Documentation: Spiral Matrix

Overview

The Spiral Matrix problem entails traversing a given matrix in a spiral order, starting from the top-left corner and moving clockwise to visit all elements of the matrix.

Problem Statement

Given an (m times n) matrix of integers, the task is to return all elements of the matrix in spiral order.

Example

Input:

[[1,2,3],

[4,5,6],

[7,8,9]]

Output:

[1,2,3,6,9,8,7,4,5]

Constraints

- (m) equals the length of the matrix.
- (n) equals the length of each row in the matrix.
- (1 leq m, n leq 10)
- (-100 leq text{{matrix}}[i][j] leq 100)

Solution Approach

The problem can be solved by traversing the matrix layer by layer. We define four variables to represent the boundaries of the matrix: `row_begin`, `row_end`, `col_begin`, and `col_end`. We traverse the matrix in four directions: right, down, left, and up, and update the boundary variables accordingly until all elements are visited.

Implementation

The solution is implemented using a Python class named `Solution`, which contains a method `spiralOrder`. This method takes a 2D list `matrix` as input and returns a list containing the elements of the matrix in spiral order.

Example Usage

```
sol = Solution()

matrix1 = [[1,2,3],[4,5,6],[7,8,9]]

print(sol.spiralOrder(matrix1))  # Output: [1,2,3,6,9,8,7,4,5]


matrix2 = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

print(sol.spiralOrder(matrix2))  # Output: [1,2,3,4,8,12,11,10,9,5,6,7]
```

Complexity Analysis

- Time Complexity:(O(m times n)) - We traverse each element of the matrix once.
- Space Complexity:(O(1)) - The space used is independent of the input size; only a constant amount of additional space is required.