NOI.PH Training: Extras

Hyperbola Techniques

Kevin Charles Atienza

Contents

1 Introduction

Here, we'll discuss a couple of techniques that were/are quite common in Project Euler, especially during the time I was active in it as a problem setter.

Here are some motivating problems:

Problem 1.1. Given n, compute the sum $\sum_{x=1}^{n} \sigma(x)$ where $\sigma(x)$ is the sum of divisors of x. Compute it in $\mathcal{O}(n^{1/2})$.

Problem 1.2. Given n, find the sum of all squarefree numbers from 1 to n in $\mathcal{O}(n^{4/9})$ time.

Problem 1.3. Given n and m, find the number of lattice points visible from the origin in the rectangle $[0, n] \times [0, m]$. A lattice point P is visible from the origin O if there are no lattice points in the interior of the line segment OP.

Compute it in $\mathcal{O}(\max(n, m)^{1-\varepsilon})$ time for some $\varepsilon > 0$.

You'll notice that there are weird magic exponents appearing here. You'll come to understand why they're the way they are, but for now, notice that what they have in common is that they're all less than 1, so our solutions are all *sublinear*. (Later on, you'll even encounter log factors, such as $\mathcal{O}(n^{2/3}(\log n)^{1/3})$!

1.1 How to read

As usual, it's instructive to try the problems out before reading about them! If you don't, then you may be able to learn the "raw" forms of the technique, but you may not be able to apply them to novel situations (which is more likely to happen in an actual contest setting—problem setters strive to use original problem ideas).

Also, some sections describe things in very general settings. If it's all too general for you, consider skimming the section (or even skipping) on first reading.

1.2 Notation

Unless otherwise stated, the functions f we'll consider here will be functions taking on positive integer arguments, with the additional property that $f(1) \neq 0$.

For any function f taking on positive integer arguments,

- We extend it to 0 by defining $f(0) = 0$.
- Also, we extend it to non-integer arguments using the rule $f(x) = f(\lfloor x \rfloor)$. This will allow us to clean up our notation a bit by reducing the amount of floor symbols.
- Also, we define a new function ∇f, with $\nabla f(n) := f(n) - f(n-1)$.[1] Thus, we have that

$$\sum_{k=a+1}^{b} \nabla f(k) = f(b) - f(a),$$

[1] This is usually called the "backward difference" of f.

or the so-called "Fundamental Theorem of the Calculus of Finite Differences."[2]

Note also that

$$\sum_{k=1}^{n} \nabla f(k) = f(n).$$

[2] or at least the gist of it, haha. So-named because of the similarity to $\int_a^b df(x) = f(b) - f(a)$ for a differentiable f.

2 The "hyperbola trick"

2.1 The basic hyperbola trick

Perhaps the simplest problem solvable with the tricks introduced here is this:

Problem 2.1. Given n, compute

$$\sum_{x=1}^{n} \left\lfloor \frac{n}{x} \right\rfloor.$$

There is an obvious $\mathcal{O}(n)$ solution, but the goal is to compute it much faster than that.

There are several well-known ways to do so. I suggest you try it out before proceeding.

I will begin by describing a solution that involves an elegant reformulation. First, we replace the single sum by a double sum. This might seem to complicate things, but just follow along for now:

$$\begin{aligned}\sum_{x=1}^{n} \left\lfloor \frac{n}{x} \right\rfloor &= \sum_{x=1}^{n} \sum_{y=1}^{\lfloor n/x \rfloor} 1 \\ &= \sum_{\substack{x,y \\ 1\le x\le n \\ 1\le y\le \lfloor n/x \rfloor}} 1 \\ &= \sum_{\substack{x,y \\ x,y>0 \\ xy\le n}} 1.\end{aligned}$$

But the last one has a rather nice characterization: it is simply the number of lattice points in the region in the first quadrant bounded by the hyperbola $xy = n$ and the axes!

Armed with this knowledge, we can now compute it by exploiting the symmetry of this region about the line $x = y$. We split the sum into three parts, $x > y$, $x = y$ and $x < y$, and note that the sum for $x > y$ and $x < y$ are the same:

$$\begin{aligned}\sum_{\substack{x,y>0 \\ xy\le n}} 1 &= \sum_{\substack{x,y>0 \\ xy\le n \\ x>y}} 1 + \sum_{\substack{x,y>0 \\ xy\le n \\ x=y}} 1 + \sum_{\substack{x,y>0 \\ xy\le n \\ x<y}} 1 \\ &= \sum_{\substack{x,y>0 \\ xy\le n \\ x<y}} 1 + \sum_{\substack{x,y>0 \\ xy\le n \\ x=y}} 1 + \sum_{\substack{x,y>0 \\ xy\le n \\ x<y}} 1 \\ &= 2\sum_{\substack{x,y>0 \\ xy\le n \\ x<y}} 1 + \sum_{\substack{x,y>0 \\ xy\le n \\ x=y}} 1 \\ &= 2\sum_{\substack{x,y>0 \\ xy\le n \\ x<y}} 1 + \sum_{\substack{x>0 \\ x^2\le n}} 1 \\ &= 2\sum_{\substack{x,y>0 \\ xy\le n \\ x<y}} 1 + \lfloor \sqrt{n} \rfloor.\end{aligned}$$

We can now focus on the remaining sum. But since $x < y$ and $xy \le n$, it's easily seen that

$x \le \sqrt{n}$, thus, we can loop across all possible values of x and it will be better than $\mathcal{O}(n)$!

$$\begin{aligned}\sum_{\substack{x,y>0\\ xy\le n\\ x<y}} 1 &= \sum_{x=1}^{\lfloor\sqrt{n}\rfloor} \sum_{\substack{y>0\\ x<y\le n/x}} 1 \\ &= \sum_{x=1}^{\lfloor\sqrt{n}\rfloor} \left(\left\lfloor \frac{n}{x} \right\rfloor - x\right).\end{aligned}$$

There are only $\mathcal{O}(\sqrt{n})$ terms in this sum, hence, the answer can be computed in $\mathcal{O}(\sqrt{n})$ time!

For curiosity, writing the full formula, we get:

$$\begin{aligned}\sum_{x=1}^{n} \left\lfloor \frac{n}{x} \right\rfloor &= 2\sum_{x=1}^{\lfloor\sqrt{n}\rfloor} \left(\left\lfloor \frac{n}{x} \right\rfloor - x\right) + \lfloor\sqrt{n}\rfloor \\ &= 2\sum_{x=1}^{\lfloor\sqrt{n}\rfloor} \left\lfloor \frac{n}{x} \right\rfloor - \lfloor\sqrt{n}\rfloor^2 - \lfloor\sqrt{n}\rfloor + \lfloor\sqrt{n}\rfloor \\ &= 2\sum_{x=1}^{\lfloor\sqrt{n}\rfloor} \left\lfloor \frac{n}{x} \right\rfloor - \lfloor\sqrt{n}\rfloor^2.\end{aligned}$$

Geometrically, the sum on the left (without the 2) indicates the number of lattice points in the region from $x = 1$ up to $x = \sqrt{n}$ only, while the factor of 2 is there to also count the number of lattice points in the *mirror image* of the region about the line $y = x$, and $\lfloor\sqrt{n}\rfloor^2$ is subtracted to take care of the double-counted region, which is just the square $[1, \sqrt{n}]^2$.

