Plan for today

1. Prime numbers
2. Divisibility
3. Strong induction
Prime numbers

Which of the following are prime numbers:

0, 1, 2, 3, 4

A. All of them
B. 0, 1, 2, 3
C. 1, 2, 3
D. 1, 3
E. None of the above options.
Prime numbers

Which of the following are prime numbers:

0, 1, 2, 3, 4

A. All of them
B. 0, 1, 2, 3
C. 1, 2, 3
D. 1, 3
E. None of the above options.

Definition: A prime number is an integer greater than 1 whose only positive integer divisors are 1 and itself.

Prime factors?

I have nothing to do, so I'm trying to calculate the prime factors of the time each minute before it changes. It was easy when I started at 1:00, but with each hour the number gets bigger. I wonder how long I can keep up.

253 is 11 x 23

What?

I'm factoring the time.

Hey!

Think fast.

xkcd.com
How to decide if a number is prime?

Fundamental Theorem of Arithmetic

Every positive integer greater than 1 is divisible by a prime number.
Every positive integer greater than 1 is divisible by a prime number.

Do you believe this statement?
Fundamental Theorem of Arithmetic

Every positive integer greater than 1 is divisible by a prime number.

Do you believe this statement?

How do we prove the statement?
Fundamental Theorem of Arithmetic

Every positive integer greater than 1 is divisible by a prime number.

Do you believe this statement?

How do we prove the statement?

- **Base case:**

- **Induction step:** declare variables!
  - Assumption: previous term
  - Subgoal: next term
Comparing the two kinds of induction

When proving the theorem $\forall n \geq a \ P(n)$,

<table>
<thead>
<tr>
<th>Regular induction</th>
<th>Strong induction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base case:</strong> WTS $P(a)$</td>
<td><strong>Base case:</strong> WTS $P(a)$</td>
</tr>
<tr>
<td><strong>Induction step:</strong></td>
<td><strong>Induction step:</strong></td>
</tr>
<tr>
<td>Let $n \geq a$</td>
<td>Let $n \geq a$</td>
</tr>
<tr>
<td>Assume $P(n)$</td>
<td>Assume $P(1), P(2), \ldots, P(n)$</td>
</tr>
<tr>
<td>WTS $P(n+1)$</td>
<td>WTS $P(n+1)$</td>
</tr>
</tbody>
</table>
Why do primes matter?

- Number theory: primes are basic building blocks of integers
- Cryptography: finding large primes is relatively easy, factoring into primes is hard.
Example: Turing’s code 1.0

- **The set-up**
  - Assign each symbol to a number.
  - Sender and receiver agree on **secret key**, some large **prime** number $k$.
- Sender translates message to string of numbers.
- Sender pads it with enough trailing digits to make the message **prime**.
- Sender **encrypts** message by computing

$$M = m \cdot k$$

- Receiver receives $M$
- Receiver **decrypts** message by computing

$$m = \frac{M}{k}$$
1.0 Example

The set-up

- $A = 01$, $B = 02$, $C = 03$, $Y = 25$, $Z = 26$
- Secret key $k = 991$. 

Message: "SPIS" becomes $1916091901$ then pad it to get a prime number $1916091901$

Encrypt the message $M = (1916091901) \cdot (991) = 1898847073891$

Receiver receives $M$

Receiver decrypts message by computing $m = 1898847073891 \div 991 = 1916091901$

and then using dictionary $1916091901$ becomes SPISA
1.0 Example

- The set-up
  - \( A = 01, B = 02, C = 03, \ldots, Y = 25, Z = 26 \)
  - Secret key \( k = 991 \).
- Message: “SPIS” becomes \[ \begin{array}{c} 19 \end{array} \begin{array}{c} 16 \end{array} \begin{array}{c} 09 \end{array} \begin{array}{c} 19 \end{array} \]
1.0 Example

- **The set-up**
  - $A = 01$, $B = 02$, $C = 03$, \ldots, $Y = 25$, $Z = 26$
  - Secret key $k = 991$.
- Message: “SPIS” becomes $\boxed{19\ 16\ 09\ 19}$
- Then pad it to get a prime number $1916091901$
1.0 Example

- **The set-up**
  - $A = 01$, $B = 02$, $C = 03$, \ldots, $Y = 25$, $Z = 26$
  - Secret key $k = 991$.
- Message: “SPIS” becomes $19\ 16\ 09\ 19$
- then pad it to get a prime number
  
  $1916091901$

- Encrypt the message
  
  $M = (1916091901) \cdot (991) = 1898847073891$
1.0 Example

- **The set-up**
  - \( A = 01, B = 02, C = 03, \ldots, Y = 25, Z = 26 \)
  - Secret key \( k = 991 \).

