Number Theory

> *Mathematics is the queen of sciences and number theory is the queen of mathematics.*
>
> – Carl Friedrich Gauss

Number Theory

Mathematics is the queen of sciences and number theory is the queen of mathematics.

– Carl Friedrich Gauss

But why is it computer science?

- It turns out to be critical for cryptography!

Number Theory

Mathematics is the queen of sciences and number theory is the queen of mathematics.

– Carl Friedrich Gauss

But why is it computer science?

- It turns out to be critical for cryptography!

  No one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems unlikely that anyone will do so for many years.

  – G.H. Hardy

Division

For $a, b \in Z$, $a \neq 0$, a *divides* b if there is some $c \in Z$ such that $b = ac$.

- Notation: $a \mid b$
- Examples: $3 \mid 9$, $3 \not\mid 7$

If $a \mid b$, then a is a *factor* of b, b is a *multiple* of a.

Theorem 1: If $a, b, c \in Z$, then

1. if $a \mid b$ and $a \mid c$ then $a \mid (b + c)$.
2. If $a \mid b$ then $a \mid (bc)$
3. If $a \mid b$ and $b \mid c$ then $a \mid c$ (divisibility is transitive).

Proof: How do you prove this? Use the definition!

Division

For $a, b \in Z$, $a \neq 0$, a *divides* b if there is some $c \in Z$ such that $b = ac$.

- Notation: $a \mid b$
- Examples: $3 \mid 9$, $3 \not\mid 7$

If $a \mid b$, then a is a *factor* of b, b is a *multiple* of a.

Theorem 1: If $a, b, c \in Z$, then

1. if $a \mid b$ and $a \mid c$ then $a \mid (b + c)$.
2. If $a \mid b$ then $a \mid (bc)$
3. If $a \mid b$ and $b \mid c$ then $a \mid c$ (divisibility is transitive).

Proof: How do you prove this? Use the definition!

- E.g., if $a \mid b$ and $a \mid c$, then, for some d_1 and d_2,

$$b = ad_1 \text{ and } \ c = ad_2.$$

- That means $b + c = a(d_1 + d_2)$
- So $a \mid (b + c)$.

Other parts: homework.

Division

For $a, b \in Z$, $a \neq 0$, a *divides* b if there is some $c \in Z$ such that $b = ac$.

- Notation: $a \mid b$
- Examples: $3 \mid 9$, $3 \not\mid 7$

If $a \mid b$, then a is a *factor* of b, b is a *multiple* of a.

Theorem 1: If $a, b, c \in Z$, then

1. if $a \mid b$ and $a \mid c$ then $a \mid (b + c)$.
2. If $a \mid b$ then $a \mid (bc)$
3. If $a \mid b$ and $b \mid c$ then $a \mid c$ (divisibility is transitive).

Proof: How do you prove this? Use the definition!

- E.g., if $a \mid b$ and $a \mid c$, then, for some d_1 and d_2,

$$b = ad_1 \text{ and } \ c = ad_2.$$

- That means $b + c = a(d_1 + d_2)$
- So $a \mid (b + c)$.

Other parts: homework.

Corollary 1: If $a \mid b$ and $a \mid c$, then $a \mid (mb + nc)$ for all $m, n \in Z$.

The division algorithm

Theorem 2: For $a \in Z$ and $d \in N$, $d > 0$, there exist unique $q, r \in Z$ such that $a = q \cdot d + r$ and $0 \leq r < d$.

- r is the remainder when a is divided by d

Notation: $r \equiv a \pmod{d}$; $a \bmod d = r$

The division algorithm

Theorem 2: For $a \in Z$ and $d \in N$, $d > 0$, there exist unique $q, r \in Z$ such that $a = q \cdot d + r$ and $0 \leq r < d$.

- r is the remainder when a is divided by d

Notation: $r \equiv a \pmod{d}$; $a \bmod d = r$

Examples:

- Dividing 101 by 11 gives a quotient of 9 and a remainder of 2, so $101 \equiv 2 \pmod{11}$ and $101 \bmod 11 = 2$.
- Dividing 18 by 6 gives a quotient of 3 and a remainder of 0, so $18 \equiv 0 \pmod{6}$ and $18 \bmod 6 = 0$.

The division algorithm

Theorem 2: For $a \in Z$ and $d \in N$, $d > 0$, there exist unique $q, r \in Z$ such that $a = q \cdot d + r$ and $0 \leq r < d$.

- r is the remainder when a is divided by d

Notation: $r \equiv a \pmod{d}$; $a \bmod d = r$

Examples:

- Dividing 101 by 11 gives a quotient of 9 and a remainder of 2, so $101 \equiv 2 \pmod{11}$ and $101 \bmod 11 = 2$.
- Dividing 18 by 6 gives a quotient of 3 and a remainder of 0, so $18 \equiv 0 \pmod{6}$ and $18 \bmod 6 = 0$.

Proof: The proof is constructive: We define q, r explicitly:
Let $q = \lfloor a/d \rfloor$ and define $r = a - q \cdot d$.

- $\lfloor a/d \rfloor$ is the largest integer $\leq a/d$
- it's what you get when you divide a by d, ignoring the remainder; r is the remainder

Now use algebra:

- So $a = q \cdot d + r$. Clearly $q \in Z$. But why is $0 \leq r < d$?
  - By definition of $\lfloor \cdot \rfloor$, since $q = \lfloor a/d \rfloor$, we have $q \leq a/d < q + 1$.
  - Since $d > 0$, multiplying through by d, we have $qd \leq a < qd + d$.
  - subtracting qd, we have $0 \leq a - qd = r < d$

But why are q and r unique?

Now use algebra:

- So $a = q \cdot d + r$. Clearly $q \in Z$. But why is $0 \leq r < d$?
  - By definition of $\lfloor \cdot \rfloor$, since $q = \lfloor a/d \rfloor$, we have $q \leq a/d < q + 1$.
  - Since $d > 0$, multiplying through by d, we have $qd \leq a < qd + d$.
  - subtracting qd, we have $0 \leq a - qd = r < d$

But why are q and r unique?

