*	*	*	*	*	*	*
*	V					*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

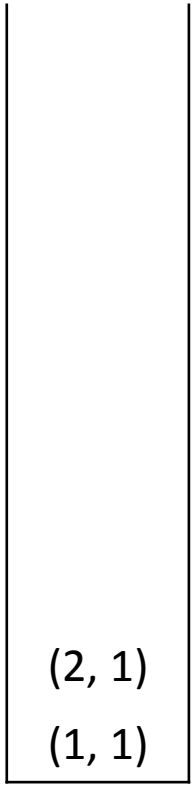
*	*	*	*	*	*	*
*	V	V				*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 1)
  - (3,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

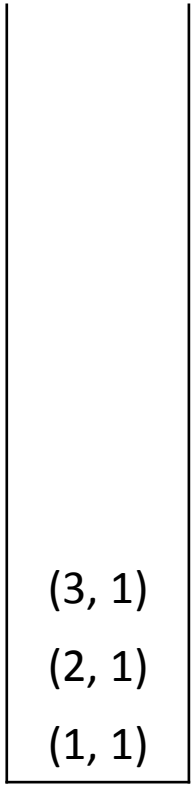
*	*	*	*	*	*	*
*	V	V				*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 1)
  - (3,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

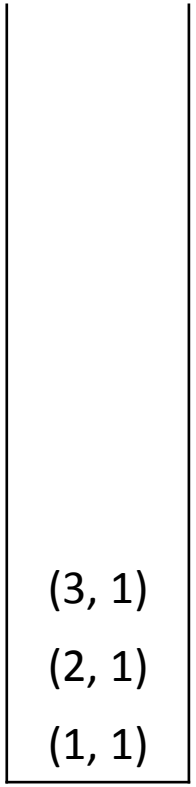
*	*	*	*	*	*	*
*	V	V	V			*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (3, 1)
  - (4,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

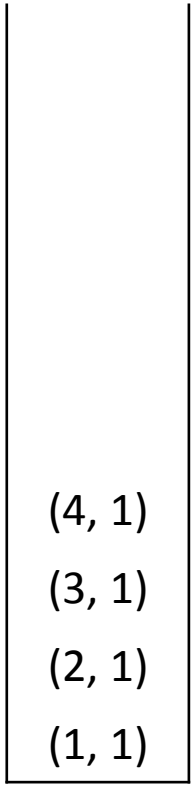
*	*	*	*	*	*	*
*	V	V	V			*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (3, 1)
  - (4,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

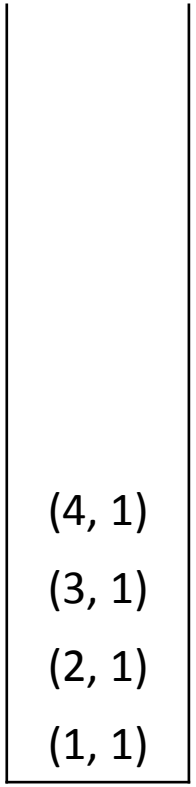
*	*	*	*	*	*	*
*	V	V	V	V		*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (4, 1)
  - (5,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

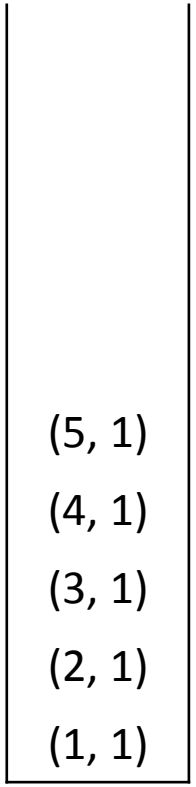
*	*	*	*	*	*	*
*	V	V	V	V		*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (4, 1)
  - (5,1)? yes
- Push on Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 1)
  - (6,1)? no ; (5,2)? no ; (4,1) ? no ; (5,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 1)
  - (6,1)? no ; (5,2)? no ; (4,1) ? no ; (5,0) ? no
- Pop the Stack but keep (5,1) marked as visited!
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (4, 1)
  - (5,1)? no ; (4,2)? no ; (3,1) ? no ; (4,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (4, 1)
  - (5,1)? no ; (4,2)? no ; (3,1) ? no ; (4,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

