# Maze Traversal

Joseph Chuang-Chieh Lin (林莊傑)

Department of Computer Science & Engineering,
National Taiwan Ocean University

Fall 2025

# Outline



# Outline

# Maze

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

- 0: path; 1: barriers.

# Maze ▷ a path from the upper-left corner to the lower-right corner?

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

- 0: path; 1: barriers.

# Moves

| | | |
|---|---|---|
| NW ↖ [i-1][j-1] | N ↑ [i-1][j] | NE ↗ [i-1][j+1] |
| W ← [i][j-1] | X [i][j] | E → [i][j+1] |
| SW ↙ [i+1][j-1] | S ↓ [i-1][j] | SE ↘ [i+1][j+1] |

```
typedef struct {
    short int vert; // vertical direction
    short int horiz; // horizontal direction
} offsets;
offsets move[8];
```

# Moves (2/2)

| name | dir | move[dir].vert | move[dir].horiz |
|---|---|---|---|
| N ↑ | 0 | −1 | 0 |
| NE ↗ | 1 | −1 | 1 |
| E → | 2 | 0 | 1 |
| SE ↘ | 3 | 1 | 1 |
| S ↓ | 4 | 1 | 0 |
| SW ↙ | 5 | 1 | −1 |
| W ← | 6 | 0 | −1 |
| NW ↖ | 7 | −1 | −1 |

- `next_row = row + move[dir].vert;`
- `next_col = col + move[dir].horiz;`

# Outline



# Initial Attempt for Maze Traversal

- A two-dimensional array `mark` for recording the maze positions that are already checked.
- Use a stack to save current path and direction.
- Return and try another path if we take a hopeless path.
- The stack size: $m \times n$ (# of positions).

```
#define MAX_STACK_SIZE 100

typedef struct {
    short int row; // can be of other types...
    short int col;
    short int dir;
} element;

element stack a[MAX_STACK_SIZE];
```

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | | | | | | | |
| col | | | | | | | |
| dir | | | | | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | **0** | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | | | | | | |
| col | 1 | | | | | | |
| dir | 3 | | | | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | **0** | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | | | | | |
| col | 1 | 2 | | | | | |
| dir | 3 | 2 | | | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | **0** | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | | | | |
| col | 1 | 2 | 3 | | | | |
| dir | 3 | 2 | 2 | | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | **0** | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | 2 | | | |
| col | 1 | 2 | 3 | 4 | | | |
| dir | 3 | 2 | 2 | 1 | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | **0** | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | | | | |
| col | 1 | 2 | 3 | | | | |
| dir | 3 | 2 | 2 | | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | **0** | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | 2 | | | |
| col | 1 | 2 | 3 | 4 | | | |
| dir | 3 | 2 | 2 | 5 | | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | **0** | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | 2 | 3 | | |
| col | 1 | 2 | 3 | 4 | 3 | | |
| dir | 3 | 2 | 2 | 5 | 3 | | |

# An algorithm using a stack

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| name | dir | .vert | .horiz |
|---|---|---|---|
| N | 0 | $-1$ | 0 |
| NE | 1 | $-1$ | 1 |
| E | 2 | 0 | 1 |
| SE | 3 | 1 | 1 |
| S | 4 | 1 | 0 |
| SW | 5 | 1 | $-1$ |
| W | 6 | 0 | $-1$ |
| NW | 7 | $-1$ | $-1$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | 2 | 3 | 4 |
| col | 1 | 2 | 3 | 4 | 3 | 4 |
| dir | 3 | 2 | 2 | 5 | 3 | 3 |

# path()

```
void path() { /* output a path through the maze if such a path exists */
    int i, row, col, next_row, next_col, dir, found = false;
    element position;
    mark[1][1] = 1; top = 0; stack[0].row = 1; stack[0].col = 1; stack[0].dir = 0;
    while (top > -1 && !found) {
        position = pop(&top);
        row = position.row; col = position.col; dir  = position.dir;
        while (dir < 8 && !found) { /* move in direction dir  */
            next_row = row + move[dir].vert;
            next_col = col + move[dir].horiz;
            if (next_row == EXIT_ROW && next_col == EXIT_COL) /* see if the step goes to the exit */
                found = true;
            else if (maze[next_row][next_col] == 0 && mark[next_row][next_col]==0) {
                mark[next_row][next_col] = 1; /* mark the position if it has not been visited yet */
                position.row = row; position.col = col; position.dir = ++dir; // update the current position
                push(&top, position); // proceed to the next stage!
                row = next_row; col = next_col; dir = 0;
            } else ++dir; // neither the exit nor unvisited direction, try next direction!
        }
    }
    if (found) {
        printf("The path is :\n"); printf("row col\n");
        for (i = 0; i <= top; i++)
            printf("%2d%5d", stack[i].row, stack[i].col);
        printf("%2d%5d\n", row, col);
        printf("%2d%5d\n", EXIT_ROW, EXIT_COL);
    } else printf("The maze does not have a path\n");
}
```

# A Recursive Example

- An example of maze traversal by recursion.


# A Recursive Example

- An example of maze traversal by recursion.
  - Recall: System Stack.

# A Recursive Example

- An example of maze traversal by recursion.
  - Recall: System Stack.
- **Exercise:** Try to modify it to consider 8 directions.

# Example: Sudoku Generation & Solver

- Instance Generation: →link.
- Solver: Exercise!

# Discussions