**Challenges** | Matrix (2D Array) Traversal

|  |  |  |  |
| --- | --- | --- | --- |
| 🍌 | 🍎 | 😀 | 🐉 |
| 👺 | 🍺 | 🍩 | 🚴‍♂️ |
| 🚘 | 🦑 | 🚆 | 🏝️ |
| 🌆 | 🛹 | 🕺 | 🍕 |

For each of the following challenges create a function/method that takes an m × n matrix (2D array) and returns an array containing a record of the given traversal. An approach might be to create separate module files (Python), or class files (Java) for each of the grouped challenges, and access these from a central file/method. Your solution[s] should be dynamic, in that they must be capable of taking any size matrix and use simple constructs that are common to all programming languages.

## Row-wise traversal (row by row)

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a number game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a number game  Description automatically generated |
| Top-left to bottom-right | Top-right to bottom-left | Bottom-right to top-left | Bottom-left to top-right |

## Column-wise traversal (column by column)

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a number game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a number game  Description automatically generated |
| Top-left to bottom-right | Top-right to bottom-left | Bottom-right to top-left | Bottom-left to top-right |

## Diagonal traversal (corner to corner)

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated |
| Top-left to bottom-right | Top-right to bottom-left | Bottom-right to top-left | Bottom-left to top-right |

## Horizontal snake traversal (row by row)

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a number game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated |
| Top-left to bottom-left | Top-right to bottom-right | Bottom-right to top-right | Bottom-left to top-left |

## Vertical snake traversal (column by column)

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a game  Description automatically generated |
| Top-left to top-right | Top-left to top-right | Bottom-right to bottom-left | Bottom-left to top-left |

## Boundary traversal

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated |
| Top | Right | Bottom | Left |
| A screenshot of a game  Description automatically generated | A screenshot of a number game  Description automatically generated |  |  |
| Clockwise | Anticlockwise |  |  |

## Spiral traversal (column by column)

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated |
| Outside in clockwise | Outside in anticlockwise | Inside out clockwise | Inside out anticlockwise |

## Zigzag traversal

|  |  |  |  |
| --- | --- | --- | --- |
| A screenshot of a number game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a number game  Description automatically generated |
| Top-left to bottom-right, column first | Top-right to bottom-left, column first | Bottom-right to top-left, column first | Bottom-left to top-right, column first |
| A screenshot of a number game  Description automatically generated | A screenshot of a number game  Description automatically generated | A screenshot of a game  Description automatically generated | A screenshot of a game  Description automatically generated |
| Top-left to bottom-right,  row first | Top-right to bottom-left,  row first | Bottom-right to top-left,  row first | Bottom-left to top-right,  row first |

## Additional challenges (courtesy of ChatGPT)

* **Islands**: Given a matrix that represents the distribution of land and water in a given area, count the number of islands (connected land cells).
* **Adjacent submatrix**: Locate given element[s] in the matrix, and its adjacent elements.
* **Connected submatrix**: Locate given element[s] in the matrix, and elements deemed connected, for example odd or even.
* **Knight's Tour**: Given a chessboard matrix, show the path a given knight would take to visit every square on the board exactly once.
* **Knight's shortest path**: Extend the Knight's Tour problem by finding the shortest path from a starting position to an ending position on the chessboard.
* **Path**: Given matrix that represents a maze, find a path from the top-left to the bottom-right corner.
* **Unique paths**: Given matrix that represents a maze, find the number of unique paths from the top-left to the bottom-right corner, moving only right or down.
* **Magic square**: Determine if a given square matrix is a magic square, meaning the sum of each row, column, and diagonal is the same.
* **Multiplication**: Multiply two matrices.
* **Longest increasing path**: Given a matrix of numbers, find the longest path where a path is a sequence of cells with increasing values.
* **Sparse matrix compression**: Compress a sparse matrix (containing a sizeable number of zeros), converting it to a more memory-efficient representation.
* **Binary connectivity**: Given a matrix that represents a binary image, find the number of connected components of 1s.
* **Rotation**: Rotate a given matrix by 90 degrees clockwise or counterclockwise.