Daily Coding Problem #128

Problem

The Tower of Hanoi is a puzzle game with three rods and n disks, each a different size.

All the disks start off on the first rod in a stack. They are ordered by size, with the largest disk on the bottom and the smallest one at the top.

The goal of this puzzle is to move all the disks from the first rod to the last rod while following these rules:

- You can only move one disk at a time.
- A move consists of taking the uppermost disk from one of the stacks and placing it on top of another stack.
- You cannot place a larger disk on top of a smaller disk.

Write a function that prints out all the steps necessary to complete the Tower of Hanoi. You should assume that the rods are numbered, with the first rod being 1, the second (auxiliary) rod being 2, and the last (goal) rod being 3.

For example, with n = 3, we can do this in 7 moves:

```
Move 1 to 3
Move 1 to 2
Move 3 to 2
Move 1 to 3
Move 2 to 1
Move 2 to 3
Move 1 to 3
```

Solution

The goal of the Tower of Hanoi is to get all n disks from the source peg to the target peg, using a spare peg and abiding by all the constraints.

Let's try thinkinkg about this problem recursively. First consider the base cases:

- If there are `0` disks, then do nothing, since we are done.
- If there is `1` disk, then we can move the single disk from the source peg to the target peg directly.

Now, let's assume we have an existing `tower_of_hanoi` function that can move `n` disks from a source peg to a target peg using a spare stack. The recurrence would then look like this:

- If there is more than `1` disk, then we can do the following:
    - Recursively move `n - 1` disks from the source stack to the spare stack
    - Move the last (biggest) disk from the source stack to the target stack
    - Recursively move all `n - 1` disks from the spare stack to the target stack

We're able to recursively move the disks because it doesn't break any constraints: we can just treat the base disk as if it wasn't there.

In our code, we'll call our source stack a, spare stack b, and target stack c.

```
def tower_of_hanoi(n, a="1", b="2", c="3"):
    if n >= 1:
        tower_of_hanoi(n - 1, a, c, b)
        print("Move {} to {}".format(a, c))
        tower_of_hanoi(n - 1, b, a, c)
```

This will run in $O(2^n)$ time, since for each call we're recursively calling ourselves twice. This should also take O(n) space since the function call stack goes n calls deep.
