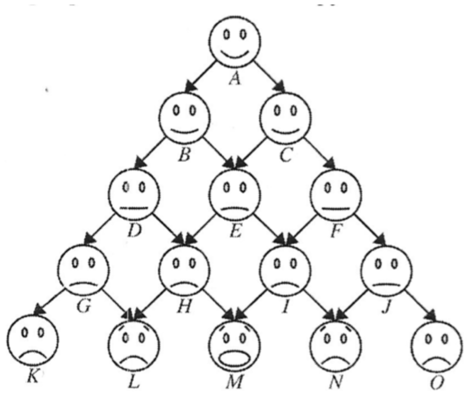
Project 2: Human Pyramids

CS 1410

Background



A human pyramid is a way of stacking people vertically in a triangle. With the exception of the people in the bottom row, each person splits their weight evenly on the two people below them in the pyramid. Their “weight”, however, includes not only their own body weight but the weight they are supporting above them. For example, in the pyramid above, person A splits her weight across people B and C, and person H splits his weight—plus the accumulated weight of the people he’s supporting—onto people L and M. In this question, you'll explore just how much weight that is.

For simplicity we will assume that everyone in the pyramid weighs exactly 200 pounds. Person A at the top of the pyramid has no weight on her back. People B and C are each carrying half of person A's weight. That means that each of them is shouldering 100 pounds.

Now, let's look at the people in the third row. Let’s begin by focusing on person E. How much weight is she supporting? Well, she’s directly supporting half the weight of person B (100 pounds) and half the weight of person E (100 pounds), so she’s supporting at least 200 pounds. On top of this, she’s feeling some of the weight that people B and C are carrying. Half of the weight that person B is shouldering (50 pounds) gets transmitted down onto person E and half the weight that person C is shouldering (50 pounds) similarly gets sent down to person E, so person E ends up feeling an extra 100 pounds. That means she’s supporting a net total of 300 pounds.

Not everyone in that third row is feeling the same amount, though. Look at person D for example. The only weight on person D comes from person B. Person D therefore ends up supporting

- half of person B’s body weight (100 pounds), plus
- half of the weight person Bis holding up (50 pounds),

so person D ends up supporting 150 pounds, only half of what E is feeling!

Going deeper in the pyramid, how much weight is person H feeling? Well, person H is supporting

> half of person D's body weight (100 pounds),
> half of person E's body weight (100 pounds), plus
> half of the weight person D is holding up (75 pounds), plus
> half of the weight person E is holding up (150) pounds.

The net effect is that person H is carrying 425 pounds—ouch! A similar calculation shows that person I is also carrying 425 pounds—can you see why? Compare this to person G. Person G is supporting

- half of person D's body weight (100 pounds), plus
- half of the weight person D is holding up (75 pounds)

or a net total of 175 pounds.

Finally, let's look at person M in the middle of the bottom row. How is she doing? Well, she's supporting

- half of person H's body weight (100 pounds),
- half of person I's body weight (100 pounds),
- half of the weight person H is holding up (212.5 pounds), and
- half of the weight person I is holding up (215.5 pounds),

for a net total of 625 pounds!

So, keep in mind that the first and last people in each row calculate their weight differently than people in interior positions, and the base case for the recursion is the person at the top of the pyramid.

Requirements

1. Write a recursive function (using no loops), `weightOn(r,c)`, which returns the weight on the back of the person in row `r` and and column `c`. Rows and columns are 0-based, so the top position is (0,0), for example, and person H is in position (3,1). The following also hold:

    ```
    weightOn(0,0) == 0.00
    weightOn(3,1) == 425.00
    ```

    Weights should be floating-point numbers.

2. When run as a main module, accept the number of rows to process via **sys.argv[1]**, and then print each row as a line at a time as you compute them, using 2 decimals:

    ```
    $ python3 pyramid.py 7

    0.00
    100.00 100.00
    150.00 300.00 150.00
    175.00 425.00 425.00 175.00
    187.50 500.00 625.00 500.00 187.50
    193.75 543.75 762.50 762.50 543.75 193.75
    196.88 568.75 853.12 962.50 853.12 568.75 196.88

    Elapsed time: 0.00017461799999995975 seconds
    ```

    Name your file *pyramid.py*. Use `time.perf_counter` to time your main function.

3. After finishing part 2, you will notice that it takes a long time for a large number of rows because many recursive calls are *repeated*. For example, on my machine, processing 23 rows took 7.47 seconds. To avoid calling `weightOn` for the same row and column more than once, save them in a dictionary named `cache` at the module level. The key for the dictionary is the tuple (row,column) and the value is the weight on the person in that (row,column) position. The first thing that `weightOn` should do is check to see if there is a previously computed entry for the key (r,c) in `cache`. If there is, simply return it. Otherwise, compute the weight recursively and save the result in cache before returning it. With caching it took only 0.0009 seconds to process 23 rows!

Your function `weightOn` and your cache (which must be named `cache`) will be tested on arbitrary values.