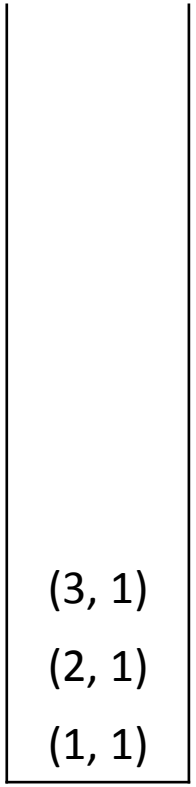
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (3, 1)
  - (4,1)? no ; (3,2)? no ; (2,1) ? no ; (3,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

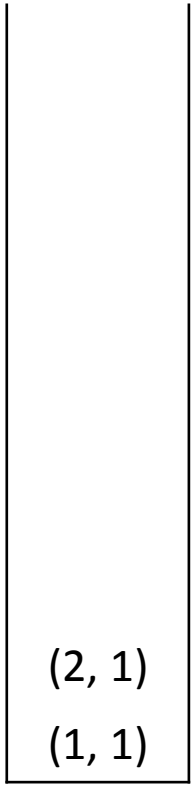
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (3, 1)
  - (4,1)? no ; (3,2)? no ; (2,1) ? no ; (3,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

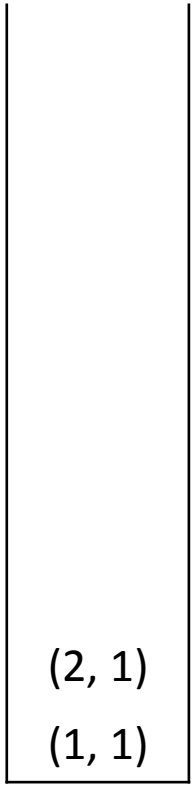
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 1)
  - (3,1)? no ; (2,2)? no ; (1,1) ? no ; (2,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

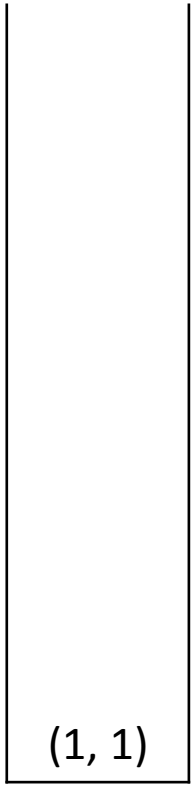
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 1)
  - (3,1)? no ; (2,2)? no ; (1,1) ? no ; (2,0) ? no
- Pop the Stack
- Assume Search Order: Right, Down, Left Down

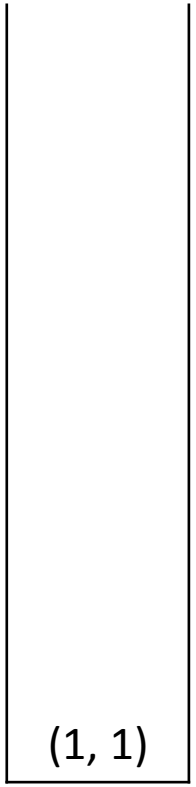
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2,1)? no ; (1,2)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

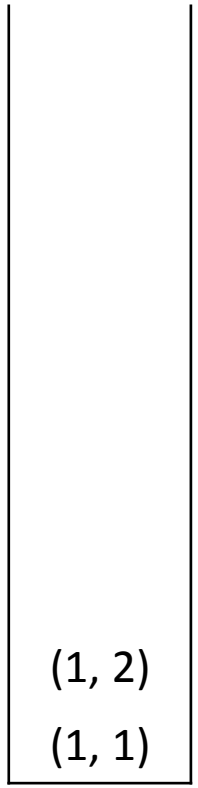
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2,1)? no ; (1,2)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

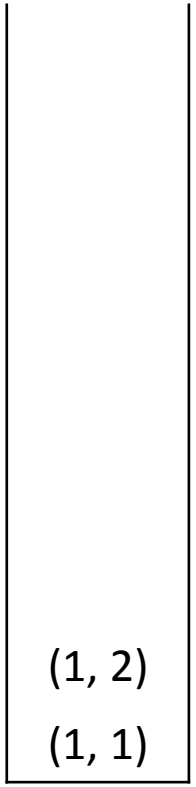
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (2,2)? no ; (1,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

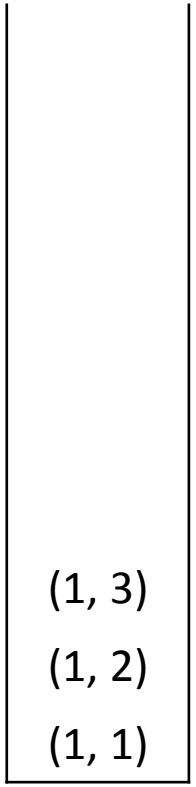
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (2,2)? no ; (1,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

