2435. Paths in Matrix Whose Sum Is Divisible by K

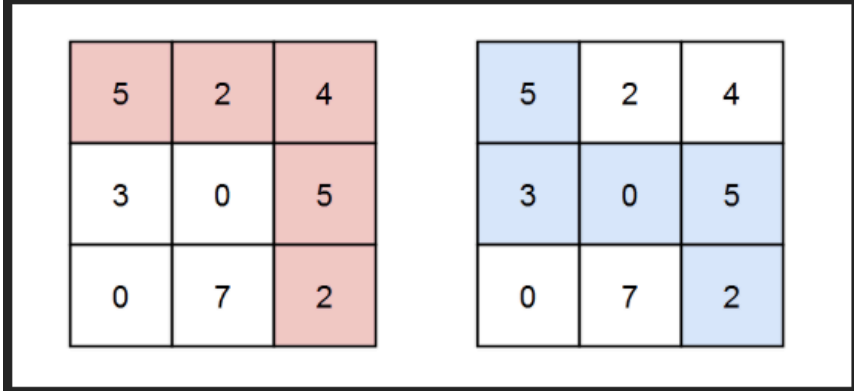
Solved

Hard | Topics | Companies | Hint

You are given a **0-indexed** `m x n` integer matrix `grid` and an integer `k`. You are currently at position `(0, 0)` and you want to reach position `(m - 1, n - 1)` moving only **down** or **right**.

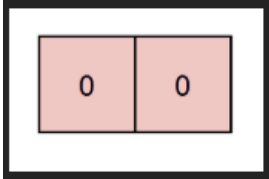
Return *the number of paths where the sum of the elements on the path is divisible by* `k`. Since the answer may be very large, return it **modulo** $10^9 + 7$.

Example 1:



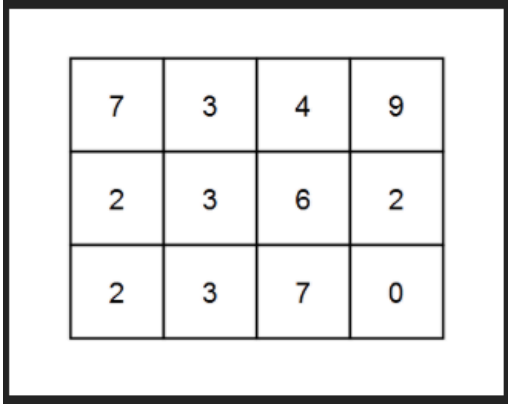

```
Input: grid = [[5,2,4],[3,0,5],[0,7,2]], k = 3
Output: 2
Explanation: There are two paths where the sum of the elements on the
path is divisible by k.
The first path highlighted in red has a sum of 5 + 2 + 4 + 5 + 2 = 18
which is divisible by 3.
The second path highlighted in blue has a sum of 5 + 3 + 0 + 5 + 2 = 15
which is divisible by 3.
```

Example 2:




```
Input: grid = [[0,0]], k = 5
Output: 1
Explanation: The path highlighted in red has a sum of 0 + 0 = 0 which
is divisible by 5.
```

Example 3:




```
Input: grid = [[7,3,4,9],[2,3,6,2],[2,3,7,0]], k = 1
Output: 10
Explanation: Every integer is divisible by 1 so the sum of the elements
on every possible path is divisible by k.
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 5 * 10^4`
- `1 <= m * n <= 5 * 10^4`
- `0 <= grid[i][j] <= 100`
- `1 <= k <= 50`

Python:

```python
class Solution:
    def numberOfPaths(self, grid: List[List[int]], k: int) -> int:
        MOD = 10**9 + 7
        m, n = len(grid), len(grid[0])

        prev = [[0]*k for _ in range(n)]
        curr = [[0]*k for _ in range(n)]

        s = 0
        for j in range(n):
            s = (s + grid[0][j]) % k
            prev[j][s] = 1

        s = grid[0][0] % k

        for i in range(1, m):
            s = (s + grid[i][0]) % k
            curr[0] = [0]*k
            curr[0][s] = 1

            for j in range(1, n):
                curr[j] = [0]*k
                val = grid[i][j]
                for r in range(k):
                    nr = (r + val) % k
                    curr[j][nr] = (prev[j][r] + curr[j - 1][r]) % MOD

            prev, curr = curr, prev

        return prev[n - 1][0]
```

JavaScript:

```javascript
/**
 * @param {number[][]} grid
 * @param {number} k
 * @return {number}
 */
var numberOfPaths = function(grid, k) {
    const MOD = 1e9 + 7;
    const m = grid.length, n = grid[0].length;

    let prev = Array.from({ length: n }, () => Array(k).fill(0));
    let curr = Array.from({ length: n }, () => Array(k).fill(0));
```

```
    let sum = 0;
    for (let j = 0; j < n; j++) {
        sum = (sum + grid[0][j]) % k;
        prev[j][sum] = 1;
    }

    sum = grid[0][0] % k;

    for (let i = 1; i < m; i++) {
        sum = (sum + grid[i][0]) % k;
        curr[0].fill(0);
        curr[0][sum] = 1;

        for (let j = 1; j < n; j++) {
            curr[j].fill(0);
            const val = grid[i][j];
            for (let r = 0; r < k; r++) {
                const nr = (r + val) % k;
                curr[j][nr] = (prev[j][r] + curr[j - 1][r]) % MOD;
            }
        }

        const temp = prev;
        prev = curr;
        curr = temp;
    }

    return prev[n - 1][0];
};
```

Java:

```
class Solution {
    private static final int MOD = (int) 1e9 + 7;

    public int numberOfPaths(int[][] grid, int k) {
        int m = grid.length, n = grid[0].length;
        int[][] prev = new int[n][k];
        int[][] curr = new int[n][k];

        int sum = 0;
        for (int j = 0; j < n; j++) {
            sum = (sum + grid[0][j]) % k;
            prev[j][sum] = 1;
```

```
        }

        sum = grid[0][0] % k;

        for (int i = 1; i < m; i++) {
            sum = (sum + grid[i][0]) % k;
            Arrays.fill(curr[0], 0);
            curr[0][sum] = 1;

            for (int j = 1; j < n; j++) {
                Arrays.fill(curr[j], 0);
                int val = grid[i][j];

                for (int r = 0; r < k; r++) {
                    int nr = (r + val) % k;
                    curr[j][nr] = (prev[j][r] + curr[j - 1][r]) % MOD;
                }
            }

            int[][] tmp = prev;
            prev = curr;
            curr = tmp;
        }

        return prev[n - 1][0];
    }
}
```