- Suppose $q \cdot d + r = q' \cdot d + r'$ with $q', r' \in Z$ and $0 \leq r' < d$.
- Then $(q' - q)d = (r - r')$ with $-d < r - r' < d$.
- The lhs is divisible by d so $r = r'$ and we're done.

Primes

- If $p \in N$, $p > 1$ is *prime* if its only positive factors are 1 and p.
- $n \in N$ is *composite* if $n > 1$ and n is not prime.
  - If n is composite then $a \mid n$ for some $a \in N$ with $1 < a < n$
  - Can assume that $a \leq \sqrt{n}$.
    - **Proof:** If $a|n$, then $n = ac$ for some c. If $a \leq \sqrt{n}$, then we are done. If $a > \sqrt{n}$, then we must have $c < \sqrt{n}$. For if $c \geq \sqrt{n}$, then $ac > \sqrt{n}\sqrt{n} = n$, a contradiction. Thus, $c < \sqrt{n}$, and $c|n$, so n has a factor that is at most $\sqrt{n}$.

Primes: $2, 3, 5, 7, 11, 13, \ldots$
Composites: $4, 6, 8, 9, \ldots$

Primality testing

How can we tell if $n \in N$ is prime?

Primality testing

How can we tell if $n \in N$ is prime?

The naive approach: check if $k \mid n$ for every $1 < k < n$.

- But at least 10^{m-1} numbers are $\leq n$, if n has m digits
  - 1000 numbers less than 1000 (a 4-digit number)
  - 1,000,000 less than 1,000,000 (a 7-digit number)

So the algorithm is *exponential time*!

We can do a little better

- Skip the even numbers
- That saves a factor of 2 $\longrightarrow$ not good enough
- Try only primes (Sieve of Eratosthenes)
  - Still doesn't help much

Primality testing

How can we tell if $n \in N$ is prime?

The naive approach: check if $k \mid n$ for every $1 < k < n$.

- But at least 10^{m-1} numbers are $\leq n$, if n has m digits
  - 1000 numbers less than 1000 (a 4-digit number)
  - 1,000,000 less than 1,000,000 (a 7-digit number)

So the algorithm is *exponential time*!

We can do a little better

- Skip the even numbers
- That saves a factor of 2 $\longrightarrow$ not good enough
- Try only primes (Sieve of Eratosthenes)
  - Still doesn't help much

We can do much better:

- There is a polynomial time *randomized* algorithm
  - We will discuss this when we talk about probability
- In 2002, Agarwal, Saxena, and Kayal gave a (nonprobabilistic) polynomial time algorithm
  - Saxena and Kayal were undergrads in 2002!

The Fundamental Theorem of Arithmetic

Theorem 3: Every natural number $n > 1$ can be uniquely represented as a product of primes, written in nondecreasing size.

- Examples: $54 = 2 \cdot 3^3$, $100 = 2^2 \cdot 5^2$, $15 = 3 \cdot 5$.

Proving that that n can be written as a product of primes is easy (by strong induction):

- Base case: 2 is the product of primes (just 2)
- Inductive step: If $n > 2$ is prime, we are done. If not, $n = ab$.
  - Must have $a < n$, $b < n$.
  - By I.H., both a and b can be written as a product of primes
  - So n is product of primes

Proving uniqueness is harder.

- We'll do that in a few days . . .

An Algorithm for Prime Factorization

Fact: If a is the smallest number > 1 that divides n, then a is prime.

Proof: By contradiction. (Left to the reader.)

- A *multiset* is like a set, except repetitions are allowed
  - $\{\{2, 2, 3, 3, 5\}\}$ is a multiset, not a set

PF(n): A prime factorization procedure

Input: $n \in N^+$
Output: PFS - a multiset of n's prime factors
PFS $:= \emptyset$
for $a = 2$ to $\lfloor \sqrt{n} \rfloor$ do
    if $a \mid n$ **then** PFS $:=$ PF(n/a) $\cup \{\{a\}\}$ **return** PFS
if PFS $= \emptyset$ **then** PFS $:= \{\{n\}\}$ [n is prime]

Example:
$$\begin{aligned}\mathrm{PF}(7007) &= \{\{7\}\} \cup \mathrm{PF}(1001)\\ &= \{\{7,7\}\} \cup \mathrm{PF}(143)\\ &= \{\{7,7,11\}\} \cup \mathrm{PF}(13)\\ &= \{\{7,7,11,13\}\}.\end{aligned}$$

The Complexity of Factoring

Algorithm PF runs in exponential time:

- We're checking every number up to $\sqrt{n}$

Can we do better?

- We don't know.
- Modern-day cryptography implicitly depends on the fact that we can't!
- There is an efficient factoring algorithm using quantum computing.

How Many Primes Are There?

Theorem 4: [Euclid] There are infinitely many primes.

Proof: By contradiction.

- Suppose that there are only finitely many primes: $p_1, \ldots, p_n$.
- Consider $q = p_1 \times \cdots \times p_n + 1$
- Clearly $q > p_1, ..., p_n$, so it can't be prime.
- So q must have a prime factor, which must be one of $p_1, \ldots, p_n$ (since these are the only primes).
- Suppose it is p_i.
  - Then $p_i \mid q$ and $p_i \mid p_1 \times \cdots \times p_n$
  - So $p_i \mid (q - p_1 \times \cdots \times p_n)$; i.e., $p_i \mid 1$ (Corollary 1)
  - Contradiction!

How Many Primes Are There?

Theorem 4: [Euclid] There are infinitely many primes.

Proof: By contradiction.

- Suppose that there are only finitely many primes: $p_1, \ldots, p_n$.
- Consider $q = p_1 \times \cdots \times p_n + 1$
- Clearly $q > p_1, ..., p_n$, so it can't be prime.
- So q must have a prime factor, which must be one of $p_1, \ldots, p_n$ (since these are the only primes).
- Suppose it is p_i.
  - Then $p_i \mid q$ and $p_i \mid p_1 \times \cdots \times p_n$
  - So $p_i \mid (q - p_1 \times \cdots \times p_n)$; i.e., $p_i \mid 1$ (Corollary 1)
  - Contradiction!

