Counting basics.

**First rule:** $n_1 \times n_2 \cdots \times n_3$. 
Counting basics.

**First rule:** \( n_1 \times n_2 \cdots \times n_3 \).

\( k \) Samples with replacement from \( n \) items: \( n^k \).
Counting basics.

**First rule:** \( n_1 \times n_2 \cdots \times n_3. \)

\( k \) Samples with replacement from \( n \) items: \( n^k. \)
Sample without replacement: \( \frac{n!}{(n-k)!} \).
Counting basics.

**First rule:** \( n_1 \times n_2 \cdots \times n_3. \)

$k$ Samples with replacement from $n$ items: \( n^k \).
Sample without replacement: \( \frac{n!}{(n-k)!} \)

**Second rule:** when order doesn’t matter divide..when possible.
Counting basics.

**First rule:** $n_1 \times n_2 \cdots \times n_3$.

$k$Samples with replacement from $n$ items: $n^k$.
Sample without replacement: $\frac{n!}{(n-k)!}$

**Second rule:** when order doesn’t matter divide...when possible.
Sample without replacement and order doesn’t matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
“$n$ choose $k$”
First rule: $n_1 \times n_2 \cdots \times n_3$.

$k$ Samples with replacement from $n$ items: $n^k$.
Sample without replacement: $\frac{n!}{(n-k)!}$

Second rule: when order doesn’t matter divide..when possible.
Sample without replacement and order doesn’t matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
“$n$ choose $k$”

One-to-one rule: equal in number if one-to-one correspondence.
First rule: \[ n_1 \times n_2 \cdots \times n_3. \]

\( k \) Samples with replacement from \( n \) items: \( n^k \).

Sample without replacement: \( \frac{n!}{(n-k)!} \).

Second rule: when order doesn’t matter divide..when possible.

Sample without replacement and order doesn’t matter: \( \binom{n}{k} = \frac{n!}{(n-k)!k!} \).

“\( n \) choose \( k \)”

One-to-one rule: equal in number if one-to-one correspondence.

Sample with replacement and order doesn’t matter: \( \binom{k+n-1}{n-1} \).
Bijection: sums to ’k’ → stars and bars.

\[ S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\} \]
Bijection: sums to 'k' $\rightarrow$ stars and bars.

$$S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\}$$

$$T = \{s \in \{'|', '*'\} : |s| = 7, \text{number of bars in } s = 2\}$$
Bijection: sums to ’k’ → stars and bars.

\[
S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\}
\]

\[
T = \{s \in \{\|, \star\} : |s| = 7, \text{number of bars in } s = 2\}
\]

\[
f((n_1, n_2, n_3)) = \star^{n_1} \| \star^{n_2} \| \star^{n_3}
\]
Bijection: sums to ’k’ → stars and bars.

\[ S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\} \]
\[ T = \{s \in \{'|','\,'\} : |s| = 7, \text{number of bars in } s = 2\} \]
\[ f((n_1, n_2, n_3)) = \star^{n_1} '|' \star^{n_2} '|' \star^{n_3} \]

Bijection:
  argument: unique \((n_1, n_2, n_3)\) from any \(s\).
Bijection: sums to ’k’ → stars and bars.

\[ S = \{ (n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5 \} \]

\[ T = \{ s \in \{ |, \star \} : |s| = 7, \text{number of bars in } s = 2 \} \]

\[ f((n_1, n_2, n_3)) = \star^{n_1} | \star^{n_2} | \star^{n_3} \]

Bijection:
argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\[ |S| = |T| = \binom{7}{2}. \]
Bijection: sums to 'k' $\rightarrow$ stars and bars.

$S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\}$

$T = \{s \in \{'|', '⋆'\} : |s| = 7, \text{number of bars in } s = 2\}$

$f((n_1, n_2, n_3)) = ↑^{n_1} \shuffle \downarrow^{n_2} \shuffle \downarrow^{n_3}$

Bijection:
- argument: unique $(n_1, n_2, n_3)$ from any $s$.

