Project Euler: Problem 421

> **Problem.** Let $s(n, m)$ denote the sum of all distinct prime factors of $n^{15} + 1$ less than or equal to m. Determine the value of
>
> $$\sum_{n=1}^{10^{11}} s(n, 10^8).$$

Solution. For sake of being general, let $L := 10^{11}$ and $N := 10^8$ so that we must determine the value instead of

$$\sum_{n=1}^{L} s(n, N).$$

Observation. Our first observation is that L is sufficiently large such that it is not possible to even iterate over each value of n in a reasonable amount of time. It *is*, however, possible to iterate over all primes less than or equal to N in a reasonable amount of time, which motivates us to make a transformation of the problem.

In particular, we shall take the sum over all primes less than or equal to N and, noticing that we only care about distinct prime factors, we can evaluate this sum as the product of each prime with the frequency for which it appears in $n^{15} + 1$ as n ranges from 1 to L. Along with the previous motivation given for such a reframing, this transformation is often used in many similar Project Euler problems, which is a good indicator that we are on the right track.

It now suffices to determine the number of $n \le L$ such that for some fixed p,

$$n^{15} + 1 \equiv 0 \pmod{p}.$$

Denote this frequency by $f_{\le L}(p)$.

Observation. Obviously, the values of $n^{15} + 1$ are periodic with a period length of p. This means that there are the same number of solutions to the indicated equation in each block of p elements. This allows us to decompose $f_{\le L}(p)$ to be

$$f_{\le L}(p) = \left\lfloor \frac{L}{p} \right\rfloor f_{\le p}(p) + f_{\le L \bmod p}(p).$$

We shall try to tackle the $f_{\le p}(p)$ term first.

Observation. The number of solutions to

$$n^{15} + 1 \equiv 0 \pmod p$$

is the same as the number of solutions to

$$n^{15} - 1 \equiv 0 \pmod p.$$

Proof. Suppose that ω is some solution to

$$\omega^{15} \equiv -1 \pmod p.$$

We can then observe that $-\omega$ is a solution to the latter equation since

$$(-\omega)^{15} \equiv -\omega^{15} \equiv 1 \pmod p.$$

Since $x \mapsto -x$ is a bijection over $\mathbb{Z}_p^\times$, the number of solutions to the first and second equation are equal. ■

For the case of $f_{\leq L \bmod p}(p)$, this also means that we can simply find all solutions to the second equation in the specified range and negate them.

This is now slightly easier to work with because we have more knowledge of solving for when values are congruent to the identity element due to Fermat's little theorem and group theory.

Observation. Observe that since $15 = 3 \cdot 5$, we know that if

$$n^{15} \equiv 1 \pmod p,$$

then n must have an order of either 1, 3, 5, or 15 in $\mathbb{Z}_p^\times$ (since 15 must be a multiple of the order). This tells us that there at least exists a subgroup of $\mathbb{Z}_p^\times$ that is of the same order, and one can verify that this subgroup is cyclic. Ranging n over all possible values, we see that there are (and assert that there are only) either $1, 3, 5$, or 15 solutions to this equation depending on whether or not subgroups of the corresponding orders exist for the group.

This turns our focus to determining whether the group has a subgroup of certain order. In particular, we have the order of the group as $|\mathbb{Z}_p^\times| = p - 1$ and a desired subgroup order, say a, and we want to determine whether there is such a subgroup in this group. In other words, we wish for a sort of converse to Lagrange's theorem, which can be achieved using the following theorem (one could also apply the more general Sylow's first theorem).

> **Theorem** (Cauchy's Theorem)**.** Let G be a finite group and suppose some prime p divides $|G|$. There then exists a subgroup of G with order p.

Applying it to our case here, we see that there exists subgroups of order 3 and 5 iff $3 \mid p - 1$ and $5 \mid p - 1$ respectively. Note that there exists a subgroup of order 15 iff both such subgroups of order 3 and 5 exist (as this subgroup is just their direct product). Thus, we have the following realization.

Lemma. There are $\gcd(15, p - 1)$ solutions to

$$n^{15} \equiv 1 \pmod{p}$$

for $1 \leq n < p$.

