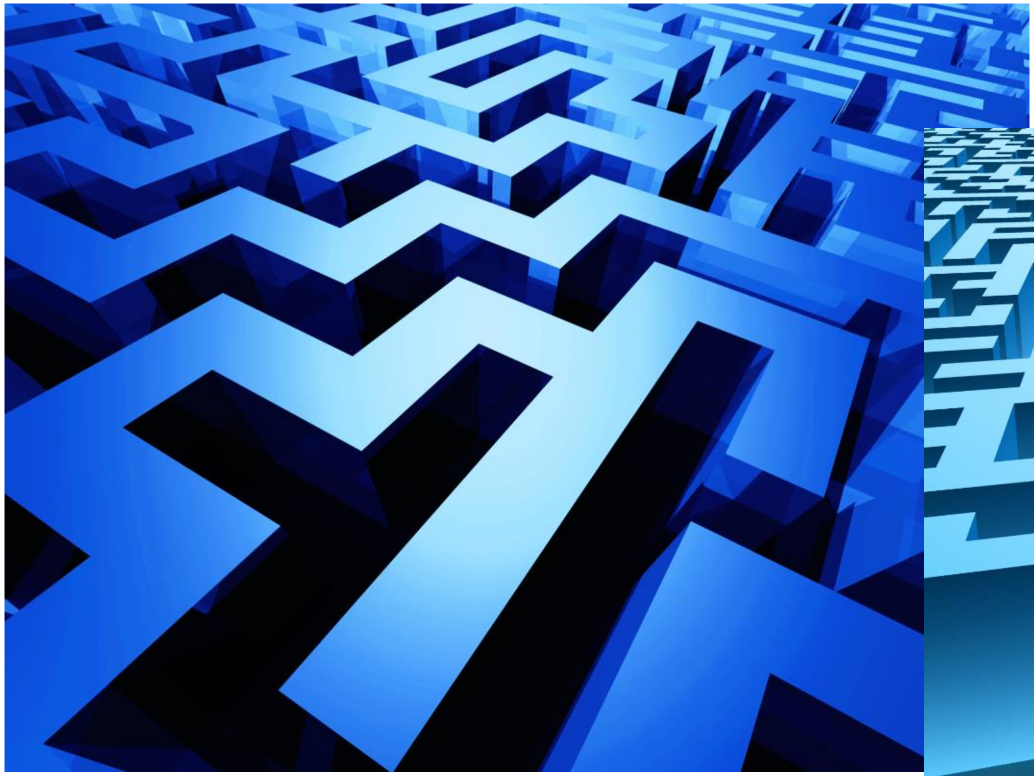
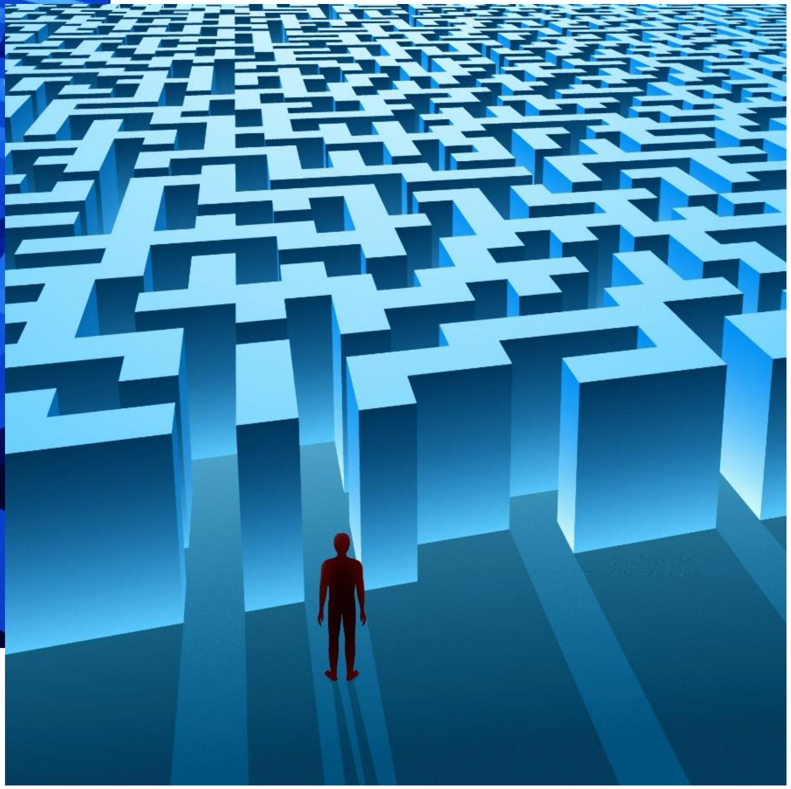
Maze Problem

- Using stack -

Maze Problem [1]

- What is Maze?
  - A rectangular area with an entrance and an exit
  - The interior of maze contains obstacles

Maze Problem [2]

