2536. Increment Submatrices by One

Solved

Medium | Topics | Companies | Hint

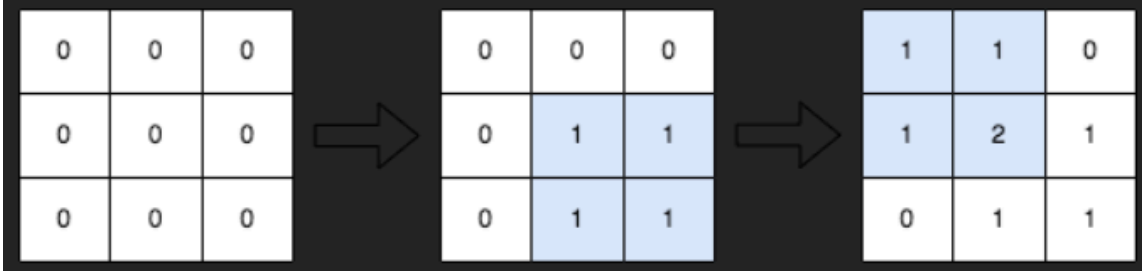
You are given a positive integer `n`, indicating that we initially have an `n x n` **0-indexed** integer matrix `mat` filled with zeroes.

You are also given a 2D integer array `query`. For each `query[i] = [row1i, col1i, row2i, col2i]`, you should do the following operation:

- Add `1` to **every element** in the submatrix with the **top left** corner `(row1i, col1i)` and the **bottom right** corner `(row2i, col2i)`. That is, add `1` to `mat[x][y]` for all `row1i <= x <= row2i` and `col1i <= y <= col2i`.

Return *the matrix* `mat` *after performing every query.*

Example 1:




```
Input: n = 3, queries = [[1,1,2,2],[0,0,1,1]]
Output: [[1,1,0],[1,2,1],[0,1,1]]
Explanation: The diagram above shows the initial matrix, the
matrix after the first query, and the matrix after the second
query.
- In the first query, we add 1 to every element in the submatrix
with the top left corner (1, 1) and bottom right corner (2, 2).
- In the second query, we add 1 to every element in the
submatrix with the top left corner (0, 0) and bottom right
corner (1, 1).
```

Example 2:


```
Input: n = 2, queries = [[0,0,1,1]]
Output: [[1,1],[1,1]]
Explanation: The diagram above shows the initial matrix and the
matrix after the first query.
- In the first query we add 1 to every element in the matrix.
```

Constraints:

- $1 <= n <= 500$
- $1 <= queries.length <= 10^4$
- $0 <= row1_i <= row2_i < n$
- $0 <= col1_i <= col2_i < n$

Python:

```
class Solution:
    def rangeAddQueries(self, n: int, queries: list[list[int]]) -> list[list[int]]:
        diff = [[0] * (n + 1) for _ in range(n + 1)]

        for r1, c1, r2, c2 in queries:
            diff[r1][c1] += 1
            diff[r2 + 1][c1] -= 1
            diff[r1][c2 + 1] -= 1
            diff[r2 + 1][c2 + 1] += 1

        mat = [[0] * n for _ in range(n)]
        for i in range(n):
            for j in range(n):
                above = mat[i - 1][j] if i > 0 else 0
                left = mat[i][j - 1] if j > 0 else 0
                diag = mat[i - 1][j - 1] if i > 0 and j > 0 else 0
                mat[i][j] = diff[i][j] + above + left - diag
```

```
        return mat
```

JavaScript:

```
const rangeAddQueries = (n, queries) => {
    let diff = Array.from({ length: n + 1 }, () => Array(n + 1).fill(0));

    for (let [r1, c1, r2, c2] of queries) {
        diff[r1][c1]++;
        diff[r2 + 1][c1]--;
        diff[r1][c2 + 1]--;
        diff[r2 + 1][c2 + 1]++;
    }

    let mat = Array.from({ length: n }, () => Array(n).fill(0));
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            const above = mat[i - 1]?.[j] ?? 0;
            const left = mat[i]?.[j - 1] ?? 0;
            const diag = mat[i - 1]?.[j - 1] ?? 0;
            mat[i][j] = diff[i][j] + above + left - diag;
        }
    }

    return mat;
};
```

Java:

```
class Solution {
    public int[][] rangeAddQueries(int n, int[][] queries) {
        int[][] diff = new int[n + 1][n + 1];

        for (int[] q : queries) {
            int r1 = q[0], c1 = q[1], r2 = q[2], c2 = q[3];
            diff[r1][c1]++;
            diff[r2 + 1][c1]--;
            diff[r1][c2 + 1]--;
            diff[r2 + 1][c2 + 1]++;
        }

        int[][] mat = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int above = i > 0 ? mat[i - 1][j] : 0;
                int left = j > 0 ? mat[i][j - 1] : 0;
```

```
            int diag = i > 0 && j > 0 ? mat[i - 1][j - 1] : 0;
            mat[i][j] = diff[i][j] + above + left - diag;
        }
      }

      return mat;
    }
}
```