1260. Shift 2D Grid

Easy 964 240 Add to List Share

Given a 2D `grid` of size `m x n` and an integer `k`. You need to shift the `grid` `k` times.

In one shift operation:

- Element at `grid[i][j]` moves to `grid[i][j + 1]`.
- Element at `grid[i][n - 1]` moves to `grid[i + 1][0]`.
- Element at `grid[m - 1][n - 1]` moves to `grid[0][0]`.

Return the *2D grid* after applying shift operation `k` times.

Example 1:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \to \begin{bmatrix} 9 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

```
Input: grid = [[1,2,3],[4,5,6],[7,8,9]], k = 1
Output: [[9,1,2],[3,4,5],[6,7,8]]
```

Example 2:

$$\begin{bmatrix} 3 & 8 & 1 & 9 \\ 19 & 7 & 2 & 5 \\ 4 & 6 & 11 & 10 \\ 12 & 0 & 21 & 13 \end{bmatrix} \to \begin{bmatrix} 13 & 3 & 8 & 1 \\ 9 & 19 & 7 & 2 \\ 5 & 4 & 6 & 11 \\ 10 & 12 & 0 & 21 \end{bmatrix} \to \begin{bmatrix} 21 & 13 & 3 & 8 \\ 1 & 9 & 19 & 7 \\ 2 & 5 & 4 & 6 \\ 11 & 10 & 12 & 0 \end{bmatrix}$$

$$\to \begin{bmatrix} 0 & 21 & 13 & 3 \\ 8 & 1 & 9 & 19 \\ 7 & 2 & 5 & 4 \\ 6 & 11 & 10 & 12 \end{bmatrix} \to \begin{bmatrix} 12 & 0 & 21 & 13 \\ 3 & 8 & 1 & 9 \\ 19 & 7 & 2 & 5 \\ 4 & 6 & 11 & 10 \end{bmatrix}$$

```
Input: grid = [[3,8,1,9],[19,7,2,5],[4,6,11,10],
[12,0,21,13]], k = 4
Output: [[12,0,21,13],[3,8,1,9],[19,7,2,5],[4,6,11,10]]
```

Example 3:

```
Input: grid = [[1,2,3],[4,5,6],[7,8,9]], k = 9
Output: [[1,2,3],[4,5,6],[7,8,9]]
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m <= 50`
- `1 <= n <= 50`
- `-1000 <= grid[i][j] <= 1000`
- `0 <= k <= 100`

Accepted 58,290 | Submissions 87,507

Seen this question in a real interview before? Yes No

Companies

```java
class Solution {
    public List<List<Integer>> shiftGrid(int[][] grid, int k) {
        int m = grid.length;
        int n = grid[0].length;
        int total = m * n;
        // if shifting total times, it shifts back to orignal state
        k = k % (total);
        List<List<Integer>> result = new ArrayList<>();
        for(int i = 0; i < m; i++) {
            List<Integer> list = new ArrayList<>();
            result.add(list);
            for (int j = 0; j < n; j++) {
                // i * n + j original place index in 1D array
                // i * n + j - k  is to get value k steps before
                int index = ((i * n + j) - k + total) % total;
                list.add(grid[index / n][index % n]);
            }
        }
        return result;
    }
}
```

Your previous code was restored from your local storage. Reset to default