- Message: “SPIS” becomes \( 19 \ | \ 16 \ | \ 09 \ | \ 19 \)

- then pad it to get a prime number

\[
1916091901
\]

- Encrypt the message

\[
M = (1916091901) \cdot (991) = 1898847073891
\]

- Receiver receives \( M \)
1.0 Example

- **The set-up**
  - $A = 01, B = 02, C = 03, \ldots, Y = 25, Z = 26$
  - Secret key $k = 991$.

- Message: “SPIS” becomes $19\ 16\ 09\ 19$

- then pad it to get a prime number

  $1916091901$

- Encrypt the message

  $M = (1916091901) \cdot (991) = 1898847073891$

- Receiver receives $M$

- Receiver decrypts message by computing

  $m = \frac{1898847073891}{991} = 1916091901$

  and then using dictionary $19\ 16\ 09\ 19\ 01$ becomes SPISA
1.0 Problem

Suppose the same key was used twice.

An eavesdropper would see

\[ M_1 \quad \text{and} \quad M_2 \]
Suppose the same key was used \textit{twice}.

An eavesdropper would see

\[ M_1 = m_1 k \quad \text{and} \quad M_2 = m_2 k \]

But, $m_1$, $m_2$, $k$ are all prime.
1.0 Problem

Suppose the same key was used twice.

An eavesdropper would see

\[ M_1 = m_1 k \quad \text{and} \quad M_2 = m_2 k \]

But, \( m_1, m_2, k \) are all prime.

So,

\[ \gcd(M_1, M_2) = k \]
1.0 Problem

Suppose the same key was used twice.

An eavesdropper would see

\[ M_1 = m_1 k \quad \text{and} \quad M_2 = m_2 k \]

But, \( m_1, m_2, k \) are all prime.

So,

\[ \gcd(M_1, M_2) = k \]

And, there is a fast way to find the GCD of two numbers (Euclidean algorithm) ... compromising the (no longer) secret key.
Example: Turing's code 2.0


Sender and receiver agree on secret key, some large prime number $k$. 

Sender and receiver also agree on a public modulus, large prime $p$. 

Sender translates message to a number between 0 and $p - 1$. 

Sender encrypts message by computing $M = (m \cdot k) \mod p$. 

Receiver receives $M$. 

Receiver decrypts message how? 

Want to divide integers, mod $p$. 

What does this mean?
Example: Turing’s code 2.0

Idea: Work \textit{mod} some prime \( p \).

- \textit{The set-up}
  - Assign each symbol to a number.
  - Sender and receiver agree on a secret key, some large prime number \( k \).
  - \textit{Sender and receiver also agree on a public modulus, large prime} \( p \).
Example: Turing’s code 2.0


- **The set-up**
  - Assign each symbol to a number.
  - Sender and receiver agree on a **secret key**, some large prime number $k$.
  - *Sender and receiver also agree on a **public modulus**, large prime $p$.*

- Sender translates message to a number between 0 and $p - 1$

- Sender encrypts message by computing

  $$M = (m \cdot k) \mod p$$
Example: Turing’s code 2.0

*Idea:* Work \( \text{mod} \) some prime \( p \).

- **The set-up**
  - Assign each symbol to a number.
  - Sender and receiver agree on a **secret key**, some large prime number \( k \).
  - *Sender and receiver also agree on a** public modulus**, large prime \( p \).

- Sender translates message to a number between 0 and \( p - 1 \)

- Sender encrypts message by computing

\[
M = (m \cdot k) \mod p
\]

- Receiver receives \( M \)

- Receiver decrypts message **How**?
  - Want to divide integers, \( \text{mod} \ p \)
  - What does this mean?
What is \( \frac{2^5}{5} \mod 7 \)?

A. Does not exist.
B. \( \frac{2}{5} \)
C. 1
D. 6
E. 4
What is $\frac{2}{3} \mod 6$?

A. Does not exist.
B. $\frac{2}{5}$
C. 1
D. 6
E. 4
Multiplicative inverses mod $p$

What is $2^{-1} \mod 7$?

*Hint: See clock.*

A. Does not exist.  
B. $\frac{2}{5}$  
C. 1  
D. 6  
E. 4
Why primes?

Theorem: if $p$ is a prime, then for each $n$ ($0 < n < p$) there is some $r$ ($0 < r < p$) such that
\[(n \cdot r) \% p = 1.\]

This is not true for nonprimes.
Why primes?

**Theorem:** if $p$ is a prime, then for each $n$ $(0 < n < p)$ such that $(0 < r < p)$ there is some $r$ such that

$$(n \cdot r) \mod p = 1.$$

This is not true for nonprimes.

