Daily Coding Problem #62

Problem

This problem was asked by Facebook.

There is an N by M matrix of zeroes. Given N and M, write a function to count the number of ways of starting at the top-left corner and getting to the bottom-right corner. You can only move right or down.

For example, given a 2 by 2 matrix, you should return 2, since there are two ways to get to the bottom-right:

- Right, then down
- Down, then right

Given a 5 by 5 matrix, there are 70 ways to get to the bottom-right.

Solution

Notice that, to get to any cell, we only have two ways: either directly from above, or from the left, unless we can't go up or left anymore, in which case there's only one way. This leads to the following recurrence:

- If either N or M is 1, then return 1
- Otherwise, `f(n, m) = f(n - 1, m) + f(n, m - 1)`

This is very similar to the staircase problem from Daily Coding Problem #12.

The recursive solution would look like this:

```
def num_ways(n, m):
    if n == 1 or m == 1:
        return 1
    return num_ways(n - 1, m) + num_ways(n, m - 1)
```

However, just like in the staircase problem (or fibonacci), we will have a lot of repeated subcomputations. So, let's use bottom-up dynamic programming to store those results.

We'll initialize an N by M matrix A, and each entry `A[i][j]`, will contain the number of ways we can get to that entry from the top-left. Then, once we've filled up the matrix using our recurrence (by checking directly above or directly left), we can just look at the bottom-right value to get our answer.

```
def num_ways(n, m):
    A = [[0 for _ in range(m)] for _ in range(n)]
    for i in range(n):
        A[i][0] = 1
    for j in range(m):
        A[0][j] = 1
    for i in range(1, n):
        for j in range(1, m):
            A[i][j] = A[i - 1][j] + A[i][j - 1]
    return A[-1][-1]
```

This runs in O(N * M) time and space.