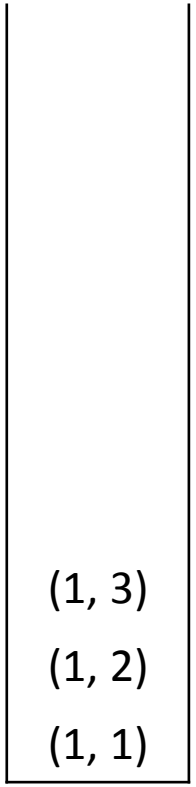
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V					*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

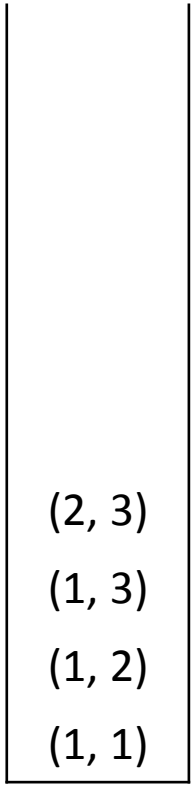
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V					*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

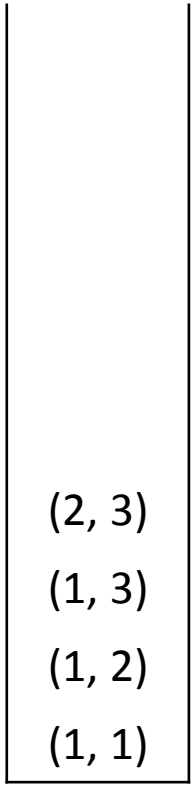
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V				*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 3)
  - (3,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

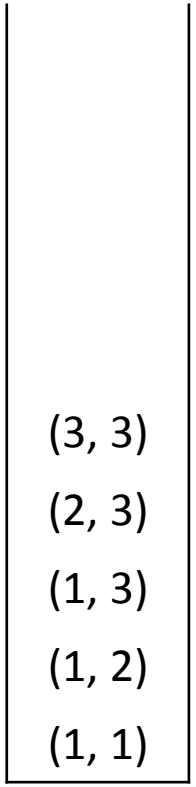
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V				*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 3)
  - (3,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

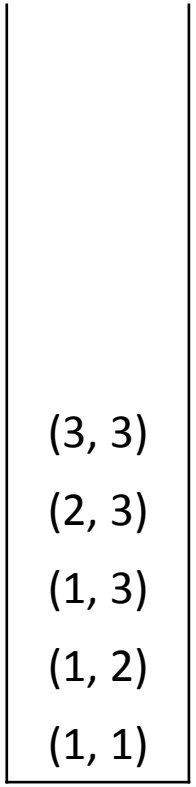
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V			*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (3, 3)
  - (4,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

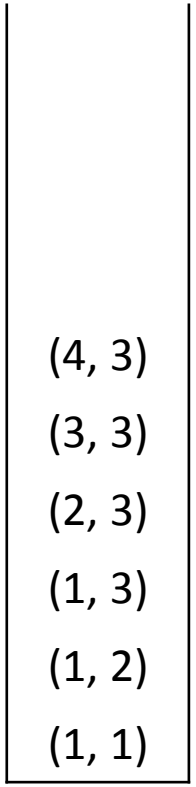
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V			*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (3, 3)
  - (4,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

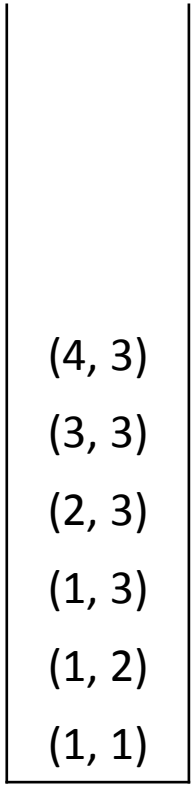
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V		*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (4, 3)
  - (5,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

