Problem 1476: Subrectangle Queries

Problem Information

Difficulty: Medium
Acceptance Rate: 86.14%
Paid Only: No
Tags: Array, Design, Matrix

Problem Description

Implement the class `SubrectangleQueries` which receives a `rows x cols` rectangle as a matrix of integers in the constructor and supports two methods:

1.` updateSubrectangle(int row1, int col1, int row2, int col2, int newValue)`

* Updates all values with `newValue` in the subrectangle whose upper left coordinate is `(row1,col1)` and bottom right coordinate is `(row2,col2)`.

2.` getValue(int row, int col)`

* Returns the current value of the coordinate `(row,col)` from the rectangle.

Example 1:

Input ["SubrectangleQueries","getValue","updateSubrectangle","getValue","getValue","updateSubrectangle","getValue","getValue"] [[[[1,2,1],[4,3,4],[3,2,1],[1,1,1]]],[0,2],[0,0,3,2,5],[0,2],[3,1],[3,0,3,2,10],[3,1],[0,2]] **Output** [null,1,null,5,5,null,10,5] **Explanation** SubrectangleQueries subrectangleQueries = new SubrectangleQueries([[1,2,1],[4,3,4],[3,2,1],[1,1,1]]); // The initial rectangle (4x3) looks like: // 1 2 1 // 4 3 4 // 3 2 1 // 1 1 1 subrectangleQueries.getValue(0, 2); // return 1 subrectangleQueries.updateSubrectangle(0, 0, 3, 2, 5); // After this update the rectangle looks like: // 5 5 5 // 5 5 5 // 5 5 5 // 5 5 5 subrectangleQueries.getValue(0, 2); // return 5 subrectangleQueries.getValue(3, 1); // return 5 subrectangleQueries.updateSubrectangle(3, 0, 3, 2, 10); // After this update the rectangle looks like: // 5 5 5 // 5 5 5 // 5 5 5 // 10 10 10 subrectangleQueries.getValue(3, 1); // return 10 subrectangleQueries.getValue(0, 2); // return 5

Example 2:

Input ["SubrectangleQueries","getValue","updateSubrectangle","getValue","getValue","updateSubrectangle","getValue"] [[[[1,1,1],[2,2,2],[3,3,3]]],[0,0],[0,0,2,2,100],[0,0],[2,2],[1,1,2,2,20],[2,2]] **Output** [null,1,null,100,100,null,20] **Explanation** SubrectangleQueries subrectangleQueries = new SubrectangleQueries([[1,1,1],[2,2,2],[3,3,3]]); subrectangleQueries.getValue(0, 0); // return 1 subrectangleQueries.updateSubrectangle(0, 0, 2, 2, 100); subrectangleQueries.getValue(0, 0); // return 100 subrectangleQueries.getValue(2, 2); // return 100 subrectangleQueries.updateSubrectangle(1, 1, 2, 2, 20); subrectangleQueries.getValue(2, 2); // return 20

Constraints:

* There will be at most `500` operations considering both methods: `updateSubrectangle` and `getValue`. * `1 <= rows, cols <= 100` * `rows == rectangle.length` * `cols == rectangle[i].length` * `0 <= row1 <= row2 < rows` * `0 <= col1 <= col2 < cols` * `1 <= newValue, rectangle[i][j] <= 10^9` * `0 <= row < rows` * `0 <= col < cols`

Code Snippets

C++:

```
class SubrectangleQueries {
public:
SubrectangleQueries(vector<vector<int>>& rectangle) {

}

void updateSubrectangle(int row1, int col1, int row2, int col2, int newValue)
{

}

int getValue(int row, int col) {

}
};

/**
```

```
* Your SubrectangleQueries object will be instantiated and called as such:
* SubrectangleQueries* obj = new SubrectangleQueries(rectangle);
* obj->updateSubrectangle(row1,col1,row2,col2,newValue);
* int param_2 = obj->getValue(row,col);
*/
```

Java:

```
class SubrectangleQueries {

public SubrectangleQueries(int[][] rectangle) {

}

public void updateSubrectangle(int row1, int col1, int row2, int col2, int
newValue) {

}

public int getValue(int row, int col) {

}
}

/**
* Your SubrectangleQueries object will be instantiated and called as such:
* SubrectangleQueries obj = new SubrectangleQueries(rectangle);
* obj.updateSubrectangle(row1,col1,row2,col2,newValue);
* int param_2 = obj.getValue(row,col);
*/
```

Python3:

```
class SubrectangleQueries:

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


def updateSubrectangle(self, row1: int, col1: int, row2: int, col2: int,
newValue: int) -> None:
```

```
def getValue(self, row: int, col: int) -> int:



# Your SubrectangleQueries object will be instantiated and called as such:
# obj = SubrectangleQueries(rectangle)
# obj.updateSubrectangle(row1,col1,row2,col2,newValue)
# param_2 = obj.getValue(row,col)
```