2.2 A more general hyperbola trick

The above derivation relies on exploiting the symmetry of a region bounded by some hyperbola, but there's another perspective on it that lends itself more easily to generalization. It only relies on the fact that the set

$$\langle n \rangle := \{\lfloor n/x \rfloor : x \in \{1, 2, \ldots, n\}\}$$

has $\mathcal{O}(\sqrt{n})$ elements. The proof is simple. Try proving it yourself!

Exercise 2.1. Prove that $|\langle n \rangle| = \mathcal{O}(\sqrt{n})$. **Hint**: For all $x \ge \sqrt{n}$, $n/x \le \sqrt{n}$.

Since this set is so important, and will be central to a lot of the algorithms we will discuss, we'll want to denote it with something. That's why we introduce the notation $\langle n \rangle$.

Let's start over then, and compute the sum in a different way.

$$\sum_{1\le x\le n} \left\lfloor \frac{n}{x} \right\rfloor$$

To take advantage of the relatively small size of $\langle n \rangle$, we'll try to compute this sum by

iterating through all possible values of $\lfloor n/x \rfloor$:

$$\begin{aligned}\sum_{1\le x\le n}\left\lfloor\frac{n}{x}\right\rfloor &= \sum_{v\in\langle n\rangle}\sum_{\substack{1\le x\le n\\ \lfloor n/x\rfloor=v}}\left\lfloor\frac{n}{x}\right\rfloor \\ &= \sum_{v\in\langle n\rangle}\sum_{\substack{1\le x\le n\\ \lfloor n/x\rfloor=v}} v \\ &= \sum_{v\in\langle n\rangle} v\sum_{\substack{1\le x\le n\\ \lfloor n/x\rfloor=v}} 1\end{aligned}$$

Now, since the outer sum iterates through only $\mathcal{O}(\sqrt{n})$ values, all will be well if we can reduce the inner sum to $\mathcal{O}(1)$. But it's actually easy to show that

$$\lfloor n/x\rfloor = v \iff \left\lfloor\frac{n}{v+1}\right\rfloor < x \le \left\lfloor\frac{n}{v}\right\rfloor.$$

Hence, the inner sum is really just $\lfloor\frac{n}{v}\rfloor - \left\lfloor\frac{n}{v+1}\right\rfloor$, and we're done! The final formula is

$$\sum_{1\le x\le n}\left\lfloor\frac{n}{x}\right\rfloor = \sum_{v\in\langle n\rangle} v\left(\left\lfloor\frac{n}{v}\right\rfloor - \left\lfloor\frac{n}{v+1}\right\rfloor\right).$$

Although it also runs in $\mathcal{O}(\sqrt{n})$, you might say this looks less elegant than the first approach, so why did we bother doing it this way? Well, I did say that this is more easily generalizable, but how so?

The reason is that it still works for other kinds of sums like

$$\sum_{x=1}^{n} x^2\left\lfloor\frac{n}{x}\right\rfloor^3.$$

Let's try to compute it. Well, it pretty much goes the same way:

$$\begin{aligned}\sum_{1\le x\le n} x^2\left\lfloor\frac{n}{x}\right\rfloor^3 &= \sum_{v\in\langle n\rangle}\sum_{\substack{1\le x\le n\\ \lfloor n/x\rfloor=v}} x^2\left\lfloor\frac{n}{x}\right\rfloor^3 \\ &= \sum_{v\in\langle n\rangle}\sum_{\substack{1\le x\le n\\ \lfloor n/x\rfloor=v}} x^2v^3 \\ &= \sum_{v\in\langle n\rangle} v^3\sum_{\substack{1\le x\le n\\ \lfloor n/x\rfloor=v}} x^2 \\ &= \sum_{v\in\langle n\rangle} v^3\sum_{x=\left\lfloor\frac{n}{v+1}\right\rfloor+1}^{\left\lfloor\frac{n}{v}\right\rfloor} x^2 \\ &= \sum_{v\in\langle n\rangle} v^3\left[f_2\left(\frac{n}{v}\right) - f_2\left(\frac{n}{v+1}\right)\right].\end{aligned}$$

(remember the convention $f(x) = f(\lfloor x\rfloor)$)

In the latter, $f_2(n)$ denotes the sum $1^2+2^2+\ldots+n^2$ which has a closed-form formula that can be computed in $\mathcal{O}(1)$, and which you also probably know![3]

[3] See https://en.wikipedia.org/wiki/Faulhaber%27s_formula#Faulhaber_polynomials.

Obviously, this approach can be extended straightforwardly to sums like this:

$$\sum_{x=1}^{n} x^a \left\lfloor \frac{n}{x} \right\rfloor^b.$$

Even more generally, it can be extended to sums of the form

$$\sum_{x=1}^{n} \nabla d(x) h(n/x)$$

for *any* functions d and h. Assuming that $d(x)$ and $h(x)$ can be computed in $\mathcal{O}(1)$ each, this sum can be computed in $\mathcal{O}(\sqrt{n})$ time. Let's derive this:

$$\begin{aligned}
\sum_{1\le x\le n} \nabla d(x) h(n/x) &= \sum_{v\in\langle n\rangle} \sum_{\substack{1\le x\le n \\ \lfloor n/x\rfloor = v}} \nabla d(x) h(n/x) \\
&= \sum_{v\in\langle n\rangle} \sum_{\substack{1\le x\le n \\ \lfloor n/x\rfloor = v}} \nabla d(x) h(v) \\
&= \sum_{v\in\langle n\rangle} h(v) \sum_{\substack{1\le x\le n \\ \lfloor n/x\rfloor = v}} \nabla d(x) \\
&= \sum_{v\in\langle n\rangle} h(v) \sum_{x=\left\lfloor \frac{n}{v+1}\right\rfloor+1}^{\left\lfloor \frac{n}{v}\right\rfloor} \nabla d(x) \\
&= \sum_{v\in\langle n\rangle} h(v) \left[d\left(\frac{n}{v}\right) - d\left(\frac{n}{v+1}\right)\right].
\end{aligned}$$

Remember that d and h can be computed in $\mathcal{O}(1)$ time. Thus, the overall sum can be computed in $\mathcal{O}(\sqrt{n})$ time!

Theorem 2.2.

$$\sum_{1\le x\le n} \nabla d(x) h(n/x) = \sum_{v\in\langle n\rangle} h(v) \left[d\left(\frac{n}{v}\right) - d\left(\frac{n}{v+1}\right)\right].$$

In other words, if you ever find yourself needing to sum something of the form

$$\sum_{x=1}^{n} f(x) h(n/x)$$

and it so happens that $\sum_{k=1}^{n'} f(k)$ (f's "antiderivative") can be computed in $\mathcal{O}(1)$ for any n', then the whole sum can be computed in $\mathcal{O}(\sqrt{n})$ time!

3 The inverse form "hyperbola trick"

In this section, we will discuss a related but slightly different problem.

3.1 The trick with gcd sums

We begin by trying to solve a new problem.

Problem 3.1. We say that a lattice point (x, y) is **visible** from the origin if the (open) line segment joining the origin and (x, y) contains no lattice points. Given n, how many lattice points in $[1, n] \times [1, n]$ are visible from the origin?

To solve this, we first need to characterize when (x, y) is visible or not. First, it is necessary that $\gcd(x, y) = 1$, because if $\gcd(x, y) = g > 1$, then $(x/g, y/g)$ is a lattice point in the segment. On the other hand, it turns out that having $\gcd(x, y) = 1$ is also sufficient; we'll leave the proof to the reader. Thus, the answer to the problem can be expressed as double sum

$$f(n) := \sum_{x=1}^{n} \sum_{\substack{y=1 \\ \gcd(x,y)=1}}^{n} 1.$$

This time, the naive solution runs in $\mathcal{O}(n^2 \log n)$,[4] so the goal is to compute it much faster than that. In fact, we can show that this can be computed in $o(n)$ (i.e., *sublinear*) time!