**Fermat’s Little Theorem:** If $p$ is a prime and $r$ is not a multiple of $p$ then

$$r^{-1} \mod p = r^{p-2} \mod p.$$
Why primes?

**Theorem:** if \( p \) is a prime, then for each \( n \) (\( 0 < n < p \)) (\( 0 < r < p \)) such that

\[
(n \cdot r) \mod p = 1.
\]

This is not true for nonprimes.

**Fermat’s Little Theorem:** If \( p \) is a prime and \( r \) is not a multiple of \( p \) then

\[
r^{-1} \mod p = r^{p-2} \mod p.
\]

Back to clock example: we saw that \( 2^{-1} \mod 7 = 4 \).

Compare with

\[
2^5 \mod 7 =
\]
Why primes?

**Theorem:** if $p$ is a prime, then for each $n$ $(0 < n < p)$ there is some $r$ $(0 < r < p)$ such that

$$(n \cdot r) \% p = 1.$$ 

This is not true for nonprimes.

**Fermat’s Little Theorem:** If $p$ is a prime and $r$ is not a multiple of $p$ then

$$r^{-1} \% p = r^{p-2} \% p.$$ 

Back to clock example: we saw that $2^{-1} \% 7 = 4$.

Compare with

$$2^5 \% 7 = 32 \% 7$$
Why primes?

**Theorem:** if $p$ is a prime, then for each $n$ $(0 < n < p)$ there is some $r$ $(0 < r < p)$ such that

$$(n \cdot r) \mod p = 1.$$

This is not true for nonprimes.

**Fermat’s Little Theorem:** If $p$ is a prime and $r$ is not a multiple of $p$ then

$$r^{-1} \mod p = r^{p-2} \mod p.$$

Back to clock example: we saw that $2^{-1} \mod 7 = 4$.

Compare with

$$2^5 \mod 7 = 32 \mod 7 = 4 \quad ☺.$$

- **The set-up**
  - Assign each symbol to a number.
  - Sender and receiver agree on a **secret key**, some large prime number $k$.
  - *Sender and receiver also agree on a **public modulus**, large prime $p$.*

- Sender translates message to a number between 0 and $p - 1$

- Sender encrypts message by computing

$$M = (m \cdot k) \mod p$$

- Receiver receives $M$

- Receiver decrypts message
Idea: Work \mod some prime $p$.

- **The set-up**
  - Assign each symbol to a number.
  - Sender and receiver agree on a **secret key**, some large prime number $k$.
  - Sender and receiver also agree on a **public modulus**, large prime $p$.

- Sender translates message to a number between 0 and $p - 1$
- Sender encrypts message by computing

$$M = (m \cdot k) \mod p$$

- Receiver receives $M$
- Receiver decrypts message by computing

$$m = (M \cdot k^{-1}) \mod p = (M \cdot k^{p-2}) \mod p$$
Example

- **The set-up**
  - \( A = 01, B = 02, C = 03, \ldots, Y = 25, Z = 26; \)
    - send one letter at a time (or two).
  - Secret key \( k = 991. \)
  - Public key \( p = 3571. \)
Example

- **The set-up**
  - $A = 01$, $B = 02$, $C = 03$, $\ldots$, $Y = 25$, $Z = 26$; send one letter at a time (or two).
  - Secret key $k = 991$.
  - Public key $p = 3571$.

- **Message**: “SPIS” becomes four messages
  - \[19\] then \[16\] then \[09\] then \[19\] (each less than $3571 - 1$).
Example

- **The set-up**
  - $A = 01, B = 02, C = 03, \ldots, Y = 25, Z = 26$; send one letter at a time (or two).
  - Secret key $k = 991$.
  - Public key $p = 3571$.

- Message: “SPIS” becomes four messages
  - 19 then 16 then 09 then 19 (each less than 3571 – 1).

- Encrypt the (first part of the) message

\[ M = (19 \cdot 991) \mod 3571 = 974 \]
Example

- **The set-up**
  - $A = 01$, $B = 02$, $C = 03$, ..., $Y = 25$, $Z = 26$; send one letter at a time (or two).
  - Secret key $k = 991$.
  - Public key $p = 3571$.

- Message: “SPIS” becomes four messages: 19 then 16 then 09 then 19 (each less than 3571 – 1).

- Encrypt the (first part of the) message

  $$M = (19 \cdot 991) \mod 3571 = 974$$

- Receiver receives $M$
Example

- **The set-up**
  - $A = 01$, $B = 02$, $C = 03$, ..., $Y = 25$, $Z = 26$; send one letter at a time (or two).
  - Secret key $k = 991$.
  - Public key $p = 3571$.