The residual term, $f_{\leq L \bmod p}(p)$, is a lot more unwieldy. Since L is arbitrary, we must find some way of computing efficiently $f_{\leq k}(p)$ for any arbitrary k such that $1 \leq k < p$. Indeed, one must be cognizant of our entire solution time complexity when looking for a viable solution. Iterating over all primes less than or equal to $N := 10^8$ alone takes $O(N/\log N)$ time, so we mandate that computing the answer for each prime must take sublinear time in order for our program to run in reasonable enough time. Thus, if we're to find all solutions to

$$n^{15} \equiv \pm 1 \pmod{p}$$

we must be smarter than just iterating over all elements of $\mathbb{Z}_p^\times$. Indeed, we shall use our previous work on the existence of cyclic subgroups to help us in this case.

Observation. Suppose we have a generator g of $\mathbb{Z}_p^\times$. Consider elements of the form $g^{k(p-1)/q}$ for $0 \leq k < q$, where q is some prime dividing $p - 1$. We can then see that the set of these elements forms a subgroup of order q. If we take $q = 3$ or $q = 5$, these are precisely the subgroups that we want to compute. We have that q is constant with respect to p and calculating $g^{(p-1)/q}$ only takes $O(\log p)$ time, so this gives us indeed a sublinear way of getting all solutions modulo p.

Now we turn to finding a generator of $\mathbb{Z}_p^\times$. This, too, must be determined in sublinear time, so we still cannot just linearly iterate over all elements. It should be noted, though, that there is a very efficient approach to checking whether an element is indeed a generator of the group. Recall the following result:

> **Fact.** An element g is a generator if and only if for *each* prime factor q of $p - 1$,
>
> $$g^{(p-1)/q} \not\equiv 1 \pmod{p}.$$

Since there are asymptotically $O(\log \log p)$ distinct prime factors (see the omega function) and we can factor $p - 1$ in roughly $O(\log p)$ time with the help of some precomputation, this gives us a very fast way of verifying that an element is in fact a generator.

The fact that we can quickly verify elements but cannot linearly iterate over all of them in time motivates the following idea:

Idea. We can randomly sample values from $\mathbb{Z}_p^\times$ until we reach a generator.

Our intuition for this is aided by the following theorem:

> **Theorem.** There are $\phi(p - 1)$ generators in $\mathbb{Z}_p^\times$.

With this, there is a $\phi(p-1)/(p-1)$ probability of sampling a generator, so we should then expect $(p-1)/\phi(p-1)$ samples before arriving at a generator (note that this assumes sampling with replacement; sampling without replacement would likely achieve better odds at the honestly small cost of storing past samples in perhaps a hashmap). Luckily, this probability is actually decently large. Direct computation of primes up to 10^6 tells us that the average value of this characteristic is actually roughly 0.374. It also also known that

$$\sum_{k=2}^{n} \phi(n) \sim \frac{1}{2\zeta(2)} n^2 + O(n \log n).$$

Although there is no rigorous basis for this, we can expect that the probability of sampling a random generator is probably close to this value in the limit, so all in all we expect roughly $2\zeta(2) \approx 3.3$ samples per prime, which is absolutely within reason, and we can just treat this as a constant factor for time complexity.

In total, this means that for each prime p, we can determine the set of solutions to

$$n^{15} \equiv 1 \pmod{p}$$

in $O(\log p \log\log p)$ time. Negating these n will then give us the desired solutions to the original congruence. This means that our full solution should run in roughly $O(N \log\log N)$ time.

While our ability to generate the set does seem to make some of our work determining $f_{\leq p}(p)$ slightly redundant, it should be noted that it's better to use our previous work for efficiency and perhaps even some cleanliness. It's also just really interesting.

Remark. We shall make some final remarks on implementation, specifically on the size of the answer. Since the problem does not specify that we should take the answer modulo some reasonably large prime smaller than the size of a 64-bit integer, we are inclined to believe that the full answer should fit, but it's good practice to show that this is true anyway.

In particular, we can intuit the following upper bound:

$$\begin{aligned}\textsf{answer} &:= \sum_{p \leq N} p \left(\left\lfloor \frac{L}{p} \right\rfloor f_{\leq p}(p) + f_{\leq L \bmod p}(p) \right) \\ &\leq \sum_{p \leq N} 15p \left(\frac{L}{p} + 1 \right) \\ &= 15 \sum_{p \leq N} (L + p) \,.\end{aligned}$$

Since the sum is taken over all primes less than N, we can transform this sum to be

$$15L \cdot \pi(N) + 15 \sum_{p \leq N} p.$$

Indeed, this sum is less than $2^{64} - 1$ (which can be quickly verified with Mathematica; the explicit value is 8646370646855809140), so if we use an unsigned long long integer type, we shouldn't run into integer overflow.