When faced with a sum across values with gcd equal to 1, a rather funny trick surprisingly works: *ignore the gcd condition*! Well, let's do that, and define a new sum

$$h(n) := \sum_{x=1}^{n} \sum_{y=1}^{n} 1$$

Notice that $h(n)$ is easily computable—it's just n^2 :)—but how does this actually help us solve the original problem?

The trick is to decompose $h(n)$ into several sums, looping across all possible gcd values of x and y, like so:

$$\begin{aligned} h(n) &= \sum_{x=1}^{n} \sum_{y=1}^{n} 1 \\ &= \sum_{g=1}^{n} \sum_{x=1}^{n} \sum_{\substack{y=1 \\ \gcd(x,y)=g}}^{n} 1. \end{aligned}$$

Now, it seems to have become more complicated—after all, there are now *three* nested sums—but as before, just go along with it for now.

The fact that $\gcd(x, y) = g$ means that $x = gx'$ and $y = gy'$ for some integers x' and y'

[4] or $\mathcal{O}(n^2)$ if you use some dynamic programming tricks, or other clever tricks with Euclid's algorithm.

such that $\gcd(x', y') = 1$. Thus, let's iterate through x' and y' instead:

$$\begin{aligned}\sum_{g=1}^{n} \sum_{1\le x\le n} \sum_{\substack{1\le y\le n \\ \gcd(x,y)=g}} 1 &= \sum_{g=1}^{n} \sum_{1\le gx'\le n} \sum_{\substack{1\le gy'\le n \\ \gcd(gx',gy')=g}} 1 \\ &= \sum_{g=1}^{n} \sum_{1\le x'\le n/g} \sum_{\substack{1\le y'\le n/g \\ \gcd(x',y')=1}} 1 \\ &= \sum_{g=1}^{n} f(n/g).\end{aligned}$$

This establishes a relationship between $f(n)$ and $h(n)$, namely,

$$h(n) = \sum_{1\le g\le n} f(n/g).$$

Notice that there's one summation remaining! Also, we're now in a little bit more familiar territory, since we're again dealing with values in $\langle n\rangle$:)

Now, you might think that we can use the results of the previous section to solve the problem; unfortunately, this is slightly different. Previously, the right side contains all known values, and the left side is the unknown. But this time, it's the reverse; f is the unknown, and h is known! For this reason, we'll call this the **inverse form**.

Fortunately, a very similar trick can be performed if we rewrite the inverse form as

$$f(n) = h(n) - \sum_{2\le g\le n} f(n/g).$$

Now, noticing that $n/g < n$, we could assume for now that these values have already been computed—for example, via memoization or dynamic programming—and so we can proceed to use a similar trick as above! We get

$$f(n) = h(n) - \sum_{\substack{v\in\langle n\rangle \\ v<n}} f(v) \left[\left\lfloor \frac{n}{v} \right\rfloor - \left\lfloor \frac{n}{v+1} \right\rfloor \right].$$

Thus, this achieves a running time of $\mathcal{O}(\sqrt{n})$ for a single $f(n)$...however, this assumes that $f(n')$ has already been computed for $n' < n$. We need to do the same computation for all n' from 1 to n. The running time then becomes

$$\mathcal{O}\left(\sum_{n'=1}^{n} \sqrt{n'} \right) = \mathcal{O}(n^{3/2}).^{5}$$

This is an improvement over the naive quadratic solution!

We can do better. Note that to compute $f(n)$, we only ever need the values $f(v)$ for $v \in \langle n\rangle$. Furthermore, for any $v \in \langle n\rangle$, we have $\langle v\rangle \subset \langle n\rangle$ (why?), so to compute $f(v)$, we already have what we need! So it turns out that we actually only ever need to calculate $f(v)$ for $v \in \langle n\rangle$, no matter how deep the recursion. And $\langle n\rangle$ is quite small, so this should improve the running time!

[5] You derive this by replacing the sum on the left with the integral $\int_1^n \sqrt{x}\,dx$, or just naively replacing $\sqrt{n'}$ with the upper bound $\sqrt{n}$.

So after restricting the computation to just $v \in \langle n \rangle$, what's the new complexity? Well, naively just adding up the cost of computing $f(n/1)$, $f(n/2)$, $f(n/3)$, etc., we get the complexity

$$\mathcal{O}\left(\sum_{g=1}^{n} \sqrt{n/g}\right) = \mathcal{O}(n),^{6}$$

which is already a further improvement!

But again, remember that $\lfloor n/x \rfloor$ takes only on a few values once x becomes moderately large, so in fact, the previous computation still overcounts the complexity. Let's compute it more accurately. Note that we can view $\langle n \rangle$ as consisting of two parts: the "large" values and the "small" values. Let's declare "small" as being less than $\sqrt{n}$, and then split the sum into two parts:

$$\mathcal{O}\left(\sum_{v \in \langle n \rangle} \sqrt{v}\right) = \mathcal{O}\left(\sum_{\substack{v \in \langle n \rangle \\ v \geq s}} \sqrt{v} + \sum_{\substack{v \in \langle n \rangle \\ v < s}} \sqrt{v}\right).$$

Here, s will stand for a number close to $\sqrt{n}$.

The first sum corresponds to computing $f(n/g)$ for $\lfloor n/g \rfloor \geq s$. We note that $\lfloor n/g \rfloor \geq s$ if and only if $g \leq \lfloor n/s \rfloor$, so the total is

$$\mathcal{O}\left(\sum_{g=1}^{\lfloor n/s \rfloor} \sqrt{n/g}\right) = \mathcal{O}(n/s^{1/2}).$$

The second sum corresponds to computing $f(1), f(2), \ldots, f(s-1)$ since these are all the distinct values. Hence, the total complexity is

$$\mathcal{O}\left(\sum_{v=1}^{s-1} \sqrt{v}\right) = \mathcal{O}(s^{3/2}).$$

Combining the two parts, we get a total complexity of

$$\mathcal{O}(n/s^{1/2} + s^{3/2}).$$

Now, as I said, s is close to $\sqrt{n}$, so in fact, the complexity is just $\mathcal{O}(n^{3/4})$, which is now *sublinear* in n :D

In fact, one can also show that choosing s to be anything else other than close to $\sqrt{n}$ will yield a worse complexity. I invite you to show why this is so.

3.2 The sieve improvement

This can be improved even further!

Let's step back a bit to our original relation,

$$h(n) = \sum_{1 \leq g \leq n} f(n/g).$$

[6] again, you replace the sum with an integral to derive this.

An interesting thing happens when we look at ∇h:

$$\begin{aligned}\nabla h(n) &= h(n) - h(n-1) \\ &= \sum_{1 \le g \le n} f\left(\frac{n}{g}\right) - \sum_{1 \le g \le n-1} f\left(\frac{n-1}{g}\right) \\ &= \sum_{1 \le g \le n} \left[f\left(\frac{n}{g}\right) - f\left(\frac{n-1}{g}\right) \right].\end{aligned}$$

(Remember that we set $f(0) = 0$ by convention, so we can combine the sums like this.)

