CSCI 2200

Foundations of Computer Science

Lecture 10: A Taste of Number Theory

CSCI 2200

Foundations of Computer Science

Lecture 10: A Taste of Number Theory

"Baba Yetu" performed by Ron Ragin & the Stanford Talisman, composed by Christopher Tin for *Civilization IV*

Today's tasks

- Gentle reminder: Exam 2 on Wed. 2/26
- HW questions
- Divisibility & GCD
- Modular arithmetic
- Primes & cryptography

But first: What is number theory?

- “Number theory is the queen of mathematics.”
  --Carl Friedrich Gauss
- Number theory is the branch of pure math that is mostly concerned about integers and elements that can be constructed with them (e.g. rational numbers).
  - Basically, it’s Discrete Math++
  - A <u>lot</u> of number theory is concerned with prime numbers.
  - One of the Millenium Prize Problems (the Riemann-Zeta Hypothesis) is intimately tied to the distribution of primes.

Divisibility, quotients, & remainders

- As previously discussed, a is divisible by d means that there is some $k \in \mathbb{Z}$ such that $a = kd$.
  - We often write $d \mid a$, read as "d divides a."
- What if $d \nmid a$ (i.e. a is not divisible by d)?
  - Then $a = kd + r$, where r is the remainder.
  - With a and d fixed, how many integer solutions are there?
    - Infinite.
  - But what if we insist that $0 \leq r < d$?
    - There is <u>exactly one</u> solution!

The Quotient-Remainder Theorem

- Given $a \in \mathbb{Z}$ and $d \in \mathbb{N}$, then there is exactly one pair of integers (k, r) that satisifies $a = kd + r$ and $0 \leq r < d$.
- Put another way: When you divide an integer by a positive integer, there is <u>one</u> quotient and <u>one</u> remainder. Just like in grade school.

Quick divisibility facts

- $\forall d \in \mathbb{N}, d \mid 0$
- If $d \mid n$ and $e \mid n$, then $de \mid n$.
- If $d \mid m$ and $m \mid n$, then $d \mid n$.
- If $d \mid n$ and $d \mid m$, then $d \mid (m + n)$.
- If $d \mid (m + n)$ and $d \mid m$, then $d \mid n$.
- If $d \mid n$, then $\forall k \in \mathbb{N}, kd \mid kn$.

Greatest common divisor

- What are the divisors of 30?
  - {1,2,3,5,6,15,30}
- What are the divisors of 42?
  - {1,2,3,6,7,14,21,42}
- 6 is the largest number in both lists, so gcd(30,42) = 6.

- How can we calculate this in general?

GCD algorithms

- The obvious approach: Generate both of those lists. How long will that take?
  - If you check all of the numbers from 1 to n, is that $O(n)$?
    - No, because we define runtime in terms of the length of the input. If it takes b bits to write n, then $n \approx 2^b$, which means we've got exponential runtime.
    - Even if you stop at $\sqrt{n}$, that's still exponential.
- We need another approach.

A surprising fact

- If $n \geq m$, then $\mathbf{gcd}(n, m) = \mathbf{gcd}(m, r)$, where r is the remainder of $n \div m$.
  - Proof: $n = km + r$ - by the Q-R theorem, r is unique.
  - Let $D = \mathbf{gcd}(n, m)$ and $d = \mathbf{gcd}(m, r)$.
  - Since $d \mid m$ and $d \mid r$, $d \mid n$ as well. But since D is the greatest common divisor of n and m, $d \leq D$.
  - Observe that $r = n - km$. Since $D \mid n$ and $D \mid m$, then $D \mid r$ as well. But since d is the greatest common divisor of m and r, $D \leq d$.
  - $d \leq D \ \wedge \ D \leq d \leftrightarrow d = D \ \blacksquare$

Euclid's GCD Algorithm

```
def gcd(n,m):     # assume n >= m >= 1
    if m == 1:
        return 1
    return gcd(m,n%m)
```

This algorithm is <u>linear</u> in the number of bits of the input.

GCD facts

- When $\gcd(m, n) = 1$, we say that m and n are *relatively prime*.
- If $\gcd(a, b) = 1$ and $\gcd(a, c) = 1$, then $\gcd(a, bc) = 1$.
- If $d \mid mn$ and $\gcd(d, m) = 1$, then $d \mid n$.
- $\forall k \in \mathbb{N}, \gcd(km, kn) = k \cdot \gcd(m, n)$
- If $d \mid m$ and $d \mid n$, then $d \mid \gcd(m, n)$.

Modular arithmetic

Calculations on a clock

- It's 9am. What time will it be 6 hours from now?
  - 3pm. So 9 + 6 = 3?
  - Yes! At least when we're working *mod 12*.
- The overall idea is that, when working mod d, the only integers that exist are $0, 1, \ldots, d-1$.
  - We'll say that $a \equiv b \pmod{d} \leftrightarrow d \mid (a-b)$.

Modular arithmetic identities

- If $a \equiv b \ (mod\ d)$ and $r \equiv s \ (mod\ d)$, then:
  - $ar \equiv bs \ (mod\ d)$
  - $a + r \equiv b + s \ (mod\ d)$
  - $\forall k \in \mathbb{N}, a^k \equiv b^k \ (mod\ d)$

- In other words, addition and multiplication work in modular equations just like regular ones.

