Spiral Matrix II Documentation

Overview

The "Spiral Matrix II" problem is a computational challenge aimed at generating an (n times n) matrix filled with elements from 1 to (n^2) in a spiral order.

Problem Statement

Given a positive integer (n), the task is to generate an (n times n) matrix filled with elements in a spiral order. The spiral starts from the top-left corner and moves clockwise, filling the matrix with sequential integers from 1 to (n^2).

Example

Input:

n = 3

Output:

```
 [[1, 2, 3],

 [8, 9, 4],

 [7, 6, 5]]
```

Input:

n = 1

Output:

```
[[1]]
```

Constraints

- 1 <= n <= 20

Approach

To solve this problem, a class named `Solution` is implemented, which contains a method `generateMatrix(self, n: int) -> List[List[int]]`. This method generates the desired matrix using the following steps:

1. Initialize an (n times n) matrix with all elements set to 0.
2. Define variables to represent the boundaries of the matrix (top, bottom, left, and right) and initialize them accordingly.
3. Start filling the matrix in a spiral order, incrementing the numbers from 1 to (n^2) until the matrix is completely filled.
    - Fill the top row from left to right.
    - Fill the rightmost column from top to bottom.
    - Fill the bottom row from right to left.
    - Fill the leftmost column from bottom to top.
4. Repeat steps 3 until all elements in the matrix are filled.

Complexity Analysis

- **Time Complexity:** The time complexity of the solution is (O(n^2)) since each cell of the (n times n) matrix is filled exactly once.
- **Space Complexity:** The space complexity is (O(n^2)) as we are creating an (n times n) matrix.

Example Usage

```
sol = Solution()

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

print(sol.generateMatrix(1))  # Output: [[1]]
```