Now, consider the value $f\left(\frac{n}{g}\right) - f\left(\frac{n-1}{g}\right)$. Note that the difference between $\lfloor \frac{n}{g} \rfloor$ and $\lfloor \frac{n-1}{g} \rfloor$ is either 0 or 1. Furthermore:

$$f\left(\frac{n}{g}\right) - f\left(\frac{n-1}{g}\right) = \begin{cases} \nabla f\left(\frac{n}{g}\right) & \text{if } \lfloor \frac{n}{g} \rfloor \neq \lfloor \frac{n-1}{g} \rfloor, \\ 0 & \text{if } \lfloor \frac{n}{g} \rfloor = \lfloor \frac{n-1}{g} \rfloor. \end{cases}$$

Hence, we only need to look at the cases where $\lfloor \frac{n}{g} \rfloor \neq \lfloor \frac{n-1}{g} \rfloor$. When does that happen? That happens when $\lfloor \frac{n}{g} \rfloor$ is some integer, but if we decrease the numerator by 1, i.e., $\lfloor \frac{n-1}{g} \rfloor$, we get the next lower integer. But that happens precisely when n is a multiple of g! Hence, we have

$$\nabla h(n) = \sum_{g \mid n} \nabla f\left(n/g\right).$$

The argument just shown is actually reversible, and thus, proves the following theorem:

Theorem 3.1. For any f and h, the following two are equivalent:

- For all n, $h(n) = \sum_{1 \le g \le n} f(n/g)$.
- For all n, $\nabla h(n) = \sum_{g \mid n} \nabla f\left(n/g\right)$.

This relationship between ∇h and ∇f allows us to perform a sieve-like procedure to compute $\nabla f(1), \nabla f(2), \ldots, \nabla f(n)$ in $\mathcal{O}(n \log n)$ time. Afterwards, $f(1), f(2), \ldots, f(n)$ can be computed in $\mathcal{O}(n)$ using prefix sums!

1. First, set $f[k] := \nabla h(k)$ for $1 \le k \le n$.
2. For every g from 1 to n, subtract $f[g]$ from $f[k]$ for all proper multiples k of g.
3. Compute the prefix sums of f.

So the overall complexity is dominated by the second step, which is

$$\mathcal{O}\left(\frac{n}{1} + \frac{n}{2} + \ldots + \frac{n}{n}\right) = \mathcal{O}(n \log n).$$[7]

This is not really an improvement (yet) over $\mathcal{O}(n^{3/4})$. But remember that in the $\mathcal{O}(n^{3/4})$ algorithm, we had a step where we needed to compute $f(1), f(2), \ldots, f(s-1)$. Previously, we

[7] Again, you derive this with an integral.

simply used the same hyperbola+recursion trick, but what if we use this new sieve technique instead?

Then the complexity becomes $\mathcal{O}(n/s^{1/2} + s \log s)$. Again, we want to choose s to minimize this. This time, $s = \Theta(\sqrt{n})$ is not the optimal choice, but rather, something which will give $\mathcal{O}(n/s^{1/2}) = \mathcal{O}(s \log s)$. The choice $s = \Theta((n/\log n)^{2/3})$ works,[8] and the overall complexity becomes $\mathcal{O}(n^{2/3}(\log n)^{1/3})$, which is now an improvement over $\mathcal{O}(n^{3/4})$!

As a side note, in some (many?) cases, the sieve above can be improved from $\mathcal{O}(n \log n)$ to $\mathcal{O}(n \log \log n)$. For instance, in cases where ∇h is *multiplicative*[9], so that its value is completely dependent on its prime power factors, we can perform an Eratosthenes-like sieve to compute everything in $\mathcal{O}(n \log \log n)$. (There are even instances where the sieve improvement can be done even when ∇h is not multiplicative.) Applying such an improved sieve yields a very slightly better complexity of $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ which is actually sometimes significant, believe it or not! So the lesson here is to always attempt to find a fast sieve to improve the algorithm.

3.3 A generalization

The above algorithm allows us to compute $f(n)$ in $\mathcal{O}(n^{2/3}(\log n)^{1/3})$ and is based purely on the relationship

$$h(n) = \sum_{1 \le g \le n} f(n/g)$$

and the assumption that $h(n')$ can be computed in $\mathcal{O}(1)$ time. Slightly more generally, a similar algorithm exists if we start with a more general relationship like

$$h(n) = \sum_{1 \le g \le n} \nabla d(g) f(n/g)$$

for *any* h and d, and assuming that $h(n')$ and $d(n')$ can be computed in $\mathcal{O}(1)$ time for any n'. The derivation is pretty much the same, so we leave it to the reader. The "sieve" part of this generalized algorithm relies on the following generalization of Theorem 3.1:

Theorem 3.2. For any functions f, a and h, the following two are equivalent:

- For all n, $h(n) = \sum_{1 \le g \le n} a(g) f(n/g)$.
- For all n, $\nabla h(n) = \sum_{g \mid n} a(g) \nabla f\left(n/g\right)$.

Theorem 3.1 is a special case with $a(g) = 1$. The proof of this is basically the same as the proof of Theorem 3.1 just with $a(g)$ sprinkled all over.

As an example application, suppose we modify the previous problem: instead of counting the visible points, we want to find the sum of $|x - y|$ across all visible points (x, y). In other words, we want to compute the sum

$$f(n) := \sum_{x=1}^{n} \sum_{\substack{y=1 \\ \gcd(x,y)=1}}^{n} |x - y|.$$

[8] In practice, you can increase or decrease s by some constant since the optimal splitting point is actually dependent on your implementation. This move is justified by the hidden constant behind the Θ notation.

[9] A function f is multiplicative if $f(xy) = f(x)f(y)$ for all coprime pairs (x, y).

Again, using the "*ignore the gcd condition*" trick, we get

$$h(n) := \sum_{x=1}^{n} \sum_{y=1}^{n} |x - y|.$$

Although this is a bit more complicated, it should still be possible to compute $h(n)$ in $\mathcal{O}(1)$.

Now, we again split the sum $h(n)$ across the different gcd values of (x, y):

$$\begin{aligned}
h(n) &= \sum_{g=1}^{n} \sum_{1 \le x \le n} \sum_{\substack{1 \le y \le n \\ \gcd(x,y)=g}} |x - y| \\
&= \sum_{g=1}^{n} \sum_{1 \le gx' \le n} \sum_{\substack{1 \le gy' \le n \\ \gcd(gx',gy')=g}} |gx' - gy'| \\
&= \sum_{g=1}^{n} g \sum_{1 \le x' \le n/g} \sum_{\substack{1 \le y' \le n/g \\ \gcd(x',y')=1}} |x' - y'| \\
&= \sum_{g=1}^{n} g \cdot f(n/g).
\end{aligned}$$

This is now of the generalized form just mentioned, with $\nabla d(g) = g$, hence $d(g) = g(g+1)/2$, which can be computed in $\mathcal{O}(1)$. We leave it up to you to derive an $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ algorithm from this relationship.

4 Similar sums involving other exponents

There are other kinds of sums that can be solved with similar techniques but are not yet covered by the previous cases.

For example, let us consider the sum

$$\sum_{x=1}^{n} \left\lfloor n/x^2 \right\rfloor.$$

Actually, rather than doing only this sum, let's go ahead and generalize it immediately:

$$\sum_{x=1}^{n} \nabla d(x) h(n/x^2).$$

Although this is not covered by the previous cases, pretty much the same technique works. But this time, rather than looking at the set $\langle n \rangle$, we must look at the following set instead:

$$\langle n \rangle_2 := \left\{ \lfloor n/x^2 \rfloor : x \in \{1, 2, \ldots, \lfloor n^{1/2} \rfloor\} \right\}.$$

With similar techniques, one can easily show that $\langle n \rangle_2$ contains $\mathcal{O}(n^{1/3})$ terms.

