Problem 308: Range Sum Query 2D - Mutable

Problem Information

Difficulty: Medium
Acceptance Rate: 45.20%
Paid Only: Yes
Tags: Array, Design, Binary Indexed Tree, Segment Tree, Matrix

Problem Description

Given a 2D matrix `matrix`, handle multiple queries of the following types:

1. **Update** the value of a cell in `matrix`. 2. Calculate the **sum** of the elements of `matrix` inside the rectangle defined by its **upper left corner** `(row1, col1)` and **lower right corner** `(row2, col2)`.

Implement the NumMatrix class:

* `NumMatrix(int[][] matrix)` Initializes the object with the integer matrix `matrix`. * `void update(int row, int col, int val)` **Updates** the value of `matrix[row][col]` to be `val`. * `int sumRegion(int row1, int col1, int row2, int col2)` Returns the **sum** of the elements of `matrix` inside the rectangle defined by its **upper left corner** `(row1, col1)` and **lower right corner** `(row2, col2)`.

Example 1:

Input ["NumMatrix", "sumRegion", "update", "sumRegion"] [[[[3, 0, 1, 4, 2], [5, 6, 3, 2, 1], [1, 2, 0, 1, 5], [4, 1, 0, 1, 7], [1, 0, 3, 0, 5]]], [2, 1, 4, 3], [3, 2, 2], [2, 1, 4, 3]] **Output** [null, 8, null, 10] **Explanation** NumMatrix numMatrix = new NumMatrix([[3, 0, 1, 4, 2], [5, 6, 3, 2, 1], [1, 2, 0, 1, 5], [4, 1, 0, 1, 7], [1, 0, 3, 0, 5]]); numMatrix.sumRegion(2, 1, 4, 3); // return 8 (i.e. sum of the left red rectangle) numMatrix.update(3, 2, 2); // matrix changes from left image to right image numMatrix.sumRegion(2, 1, 4, 3); // return 10 (i.e. sum of the right red rectangle)

Constraints:

* `m == matrix.length` * `n == matrix[i].length` * `1 <= m, n <= 200` * `-1000 <= matrix[i][j] <= 1000` * `0 <= row < m` * `0 <= col < n` * `-1000 <= val <= 1000` * `0 <= row1 <= row2 < m` * `0 <= col1 <= col2 < n` * At most `5000` calls will be made to `sumRegion` and `update`.

Code Snippets

C++:

```
class NumMatrix {
public:
NumMatrix(vector<vector<int>>& matrix) {

}

void update(int row, int col, int val) {

}

int sumRegion(int row1, int col1, int row2, int col2) {

}
};

/**
* Your NumMatrix object will be instantiated and called as such:
* NumMatrix* obj = new NumMatrix(matrix);
* obj->update(row,col,val);
* int param_2 = obj->sumRegion(row1,col1,row2,col2);
*/
```

Java:

```
class NumMatrix {

public NumMatrix(int[][] matrix) {

}

public void update(int row, int col, int val) {

}
```

```
public int sumRegion(int row1, int col1, int row2, int col2) {

}
}

/**
* Your NumMatrix object will be instantiated and called as such:
* NumMatrix obj = new NumMatrix(matrix);
* obj.update(row,col,val);
* int param_2 = obj.sumRegion(row1,col1,row2,col2);
*/
```

Python3:

```
class NumMatrix:

def __init__(self, matrix: List[List[int]]):


def update(self, row: int, col: int, val: int) -> None:


def sumRegion(self, row1: int, col1: int, row2: int, col2: int) -> int:



# Your NumMatrix object will be instantiated and called as such:
# obj = NumMatrix(matrix)
# obj.update(row,col,val)
# param_2 = obj.sumRegion(row1,col1,row2,col2)
```