$|S| = |T| = \binom{7}{2}.$

What if someone gets zero?
Bijection: sums to 'k' → stars and bars.

\[ S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\} \]
\[ T = \{s \in \{\text{'|','*'}\} : |s| = 7, \text{number of bars in } s = 2\} \]
\[ f((n_1, n_2, n_3)) = \star^{n_1} \text{'|'} \star^{n_2} \text{'}\star^{n_3} \]

Bijection:
  argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\(|S| = |T| = \binom{7}{2}\).

What if someone gets zero? '***|**' versus '***||**'
Bijection: sums to ’k’ → stars and bars.

\[ S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\} \]
\[ T = \{s \in \{\,'\,'\,\,'\,*\}\ : |s| = 7, \text{number of bars in } s = 2\} \]
\[ f((n_1, n_2, n_3)) = \star^{n_1} \ ' \ ' \ \star^{n_2} \ ' \ ' \ \star^{n_3} \]

Bijection:
- argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\[ |S| = |T| = \binom{7}{2}. \]

What if someone gets zero? ’***|**’ versus ’***|**’

Sure can count number of ’***|**’ + number of ’**|*|**’.
Bijection: sums to 'k' → stars and bars.

\[ S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\} \]
\[ T = \{s \in \{'|','*'\} : |s| = 7, \text{number of bars in } s = 2\} \]
\[ f((n_1, n_2, n_3)) = \star^{n_1} '|| \star^{n_2} '| \star^{n_3} \]

Bijection:
  - argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\[ |S| = |T| = \binom{7}{2}. \]

What if someone gets zero? '***|**' versus '***||**'

Sure can count number of '***|**' + number of '**|*||**'.
  - Second pattern is complicated: bars at least one apart.
Bijection: sums to ’k’ → stars and bars.

\[ S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\} \]

\[ T = \{s \in \{'|', '⋆'\} : |s| = 7, \text{number of bars in } s = 2\} \]

\[ f((n_1, n_2, n_3)) = ⋆^{n_1} '|' ⋆^{n_2} '|' ⋆^{n_3} \]

Bijection:
argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\[ |S| = |T| = \binom{7}{2}. \]

What if someone gets zero? ’***|**’ versus ’***||**’

Sure can count number of ’***|**’ + number of ’**|*|**’.

Second pattern is complicated: bars at least one apart.

For four number which is three bars:
Bijection: sums to ’k’ \( \rightarrow \) stars and bars.

\[
S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\}
\]

\[
T = \{s \in \{\prime, \prime, \star\} : |s| = 7, \text{number of bars in } s = 2\}
\]

\[
f((n_1, n_2, n_3)) = \star^{n_1} \prime \star^{n_2} \prime \star^{n_3}
\]

Bijection:
- argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\[
|S| = |T| = \binom{7}{2}.
\]

What if someone gets zero? ’***|**’ versus ’***|*|**’

Sure can count number of ’***|**’ + number of ’**|*|**’.

Second pattern is complicated: bars at least one apart.

For four number which is three bars:
’***|**’ - two bars on top of each other.
Bijection: sums to ’k’ \( \rightarrow \) stars and bars.

\[
S = \{(n_1, n_2, n_3) : n_1 + n_2 + n_3 = 5\}
\]
\[
T = \{s \in \{',', '*'\} : |s| = 7, \text{number of bars in } s = 2\}
\]
\[
f((n_1, n_2, n_3)) = *^{n_1} ' | ' *^{n_2} ' | ' *^{n_3}
\]

Bijection:
- argument: unique \((n_1, n_2, n_3)\) from any \(s\).

\(|S| = |T| = \binom{7}{2}.
\]

What if someone gets zero? ’***|**’ versus ’***||**’

Sure can count number of ’***|**’ + number of ’**|*|**’.

Second pattern is complicated: bars at least one apart.

For four number which is three bars:
- ’***|**’ - two bars on top of each other. Which two?
Mark what's correct.
(A) ways to split n dollars among k: \( \binom{n+k-1}{k-1} \)
(B) ways to split k dollars among n: \( \binom{k+n-1}{n-1} \)
(C) ways to split 5 dollars among 3: \( \binom{7}{5} \)
(D) ways to split 5 dollars among 3: \( \binom{5+3-1}{3-1} \)
Stars and Bars Poll

Mark what's correct.
(A) ways to split n dollars among k: \( \binom{n+k-1}{k-1} \)
(B) ways to split k dollars among n: \( \binom{k+n-1}{n-1} \)
(C) ways to split 5 dollars among 3: \( \binom{7}{5} \)
(D) ways to split 5 dollars among 3: \( \binom{5+3-1}{3-1} \)
All correct.
Balls in bins.