- Message: “SPIS” becomes four messages 19 then 16 then 09 then 19 (each less than $3571 - 1$).

- Encrypt the (first part of the) message

\[ M = (19 \cdot 991) \mod 3571 = 974 \]

- Receiver receives $M$

- Receiver decrypts message by computing

\[ m = (974 \cdot 991^{-1}) \mod 3571 = (974 \cdot 209) \mod 3571 = 19 \]
Plaintext attack: If an eavesdropper can get both $m, M$ for some message then, since $M = (m \cdot k) \mod p,$

$$(m^{p-2} \cdot M) \mod p = (m^{p-2} \cdot mk) \mod p$$
Problem

Plaintext attack: If an eavesdropper can get both $m, M$ for some message then, since $M = (m \cdot k) \mod p$,

$$(m^{p-2} \cdot M) \mod p = (m^{p-2} \cdot mk) \mod p \overset{\text{Fermat}}{=} (m^{-1} m \cdot k) \mod p$$
Plaintext attack: If an eavesdropper can get both $m, M$ for some message then, since $M = (m \cdot k) \mod p$,

$$
\left( m^{p-2} \cdot M \right) \mod p = \left( m^{p-2} \cdot mk \right) \mod p \overset{\text{Fermat}}{=} \left( m^{-1} m \cdot k \right) \mod p = k \mod p
$$

So the secret key is revealed!
Problem

**Plaintext attack**: If an eavesdropper can get both $m, M$ for some message then, since $M = (m \cdot k) \mod p$,

$$(m^{p-2} \cdot M) \mod p = (m^{p-2} \cdot mk) \mod p \stackrel{\text{Fermat}}{=} (m^{-1} m \cdot k) \mod p = k \mod p$$

So the secret key is revealed!

*How do we fix this . . . Cryptography!*
Relatively prime numbers

Which of the following pairs of numbers are relatively prime:

A. 2, 6
B. 5, 8
C. 6, 10
D. 0, 5
E. None of the above.

Definition: Two positive integers are relatively prime if their GCD is 1.
Which of the following pairs of numbers are relatively prime:

A. 2, 6
B. 5, 8
C. 6, 10
D. 0, 5
E. None of the above.

**Definition:** Two positive integers are relatively prime if their GCD is 1.
Example: LCG

In Week 1’s Python Warmup, you wrote a linear congruential pseudo random number generator.

With parameters

- $m$ (positive), modulus
- $a$ (positive, less than $m$), multiplier
- $c$ (nonnegative, less than $m$), increment
- $x_0$ (nonnegative, less than $m$), seed

Define:

$$x_{n+1} = (ax_n + c) \mod m$$

Java's java.util.Random is LCG with $m = 2^{48}$, $a = 2^{52} - 1$, $c = 11$ (only outputs bits 47.

Python uses different generator: Mersenne Twister.
Example: LCG

In Week 1’s Python Warmup, you wrote a linear congruential pseudo random number generator.

With parameters

- \( m \) (positive), modulus
- \( a \) (positive, less than \( m \)), multiplier
- \( c \) (nonnegative, less than \( m \)), increment
- \( x_0 \) (nonnegative, less than \( m \)), seed

Define:

\[
x_{n+1} = (ax_n + c) \mod m
\]

Java’s java.util.Random is LCG with

\[
m = 2^{48} \quad a = 25214903917 \quad c = 11
\]

(only outputs bits 47...16).  *Python uses different generator: Mersenne Twister.*
Why only pseudorandom?

- Deterministic once we have all the parameters.
Why only pseudorandom?

- Deterministic once we have all the parameters.
- So, length of the sequence before it loops back to beginning, i.e. its period, is important.
Why only pseudorandom?

- Deterministic once we have all the parameters.
- So, length of the sequence before it loops back to beginning, i.e. its period, is important.
- Max length? $m$ ... why??

Hull-Dobell Theorem: achieve this max length if
- $c$ and $m$ are relatively prime
- $a - 1$ is divisible by each prime factor of $m$
- $a - 1$ is multiple of 4 if $m$ is multiple of 4.

In particular, if $m$ is a power of 2 then sufficient to have $c$ odd and $a \% 4 = 1$. 
Why only pseudorandom?

- Deterministic once we have all the parameters.
- So, length of the sequence before it loops back to beginning, i.e. its **period**, is important.
- Max length? $m$ ... why??
- Hull-Dobell Theorem: achieve this max length if
  - $c$ and $m$ are relatively prime
  - $a - 1$ is divisible by each prime factor of $m$
  - $a - 1$ is multiple of 4 if $m$ is multiple of 4.

In particular, if $m$ is a power of 2 then sufficient to have $c$ odd and $a \% 4 = 1$. 