We can now derive the sum above. The steps follow very similarly:

$$\begin{aligned}\sum_{1 \le x \le n} \nabla d(x) h(n/x^2) &= \sum_{v \in \langle n \rangle_2} \sum_{\substack{1 \le x \le n \\ \lfloor n/x^2 \rfloor = v}} \nabla d(x) h(v) \\ &= \sum_{v \in \langle n \rangle_2} h(v) \sum_{\substack{1 \le x \le n \\ \lfloor n/x^2 \rfloor = v}} \nabla d(x).\end{aligned}$$

Again, with a similar derivation, we find that

$$\lfloor n/x^2 \rfloor = v \iff \left\lfloor \sqrt{\frac{n}{v+1}} \right\rfloor < x \le \left\lfloor \sqrt{\frac{n}{v}} \right\rfloor.$$

Hence, we have:

Theorem 4.1.

$$\sum_{1 \le x \le n} \nabla d(x) h(n/x^2) = \sum_{v \in \langle n \rangle_2} h(v) \left[d\left(\sqrt{\frac{n}{v}}\right) - d\left(\sqrt{\frac{n}{v+1}}\right) \right].$$

The complexity is now $\mathcal{O}(n^{1/3})$:)

More generally, sums of the form

$$\sum_{x=1}^{n} \nabla f(x) h(n/x^k)$$

can be computed in $\mathcal{O}(n^{1/(k+1)})$ time assuming f and h can be computed in $\mathcal{O}(1)$; we just need to look at the sets $\langle n \rangle_k$!

The related **generalized inverse form**, i.e., computing f from the relationship

$$h(n) = \sum_{g=1}^{n} \nabla d(g) f(n/g^k),$$

also has a sublinear algorithm counterpart, derived in pretty much the same way: with a recursive algorithm that runs in $\mathcal{O}(n^{1/(k+1)})$ for a single $f(n)$, and a sieve-like algorithm for the first s values. Combining them with the correct splitting point s yields an $\mathcal{O}(n^{(k+1)/(k^2+k+1)})$ algorithm. (This doesn't even have a log factor!) Without the sieve, the algorithm would just be $\mathcal{O}(n^{(k+2)/(k+1)^2})$.[10]

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$\dots$
no sieve	$n^{3/4}$	$n^{4/9}$	$n^{5/16}$	$n^{6/25}$	$n^{7/36}$	$\dots$
with sieve	$(\log n)^{1/3}n^{2/3}$	$n^{3/7}$	$n^{4/13}$	$n^{5/21}$	$n^{6/31}$	$\dots$

The whole derivation is pretty much the same as before, but as an example, let's derive it for $k=2$. Our goal is to find an $\mathcal{O}(n^{4/9})$ algorithm without the sieve, and an $\mathcal{O}(n^{3/7})$ algorithm with the sieve.

First, we have

$$h(n) = \sum_{g=1}^{n} \nabla d(g) f(n/g^2).$$

If we rewrite it as

$$f(n) = \frac{1}{\nabla d(1)} \left[h(n) - \sum_{g=2}^{n} \nabla d(g) f(n/g^2) \right],$$

then we can use the previous algorithm to compute a single $f(n)$ in $\mathcal{O}(n^{1/3})$ time. This time, note that we only need to compute $f(v)$ for $v \in \langle n \rangle_2$. As before, we can view $\langle n \rangle_2$ as consisting of two parts, the "large" values and the "small" values. We'll arbitrarily declare "small" as $v < s$ for some s. Thus, the total complexity becomes

$$\mathcal{O}\left(\sum_{v \in \langle n \rangle_2} v^{1/3} \right) = \mathcal{O}\left(\sum_{\substack{v \in \langle n \rangle_2 \\ v \geq s}} v^{1/3} + \sum_{\substack{v \in \langle n \rangle_2 \\ v < s}} v^{1/3} \right).$$

For the former, we simply compute each value individually, and noting that $\lfloor n/g^2 \rfloor \geq s$ if and only if $g \leq \sqrt{n/s}$, we have a total complexity of

$$\mathcal{O}\left(\sum_{g=1}^{\sqrt{n/s}} (n/g^2)^{1/3} \right) = \mathcal{O}\left(\frac{n^{1/2}}{s^{1/6}} \right).$$

For the latter, assuming we don't have a sieve yet, then we also compute each value individually the same way, giving a total complexity of

$$\mathcal{O}\left(\sum_{v=1}^{s-1} v^{1/3} \right) = \mathcal{O}\left(s^{4/3} \right).$$

Thus, the overall complexity is

$$\mathcal{O}\left(\frac{n^{1/2}}{s^{1/6}} + s^{4/3} \right)$$

which can be minimized by setting $s = \Theta(n^{1/3})$,[11] giving the complexity $\mathcal{O}(n^{4/9})$, which is what we expected.

Now, let's try to derive the sieve. To do so, we again consider ∇h. With similar arguments, we can prove the following:

[10] Note that $\frac{k+2}{(k+1)^2} > \frac{k+1}{k^2+k+1}$, so using the sieve is a theoretical improvement. Obviously, as k increases, the relative advantage also decreases, but I think it's still noticeable in practice.

[11] again, adjust s to suit your implementation.

Theorem 4.2. For any f, a and h, the following two are equivalent:

- For all n, $h(n) = \sum_{1 \le g \le n} a(g) f(n/g^2)$.
- For all n, $\nabla h(n) = \sum_{g^2 | n} a(g) \nabla f\left(n/g^2\right)$.

Again, we can convert this to a sieve-like procedure, but now, we iterate across all multiples of g^2 instead of g. Hence, the overall complexity of the sieve is

$$\mathcal{O}\left(\frac{n}{1^2} + \frac{n}{2^2} + \ldots\right) = \mathcal{O}(n).$$

(Note that there are no log factors since the infinite sum $\frac{1}{1^2} + \frac{1}{2^2} + \ldots$ converges to a finite value!) Thus, replacing the "small" part of the algorithm with this sieve, the overall complexity becomes

$$\mathcal{O}\left(\frac{n^{1/2}}{s^{1/6}} + s\right),$$

which can be optimized by setting $s = \Theta(n^{3/7})$, giving an overall complexity of $\mathcal{O}(n^{3/7})$, which is what we expected as well!

The derivation for general k is pretty much the same.

5 Connections with the Möbius function

The approaches derived above have some connection with the so-called **Möbius function**, μ, defined as

$$\mu(x) = \begin{cases} (-1)^k & \text{if } x \text{ is squarefree and has exactly } k \text{ prime factors,} \\ 0 & \text{if } x \text{ is not squarefree.} \end{cases}$$

An absolute lot can be said about the nice properties of $\mu(x)$, but for now, we'll just focus on a few of them.

First, we begin with the famous **Möbius inversion formula**:

Theorem 5.1. For any f and h, the following two are equivalent:

- For all n, $h(n) = \sum_{g|n} f(g)$.
- For all n, $f(n) = \sum_{g|n} h(g)\mu(n/g)$.

There are many ways to prove this identity, and one of them is presented in the appendix, but for now, let's take this as a given.

Using this, and Theorem 3.2, we'll derive an analog of Theorem 5.1 that applies to the sums we've been considering:

Theorem 5.2. For any f and h, the following two are equivalent:

- For all n, $h(n) = \sum_{1 \le g \le n} f(n/g)$.
- For all n, $f(n) = \sum_{1 \le g \le n} h(n/g)\mu(g)$.