Largest currently-known prime (as of 2/20):

- $2^{82,589,933} - 1$: 24,862,048 digits
- Check www.utm.edu/research/primes

Primes of the form $2^p - 1$ where p is prime are called *Mersenne primes*.

- Search for large primes focuses on Mersenne primes

The distribution of primes

There are quite a few primes out there:

- Roughly one in every $\log(n)$ numbers is prime

Formally: let $\pi(n)$ be the number of primes $\leq n$:

Prime Number Theorem: $\pi(n) \sim n/\log(n)$; that is,

$$\lim_{n\to\infty} \pi(n)/(n/\log(n)) = 1$$

Why is this important?

- Cryptosystems like RSA use a secret key that is the product of two large (100-digit) primes.
- How do you find two large primes?
  - Roughly one of every 100 100-digit numbers is prime
  - To find a 100-digit prime;
    - Keep choosing odd numbers at random
    - Check if they are prime (using fast randomized primality test)
    - Keep trying until you find one
    - Roughly 100 attempts should do it

(Some) Open Problems Involving Primes

- Are there infinitely many Mersenne primes?
- *Goldbach's Conjecture*: every even number greater than 2 is the sum of two primes.
  - E.g., $6 = 3 + 3$, $20 = 17 + 3$, $28 = 17 + 11$
  - This has been checked out to 4×10^{18} (as of 2020)
  - True for *almost* all even numbers
    - the fraction of even numbers for which it's true tends to 1
  - Every sufficiently large integer ($> 10^{43{,}000}$!) is the sum of four primes
- Two prime numbers that differ by two are *twin primes*
  - E.g.: (3,5), (5,7), (11,13), (17,19), (41,43)
  - also $4,648,619,711,505 \times 2^{1290000} \pm 1$!
    - largest known as of 2/20

  Are there infinitely many twin primes?

All these conjectures are believed to be true, but no one has proved them.

Greatest Common Divisor (gcd)

Definition: For $a \in Z$ let $D(a) = \{k \in N : k \mid a\}$

- $D(a) = \{\text{divisors of } a\}$.

Claim. $|D(a)| < \infty$ if (and only if) $a \neq 0$.

Proof: If $a \neq 0$ and $k \mid a$, then $0 < k < a$.

Greatest Common Divisor (gcd)

Definition: For $a \in Z$ let $D(a) = \{k \in N : k \mid a\}$

- $D(a) = \{\text{divisors of } a\}$.

Claim. $|D(a)| < \infty$ if (and only if) $a \neq 0$.

Proof: If $a \neq 0$ and $k \mid a$, then $0 < k < a$.

Definition: For $a, b \in Z$, $CD(a, b) = D(a) \cap D(b)$ is the set of common divisors of a, b.

Definition: The *greatest common divisor* of a and b is

$$\gcd(a, b) = \max(CD(a, b)).$$

Greatest Common Divisor (gcd)

Definition: For $a \in Z$ let $D(a) = \{k \in N : k \mid a\}$

- $D(a) = \{\text{divisors of } a\}$.

Claim. $|D(a)| < \infty$ if (and only if) $a \neq 0$.

Proof: If $a \neq 0$ and $k \mid a$, then $0 < k < a$.

Definition: For $a, b \in Z$, $CD(a, b) = D(a) \cap D(b)$ is the set of common divisors of a, b.

Definition: The *greatest common divisor* of a and b is

$$\gcd(a, b) = \max(CD(a, b)).$$

Examples:

- $\gcd(6, 9) = 3$
- $\gcd(13, 100) = 1$
- $\gcd(6, 45) = 3$

Greatest Common Divisor (gcd)

Definition: For $a \in Z$ let $D(a) = \{k \in N : k \mid a\}$

- $D(a) = \{\text{divisors of } a\}$.

Claim. $|D(a)| < \infty$ if (and only if) $a \neq 0$.

Proof: If $a \neq 0$ and $k \mid a$, then $0 < k < a$.

Definition: For $a, b \in Z$, $CD(a, b) = D(a) \cap D(b)$ is the set of common divisors of a, b.

Definition: The *greatest common divisor* of a and b is

$$\gcd(a, b) = \max(CD(a, b)).$$

Examples:

- $\gcd(6, 9) = 3$
- $\gcd(13, 100) = 1$
- $\gcd(6, 45) = 3$

Efficient computation of $\gcd(a, b)$ lies at the heart of commercial cryptography.

Computing the GCD

There is a method for calculating the gcd that goes back to Euclid:

- **Recall:** if $n > m$ and q divides both n and m, then q divides $n - m$ and $n + m$.

Therefore $\gcd(n, m) = \gcd(m, n - m)$.

- Proof: Show that $CD(n, m) = CD(m, n - m)$; i.e. show that q divides both n and m iff q divides both m and $n - m$. (If q divides n and m, then q divides $n - m$ by the argument above. If q divides m and $n - m$, then q divides $m + (n - m) = n$.)
- This allows us to reduce the gcd computation to a simpler case.