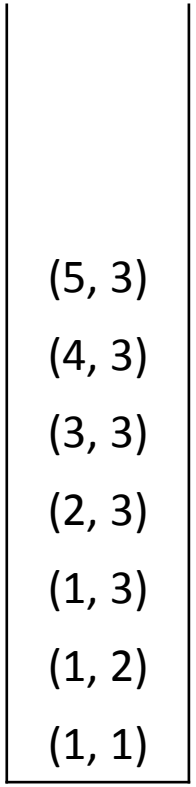
*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V		*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (4, 3)
  - (5,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V	V	*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 3)
  - (6,3)? no ; (5,4) YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V	V	*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 3)
  - (6,3)? no ; (5,4) YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V	V	*
*		*		*	V	*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 4)
  - (6,4)? no ; (5,5) YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V	V	*
*		*		*	V	*
*				*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 4)
  - (6,4)? no ; (5,5) YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Right, Down, Left Down

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V	V	*
*		*		*	V	*
*				*	V	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Is Top of Stack (5, 5) the End Position?

  YES!!!!

- Break out of loop
- Note: Stack has path from END to START

*	*	*	*	*	*	*
*	V	V	V	V	V	*
*	V	*	*	*	*	*
*	V	V	V	V	V	*
*		*		*	V	*
*				*	V	*
*	*	*	*	*	*	*

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

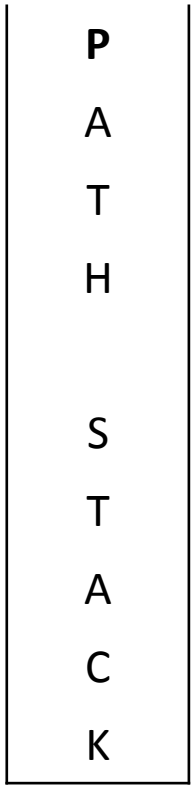
CS 211 - Project 3

Maze Solving Example 2

5x5 Maze

- Adding 2 rows and columns to put a border around the maze.
- Green cell is Start position
- Blue cell is End position
- Yellow cells are from input file

*	*	*	*	*	*	*
*	S					*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*

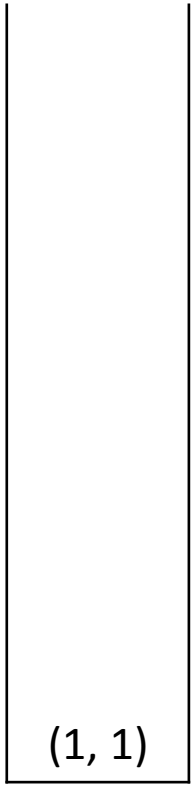


Depth First Search Algorithm

- First: Push start position on the Stack
- Mark start position as Visited

- Assume Search Order: Down, Right, Left, Down

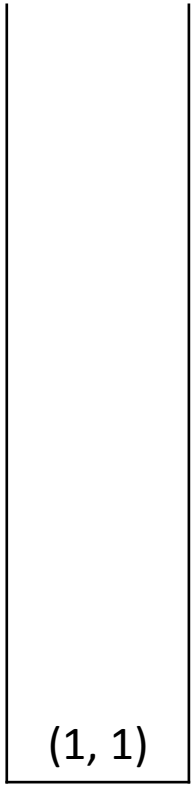
*	*	*	*	*	*	*
*	V					*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (1,2)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

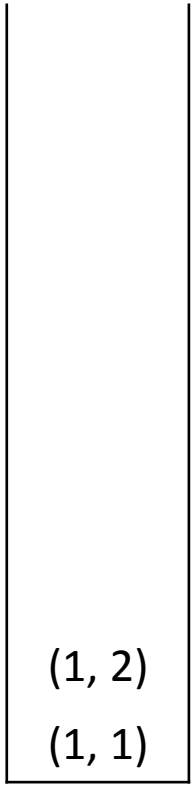
*	*	*	*	*	*	*
*	V					*
*		*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2,1)? no ; (1,2)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

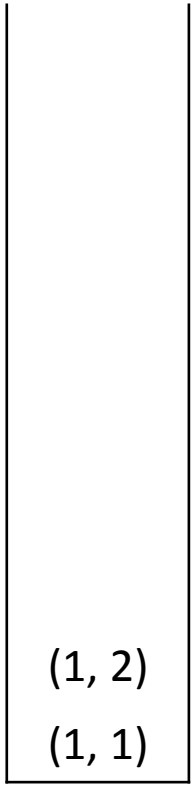
*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (1,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