Silly math tricks

- What is the last digit of 7^{2025}?
  - Observe that $7^2 = 49 \equiv -1 \ (mod\ 10)$.
  - Observe that $2025 = 2 \times 1012 + 1$.
  - Observe that $7^{2025} = (7^2)^{1012} \cdot 7$.
  - $(7^2)^{1012} \cdot 7 \ (mod\ 10) \equiv (-1)^{1012} \cdot 7 \ (mod\ 10) \equiv 1 \cdot 7 \ (mod\ 10)$

Modular division

- $15 \cdot 6 \equiv 13 \cdot 6 \ (mod\ 12)$
  - But $15 \not\equiv 13 \ (mod\ 12)$!
- $15 \cdot 6 \equiv 2 \cdot 6 \ (mod\ 13)$
  - And $15 \equiv 2 \ (mod\ 13)$.

- The rule is: If $ac \equiv bc \ (mod\ d)$ and $\gcd(c, d) = 1$, then $a \equiv b \ (mod\ d)$.
  - If $\gcd(c, d) \neq 1$, we cannot draw a conclusion.
  - Note that if d is prime, division just works.

Multiplicative inverses

- If $3 \times n = 1$, what is n?
  - What if I insist that $n \in \mathbb{N}$?
  - There's no such integer!
- But… what if we were working mod 7?
  - $3 \times 5 = 15 \equiv 1\ (mod\ 7)$
- Any time the modulus is prime, then every value will have a multiplicative inverse, and it is easy to find.

Fermat's Little Theorem

- If p is prime, then $\forall k, k^p \equiv k \ (mod\ p)$.
  - Proof is sketched out in your textbook.
  - Immediate corollary: As long as $k \not\equiv 0$, then $k^{p-1} \equiv 1 \ (mod\ p)$, and the multiplicative inverse of k is k^{p-2}.

- But, if the modulus m is composite, then:
  - Not every number has a multiplicative inverse, and...
  - ... finding one depends on factoring m, which is a very time-consuming problem.

RSA public-key cryptography

Cryptography: General idea

- Send a private message from A to B assuming that a third party E is eavesdropping on the transmission.

- The goal is to mathematically transform the message in a way that B can undo but E cannot.

- One approach is to agree upon a bitmask ahead of time – you just toggle a set of randomly chosen bits, and then no one can decode it without the bitmask/password.
    - But you have to agree ahead of time – inconvenient.

Public key cryptography

- B publishes a “lock” that anyone can use to encrypt a message.
- B maintains the “key” to this lock, and keeps it secret from everyone.
- For this to work, the “locking” process needs to be difficult to reverse-engineer without the key.
- What might this “lock” and “key” mechanism look like?

Key idea: The power of 1

- If you raise a value to the power of 1, what happens?
  - Right, nothing.
  - So what if you have two values that *multiply together to 1*?
  - And what if you use those values as *exponents*?

RSA, briefly

- Since our plaintext message is just bits, we can represent it as a single number, p.
- B publishes an *encryption exponent* e and a modulus m. He keeps the *decryption exponent* d secret.
- To encrypt a message, A simply calculates the ciphertext as $c = p^e \ (mod\ m)$.
- To decrypt the message, B calculates $c^d \equiv (p^e)^d \ (mod\ m)$.
  - What does this tell you about e and d?
  - $ed \equiv 1$ ... but not $(mod\ m)$!

The right modulus

- If our modulus m were prime, then (per Fermat) it would be trivially easy for E to calculate d from e and m.
- If m has lots of small factors, then it is relatively easy to break it down to a manageable size quickly.
- The ideal case turns out to be where $m = ab$, where a and b are both LARGE prime numbers (with a few other conditions attached). This makes factoring m very challenging, and thus makes it very hard to figure out d from e and m.

A toy example

- Let $a = 11, b = 19$. Then $m = 209$.
- In order for there to be an inverse, we need to select e to be relatively prime to $(11 - 1)(19 - 1) = 180$. Let's use 7.
- d becomes the multiplicative inverse of 7 *mod 180*. This can be calculated quickly using a variant of Euclid's GCD algorithm. It turns out to be 179 here.
- So B publishes the *public key* (the "lock"): $(209, 7)$

A toy example

- $m = 209, \mathrm{e} = 7$ (public) $d = 179$ (secret)
- A wants to send the message: 50.
- They calculate $50^7 \ (mod\ 209) \equiv 107$, and send 107 as the encrypted message.
  - If this is intercepted, there's isn't much E can do with it.
- B receives this and calculates $107^{179} \ (mod\ 209) \equiv 50$.

Important ideas we haven't discussed

- Why was our inverse mod 180 instead of mod 209?
  - More number theory. See: Euler's theorem and Euler's totient function.
- What is it about factoring that makes it hard, and how do we guarantee that?
  - Not sure, and <u>we don't</u>. We just don't have any polynomial-time (non-quantum) algorithms for doing so.
- How on Earth are we calculating numbers like 107^{179}?
  - Well, $179 = 128 + 32 + 16 + 2 + 1$...

Questions?