Daily Coding Problem #122

Problem

This question was asked by Zillow.

You are given a 2-d `matrix` where each cell represents number of coins in that cell. Assuming we start at `matrix[0][0]`, and can only move right or down, find the maximum number of coins you can collect by the bottom right corner.

For example, in this matrix

```
0 3 1 1
2 0 0 4
1 5 3 1
```

The most we can collect is 0 + 2 + 1 + 5 + 3 + 1 = 12 coins.

Solution

Notice the recursive structure of this problem.

- If we're at the bottom-right corner of the matrix, the most amount of coins we can collect is that cell.
- If we're at the bottom of the matrix, the most amount of coins we can collect is that cell and amount we can collect by movinrg right.
- If we're at the rightmost part the matrix, the most amount of coins we can collect is that cell and amount we can collect by moving down.
- Otherwise, the most amount of coins we can collect is that cell and the max of either the most amount we can collect from moving right or the most amount we can collect from moving down.

For example, in the following matrix,

```
0 2 3
5 4 9
6 7 1
```

- In the 1 cell, bottom-right cell, the most we can collect is that cell: 1.
- In the 7 cell, a bottom cell, we can only collect by taking coins from the right: 7 + 1.
- In the 9 cell, a right-most cell, we can only collect by taking coins from below 9 + 1.
- In the 4 cell, a non-edge cell, we can collect 4 plus the max of going right or going down: 4 + max(9 + 1, 7 + 1) == 14.

This leads itself to the following code:

```
def collect_coins(matrix, r=0, c=0, cache=None):
    if cache is None:
        cache = {}

    num_rows = len(matrix)
    num_cols = len(matrix[0])

    is_bottom = r == num_rows - 1
    is_rightmost = c == num_cols - 1

    if (r, c) not in cache:


        if is_bottom and is_rightmost:
            cache[r, c] = matrix[r][c]
        elif is_bottom:
            cache[r, c] = matrix[r][c] + collect_coins(matrix, r, c + 1, cache)
        elif is_rightmost:
            cache[r, c] = matrix[r][c] + collect_coins(matrix, r + 1, c, cache)
        else:
            cache[r, c] = matrix[r][c] + max(collect_coins(matrix, r + 1, c, cache),
                                             collect_coins(matrix, r, c + 1, cache))


    return cache[r, c]
```

This is O(MN) both space and time since we compute the solution for each cell at most once.

© Daily Coding Problem 2019

Privacy Policy