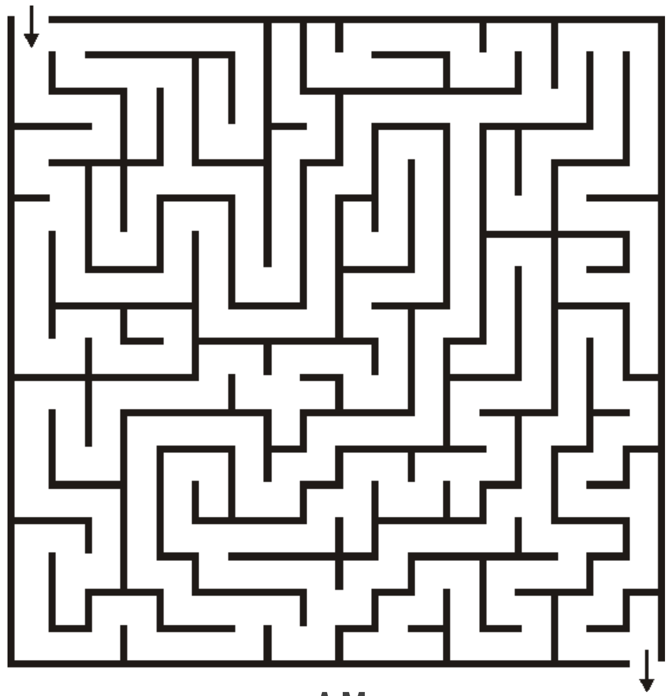
- Suppose that maze is to be modeled as an n x m matrix
  - Position (1, 1) : entrance
  - Position (n, m) : exit
  - Each maze position : row and column intersection
  - Position (i, j) = 1 iff there is an obstacle
  - Position (i, j) = 0 otherwise

Entrance

E	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	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	X

- Matrix representation of maze -

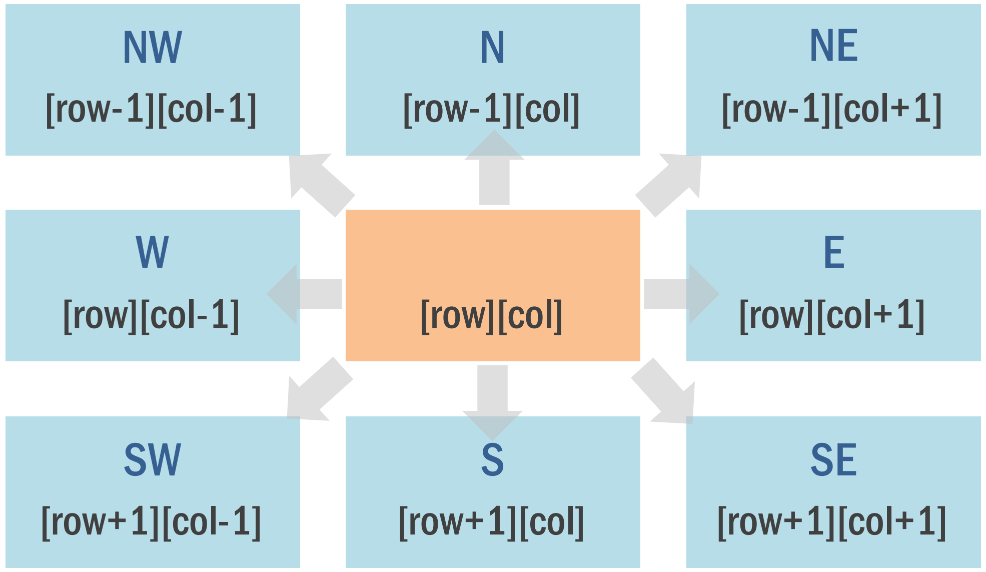
Exit



- A Maze -

Maze Problem [3]