Proof. In what follows, each formula $\phi(n)$ in each line is expected to be read: "For all n, $\phi(n)$."

$$
\begin{aligned}
& h(n) = \sum_{1 \le g \le n} f(n/g) && \\
\iff \quad & \nabla h(n) = \sum_{g \mid n} \nabla f(n/g) && \text{Theorem 3.2} \\
\iff \quad & \nabla h(n) = \sum_{g \mid n} \nabla f(g) && \text{Renaming } g \to n/g \\
\iff \quad & \nabla f(n) = \sum_{g \mid n} \nabla h(g)\mu(n/g) && \text{Theorem 5.1} \\
\iff \quad & \nabla f(n) = \sum_{g \mid n} \nabla h(n/g)\mu(g) && \text{Renaming } g \to n/g \\
\iff \quad & f(n) = \sum_{1 \le g \le n} h(n/g)\mu(g) && \text{Theorem 3.2}
\end{aligned}
$$

□

Theorem 5.2 provides an alternative way of solving a special case of the inverse form problem, i.e., computing f from the equation

$$h(n) = \sum_{1 \le g \le n} f(n/g).$$

By inverting it, it becomes a "direct" problem:

$$f(n) = \sum_{1 \le g \le n} \mu(g) h(n/g).$$

Unfortunately, we can't say yet that f can be computed in $\mathcal{O}(\sqrt{n})$. This is because in this case, μ has to be treated as the ∇ of some function, say M, and $M(n')$ itself has to be computable in $\mathcal{O}(1)$ for any n', before we get an $\mathcal{O}(\sqrt{n})$ algorithm. Unfortunately, M is quite complicated!

As a side note, the function M such that $\nabla M = \mu$ is called the *Mertens function*, and is simply the partial sum of the Möbius function:

$$M(n) = \sum_{1 \le g \le n} \mu(g).$$

5.1 The Mertens function

Fortunately, not all is lost. It turns out that we can further analyze the Mertens function the same way we've been doing before. The key is to look at the definition of M again:

$$M(n) = \sum_{1 \le g \le n} \mu(g).$$

Now, if we denote $1(n)$ as the "constant" function $1(n) = 1$,[12] then we can also write this as

$$M(n) = \sum_{1 \le g \le n} 1(n/g)\mu(g).$$

We can now invert it using Theorem 5.2, and we get the following rather cool formula for M:

$$1(n) = \sum_{1 \le g \le n} M(n/g)$$

[12] I quote "constant" since $1(0) = 0$ from our convention, hence it's not purely constant.

or simply

$$1 = \sum_{1 \le g \le n} M(n/g).$$

This formula is now of the inverse form, and using the same techniques, we can compute $M(v)$ for all $v \in \langle n \rangle$ in $\mathcal{O}(n^{2/3}(\log\log n)^{1/3})$ total time! (Note that one can easily compute $\{M(v) : 1 \le v \le s\}$ in $\mathcal{O}(s \log\log s)$ using a normal prime sieve since μ is multiplicative.)

Finally, after having computed $M(v)$ for $v \in \langle n \rangle$, we can now proceed with computing f:

$$\begin{aligned} f(n) &= \sum_{1 \le g \le n} \mu(g) h(n/g) \\ &= \sum_{1 \le g \le n} \nabla M(g) h(n/g) \\ &= \sum_{v \in \langle n \rangle} h(v) \left[M\left(\frac{n}{v}\right) - M\left(\frac{n}{v+1}\right) \right]. \end{aligned}$$

Fortunately, $\lfloor \frac{n}{v} \rfloor \in \langle n \rangle$ and $\lfloor \frac{n}{v+1} \rfloor \in \langle n \rangle$, so all the needed values of M have already been computed!

5.2 Generalized inversion

Finally, we mention that Theorems 5.1 and 5.2 are actually special cases of more general theorems from the theory of Dirichlet series.

To state the theorems, we first define the function $\varepsilon(n)$ to be

$$\varepsilon(n) := \begin{cases} 1 & \text{If } n = 1, \\ 0 & \text{If } n \neq 1. \end{cases}$$

We say that two functions b and d are **Dirichlet inverses** of each other if for all n,

$$\sum_{g \mid n} b(g) d(n/g) = \varepsilon(n).$$

Then the generalizations of Theorems 5.1 and 5.2 can be stated as follows:

Theorem 5.3. Suppose b and d are Dirichlet inverses. Then for any f and h, the following two are equivalent:

- For all n, $h(n) = \sum_{g \mid n} f(n/g) b(g)$.
- For all n, $f(n) = \sum_{g \mid n} h(n/g) d(g)$.

Theorem 5.4. Suppose b and d are Dirichlet inverses. Then for any f and h, the following two are equivalent:

- For all n, $h(n) = \sum_{1 \le g \le n} f(n/g) b(g)$.

- For all n, $f(n) = \sum_{1 \le g \le n} h(n/g)d(g)$.

Theorems 5.1 and 5.2 then follow from these and from the fact that the Möbius function and the "constant" function 1 are Dirichlet inverses.

6 Problems

6.1 Non-coding problems

Hints can be found in the footnotes. Also, it is understood that the computations are done modulo $10^9 + 7$; this is to handle rapidly growing expressions (such as 2^n).

N1 Let x be an integer and y be a real number. Show that $x \le y$ iff $x \le \lfloor y \rfloor$.

N2 Let x and y be integers. Show that $x \le \lfloor n/y \rfloor$ iff $y \le \lfloor n/x \rfloor$.

N3 Show that $\lfloor n/x \rfloor = v$ if and only if $\left\lfloor \frac{n}{v+1} \right\rfloor < x \le \lfloor \frac{n}{v} \rfloor$.[13]

N4 Show that $\lfloor n/x^2 \rfloor = v$ if and only if $\left\lfloor \sqrt{\frac{n}{v+1}} \right\rfloor < x \le \left\lfloor \sqrt{\frac{n}{v}} \right\rfloor$.

N5 Let n, g and h be integers. Show that $\left\lfloor \frac{\lfloor \frac{n}{g} \rfloor}{h} \right\rfloor = \left\lfloor \frac{n}{gh} \right\rfloor$.

N6 Which among n, g and h can be non-integers (but still nonnegative real numbers) such that the previous statement still holds for all n, g, h? Show why this is so.

N7 Show that if $v \in \langle n \rangle$, then $\langle v \rangle \subset \langle n \rangle$.

N8 Show that $\lfloor \frac{n}{g} \rfloor \ne \lfloor \frac{n-1}{g} \rfloor$ if and only if n is a multiple of g. Prove it more formally than the "intuitive explanation" given above.

N9 Show that the set $\langle n \rangle_k := \left\{ \lfloor n/x^k \rfloor : x \in \{1, 2, \ldots, \lfloor n^{1/k} \rfloor\} \right\}$ has $\mathcal{O}(n^{1/(k+1)})$ elements. *Bonus*: Find a formula for the exact count.

N10 Fix the integer n. For every $x \in [1, n]$, define x' as $\lfloor n/x \rfloor$. Show that $x''' = x'$. Also, show that $x \in \langle n \rangle$ iff $x'' = x$.