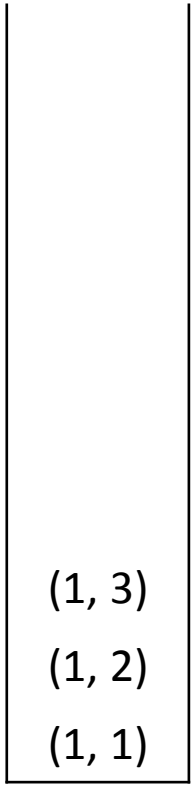
*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*						*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (1,3)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

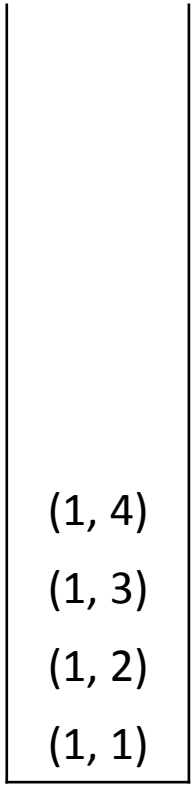
*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V					*
*		*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (1,4)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

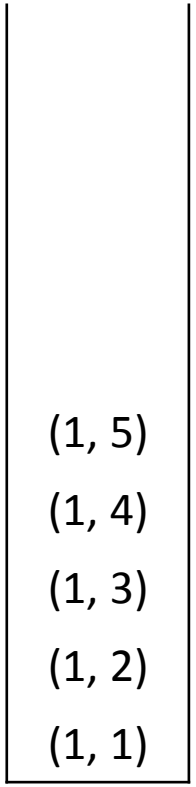
*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V					*
*	V	*		*		*
*				*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 4)
  - (1,5)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V					*
*	V	*		*		*
*	V			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 5)
  - (1,6)? no ; (2,5)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V					*
*	V	*		*		*
*	V	V		*	E	*
*	*	*	*	*	*	*

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

Depth First Search

- skipping over a few iterations in the loop

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 4)
  - (5,5)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V		V	V	V	*
*	V	*	V	*	V	*
*	V	V	V	*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (5, 4)
  - (5,5)? YES!!!!
- Push on the Stack and Mark as Visited
- Assume Search Order: Down, Right, Left, Down

*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V		V	V	V	*
*	V	*	V	*	V	*
*	V	V	V	*	E	*
*	*	*	*	*	*	*

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

Depth First Search Algorithm

- Is Position on Top of Stack (5, 5) the End Position
  - YES!!!!
- Break Out of Loop
- Stack contain path from END to START

*	*	*	*	*	*	*
*	V					*
*	V	*	*	*	*	*
*	V		V	V	V	*
*	V	*	V	*	V	*
*	V	V	V	*	E	*
*	*	*	*	*	*	*

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

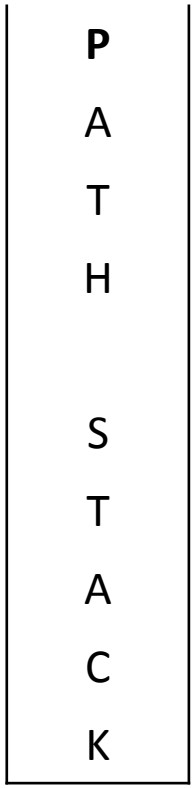
CS 211 - Project 3

Maze Solving Example 3

5x5 Maze

- Adding 2 rows and columns to put a border around the maze.
- Green cell is Start position
- Blue cell is End position
- Yellow cells are from input file

*	*	*	*	*	*	*
*	S		*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*

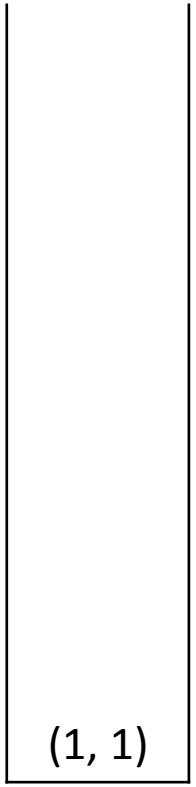


Depth First Search Algorithm

- Push Start Position on Stack
- Mark Start Position as Visited

- Search Order: Right, Down, Left, Up

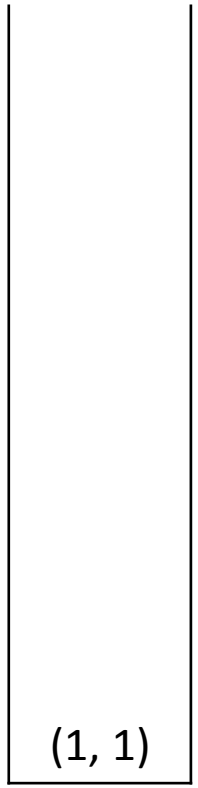
*	*	*	*	*	*	*
*	V		*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2, 1)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

