Project Euler 545 - Faulhaber's Formulas

February 10, 2016

1 Problem Statement

The sum of the kth powers of the first n positive integers can be expressed as a polynomial of degree $k+1$ with rational coefficients, the Faulhaber's Formulas:

$$\sum_{i=1}^{n} i^k = \sum_{i=1}^{k+1} a_i n^i = a_1 n + a_2 n^2 + \ldots + a_{k+1} n^{k+1}.$$

For example, for $k = 4$ we have the polynomial

$$-\frac{1}{30}n + \frac{1}{3}n^3 + \frac{1}{2}n^4 + \frac{1}{5}n^5.$$

Define $D(k)$ as the value of q_1 for the sum of the kth powers. Define $F(m)$ as the mth value for $k \geq 1$ for which $D(k) = 20010$. Find $F(10^5)$.

Given values:

- $D(4) = 30$
- $D(308) = 20010$
- $F(1) = 308$
- $F(10) = 96404$.

2 Notes

$D(k)$ is just the denominator of the kth Bernoulli number. For even values of k (set $k = 2n$), from a result related to the von-Staudt Clausen Theorem, it is known that

$$D_{2n} = \prod_{(p-1)|2n} p.$$

To achieve $D(k) = 20010$, it is helpful to look at the prime factorization of 20010. Indeed, we have

$$20010 = 2 \cdot 3 \cdot 5 \cdot 23 \cdot 29.$$

Note that $D(308)$ is the first value to achieve 20010, since

$$1|308, 2|308, 4|308, 22|308, 28|308,$$

(i.e. $p-1|308$ for each p in the prime factorization of 20010).

It follows that all candidate k such that $D(k) = 20010$ must be multiples of 308.

Thus, a basic algorithm could entail calculating $D(k)$ where k runs over multiples over 308. But this is too inefficient. To be more precise, we want to find values of the form $n = 308m$, where m does not introduce more factors of the form $p-1$, for some prime p. If m is an odd prime, then m itself will not have factors of the form $p-1$, since (nontrivial) values of $p-1$ must be even. But m multiplied by other factors can introduce $p-1$ factors. The prime factorization of 308 is

$$308 = 2^2 \cdot 7 \cdot 11.$$

For $m = 3$, note that $308m$ has 12 as a factor, which is indeed of the form $p-1$ for $p = 13$. Note that m can also be composite (this is rarer, but verified empirically for high values of m).

Note that if $308m$ doesn't satisfy our conditions, $308mn$ will not either. Thus, one strategy to filter values is to check prime values first, and then check composite ones that can be formed as products of valid prime multiples.

Using a (slightly generalized) sieve algorithm, one can get the divisors of the first n integers relatively easily. That seems key here, to quickly check for the divisibility of the $p-1$ terms.

Oddly, I don't seem to be getting much improvement in speed from my algorithm that sieves the divisors first. Also, I would expect that sieving more primes should result in an increased one-time computation, but the main loop complexity should be unaffected. That doesn't seem to be the case empirically - maybe I need to break one of my prime loops earlier. Further optimization might result in leveraging the fact that k is always of the form $308m$ when computing $D(k)$ – this way, you have to only sieve a factor of $1/308$ less primes than normal (assuming you handle the logic to check divisibility of the factors that 308 contributes).

I just achieved pretty significant speedup by removing the `set` conversion of the divisor list. It seems a little tricky to implement in practice (since I've relied on library functions), but implementing `get_divisors` as a generator would likely speed things up.

After looking at profiling, using the fact that candidates are of the form $308m$ seems to be the way to go. The behavior is a bit odd, though. One way to get the set of primes to test on is to use binary search (with the `bisect` module). Perhaps this would be faster. After implementing this, it does seem slightly faster, but not by a huge amount.

Key insight - instead of iterating through primes to check divisibility, it's far simpler to iterate through divisors. Supplementing this with Miller-Rabin primality testing for large primes, my algorithm can run almost instantaneously.