We can do even better:

- $\gcd(n, m) = \gcd(m, n - m) = \gcd(m, n - 2m) = \ldots$
- keep going as long as $n - qm \geq 0$ — $\lfloor n/m \rfloor$ steps

Consider $\gcd(6, 45)$:

- $\lfloor 45/6 \rfloor = 7$; remainder is 3 ($45 \equiv 3 \pmod 6$)
- $\gcd(6, 45) = \gcd(6, 45 - 7 \times 6) = \gcd(6, 3) = 3$

We can keep this up this procedure to compute $\gcd(n_1, n_2)$:

- If $n_1 \geq n_2$, write n_1 as $q_1 n_2 + r_1$, where $0 \leq r_1 < n_2$
  - $q_1 = \lfloor n_1/n_2 \rfloor$
- $\gcd(n_1, n_2) = \gcd(r_1, n_2)$
- Now $r_1 < n_2$, so switch their roles:
- $n_2 = q_2 r_1 + r_2$, where $0 \leq r_2 < r_1$
- $\gcd(r_1, n_2) = \gcd(r_1, r_2)$
- Notice that $\max(n_1, n_2) > \max(r_1, n_2) > \max(r_1, r_2)$
- Keep going until we have a remainder of 0 (i.e., something of the form $\gcd(r_k, 0)$ or $(\gcd(0, r_k))$
  - This is bound to happen sooner or later

Euclid's Algorithm

Input m, n [m, n natural numbers, $m \geq n$]
  $num \leftarrow m$; $denom \leftarrow n$ [Initialize num and $denom$]
  repeat until $denom = 0$
    $q \leftarrow \lfloor num/denom \rfloor$
    $rem \leftarrow num - (q * denom)$ [num mod $denom = rem$]
    $num \leftarrow denom$ [New num]
    $denom \leftarrow rem$ [New $denom$; note $num \geq denom$]
  endrepeat
Output num [$num = \gcd(m, n)$]

Example: $\gcd(84, 33)$

Iteration 1: $num = 84$, $denom = 33$, $q = 2$, $rem = 18$
Iteration 2: $num = 33$, $denom = 18$, $q = 1$, $rem = 15$
Iteration 3: $num = 18$, $denom = 15$, $q = 1$, $rem = 3$
Iteration 4: $num = 15$, $denom = 3$, $q = 5$, $rem = 0$
Iteration 5: $num = 3$, $denom = 0 \Rightarrow \gcd(84, 33) = 3$

Euclid's Algorithm: Correctness

How do we know this works?

- We need to prove that
  - (a) the algorithm terminates and
  - (b) that it correctly computes the gcd

We prove (a) and (b) simultaneously by finding appropriate loop invariants and using induction:

- Notation: Let num_k and $denom_k$ be the values of num and $denom$ at the beginning of the kth iteration.

$P(k)$ has three parts:

(1) $0 < num_{k+1} + denom_{k+1} < num_k + denom_k$

(2) $0 \leq denom_k \leq num_k$.

(3) $\gcd(num_k, denom_k) = \gcd(m, n)$

- Termination follows from parts (1) and (2): if $num_k + denom_k$ decreases and $0 \leq denom_k \leq num_k$, then eventually $denom_k$ must hit 0.
- Correctness follows from part (3).
- The induction step is proved by looking at the details of the loop.

Euclid's Algorithm: Complexity

Input m, n [m, n natural numbers, $m \geq n$]
  $num \leftarrow m$; $denom \leftarrow n$ [Initialize num and $denom$]
  repeat until $denom = 0$
    $q \leftarrow \lfloor num/denom \rfloor$
    $rem \leftarrow num - (q * denom)$
    $num \leftarrow denom$ [New num]
    $denom \leftarrow rem$ [New $denom$; note $num \geq denom$]
  endrepeat
Output num [$num = \gcd(m, n)$]

How many times do we go through the loop in Euclid's algorithm:

- Best case: Easy. Never!
- Average case: Too hard
- Worst case: Can't answer this exactly, but we can get a good upper bound.
  - See how fast *denom* goes down in each iteration.

Claim: After two iterations, *denom* is halved:

- Recall $num = q * denom + rem$. Use $denom'$ and $denom''$ to denote value of *denom* after 1 and 2 iterations. Two cases:
    1. $rem \leq denom/2 \Rightarrow denom' \leq denom/2$ and $denom'' < denom/2$.
    2. $rem > denom/2$. But then $num' = denom$, $denom' = rem$. At next iteration, $q = 1$, and $denom'' = rem' = num' - denom' < denom/2$
- How long until *denom* is ≤ 1?
    - $< 2\log_2(m)$ steps!
- After at most $2\log_2(m)$ steps, $denom = 0$.

The Extended Euclidean Algorithm

Theorem 5: For $a, b \in N$, not both 0, we can compute $s, t \in Z$ such that

$$\gcd(a, b) = sa + tb.$$

- **Example:** $\gcd(9, 4) = 1 = 1 \cdot 9 + (-2) \cdot 4$.

Proof: By strong induction on $\max(a, b)$. Suppose without loss of generality $a \leq b$.

- If $\max(a, b) = 1$, then must have $b = 1$, $\gcd(a, b) = 1$
  - $\gcd(a, b) = 0 \cdot a + 1 \cdot b$.
- If $\max(a, b) > 1$, there are three cases:
  - $a = 0$; then $\gcd(0, b) = b = 0 \cdot a + 1 \cdot b$
  - $a = b$; then $\gcd(a, b) = a = 1 \cdot a + 0 \cdot b$
  - If $0 < a < b$, then $\gcd(a, b) = \gcd(a, b - a)$. Moreover, $\max(a, b) > \max(a, b - a)$. Thus, by IH, we can compute s, t such that

$$\gcd(a, b) = \gcd(a, b - a) = sa + t(b - a) = (s - t)a + tb.$$

Note: this computation basically follows the "recipe" of Euclid's algorithm.

Example of Extended Euclidean Algorithm

Recall that $\gcd(84, 33) = \gcd(33, 18) = \gcd(18, 15) = \gcd(15, 3) = \gcd(3, 0) = 3$

We work backwards to write 3 as a linear combination of 84 and 33:

$$
\begin{aligned}
3 &= 18 - 15 \\
&\qquad \text{[Now 3 is a linear combination of 18 and 15]} \\
&= 18 - (33 - 18) \\
&= 2(18) - 33 \\
&\qquad \text{[Now 3 is a linear combination of 18 and 33]} \\
&= 2(84 - 2 \times 33)) - 33 \\
&= 2 \times 84 - 5 \times 33 \\
&\qquad \text{[Now 3 is a linear combination of 84 and 33]}
\end{aligned}
$$

Some Consequences

Definition: a and b are *relatively prime* if $\gcd(a, b) = 1$.