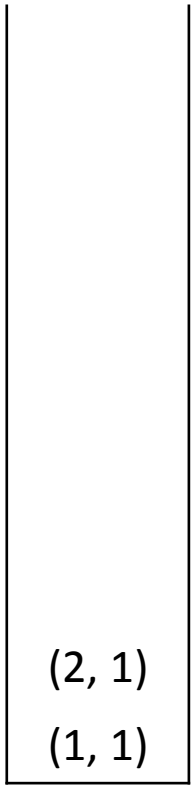
*	*	*	*	*	*	*
*	V		*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2, 1)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

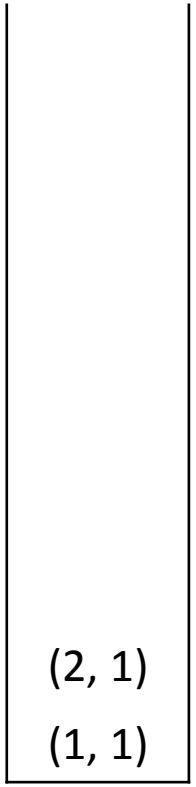
*	*	*	*	*	*	*
*	V	V	*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 1)
  - (3, 1)? no ; (2, 2)? no ; (1, 1)? no ; (2, 0)? no
- Pop the stack
- Search Order: Right, Down, Left, Up

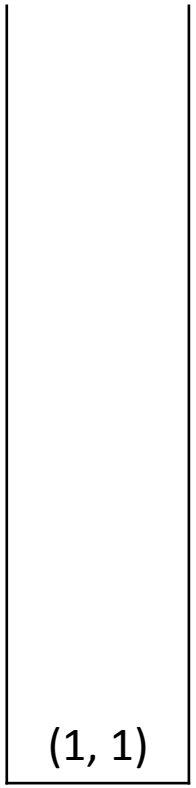
*	*	*	*	*	*	*
*	V	V	*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 1)
  - (3, 1)? no ; (2, 2)? no ; (1, 1)? no ; (2, 0)? no
- Pop the stack
- Search Order: Right, Down, Left, Up

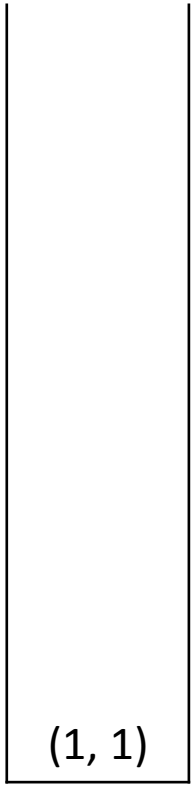
*	*	*	*	*	*	*
*	V	V	*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2, 1)? no ; (1, 2)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

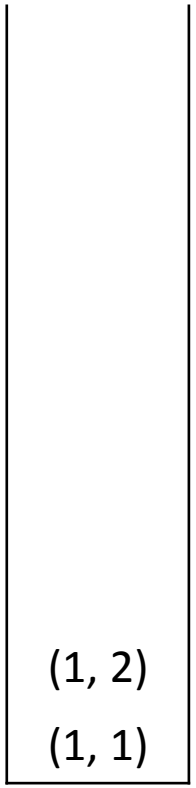
*	*	*	*	*	*	*
*	V	V	*			*
*		*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2, 1)? no ; (1, 2)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

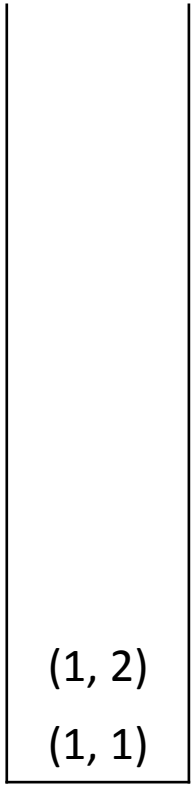
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (2, 2)? no ; (1, 3)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

