ECS 20 – Fall 2021 – P. Rogaway Number Theory

We have already done some number theory—in particular, we've talked about the **Peano Axioms**, **The Principle of Mathematical Induction**, the **Fundamental Theorem of Arithmetic**. We described how to count the number of divisors of a factored number. I mumbled something about how to see if a number in factored form is a square. There are a few more odds and ends about number theory I want to make sure that you have seen, and, for fun, I would like to do just a touch of public-key cryptography.

Let's start with:

Well-Ordering Principle: Any nonempty set of the **natural numbers** has a least element.

You can prove this by **induction**. Let S be a subset of natural numbers with **no** least element. Let $P(n)$ be the proposition that n is not in S.
Basis: $P(0)$. Note that $0 \notin S$ since 0 is the smallest natural number, so if it were in S, then it would have to be the smallest element of S.
Inductive step: Suppose $P(0)$ and $P(1)$ and … and $P(n)$, meaning that S contains none of 0, 1, …, n. Then it can't contain n+1 either, for if it die, that would be the smallest element of S, and S has no smallest element.
The conclusion is $(\forall n)P(n)$, which means that S is empty, a contradiction.

Often one can use the well-ordering principle **in place of** induction. (In fact, some books claim—incorrectly—that the well-ordering principle and the principle of mathematical induction are equivalent. The technical mean of that should be that you could use each to establish the other within the axioms of PA. While the axiom of induction is enough to show well-ordering, the converse is **not** true.) Let's think back to the result that you could dispense any number of envelopes $n \geq 44$ with packets of 5 and 12 envelopes. We did this by induction. Let's try to do it by well-ordering.

Suppose for contradiction that we can't do with the claim say. Let C be the set of all counterexamples to our claim—a set of numbers, each at least 44, where $n \in C$ means that you can't write n as $12a + 5b$ for any nonnegative a, b. By the well-ordering principle, C has a least element. Call it c. We know that $c \geq 60$ because we checked all the numbers between 44 and 59 inclusive and you could write them as $12a+5b$ for some a, b. Now consider $c – 5$. We know that $c – 5 \geq 55$ and c-5 is not a counterexample to our claim, so you can write it as $12a+5b$. But then you can write c itself as $12a + 5(b+1)$.

The Division Theorem. If n is any integer and d is a positive integer, there exist unique integers q and r such that $n = dq + r$ and $0 \leq r < d$.

One can prove the Division Theorem from the well-ordering principle. The trick is to define $R = \{n – iq\colon i \in \mathrm{N} \text{ and } n – iq > 0\}$. Then R is a nonempty subset of the natural, so it has a least element. That least element is the r, and the i that goes with it is the d, and you can establish that they're unique.

The remainder when you divide n by d has a nice notation in Python 3 – it's $q = n$ // d (the "floor division operator"). The remainder has a nice notation in lots of languages – it's $r = n$ % d or $r = n \bmod d$.

3 // 2 = 1	-3//2 = -2	3%2 = 1	-3%2 = 1
2 // 3 = 0	-2//3 = -1	2%3 = 0	-2%3 = 1

The name "floor division" provides an easy way to understand (and to compute) stuff like -3/2. Think of computing the quotient in the real, and then go to the left—take the floor—until you hit an integer.

Notation: $d \mid n$ if there exists i such that $di = n$.
Alternative, if as $n \bmod d = 0$

$\mathbb{Z}_n = \{0,1,\ldots n\text{-}1\}$ with an operation of addition modulo n.
This is called "the group of integers modulo n"

Lots of variant notations, which actually correspond to **different ways of thinking about mod.** Recall the circle of n points, moving right for increment and left for decrement, as a way of thinking of arithmetic in the world mod n. Conceptualizations:

23 mod 5 = 3	as a **binary operator**
$23 \equiv 3 \pmod 5$	23 is in the **same equivalence class** as 3 with respect to $a \equiv b$ true when $5 \mid a - b$
$1+4 = 0$ in $\mathbb{Z}_5$	We are working in **the group** $\mathbb{Z}_5$. I like this way.

Greatest common divisors

For a and b positive numbers, let's define
gcd(a, b) = the maximal integer d such that $d \mid a$ and $d \mid b$.

Well defined because d is at most min(a, b)

Define gcd(a,0) = gcd(0, a) = a for any nonzero a.

Can you figure out the gcd of a and b when they're in factored form? Sure. It's the product of p^i values where p occurs in the factorization of both a and b and i is the smaller of the two exponents. It is 1 if there are no common primes.

Of course gcd(a, b) = gcd(b, a)

Claim: gcd(a, b) = gcd($a - b$, b) = gcd(a mod b, b)
This gives rise to an efficient algorithm to find the gcd.
It's called **Euclid's algorithm.**

Let's prove Euclid's algorithm.
If $d \mid a$ (so $a = i\ d$) and $d \mid b$ (so $b = j\ d$) then
$a - b = i\ d - \mathrm{j}\ \mathrm{d} = d\ (i - j)$ whence $d \mid a - b$.
So all the divisors of a and b are divisors of $a - b$ and b.
Similarly, all the divisors of $a - b$ and b are divisors of a and b.
Since they have the exact same set of common divisors, their gcd's are the same.