N11 Show that if $\gcd(x, y) = 1$, then (x, y) is visible from the origin. [14]

N12 Given n, find the number of 3D lattice points in the cube $[-n, n]^3$ visible from the origin in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time. Don't reduce it to the $[1, n]^3$ case (or similar); rather, derive it by following section 3. Use the "*ignore the gcd condition*" trick. [15]

N13 Given n, show how to compute

$$\sum_{x=1}^{n} (n \bmod x)$$

in $\mathcal{O}(\sqrt{n})$ time.[16]

N14 Given n, show how to compute

$$\sum_{x=1}^{n} 2^{\lfloor n/x \rfloor + x}$$

in $\mathcal{O}(\sqrt{n} \log n)$ time.

[13] *Hint*: $a = \lfloor b \rfloor$ is the same as $a \le b < a + 1$ if a is an integer.

[14] *Hint*: Prove the contrapositive, i.e., that having a point inside the segment means that $\gcd(x, y) \ne 1$. To do so, let (x', y') be the closest lattice point to the origin inside the segment. Let $q = \lfloor x/x' \rfloor$ (or $\lfloor y/y' \rfloor$ if $x' = 0$), and consider the point $(x - qx', y - qy')$.

[15] *Hint*: $h(n) = (2n + 1)^3 - 1$.

[16] *Hint*: Use $n = x\lfloor n/x \rfloor + (n \bmod x)$.

N15 Find the sum of $|x - y|$ across all visible points in $[1, n] \times [1, n]$ in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time. Partial points for $\mathcal{O}(n^{2/3}(\log n)^{1/3})$.

N16 Find the sum of $|x-y-11|$ across all visible points in $[1, n] \times [1, n]$ in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time. Partial points for $\mathcal{O}(n^{2/3}(\log n)^{1/3})$.

N17 Find the sum of $(x^2 + x + 1)(y^2 + y + 1)$ across all visible points in $[1, n] \times [1, n]$ in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time. Partial points for $\mathcal{O}(n^{2/3}(\log n)^{1/3})$.

N18 Compute the following in $\mathcal{O}(n^{1/2})$ time:

$$\sum_{x=1}^{n} d(x)$$

where $d(x)$ is the number of divisors of x.

N19 Compute the following in $\mathcal{O}(n^{1/2})$ time:

$$\sum_{x=1}^{n} \sigma_k(x)$$

where $\sigma_k(x)$ is the sum of kth powers of divisors of x.

N20 In item **N19**, the constant in the big-$\mathcal{O}$ notation actually depends on k. Show that that constant can be $\mathcal{O}(k^3)$.

As a side note, there are ways to have that constant be $o(k^3)$, but it involves so many other tools (Bernoulli numbers, generating functions, FFT). Feel free to find it! Ask in Discord if you'd like some hints and/or discussion.

N21 Compute the following in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time:

$$\sum_{x=1}^{n} \sum_{y=1}^{n} \gcd(x, y).$$

Partial points for $\mathcal{O}(n^{2/3}(\log n)^{1/3})$.[17]

N22 Compute the following in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time:

$$\sum_{x=1}^{n} \sum_{y=1}^{n} \operatorname{lcm}(x, y)^2.$$

Partial points for $\mathcal{O}(n^{2/3}(\log n)^{1/3})$.[18]

N23 Find the number of squarefree numbers between 1 and n in $\mathcal{O}(n^{3/7})$ time. [19] [20]

N24 Show that

$$\sum_{1 \le g \le n} \nabla d(g) f(n/g) = \sum_{1 \le g \le n} \nabla f(g) d(n/g).$$

[17] *Hint*: Iterate through all values of $\gcd(x, y)$.

[18] *Hint*: Use the fact that $\operatorname{lcm}(x, y) \cdot \gcd(x, y) = xy$.

[19] *Hint*: Use the fact that every integer has a unique decomposition as $s^2 f$ with f squarefree.

[20] *Hint 2*: Iterate through s and f in the sum $\sum_{x=1}^{n} 1$ by setting $x = s^2 f$.

N25 Show that

$$\sum_{1 \le g \le n} \mu(g) \lfloor n/g \rfloor = 1.$$[21]

N26 Show that the following two are equivalent:

- For all n, $h(n) = \prod_{1 \le g \le n} f(n/g)^{a(g)}$.
- For all n, $\dfrac{h(n)}{h(n-1)} = \prod_{g \mid n} \left(\dfrac{f(n/g)}{f(n/g-1)} \right)^{a(g)}$.

State your assumptions on f, a and h.

N27 Suppose b and d are Dirichlet inverses, and suppose that:

- For all n, $h(n) = \prod_{1 \le g \le n} f(n/g)^{b(g)}$.

Find a statement equivalent to this that relates h, f and d, and show why they're equivalent. State your assumptions on f, b, d and h.

N28 Prove Theorem 3.2.

N29 Given n, $\phi(n)$ is defined as the number of integers in $[1, n]$ coprime to n. (ϕ is called **Euler's totient function**.) Show that

$$\phi(n) = \sum_{g \mid n} g \cdot \mu(n/g).$$[22]

N30 A complex number ω is an n**th root of unity** if $\omega^n = 1$. Show that if ω is an nth root of unity and also an mth root of unity, then it is also a $\gcd(m, n)$th root of unity.[23]

N31 A complex number ω is a **primitive nth root of unity** if it is an nth root of unity but not an mth root of unity for $m < n$. Show that if ω is a primitive nth root of unity and an mth root of unity, then $n \mid m$.[24]

N32 The nth **cyclotomic polynomial**, denoted $\Phi_n(x)$, is the unique monic polynomial whose roots are precisely the primitive nth roots of unity. Show that the degree of $\Phi_n(x)$ is $\phi(n)$, and that it is given explicitly by the formula

$$\Phi_n(x) = \prod_{g \mid n} (x^g - 1)^{\mu(n/g)}.$$[25]

(Because μ can have negative values, this formula doesn't even look like it results in a polynomial! For example, try expanding it for $n = 60$.)

N33 Show that if a and b are multiplicative, then the function

$$c(n) := \sum_{g \mid n} a(g) b(n/g)$$

is also multiplicative. (c is called the **Dirichlet convolution** of a and b and is usually denoted $a * b$.)

[21] *Hint*: Use problem **N24**.
[22] *Hint*: For every $x \in [1, n]$, $\gcd(x, n) \mid n$.
[23] *Hint*: Use Bézout's identity, that $\gcd(m, n) = am + bn$ for some integers a and b.
[24] *Hint*: Consider $\gcd(n, m)$. What happens if $n \nmid m$?
[25] *Hint*: Use problem **N27**.

N34 Show that if a and b are Dirichlet inverses and a is multiplicative, then b is also multiplicative.

N35 Show that the Möbius function and the "constant" function 1 are Dirichlet inverses.

N36 Prove Theorem 5.4. You may assume Theorem 5.3.

N37 Show that $f(n)$ can be computed in $\mathcal{O}(n^{2/3}(\log n)^{1/3})$ time from the inverse form

$$h(n) = \sum_{1 \le g \le n} \nabla d(g) f(n/g),$$

assuming that $h(n')$ and $d(n')$ can be computed in $\mathcal{O}(1)$.

N38 Show that $f(n)$ can be computed in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time from the inverse form

$$h(n) = \sum_{1 \le g \le n} f(n/g),$$

assuming that $h(n')$ can be computed in $\mathcal{O}(1)$, and that ∇h is multiplicative.

N39 Show that $f(n)$ can be computed in $\mathcal{O}(n^{2/3}(\log \log n)^{1/3})$ time from the inverse form

$$h(n) = \sum_{1 \le g \le n} \nabla d(g) f(n/g),$$

assuming that $h(n')$ and $d(n')$ can be computed in $\mathcal{O}(1)$, and that ∇d is multiplicative.