- **Example:** 4 and 9 are relatively prime.
- Two numbers are relatively prime iff they have no common prime factors.

Some Consequences

Definition: a and b are *relatively prime* if $\gcd(a, b) = 1$.

- **Example:** 4 and 9 are relatively prime.
- Two numbers are relatively prime iff they have no common prime factors.

Corollary 2: If a and b are relatively prime, then there exist s and t such that $as + bt = 1$.

Some Consequences

Definition: a and b are *relatively prime* if $\gcd(a, b) = 1$.

- **Example:** 4 and 9 are relatively prime.
- Two numbers are relatively prime iff they have no common prime factors.

Corollary 2: If a and b are relatively prime, then there exist s and t such that $as + bt = 1$.

Corollary 3: If $\gcd(a, b) = 1$ and $a \mid bc$, then $a \mid c$.

Proof:

- Exist $s, t \in Z$ such that $sa + tb = 1$
- Multiply both sides by c: $sac + tbc = c$
- Since $a \mid bc$, $a \mid sac + tbc$, so $a \mid c$

Corollary 4: If p is prime and $p \mid \Pi_{i=1}^{n} a_i$, then $p \mid a_i$ for some $1 \leq i \leq n$.

Proof: By induction on n:

- If $n = 1$: trivial.

Suppose the result holds for n and $p \mid \Pi_{i=1}^{n+1} a_i$.

- note that $p \mid \Pi_{i=1}^{n+1} a_i = (\Pi_{i=1}^{n} a_i)a_{n+1}$.
- If $p \mid a_{n+1}$ we are done.
- If not, $\gcd(p, a_{n+1}) = 1$.
- By Corollary 3, $p \mid \Pi_{i=1}^{n} a_i$
- By the IH, $p \mid a_i$ for some $1 \leq i \leq n$.

Corollary 4: If p is prime and $p \mid \Pi_{i=1}^{n} a_i$, then $p \mid a_i$ for some $1 \leq i \leq n$.

Proof: By induction on n:

- If $n = 1$: trivial.

Suppose the result holds for n and $p \mid \Pi_{i=1}^{n+1} a_i$.

- note that $p \mid \Pi_{i=1}^{n+1} a_i = (\Pi_{i=1}^{n} a_i)a_{n+1}$.
- If $p \mid a_{n+1}$ we are done.
- If not, $\gcd(p, a_{n+1}) = 1$.
- By Corollary 3, $p \mid \Pi_{i=1}^{n} a_i$
- By the IH, $p \mid a_i$ for some $1 \leq i \leq n$.

Corollary 5: If p, q prime, $p \neq q$, $p \mid n$, and $q \mid n$, then $pq \mid n$.

Proof: Since $p \mid n$, then $n = pn'$.
Since $q \mid n = pn'$ and $\gcd(p, q) = 1$, we must have that $q \mid n'$ by Corollary 3, so $n' = n''q$. That means $n = pqn''$, so $pq \mid n$.

The Fundamental Theorem of Arithmetic, II

Theorem 3: Every $n > 1$ can be represented uniquely as a product of primes, written in nondecreasing size.

Proof: Still need to prove uniqueness. We first prove (by strong induction on n), that if $n = \Pi_{i=1}^{r} p_i = \Pi_{j=1}^{s} q_j$, then

$$\{\{p_1, \ldots, p_r\}\} = \{\{q_1, \ldots, q_s\}\}.$$

- Recall that the $\{\{\ldots\}\}$ notation denotes multiset
- A multiset counts repetitions, so if $\{\{p_1, \ldots, p_r\}\} = \{\{q_1, \ldots, q_s\}\}$, then $r = s$.

The Fundamental Theorem of Arithmetic, II

Theorem 3: Every $n > 1$ can be represented uniquely as a product of primes, written in nondecreasing size.

Proof: Still need to prove uniqueness. We first prove (by strong induction on n), that if $n = \Pi_{i=1}^{r} p_i = \Pi_{j=1}^{s} q_j$, then

$$\{\{p_1, \ldots, p_r\}\} = \{\{q_1, \ldots, q_s\}\}.$$

- Recall that the $\{\{\ldots\}\}$ notation denotes multiset
- A multiset counts repetitions, so if $\{\{p_1, \ldots, p_r\}\} = \{\{q_1, \ldots, q_s\}\}$, then $r = s$.

Base case: Obvious if $n = 2$.

Inductive step. Suppose OK for $n' < n$.

- Suppose that $n = \Pi_{i=1}^{r} p_i = \Pi_{j=1}^{s} q_j$.
- $p_1 \mid \Pi_{j=1}^{s} q_j$, so by Corollary 4, $p_1 \mid q_j$ for some j.
- But then $p_1 = q_j$, since both p_1 and q_j are prime.
- But then $n/p_1 = p_2 \cdots p_r = q_1 \cdots q_{j-1} q_{j+1} \cdots q_s$
- Result now follows from I.H.

Modular Arithmetic

Remember: $a \equiv b \pmod{m}$ means a and b have the same remainder when divided by m.

- Equivalently: $a \equiv b \pmod{m}$ iff $m \mid (a - b)$
- a is *congruent* to b mod m

Theorem 7: If $a_1 \equiv a_2 \pmod{m}$ and $b_1 \equiv b_2 \pmod{m}$, then

(a) $(a_1 + b_1) \equiv (a_2 + b_2) \pmod{m}$

(b) $a_1 b_1 \equiv a_2 b_2 \pmod{m}$

Proof: Suppose

- $a_1 = c_1 m + r$, $a_2 = c_2 m + r$
- $b_1 = d_1 m + r'$, $b_2 = d_2 m + r'$

So

- $a_1 + b_1 = (c_1 + d_1)m + (r + r')$
- $a_2 + b_2 = (c_2 + d_2)m + (r + r')$

$m \mid ((a_1 + b_1) - (a_2 + b_2) = ((c_1 + d_1) - (c_2 + d_2))m$

- Conclusion: $a_1 + b_1 \equiv a_2 + b_2 \pmod{m}$.

