Daily Coding Problem #210

Problem

This problem was asked by Apple.

A Collatz sequence in mathematics can be defined as follows. Starting with any positive integer:

- if n is even, the next number in the sequence is n / 2
- if n is odd, the next number in the sequence is 3n + 1

It is conjectured that every such sequence eventually reaches the number `1`. Test this conjecture.

Bonus: What input `n <= 1000000` gives the longest sequence?

Solution

This conjecture is fairly straightforward to implement iteratively. For a given input `n`, we either divide by 2 or multiply by 3 and add 1, only terminating when we reach 1.

As an example, the sequence starting with `5` would go `5 -> 16 -> 8 -> 4 -> 2 -> 1`.

```
def collatz(n):
    if n == 1:
        return True

    while n != 1:
        if n % 2 == 0:
            n = n / 2
        else:
            n = 3 * n + 1

    return True
```

For the bonus question, one way we could solve this is to write a function that finds the length of a given Collatz sequence, and then apply this function to all `n <= 1000000`.

To change it up, let's solve this one recursively. Our base case is when `n = 1`, in which case our path length is just 1. Otherwise, our result must be one more than the result of calling our function on either `n / 2` or `3 * n + 1`.

```
def get_collatz_length(n):
    if n == 1:
        return 1
    elif n % 2 == 0:
        return get_collatz_length(n / 2) + 1
    else:
        return get_collatz_length(3 * n + 1) + 1
```

However, note that if we already have calculated, say, `collatz_length(8)`, we are duplicating that work when we calculate `collatz_length(16)`. Instead, we can use memoization. Each time we call our function on a number, we'll first check our cache to see if we've already computed the

length of a Collatz sequence starting with that number, in which case we can simply use that result. And each time we compute a new length, we'll cache it.

```
lengths = {}

def get_collatz_length(n):
    if n not in lengths:
        if n == 1:
            lengths[n] = 1
        elif n % 2 == 0:
            lengths[n] = get_collatz_length(n / 2) + 1
        else:
            lengths[n] = get_collatz_length(3 * n + 1) + 1
    return lengths[n]

for i in range(1, 1000000):
    get_collatz_length(i)

print max(lengths, key=lengths.get) # 837799
```