Daily Coding Problem #156

Problem

This problem was asked by Facebook.

Given a positive integer n, find the smallest number of squared integers which sum to n.

For example, given n = 13, return 2 since $13 = 3^2 + 2^2 = 9 + 4$.

Given n = 27, return 3 since $27 = 3^2 + 3^2 + 3^2 = 9 + 9 + 9$.

Solution

One naive recursive way of solving this problem would be to do the following:

- Iterate `i` from 1 to `sqrt(n)`
- Recursively compute the minimum number of squares needed to sum to `n - i*i`
- Pick the min of those, plus 1

The base case would be when `n = 0`.

```
from math import inf

def num_squares_naive(n):
    if n == 0:
        return 0

    min_num_squares = inf

    i = 1
    while n - i*i >= 0:
        min_num_squares = min(min_num_squares, num_squares_naive(n - i*i) + 1)
        i += 1
```

```
    return min_num_squares
```

However, this takes exponential time. We can speed things up using a cache with dynamic programming, and using the same logic:

```
def num_squares(n):
    if n == 0:
        return 0

    cache = [inf for i in range(n + 1)]
    cache[0] = 0
    for i in range(1, n + 1):
        j = 1
        while j * j <= i:
            cache[i] = min(cache[i], cache[i - j*j] + 1)
            j += 1

    return cache[n]
```

Now this is O(n^2) time and O(n) space.