For multiplication:

- $a_1 b_1 = (c_1 d_1 m + r' c_1 + r d_1) m + r r'$
- $a_2 b_2 = (c_2 d_2 m + r' c_2 + r d_2) m + r r'$

$m \mid (a_1 b_1 - a_2 b_2)$

- Conclusion: $a_1 b_1 \equiv a_2 b_2 \pmod{m}$.

Bottom line: addition and multiplication carry over to the modular world.

For multiplication:

- $a_1 b_1 = (c_1 d_1 m + r' c_1 + r d_1) m + r r'$
- $a_2 b_2 = (c_2 d_2 m + r' c_2 + r d_2) m + r r'$

$m \mid (a_1 b_1 - a_2 b_2)$

- Conclusion: $a_1 b_1 \equiv a_2 b_2 \pmod{m}$.

Bottom line: addition and multiplication carry over to the modular world.

Theorem 8: $a \equiv b \pmod{m}$ is an equivalence relation on the integers.

Modular arithmetic has lots of applications.

- Here are four ...

Hashing

Problem: How can we efficiently store, retrieve, and delete records from a large database?

- For example, students records.

Assume, each record has a unique key

- E.g. student ID, Social Security #

Do we keep an array sorted by the key?

- Easy retrieval but difficult insertion and deletion.

How about a table with an entry for every possible key?

- Often infeasible, almost always wasteful.
- There are 10^{10} possible social security numbers.

Solution: store the records in an array of size N, where N is somewhat bigger than the expected number of records.

- Store record with id k in location $h(k)$
  - h is the *hash function*
  - Basic hash function: $h(k) := k \pmod{N}$.
- A collision occurs when $h(k_1) = h(k_2)$ and $k_1 \neq k_2$.
  - Choose N sufficiently large to minimize collisions
- Lots of techniques for dealing with collisions

Pseudorandom Sequences

For randomized algorithms we need a random number generator.

- Most languages provide you with a function "rand".
- There is nothing random about rand!
  - It creates an apparently random sequence deterministically
  - These are called *pseudorandom sequences*

A standard technique for creating pseudorandom sequences: the *linear congruential method*.

- Choose a modulus $m \in N^+$,
- a multiplier $a \in \{2, 3, \ldots, m-1\}$, and
- an increment $c \in Z_m = \{0, 1, \ldots, m-1\}$.
- Choose a seed $x_0 \in Z_m$
  - Typically the time on some internal clock is used
- Compute $x_{n+1} = ax_n + c \pmod{m}$.

Warning: a poorly implemented rand, such as in C, can wreak havoc on Monte Carlo simulations.

Recall that a linear congruence generator has $x_{n+1} = ax_n + c \pmod{m}$. Some common choices for a, c, and m:

- m prime, $c = 0$
- m a power of 2, a odd (often 3 or 5 mod 8)
- $c \neq 0$, m a power of an odd prime p, $a - 1$ divisible by p

(See wikipedia article on linear congruential generator for more.)

ISBN Numbers

Since 1968, most published books have been assigned a 10-digit ISBN numbers:

- identifies country of publication, publisher, and book itself

All the information is encoded in the first 9 digits

- The 10th digit is used as a parity check
- If the digits are $a_1, \ldots, a_{10}$, then we must have

$$a_1 + 2a_2 + \cdots + 9a_9 + 10a_{10} \equiv 0 \pmod{11}.$$

- This test always detects errors in single digits and transposition errors
  - Two arbitrary errors may cancel out

Similar parity checks are used in universal product codes (UPC codes/bar codes) that appear on almost all items

- The numbers are encoded by thicknesses of bars, to make them machine readable

Casting out 9s

Notice that a number is equivalent to the sum of its digits mod 9. This can be used as a way of checking your addition and of doing mindreading [come to class to hear more . . .]

Fermat's Little Theorem

Theorem 10 (Fermat's Little Theorem):

(a) If p prime and $\gcd(p, a) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

(b) For all $a \in Z$, $a^p \equiv a \pmod{p}$.

Proof. Let

$$\begin{aligned} A &= \{1, 2, \ldots, p-1\} \\ B &= \{1a \bmod p, 2a \bmod p, \ldots, (p-1)a \bmod p\} \end{aligned}$$

Claim: $A = B$.

- $0 \notin B$, since $p \not| ja$, so $B \subseteq A$.
- If $i \neq j$, then $ia \bmod p \neq ja \bmod p$, so $f : A \to B$ with $f(i) = ia \bmod p$ is an injection.
  - since $p \not| (j - i)a$
- It follows that $|A| \leq |B|$.
- Since $0 \notin B$, $B \subseteq \{0, \ldots, p-1\}$, and $|B| \geq p - 1$, we must have $A = B$!

We've just shown that $A = B$, where

- $A = \{1, 2, \ldots, p-1\}$
- $B = \{1a \bmod p, 2a \bmod p, \ldots, (p-1)a \bmod p\}$

Therefore,

$$\begin{aligned}
& \Pi_{i \in A}\, i \equiv \Pi_{i \in B}\, i \pmod{p} \\
\Rightarrow\ & (p-1)! \equiv a(2a)\cdots(p-1)a = (p-1)!\, a^{p-1} \pmod{p} \\
\Rightarrow\ & p \mid (a^{p-1}-1)(p-1)! \\
\Rightarrow\ & p \mid (a^{p-1}-1) \quad [\text{since } \gcd(p,(p-1)!) = 1] \\
\Rightarrow\ & a^{p-1} \equiv 1 \pmod{p}
\end{aligned}$$

It follows that $a^p \equiv a \pmod{p}$

- This is true even if $\gcd(p, a) \neq 1$; i.e., if $p \mid a$

Why is this being taught in a CS course?

Private Key Cryptography

Alice (aka A) wants to send an encrypted message to Bob (aka B).