“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”
Balls in bins.

“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”
“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”
“indistinguishable balls” $\equiv$ “order doesn’t matter”
“only one ball in each bin” $\equiv$ “without replacement”
“k Balls in n bins” ≡ “k samples from n possibilities.”
“indistinguishable balls” ≡ “order doesn’t matter”
“only one ball in each bin” ≡ “without replacement”

5 balls into 10 bins
Balls in bins.

“$k$ Balls in $n$ bins” ≡ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” ≡ “order doesn’t matter”

“only one ball in each bin” ≡ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Balls in bins.

“$k$ Balls in $n$ bins” ≡ “$k$ samples from $n$ possibilities.”
“indistinguishable balls” ≡ “order doesn’t matter”
“only one ball in each bin” ≡ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
  Example: 5 digit numbers.
“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”

“only one ball in each bin” $\equiv$ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
   Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
Balls in bins.

“k Balls in n bins” ≡ “k samples from n possibilities.”
“indistinguishable balls” ≡ “order doesn’t matter”
“only one ball in each bin” ≡ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
   Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
Balls in bins.

“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”

“only one ball in each bin” $\equiv$ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
  
  Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
  
  Example: Poker hands.
“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”

“only one ball in each bin” $\equiv$ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
   Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
   Example: Poker hands.

5 indistinguishable balls into 3 bins
Balls in bins.

“$k$ Balls in $n$ bins” ≡ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” ≡ “order doesn’t matter”

“only one ball in each bin” ≡ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
  Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
  Example: Poker hands.

5 indistinguishable balls into 3 bins
5 samples from 3 possibilities with replacement and no order
Balls in bins.

“$k$ Balls in $n$ bins” $\equiv$ “$k$ samples from $n$ possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”

“only one ball in each bin” $\equiv$ “without replacement”

5 balls into 10 bins
5 samples from 10 possibilities with replacement
   Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
   Example: Poker hands.

5 indistinguishable balls into 3 bins
5 samples from 3 possibilities with replacement and no order
   Dividing 5 dollars among Alice, Bob and Eve.
Mark what's correct.

k Balls in n bins.

dist = distinguishable unique = one ball in each bin.

(A) dist $\implies n^k$
(B) dist, unique $\implies n!/(n-k)!$
(C) indist, unique $\implies \binom{n}{k}$
(D) dist, $\implies n!/(n-k)!$
(E) indist, $\implies \binom{n+k-1}{k-1}$
(F) dist, unique $\implies \binom{n}{k}$
Mark what's correct.

k Balls in n bins.

dis == distinguishable unique = one ball in each bin.

(A) dis $\Rightarrow n^k$
(B) dis, unique $\Rightarrow n!/(n-k)!$
(C) indis, unique $\Rightarrow \binom{n}{k}$
(D) dis, $\Rightarrow n!/(n-k)!$
(E) indis, $\Rightarrow \binom{n+k-1}{k-1}$
(F) dis, unique $\Rightarrow \binom{n}{k}$
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?
**Sum Rule**

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
**Sum rule:** Can sum over disjoint sets.

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands?
Choose 4 cards plus one of 2 jokers!
\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute!
Same as choosing 5 cards from 54
or \[
\binom{54}{5}
\]

Theorem:
\[
\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Algebraic Proof:
Why? Just why? Especially on Tuesday!
Already have a combinatorial proof.
Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
**Sum rule: Can sum over disjoint sets.**
Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers

\[ \binom{52}{5} \]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands?

Choose 4 cards plus one of 2 jokers!

\[ \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3} \]

Wait a minute! Same as choosing 5 cards from 54 or \[ \binom{54}{5} \]

**Theorem:**

\[ \binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3} \]

**Algebraic Proof:**

Why? Just why? Especially on Tuesday!