Example: Compute gcd(360, 1000).
gcd(1000,360) = gcd(360,280) = gcd(280,80) = gcd(80,40) = gcd(40,0)=40.

If you do some extra bookkeeping when you compute the gcd you can find, when you compute gcd(a, b), two numbers x and y such that
$ax + by$ = gcd(a, b)

That the algorithm finds them tell you that they always exist ;-)

Theorem: Given numbers a and b, not both 0, we can find integers
x and y such that $ax + by$ = gcd(a ,b).

This is useful!

Numbers a and b are said to be relatively prime if gcd(a, b)=1. This is the same as saying that they have no common prime factor.

Note that if p is prime and a is in [1..p-1] then gcd(a, p)=1 – primes are relatively prime to everything less than them.

Suppose gcd(a, n) = 1.
Then the theorem above promises an x, y such that $ax + ny = 1$. Suppose we're doing this is $\mathbb{Z}_n$ (that is, take "mod n" of both sides). In that group, n=0, so ny=0, so $ax = 1$. In other words, x is the inverse to a. That is, a has a multiplicative inverse mod n.

Thus we have: Every element relatively prime to n has a multiplicative inverse in $\mathbb{Z}_n$.

Eg: In $\mathbb{Z}_{10}$ the numbers 1, 3, 7, 9 have multiplicative inverses. Make a multiplication table for them. Note closure: if a and b are both relatively

Eg: In the world of integer operations mod 2^{32}, all the odd numbers have multiplicative inverses.

$\mathbb{Z}_n^* = \{a \in [1..n]: \gcd(n, a)=1\}$ is a group.

Example: Make a multiplication table for $\mathbb{Z}_{10}^*$

A general fact about finite groups is that, for any group G, $a^{|G|} = 1.$
This is called **Lagrange's Theorem.** I'm not going to prove this, but you will certainly prove it if you take any modern algebra class (a class that covers group theory and the like).

A particularly interesting group is $\mathbb{Z}_p^* = [1..p\text{-}1]$ where p is a prime. By Lagrange's theorem, $a^{p-1} = 1$.
This actually provides a test for primality.
If $a^{p-1} \neq 1 \pmod{p}$ then you **know** that p is not prime. This is "Fermat's test for primality".

We don't know for sure that p is prime must because $a^{p-1} = 1$. Yet this is close to being true in the sense that composite numbers for which $2^{p-1} = 1$

(mod p) are rare (although there are infinitely many). There are even nasty number, **Fermat pseudoprimes**, where $a^{p-1} = 1$ (mod p) for all $a \neq 1$.

Another interesting fact about $\mathbb{Z}_p^*$ is that it's **cyclic**: there will be an element g in this group such that $\mathbb{Z}_p^* = \{g^0, g^1, \ldots, g^{p-2}\}$. In words, you can create the entire group from g by just starting with 1 and repeatedly multiplying by g.

Do an example to figure out what are the generators of $\mathbb{Z}_5^*$

If you have g and p and i, it is easy to efficiently compute g^{i-2} . You don't have to repeatedly multiply g by itself i-1 times; you can do much better. We have already illustrated much of the idea when we compute 2^{100} mod 10.

While it is easy to compute g^i mod p from g, i, and p, nobody knows an efficient algorithm for computing i from g, p, and g^i mod p . Computing it is known as the **discrete log problem**. There is a discrete log problem for any finite cyclic group. In the group we've been talking about, we think it is **hard**. That is, we think there is no efficient way to solve this problem. It is an example of supposed **one-way-function**. Something that is easy to compute in one direction, but apparently hard to compute in the other direction.

We are now ready to do a little **cryptography**!

Alice wants to communicate with Bob. They first meet and agree on a great big prime number p and a generator g for $\mathbb{Z}_p^*$. To generate a big prime number they can generate a random big number and test if it is prime. Testing if $2^{p-1} = 1$ (mod p) is actually enough to give high confidence that the random p is prime. It is also easy to test if a given value, say g=2, is a generator. If its not, Alice and Bob could just choose a different prime. The number 2 will often be a generator for a random large prime p.

Alice computes a random a in [0..p-2] and send Bob $A = g^a$. Bob compute a random b in [0..p-2] and send Bob $B = g^b$. Everything is done mod p.

$$\text{Alice} \xrightarrow{A = g^a.} \text{Bob}$$

$$\xleftarrow{B = g^b}$$

At this point Alice can compute $B^a = (g^b)^a = g^{ab}$
At this point Bob can compute $A^b = (g^a)^b = g^{ab}$
So they share the same “secret”.
On the other hand, the adversary E watching all of this knows g, p, A, and B, but, try as it may, there is no clear way for it to compute from this g^{ab} .
It can try things like AB, but that’s g^{a+b}, not g^{ab}.
It could try to compute a and b, but that’s the discrete log problem, which we think to be computationally intractable.

Now that Alice and Bob share a secret $K = g^{ab}$ they can use it to send messages to one another. For example, Alice could send a message M by transmitting M xor $H(K)$ for some (non-secret) “hash function” H.
If Alice and Bob have only a bit m they want to communicate, they could try something like m xor lsb(K), where lsb(x) denotes the rightmost (least-significant) bit of K.