- A and B might share a private key known only to them.
- The same key serves for encryption and decryption.
- Example: Caesar's cipher $f(m) = m + 3 \bmod 26$
  (shift each letter by three)
  - `WKH EXWOHU GLG LW`
  - `THE BUTLER DID IT`

This particular cryptosystem is very easy to solve

- Idea: look for common letters (E, A, T, S)

One Time Pads

Some private key systems are completely immune to cryptanalysis:

- A and B share the only two copies of a long list of random integers s_i for $i = 1, \ldots, N$.
- A sends B the message $\{m_i\}_{i=1}^{n}$ encrypted as:

$$c_i = (m_i + s_i) \bmod 26$$

- B decrypts A's message by computing $c_i - s_i \bmod 26$.

One Time Pads

Some private key systems are completely immune to cryptanalysis:

- A and B share the only two copies of a long list of random integers s_i for $i = 1, \ldots, N$.
- A sends B the message $\{m_i\}_{i=1}^{n}$ encrypted as:

$$c_i = (m_i + s_i) \bmod 26$$

- B decrypts A's message by computing $c_i - s_i \bmod 26$.

The good news: bulletproof cryptography

The bad news: horrible for e-commerce

- How do random users exchange the pad?
  - To some extent you can simulate this using a (deterministic) random number generator
    - *A* and *B* just have to share the seed

One Time Pads

Some private key systems are completely immune to cryptanalysis:

- A and B share the only two copies of a long list of random integers s_i for $i = 1, \ldots, N$.
- A sends B the message $\{m_i\}_{i=1}^{n}$ encrypted as:

$$c_i = (m_i + s_i) \bmod 26$$

- B decrypts A's message by computing $c_i - s_i \bmod 26$.

The good news: bulletproof cryptography

The bad news: horrible for e-commerce

- How do random users exchange the pad?
  - To some extent you can simulate this using a (deterministic) random number generator
    - A and B just have to share the seed
- But all this is still pretty useless if you want to use encryption for security on the internet

Public Key Cryptography

Idea of *public key cryptography* (Diffie-Hellman)

- Everyone's encryption scheme is posted publicly
  - e.g. in a "telephone book"
- If A wants to send an encoded message to B, she looks up B's public key (i.e., B's encryption algorithm) in the telephone book
- But only B has the decryption key corresponding to his public key

BIG advantage: A need not know nor trust B.

Public Key Cryptography

Idea of *public key cryptography* (Diffie-Hellman)

- Everyone's encryption scheme is posted publicly
  - e.g. in a "telephone book"
- If A wants to send an encoded message to B, she looks up B's public key (i.e., B's encryption algorithm) in the telephone book
- But only B has the decryption key corresponding to his public key

BIG advantage: A need not know nor trust B.

There seems to be a problem though:

- If we publish the encryption key, won't everyone be able to decrypt?

Key observation: decrypting might be too hard, unless you know the key

- Computing f^{-1} could be much harder than computing f

Can we find an (f, f^{-1}) pair for which this is true?

Public Key Cryptography

Idea of *public key cryptography* (Diffie-Hellman)

- Everyone's encryption scheme is posted publicly
  - e.g. in a "telephone book"
- If A wants to send an encoded message to B, she looks up B's public key (i.e., B's encryption algorithm) in the telephone book
- But only B has the decryption key corresponding to his public key

BIG advantage: A need not know nor trust B.

There seems to be a problem though:

- If we publish the encryption key, won't everyone be able to decrypt?

Key observation: decrypting might be too hard, unless you know the key

- Computing f^{-1} could be much harder than computing f

Can we find an (f, f^{-1}) pair for which this is true?

- Yes, by using number theory!

RSA: Key Generation

Generating encryption/decryption keys

- Choose two very large (hundreds of digits) primes p, q.
  - This is done using probabilistic primality testing
  - Choose a random large number and check if it is prime
  - By the prime number theorem, there are lots of primes out there
- Let $n = pq$.
- Choose $e \in N$ relatively prime to $(p-1)(q-1)$. Here's how:
  - Choose e_1, e_2 prime and slightly greater than $\sqrt{n}$
    - using fast primality testing again
  - One must be relatively prime to $(p-1)(q-1)$
    - Otherwise $e_1 e_2 \mid (p-1)(q-1)$
  - Find out which one using Euclid's algorithm
- Compute d, the inverse of e modulo $(p-1)(q-1)$.
  - Can do this using extended Euclidean algorithm
  - Find d, s such that $de + s(p-1)(q-1) = 1$.
- Publish n and e (that's your public key)
- Keep the decryption key d to yourself.

RSA: Sending encrypted messages

How does someone send you a message?

- The message is divided into blocks each represented as a number M between 0 and n. To encrypt M, send

$$C = M^e \bmod n.$$

  - Need to use fast exponentiation ($2\log(n)$ multiplications) to do this efficiently

RSA: Sending encrypted messages

How does someone send you a message?

- The message is divided into blocks each represented as a number M between 0 and n. To encrypt M, send

$$C = M^e \bmod n.$$

  - Need to use fast exponentiation ($2\log(n)$ multiplications) to do this efficiently

Example: Encrypt "stop" using $e = 13$ and $n = 2537$:

- s t o p $\leftrightarrow$ 18 19 14 15 $\leftrightarrow$ 1819 1415
- $1819^{13} \bmod 2537 = 2081$ and
  $1415^{13} \bmod 2537 = 2182$ so
- 2081 2182 is the encrypted message.
- We did not need to know $p = 43, q = 59$ for that.

Decryption

How do you decrypt a message?

- Claim: $M^{ed} \equiv M \pmod{n}$
  - So, to decrypt, raise the encrypted message (M^e) to power d
  - **Key point:** the receiver knows d (but no one else does)
  - That's because (we believe that) given n and e, computing d is hard, because factoring n is hard.

Why is this right?

Decryption

How do you decrypt a message?

- Claim: $M^{ed} \equiv M \pmod{n}$
  - So, to decrypt, raise the encrypted message (M^e) to power d
  - **Key point:** the receiver knows d (but no one else does)
  - That's because (we believe that) given n and e, computing d is hard, because factoring n is hard.

