Daily Coding Problem #158

Problem

This problem was asked by Slack.

You are given an N by M matrix of `0`s and `1`s. Starting from the top left corner, how many ways are there to reach the bottom right corner?

You can only move right and down. `0` represents an empty space while `1` represents a wall you cannot walk through.

For example, given the following matrix:

```
[[0, 0, 1],
 [0, 0, 1],
 [1, 0, 0]]
```

Return two, as there are only two ways to get to the bottom right:

- Right, down, down, right
- Down, right, down, right

The top left corner and bottom right corner will always be `0`.

Solution

This problem is similar to Daily Coding Problem #62, but with a twist: there are now obstacles in the original matrix that we can't go through. The basic idea to solve this is the following:

- Notice again that we can only get to any cell from above or from the left.
- We'll store the number of ways to get to a cell `[i][j]`, and compute that from looking at `[i - 1][j]` and `[i][j - 1]`.

- First though, we must check that those spots are not walls, since we can't move from there. If it is, then the number of ways coming from that spot is 0.

We also set the first row and column to be 1 (since we can move straight down or right), but we have to stop when we hit the first wall.

```
EMPTY = 0
WALL = 1

def num_ways(matrix):
    m, n = len(matrix), len(matrix[0])
    num_ways_matrix = [[0 for j in range(n)] for i in range(m)]

    # Fill first row
    for j in range(n):
        if matrix[0][j] == WALL:
            break
        num_ways_matrix[0][j] = 1

    # Fill first col
    for i in range(m):
        if matrix[i][0] == WALL:
            break
        num_ways_matrix[i][0] = 1

    for i in range(1, m):
        for j in range(1, n):
            from_top = num_ways_matrix[i - 1][j] if matrix[i - 1][j] != WALL else 0
            from_left = num_ways_matrix[i][j - 1] if matrix[i][j - 1] != WALL else 0

            num_ways_matrix[i][j] = from_top + from_left

    return num_ways_matrix[m - 1][n - 1]
```

This takes O(n * m) time and space.