Documentation

The NumMatrix class is a solution to efficiently calculate the sum of elements within a rectangular subregion of a given 2D matrix. It is designed to handle multiple queries, where each query specifies a subregion defined by its top-left and bottom-right corners. By leveraging precomputed data, the implementation ensures fast response times for queries, making it suitable for scenarios with numerous queries on large matrices.

The underlying concept of the NumMatrix class is based on **prefix sums**. The prefix sum of a cell (i, j) is defined as the cumulative sum of all elements in the matrix from the top-left corner (0, 0) to the cell (i, j). A separate prefix_sum matrix is constructed during initialization, storing these cumulative values. This allows the program to efficiently calculate the sum of any subregion by referencing precomputed data rather than iterating through the matrix repeatedly.

The class structure consists of two main components: the initialization method (__init__) and the query method (sumRegion). The initialization method takes the input matrix and computes the prefix_sum matrix. Each cell in the prefix_sum matrix is calculated by summing the corresponding matrix element and the cumulative sums of its preceding row and column while subtracting the overlapping area to avoid double counting. This ensures that the pre-computation step is efficient and accurate.

The sumRegion method provides the functionality to calculate the sum of any subregion specified by its corners (row1, col1) and (row2, col2). Using the precomputed prefix_sum matrix, the method applies an inclusion-exclusion formula to compute the desired sum in constant time. By adding or subtracting values from specific cells in the prefix_sum matrix, the method quickly isolates the sum of the target subregion without recalculating values.

This design is highly efficient in terms of time complexity. The initialization process runs in O(m×n)O(m times n), where mm is the number of rows and nn is the number of columns in the input matrix. This is a one-time cost incurred when creating the NumMatrix object. Subsequent queries are handled in O(1)O(1) time, as the sum is derived from precomputed values using a fixed number of arithmetic operations. This efficiency makes the solution ideal for handling a high volume of queries.

The space complexity is also noteworthy. While the prefix_sum matrix requires O(m×n)O(m times n) additional space, this is a reasonable tradeoff for the significant reduction in query processing time. The class is especially useful when the number of queries is large relative to the size of the matrix, as the precomputation overhead is amortized over multiple operations.

In summary, the NumMatrix class combines precomputation with efficient querying to deliver an optimal solution for range sum queries in a 2D matrix. Its prefix sum approach ensures quick responses, making it highly effective for applications where performance and scalability are critical. The balance between initialization cost, query efficiency, and space usage highlights the practical design of this solution.