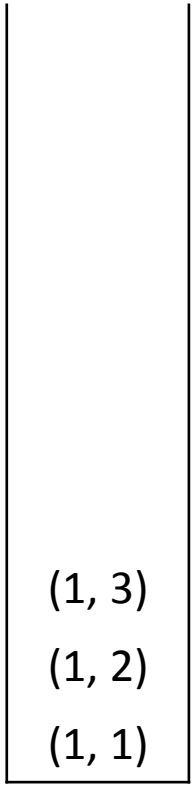
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*			*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (2, 2)? no ; (1, 3)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

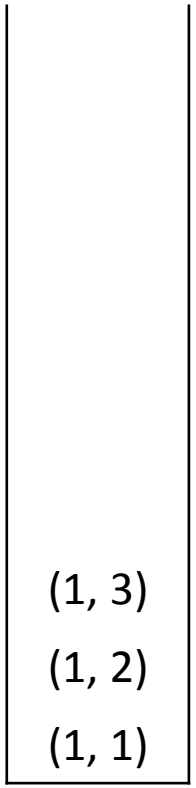
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V		*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2, 3)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

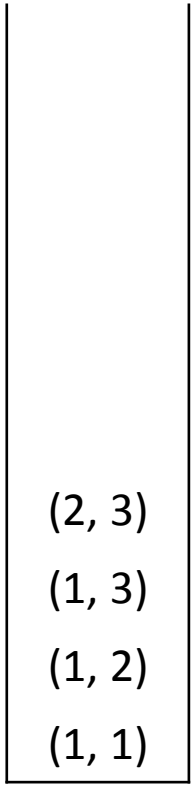
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V		*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2, 3)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

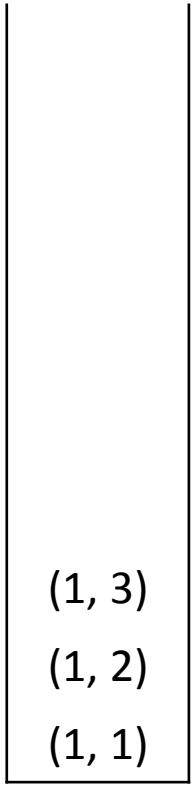
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (2, 3)
  - (3, 3)? no ; (2, 4)? no ; (1, 3)? no ; (2, 2)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

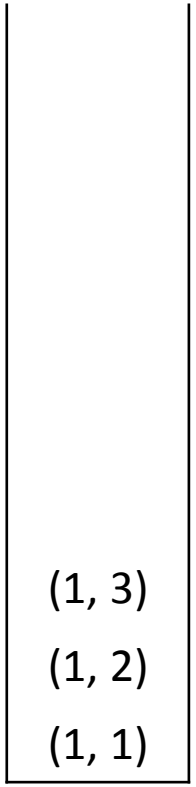
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2, 3)? no ; (1, 4)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

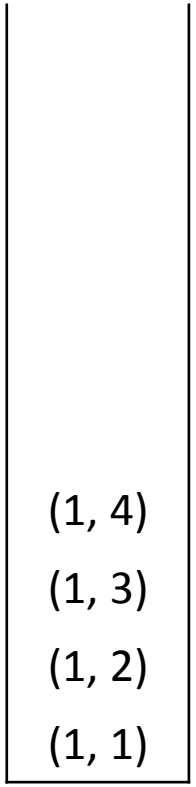
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*		*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2, 3)? no ; (1, 4)? yes
- Push on Stack and Mark as Visited
- Search Order: Right, Down, Left, Up

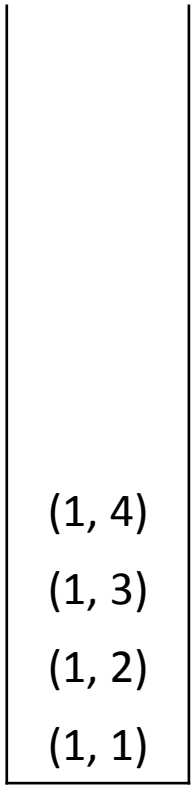
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 4)
  - (2, 4)? no ; (1, 5)? no ; (0, 4)? no ; (1, 3)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

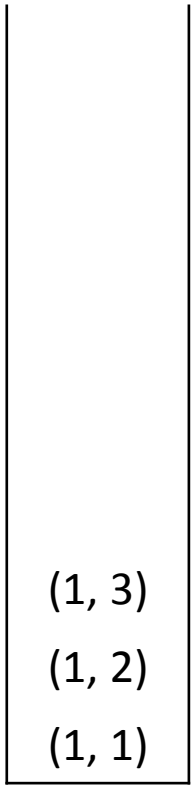
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 4)
  - (2, 4)? no ; (1, 5)? no ; (0, 4)? no ; (1, 3)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

