3391. Design a 3D Binary Matrix with Efficient Layer Tracking

Solved ●

Premium

Medium Topics Hint

You are given a `n x n x n` **binary** 3D array `matrix`.

Implement the `Matrix3D` class:

- `Matrix3D(int n)` Initializes the object with the 3D binary array `matrix`, where **all** elements are initially set to 0.
- `void setCell(int x, int y, int z)` Sets the value at `matrix[x][y][z]` to 1.
- `void unsetCell(int x, int y, int z)` Sets the value at `matrix[x][y][z]` to 0.
- `int largestMatrix()` Returns the index `x` where `matrix[x]` contains the most number of 1's. If there are multiple such indices, return the **largest** `x`.

Example 1:

> **Input:**
> ["Matrix3D", "setCell", "largestMatrix", "setCell", "largestMatrix", "setCell", "largestMatrix"]
> [[3], [0, 0, 0], [], [1, 1, 2], [], [0, 0, 1], []]
>
> **Output:**
> [null, null, 0, null, 1, null, 0]
>
> **Explanation**
>
> Matrix3D matrix3D = new Matrix3D(3); // Initializes a `3 x 3 x 3` 3D array `matrix`, filled with all 0's.
> matrix3D.setCell(0, 0, 0); // Sets `matrix[0][0][0]` to 1.
> matrix3D.largestMatrix(); // Returns 0. `matrix[0]` has the most number of 1's.
> matrix3D.setCell(1, 1, 2); // Sets `matrix[1][1][2]` to 1.
> matrix3D.largestMatrix(); // Returns 1. `matrix[0]` and `matrix[1]` tie with the most number of 1's, but index 1 is bigger.
> matrix3D.setCell(0, 0, 1); // Sets `matrix[0][0][1]` to 1.
> matrix3D.largestMatrix(); // Returns 0. `matrix[0]` has the most number of 1's.

Example 2:

> **Input:**
> ["Matrix3D", "setCell", "largestMatrix", "unsetCell", "largestMatrix"]
> [[4], [2, 1, 1], [], [2, 1, 1], []]
>
> **Output:**
> [null, null, 2, null, 3]
>
> **Explanation**
>
> Matrix3D matrix3D = new Matrix3D(4); // Initializes a `4 x 4 x 4` 3D array `matrix`, filled with all 0's.
> matrix3D.setCell(2, 1, 1); // Sets `matrix[2][1][1]` to 1.
> matrix3D.largestMatrix(); // Returns 2. `matrix[2]` has the most number of 1's.
> matrix3D.unsetCell(2, 1, 1); // Sets `matrix[2][1][1]` to 0.
> matrix3D.largestMatrix(); // Returns 3. All indices from 0 to 3 tie with the same number of 1's, but index 3 is the biggest.

Constraints:

- `1 <= n <= 100`
- `0 <= x, y, z < n`
- At most 10^5 calls are made in total to `setCell` and `unsetCell`.
- At most 10^4 calls are made to `largestMatrix`.

Seen this question in a real interview before? 1/5

Yes No

Accepted 773 /1.1K | **Acceptance Rate** 68.7 %

Topics

Hint 1

Hint 2

Discussion (3)