N40 Show how to compute

$$\sum_{x \ge 1} \nabla d(x) h(n/x + 1/3)$$

in $\mathcal{O}(\sqrt{n})$ time, assuming $h(n')$ and $d(n')$ can be computed in $\mathcal{O}(1)$ for any n'.

N41 Show how to compute the sum

$$\sum_{x \ge 1} \nabla d(x) \lfloor n/x \rfloor \lfloor m/x \rfloor$$

in $\mathcal{O}(\max(n, m)^{1-\varepsilon})$ time for some $\varepsilon > 0$, assuming $d(x)$ can be computed in $\mathcal{O}(1)$ time. How low can you get the time complexity of this?

N42 Show how to compute $f(n, m)$ from the relationship

$$h(n, m) = \sum_{x \ge 1} \nabla d(x) f(\lfloor n/x \rfloor, \lfloor m/x \rfloor)$$

in $\mathcal{O}(\max(n, m)^{1-\varepsilon})$ time for some $\varepsilon > 0$, assuming $h(n', m')$ and $d(x)$ can be computed in $\mathcal{O}(1)$ time. How low can you get the time complexity of this?

N43 Given a, b, c and d, find the number of lattice points in $[a, b] \times [c, d]$ visible from the origin in $\mathcal{O}(\max(|a|, |b|, |c|, |d|)^{1-\varepsilon})$ time for some $\varepsilon > 0$.

6.2 Coding problems

S1 Counting fractions: https://projecteuler.net/problem=72 (also try to solve it for $d \leq 10^{11}$).

S2 Counting fractions in a range: https://projecteuler.net/problem=73 (also try to solve it for $d \leq 10^{11}$).

S3 Bounded Sequences: https://projecteuler.net/problem=319

S4 Hexagonal orchards: https://projecteuler.net/problem=351

S5 Squarefree factors: https://projecteuler.net/problem=362

S6 Distinct Lines: https://projecteuler.net/problem=388

S7 Sum of squares of divisors: https://projecteuler.net/problem=401

S8 Titanic sets: https://projecteuler.net/problem=415

S9 Necklace of circles: https://projecteuler.net/problem=428 (try to solve in sublinear time)

S10 Totient sum: https://projecteuler.net/problem=432

S11 Sum of sum of divisors: https://projecteuler.net/problem=439

S12 Lattice Quadrilaterals: https://projecteuler.net/problem=453

S13 Polar polygons: https://projecteuler.net/problem=465

S14 Gcd sum: https://projecteuler.net/problem=625

S15 Lots of Cookies: https://www.hackerrank.com/contests/noi-ph-practice-page/challenges/lots-of-cookies

S16 Tagolympics: https://www.hackerrank.com/contests/up-acm-algolympics/challenges/2017-tagolympics

S17 Laser Beam: https://www.hackerrank.com/challenges/laser-beam

A Appendix: Proving the Möbius inversion formula

Rather than prove just Theorem 5.1, we'll attempt to prove the more general Theorem 5.3. To complete the proof, one just has to show that the Möbius function and the "constant" function 1 are Dirichlet inverses, and this is asked in the exercises.

Given two functions a and b, their **Dirichlet convolution**, $a * b$, is the function defined as

$$(a * b)(n) := \sum_{g|n} a(g)b(n/g).$$

It turns out that $*$ well-behaved, and it satisfies a lot of the common properties you'd expect from multiplication:[26]

- It is commutative, i.e., $a * b = b * a$.
- It is associative, i.e., $a * (b * c) = (a * b) * c$.
- It has an identity, namely the function ε we defined earlier, so $a * \varepsilon = a$ for any a.
- Every function a with $a(1) \neq 0$ has a multiplicative inverse, i.e., there exists a b such that $a * b = \varepsilon$. The term *Dirichlet inverse* we defined earlier is actually just this notion of multiplicative inverse.

These are quite easy to prove, so I'll leave them to you. (I guess the hardest one to prove is the existence of inverses. For that, I suggest defining b by induction, through increasing n. For this, the assumption that $a(1) \neq 0$ is important.)

We can shed more light on why the operator $*$ is so well-behaved once we discover its correspondence with the so-called *Dirichlet series*.

The **Dirichlet series** of the function a, denoted $\mathcal{D}_a$, is defined as

$$\mathcal{D}_a(s) := \sum_{n \geq 1} \frac{a(n)}{n^s} = \frac{a(1)}{1^s} + \frac{a(2)}{2^s} + \frac{a(3)}{3^s} + \dots$$

Obviously, you could be worrying about convergence issues and all that, but let's sidestep all that for now; just know that there are ways to handle it.[27]

Now, consider what happens when we multiply $\mathcal{D}_a(s)$ and $\mathcal{D}_b(s)$. What is the coefficient of $\frac{1}{n^s}$? Try it out on paper! The answer is amazingly $(a * b)(n)$, the Dirichlet convolution of a and b! This gives us the following:

$$\mathcal{D}_a(s)\mathcal{D}_b(s) = \mathcal{D}_{a*b}(s).$$

Right away, this explains why $*$ is commutative and associative; it follows from the commutativity and associativity of multiplying series. The existence of the identity ε follows from the fact that $\mathcal{D}_\varepsilon(s) = 1$, and the existence of inverses follows from the fact that $\frac{1}{\mathcal{D}_a(s)}$ also has a Dirichlet series expansion when $a(1) \neq 0$ (although this last one is harder to prove, haha).

Using Dirichlet series, we can now restate Theorem 5.3:

[26] This means that functions (at least those with $a(1) \neq 0$) form an *abelian group* with respect to $*$. In fact, with convolution and normal pointwise addition $(a + b)(n) = a(n) + b(n)$, they form a *ring*.

[27] One way would be to treat this as a *formal* series, i.e., a series only on form, where s doesn't really stand for a number.

Theorem A.1. Suppose $\mathcal{D}_{b*d}(s) = 1$. Then for any f and h, the following two are equivalent:

- $\mathcal{D}_h(s) = \mathcal{D}_{f*b}(s)$.
- $\mathcal{D}_f(s) = \mathcal{D}_{h*d}(s)$.

But now, after replacing any $\mathcal{D}_{a*b}(s)$ with $\mathcal{D}_a(s)\mathcal{D}_b(s)$, it becomes quite self-evident!

Proof. Let's prove the forward direction:

$$\begin{aligned}
& \mathcal{D}_h(s) = \mathcal{D}_{f*b}(s) \\
\implies \quad & \mathcal{D}_h(s) = \mathcal{D}_f(s) \cdot \mathcal{D}_b(s) \\
\implies \quad & \mathcal{D}_h(s) \cdot \mathcal{D}_d(s) = \mathcal{D}_f(s) \cdot \mathcal{D}_b(s) \cdot \mathcal{D}_d(s) \\
\implies \quad & \mathcal{D}_{h*d}(s) = \mathcal{D}_f(s) \cdot \mathcal{D}_{b*d}(s) \\
\implies \quad & \mathcal{D}_{h*d}(s) = \mathcal{D}_f(s) \cdot 1 \\
\implies \quad & \mathcal{D}_{h*d}(s) = \mathcal{D}_f(s).
\end{aligned}$$

The backward direction is very similar. □

Actually, this isn't a complete proof; I've glossed over so many details! This is mostly just to give you an understanding of why the theorem is true, and in some sense, "obvious", when looked at the right way. I will just leave it to you to fill the gaps!