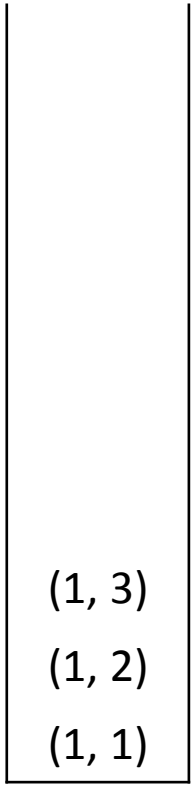
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2, 3)? no ; (1, 4)? no ; (0, 3)? no ; (1, 2)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

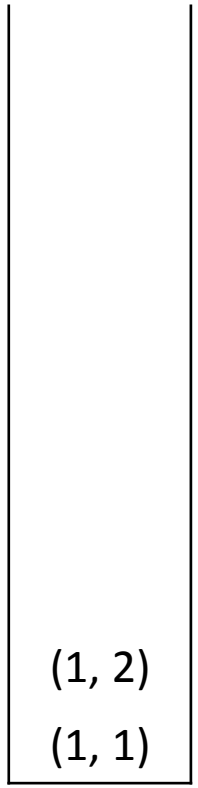
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 3)
  - (2, 3)? no ; (1, 4)? no ; (0, 3)? no ; (1, 2)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

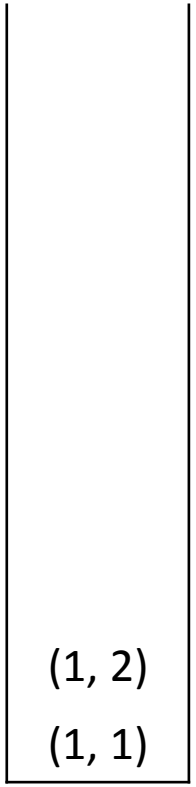
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (2, 2)? no ; (1, 3)? no ; (0, 2)? no ; (1, 1)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

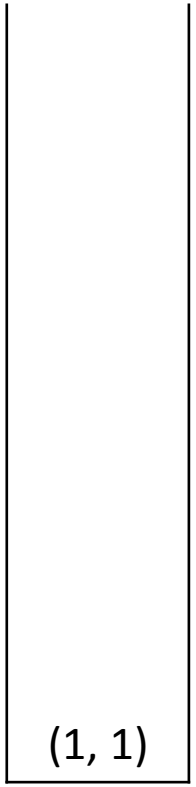
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 2)
  - (2, 2)? no ; (1, 3)? no ; (0, 2)? no ; (1, 1)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

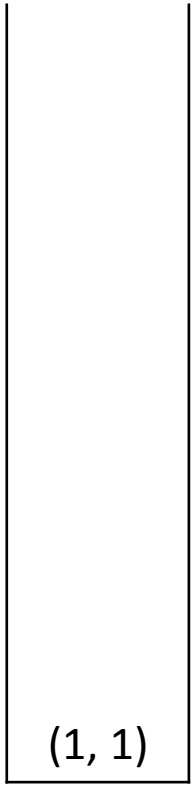
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2, 1)? no ; (1, 2)? no ; (0, 1)? no ; (1, 0)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

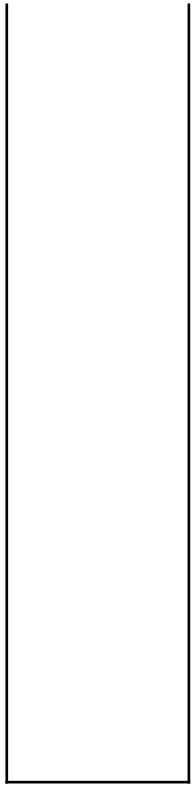
*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Find Unblocked and Unvisited Neighbor to Position on Top of Stack (1, 1)
  - (2, 1)? no ; (1, 2)? no ; (0, 1)? no ; (1, 0)? no
- Pop the Stack
- Search Order: Right, Down, Left, Up

*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*



Depth First Search Algorithm

- Stack is now Empty
- Break out of loop
- No Possible Solution
- Search Order: Right, Down, Left, Up

*	*	*	*	*	*	*
*	V	V	*			*
*	V	*	*	*	*	*
*	V	V	*			*
*	V	*		*		*
*	*			*	E	*
*	*	*	*	*	*	*