Why is this right?

- Recall that $ed \equiv 1 \pmod{(p-1)(q-1)}$
- By Fermat's Little Theorem, if $\gcd(p, M) = 1$, then $M^{ed} \equiv M \pmod{p}$
  - Since $ed = c(p-1) + 1$, so $M^{ed} = M^{c(p-1+1} = (M^{p-1})^c M \equiv M \pmod{p}$.
- This is also true if $\gcd(p, M) \neq 1$ (i.e., if $p|M$)
- Similarly $M^{ed} \equiv M \pmod{q}$.
- So $p|(M^{ed} - M)$, $q|(M^{ed} - M)$

Decryption

How do you decrypt a message?

- Claim: $M^{ed} \equiv M \pmod{n}$
  - So, to decrypt, raise the encrypted message (M^e) to power d
  - **Key point:** the receiver knows d (but no one else does)
  - That's because (we believe that) given n and e, computing d is hard, because factoring n is hard.

Why is this right?

- Recall that $ed \equiv 1 \pmod{(p-1)(q-1)}$
- By Fermat's Little Theorem, if $\gcd(p, M) = 1$, then $M^{ed} \equiv M \pmod{p}$
  - Since $ed = c(p-1) + 1$, so $M^{ed} = M^{c(p-1+1} = (M^{p-1})^c M \equiv M \pmod{p}$.
- This is also true if $\gcd(p, M) \neq 1$ (i.e., if $p|M$)
- Similarly $M^{ed} \equiv M \pmod{q}$.
- So $p|(M^{ed} - M)$, $q|(M^{ed} - M)$
  - by Collary 5, $pq|(M^{ed} - M)$
- So $M^{ed} \equiv M \pmod{n}$ (since $n = pq$)

Decryption

How do you decrypt a message?

- Claim: $M^{ed} \equiv M \pmod{n}$
  - So, to decrypt, raise the encrypted message (M^e) to power d
  - **Key point:** the receiver knows d (but no one else does)
  - That's because (we believe that) given n and e, computing d is hard, because factoring n is hard.

Why is this right?

- Recall that $ed \equiv 1 \pmod{(p-1)(q-1)}$
- By Fermat's Little Theorem, if $\gcd(p, M) = 1$, then $M^{ed} \equiv M \pmod{p}$
  - Since $ed = c(p-1)+1$, so $M^{ed} = M^{c(p-1+1} = (M^{p-1})^c M \equiv M \pmod{p}$.
- This is also true if $\gcd(p, M) \neq 1$ (i.e., if $p|M$)
- Similarly $M^{ed} \equiv M \pmod{q}$.
- So $p|(M^{ed} - M)$, $q|(M^{ed} - M)$
  - by Collary 5, $pq|(M^{ed} - M)$
- So $M^{ed} \equiv M \pmod{n}$ (since $n = pq$)

Digital Signatures

How can I send you a message in such a way that you're convinced it came from me (and can convince others).

- Want an analogue of a "certified" signature

Cool observation:

- To sign a message M, send $M^d \pmod{n}$
  - where (n, e) is my public key
- Recipient (and anyone else) can compute $(M^d)^e \equiv M \pmod{n}$, since M is public
- No one else could have sent this message, since no one else knows d.

Security is Subtle

There are lots of ways of "misapplying" RSA, even assuming that factoring is hard.

- The public key $n = pq$, the product of two large primes
- How do you find the primes?
  - Guess a big odd number n_1, check if it's prime
  - If not, try $n_1 + 2$, then $n_1 + 4$, ...
  - Within roughly $\log(n_1)$ steps, you should find a prime;
- How do you find the second prime?
  - Guess a big odd number n_2, check if it's prime
  - ...
- Suppose, instead, you started with the first prime (call it p), and checked $p + 2$, $p + 4$, $p + 6$, ..., until you found another prime q, and used that.
  - Is that a good idea? NO!!!

Security is Subtle

There are lots of ways of "misapplying" RSA, even assuming that factoring is hard.

- The public key $n = pq$, the product of two large primes
- How do you find the primes?
  - Guess a big odd number n_1, check if it's prime
  - If not, try $n_1 + 2$, then $n_1 + 4$, ...
  - Within roughly $\log(n_1)$ steps, you should find a prime;
- How do you find the second prime?
  - Guess a big odd number n_2, check if it's prime
  - ...
- Suppose, instead, you started with the first prime (call it p), and checked $p + 2$, $p + 4$, $p + 6$, ..., until you found another prime q, and used that.
  - Is that a good idea? NO!!!

If $n = pq$, then p is the first prime less than $\sqrt{n}$, and q is the first prime greater than $\sqrt{n}$.

- You can find both easily!

How Secure is RSA?

The security of RSA depends on the hardness of factoring.

- Peter Shor (now at MIT) showed in 1994 that factoring can be done in polynomial time on a quantum computer
- We don't yet have quantum computers powerful enough to factor large numbers
  - But one day we might

How Secure is RSA?

The security of RSA depends on the hardness of factoring.

- Peter Shor (now at MIT) showed in 1994 that factoring can be done in polynomial time on a quantum computer
- We don't yet have quantum computers powerful enough to factor large numbers
  - But one day we might

But even without using quantum computers, we may not be safe:

> An international team of French and U.S. researchers factored the largest RSA key size ever computed ... The researchers successfully factored RSA-240, an RSA key with 240 decimal digits and a size of 795 bits, and a same-sized discrete logarithm. The researchers used the Number Field Sieve algorithm, and the total computation time for achieving these records was approximately 4,000 core-years ... – Dec. 2019

More to Explore

If you like number theory, consider taking

- MATH 3320: Introduction to Number Theory

If you're interested in cryptography, try

- CS 4830: Introduction to Cryptography

For a brief introduction to some current number theory, check out `http://homepages.umflint.edu/~mclemanc/Files/McLemanCoolestNumbers.pdf`

- The Ten Coolest Numbers
- thanks to Rob Tirrell for pointing this out