- A Possible Representation
  - Allowable moves



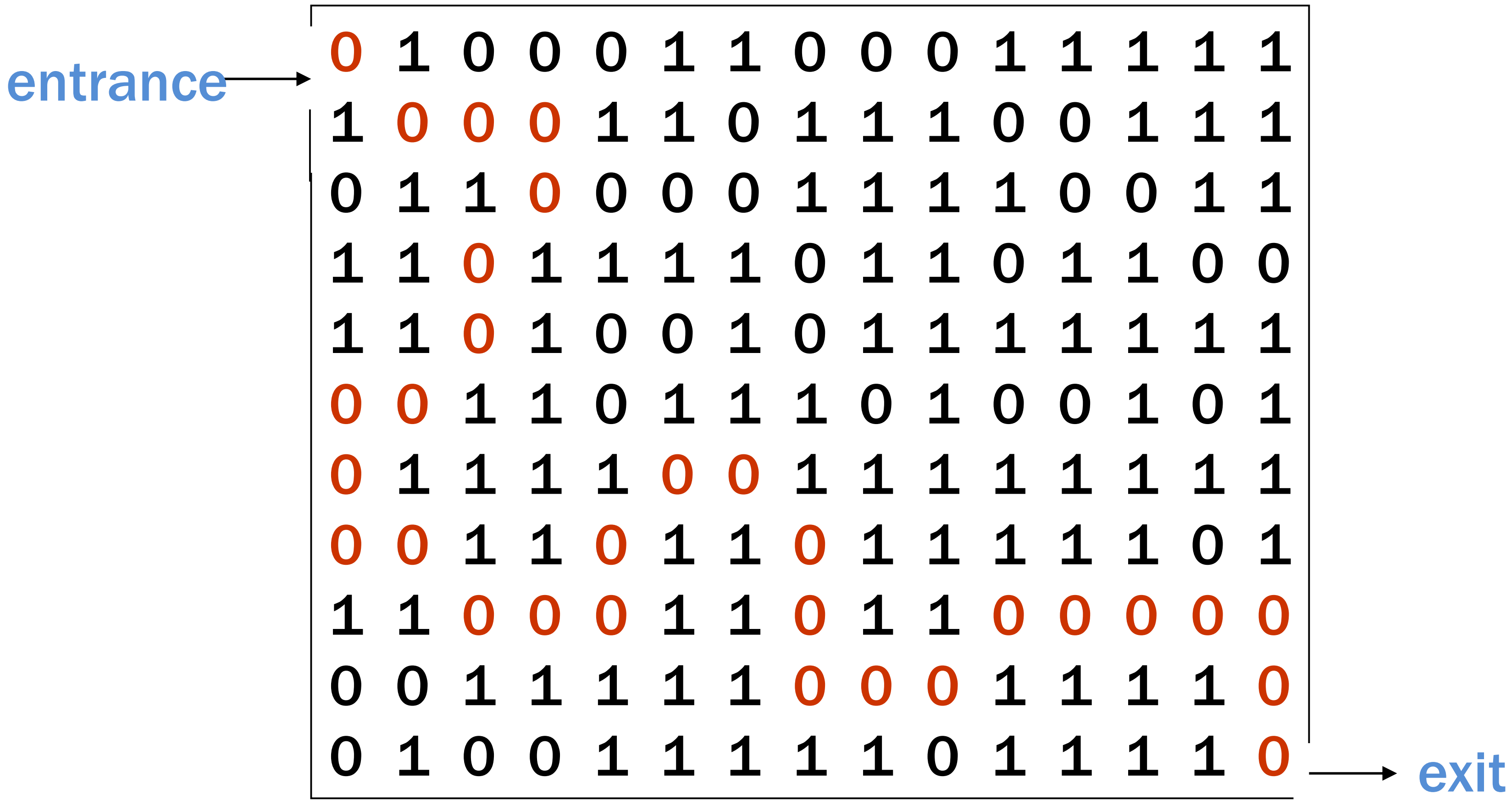
- A Possible Representation -

Dir	Index	Row	Col
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

- A Possible Implementation -

Maze Problem [4]

- Backtracking
  - Using Stack



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	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**

Quiz (1-1)

- Class Maze (1)

```
public class Maze {
    private int rows;
    private int cols;
    private int visit;
    private Cell[][] grid;
    private Stack<Cell> stack;
    private Cell ent;
    public  int[][] trace;

    // Move direction:
    private final int vert[];
    private final int horz[];
    private class Cell {
        int row;
        int col;
        char value;

        private Cell(int row, int col, char v) { // Implementation …    }

        @Override
        public String toString() {  // Implementation … }

        public Cell explore() { // Implementation … }
    }
```

Quiz (1-2)

- Class Maze (2)

```
    public static Maze create(final Character[][] data) {
        return new Maze(data);
    }

    ///////////////////////////////////////////////////////////////////////////
    // Procedures
    ///////////////////////////////////////////////////////////////////////////

    public Cell getEntrance() {
        // Implementation …
    }

    public void solve() {
        // Implementation …
    }

    private Maze(final Character[][] data) {
        // Implementation for Ctor
    }
} // End of Class Maze
```

Quiz (1-2)

- Interface IQueue (2)

```
        /**
         * Insert an element at the rear of the queue.
         * @param element to be inserted.
         */
        public void enqueue(E element);

        /**
         * Remove the front element from the queue.
         * @return element removed.
         * @exception EmptyItemException if the queue is empty.
         */
        public E dequeue() throws EmptyItemException;

}
```

Quiz (1-7)

- **Output**

```
Maze Problem: 11 x 15
E 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 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 0 X

Entrance = E[ 0, 0]
Exit     = E[10,14]

 0  1  2  3  4  1  1  9 10 11  1  1  1  1  1
 1  1  0  5  1  1  8  1  1  1 12 13  1  1  1
 0  1  1 23  6  7 19  1  1  1  1 17 14  1  1
 1  1 24  1  1  1  1 20  1  1 18  1  1 15 16
 1  1 25  1  0  0  1 21  1  1  1  1  1  1  1
28 26  1  1  0  1  1  1 22  1  0  0  1  0  1
27  1  1  1  1 33 34  1  1  1  1  1  1  1  1
 0 29  1  1 32  1  1 35  1  1  1  1  1 42  1
 1  1 30 31  0  1  1 36  1  1 39 40 41  0 43
 0  0  1  1  1  1  1  0 37 38  1  1  1  1 44
 0  1  0  0  1  1  1  1  1  0  1  1  1  0 45
```