# **Q1. Sub-matrix Sum Queries**

Given a matrix of integers **A** of size **N x M** and multiple queries **Q**, for each query, find and return the submatrix sum.

Inputs to queries are **top left (b, c)** and **bottom right (d, e)** indexes of submatrix whose sum is to find out.

**NOTE:**

* Rows are numbered from top to bottom, and columns are numbered from left to right.
* The sum may be large, so return the answer **mod 109 + 7.**
* Also, select the data type carefully, if you want to store the addition of some elements.
* Indexing given in B, C, D, and E arrays is **1-based**.
* Top Left 0-based index = (B[i] - 1, C[i] - 1)
* Bottom Right 0-based index = (D[i] - 1, E[i] - 1)

**Example Input**

Input 1:

A = [ [1, 2, 3]

[4, 5, 6]

[7, 8, 9] ]

B = [1, 2]

C = [1, 2]

D = [2, 3]

E = [2, 3]

Input 2:

A = [ [5, 17, 100, 11]

[0, 0, 2, 8] ]

B = [1, 1]

C = [1, 4]

D = [2, 2]

E = [2, 4]

**Example Output**

Output 1:

[12, 28]

Output 2:

[22, 19]

**Example Explanation**

Explanation 1:

For query 1: Submatrix contains elements: 1, 2, 4 and 5. So, their sum is 12.

For query 2: Submatrix contains elements: 5, 6, 8 and 9. So, their sum is 28.

Explanation 2:

For query 1: Submatrix contains elements: 5, 17, 0 and 0. So, their sum is 22.

For query 2: Submatrix contains elements: 11 and 8. So, their sum is 19.

# **Q2. Sum of all Submatrices**

Given a 2D Matrix **A** of dimensions **N\*N**, we need to return the sum of all possible submatrices.

**Example Input**

Input 1:

A = [ [1, 1]

[1, 1] ]

Input 2:

A = [ [1, 2]

[3, 4] ]

**Example Output**

Output 1:

16

Output 2:

40

**Example Explanation**

Example 1:

Number of submatrices with 1 elements = 4, so sum of all such submatrices = 4 \* 1 = 4

Number of submatrices with 2 elements = 4, so sum of all such submatrices = 4 \* 2 = 8

Number of submatrices with 3 elements = 0

Number of submatrices with 4 elements = 1, so sum of such submatrix = 4

Total Sum = 4+8+4 = 16

Example 2:

The submatrices are [1], [2], [3], [4], [1, 2], [3, 4], [1, 3], [2, 4] and [[1, 2], [3, 4]].

Total sum = 40

# **Q3. Maximum Submatrix**

Given a matrix **A** of size **NxM**, which is row-wise and column-wise sorted. Find a submatrix such that sum of its elements is maximum and return this sum.

**Example Input**

Input 1:

A = [[6, 8, 10, 11],

[10, 11, 12, 15]]

Input 2:

A = [[-8, 1, 1],

[-1, 6, 6],

[7, 10, 10]]

**Example Output**

Output 1:

83

Output 2:

38

**Example Explanation**

Explanation 1:

Since all values in the matrix are positive integers, their sum is taken which is 83.

Explanation 2:

Max sum is found in submatrix from [1, 0] to [2, 2].

# Q4. Search in a row wise and column wise sorted matrix

Given a matrix of integers **A** of size **N x M** and an integer **B**.

In the given matrix every row and column is sorted in non-decreasing order. Find and return the position of **B** in the matrix in the given form:

* If A[i][j] = B then return (i \* 1009 + j)
* If B is not present return -1.

**Note 1:** Rows are numbered from top to bottom and columns are numbered from left to right.  
**Note 2:** If there are multiple B in A then return the smallest value of i\*1009 +j such that A[i][j]=B.  
**Note 3:** Expected time complexity is linear  
**Note 4:** Use 1-based indexing

**Example Input**

Input 1:-

A = [[1, 2, 3]

[4, 5, 6]

[7, 8, 9]]

B = 2

Input 2:-

A = [[1, 2]

[3, 3]]

B = 3

**Example Output**

Output 1:-

1011

Output 2:-

2019

**Example Explanation**

Expanation 1:-

A[1][2] = 2

1 \* 1009 + 2 = 1011

Explanation 2:-

A[2][1] = 3

2 \* 1009 + 1 = 2019

A[2][2] = 3

2 \* 1009 + 2 = 2020

The minimum value is 2019

# **Q5. Row with maximum number of ones**

Given a binary sorted matrix **A** of size **N x N**. Find the row with the **maximum** number of **1**.

**NOTE:**

* If two rows have the maximum number of 1 then return the row which has a **lower index**.
* Rows are numbered from top to bottom and columns are numbered from left to right.
* Assume **0-based** indexing.
* Assume each row to be sorted by values.
* Expected time complexity is O(rows + columns).

**Example Input**

Input 1:

A = [ [0, 1, 1]

[0, 0, 1]

[0, 1, 1] ]

Input 2:

A = [ [0, 0, 0, 0]

[0, 0, 0, 1]

[0, 0, 1, 1]

[0, 1, 1, 1] ]

**Example Output**

Output 1:

0

Output 2:

3

**Example Explanation**

Explanation 1:

Row 0 has maximum number of 1s.

Explanation 2:

Row 3 has maximum number of 1s.