Daily Coding Problem #161

Problem

This problem was asked by Facebook.

Given a 32-bit integer, return the number with its bits reversed.

For example, given the binary number `1111 0000 1111 0000 1111 0000 1111 0000`, return `0000 1111 0000 1111 0000 1111 0000 1111`.

Solution

We can do this by iterating over every bit from 0 to 31, checking whether that bit is on, and then setting that bit on a final result:

```
NUM_BITS = 32

def reverse_bits(n):
    reversed_num = 0
    for i in range(NUM_BITS):
        j = n >> i & 1
```

```
        reversed_num += j << (NUM_BITS - i - 1)
    return reversed_num
```

This takes O(n) time where n is the number of bits.

We can also trade off time for space by creating a lookup cache of already reversed bits, let's say 8. Then we only need to iterate at 32 / 8 = 4 times if we preprocess the cache.

```
NUM_BITS = 32
NUM_BITS_IN_CACHE = 8

def reverse_naive(n, num_bits):
    reversed_num = 0
    for i in range(num_bits):
        j = n >> i & 1
        reversed_num += j << (num_bits - i - 1)
    return reversed_num

def preprocess():
    cache = {}
    for i in range(2 ** NUM_BITS_IN_CACHE):
        cache[i] = reverse_naive(i, NUM_BITS_IN_CACHE)
    return cache

preprocessed = preprocess()

def reverse_bits(n):
    result = 0
    mask = 2 ** NUM_BITS_IN_CACHE - 1
    for i in range(NUM_BITS // NUM_BITS_IN_CACHE):
        relevant = n >> (i * NUM_BITS_IN_CACHE) & mask
        result += preprocessed[relevant] << NUM_BITS - ((i + 1) *
NUM_BITS_IN_CACHE)
    return result
```

This will take $O(2^c)$ time and space to initialize and O(n / c) time to query, where c is the number of bits in the cache key.