Already have a combinatorial proof.
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker

\[
\binom{52}{5} + \binom{52}{4}
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands?

Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or \(\binom{54}{5}\).

**Theorem:** \(\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}\).

**Algebraic Proof:**

Why? Just why? Especially on Tuesday!

Already have a combinatorial proof.
Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets. No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]
Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule: Can sum over disjoint sets.**

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands?
Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands?

\[
\binom{52}{5} + 
\]
Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?

Sum rule: Can sum over disjoint sets.
No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[ \binom{52}{5} + \binom{52}{4} + \binom{52}{3}. \]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[ \binom{52}{5} + 2 \times \binom{52}{4} + \]
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}.
\]
Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.
No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? **Choose 4 cards plus one of 2 jokers!**

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute!
Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as
Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}.
\]

Wait a minute! Same as choosing 5 cards from 54
Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or

\[
\binom{54}{5}
\]

**Theorem:** \( \binom{54}{5} \)
Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
**Sum rule: Can sum over disjoint sets.**
No jokers “exclusive” or One Joker “exclusive” or Two Jokers
\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!
\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or
\[
\binom{54}{5}
\]

**Theorem:** \( \binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3} \).
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule: Can sum over disjoint sets.**

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? 
Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}.
\]

Wait a minute! Same as choosing 5 cards from 54 or

\[
\binom{54}{5}.
\]

**Theorem:** \( \binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3} \).

**Algebraic Proof:**
Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?
Sum rule: Can sum over disjoint sets.
No jokers “exclusive” or One Joker “exclusive” or Two Jokers
\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!
\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or
\[
\binom{54}{5}
\]

Theorem: \(\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}\).

Algebraic Proof: Why?
Two indistinguishable jokers in 54 card deck. How many 5 card poker hands? 

**Sum rule: Can sum over disjoint sets.**

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}.
\]

Wait a minute! Same as choosing 5 cards from 54 or

\[
\binom{54}{5}.
\]

**Theorem:** \(\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}\).

**Algebraic Proof:** Why? Just why?
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule:** Can sum over disjoint sets.

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck. How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or

\[
\binom{54}{5}
\]

**Theorem:** \(\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}\).

**Algebraic Proof:** Why? Just why? Especially on Tuesday!
Sum Rule

Two indistinguishable jokers in 54 card deck.
How many 5 card poker hands?

**Sum rule: Can sum over disjoint sets.**
No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.
How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or

\[
\binom{54}{5}
\]

**Theorem:** \(\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}\).

**Algebraic Proof:** Why? Just why? Especially on Tuesday!
Already have a *combinatorial proof.*
Sum Rule

Two indistinguishable jokers in 54 card deck. How many 5 card poker hands?

**Sum rule: Can sum over disjoint sets.**

No jokers “exclusive” or One Joker “exclusive” or Two Jokers

\[
\binom{52}{5} + \binom{52}{4} + \binom{52}{3}.
\]

Two distinguishable jokers in 54 card deck.

How many 5 card poker hands? Choose 4 cards plus one of 2 jokers!

\[
\binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}
\]

Wait a minute! Same as choosing 5 cards from 54 or

\[
\binom{54}{5}
\]

**Theorem:** \(\binom{54}{5} = \binom{52}{5} + 2 \times \binom{52}{4} + \binom{52}{3}\).

**Algebraic Proof:** Why? Just why? Especially on Tuesday!

Already have a **combinatorial proof.**
Combinatorial Proofs.

Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)
Combinatorial Proofs.

**Theorem:** \( \binom{n}{k} = \binom{n}{n-k} \)

**Proof:** How many subsets of size \( k \)?
Combinatorial Proofs.

Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)

Proof: How many subsets of size \( k \)? \( \binom{n}{k} \)
Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)

Proof: How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)

Proof: How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Choose a subset of size \( n - k \)
Combinatorial Proofs.

**Theorem:** \( \binom{n}{k} = \binom{n}{n-k} \)

**Proof:** How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Choose a subset of size \( n - k \) and what’s left out
Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)

Proof: How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Choose a subset of size \( n-k \)
and what’s left out is a subset of size \( k \).
Combinatorial Proofs.

**Theorem:** $\binom{n}{k} = \binom{n}{n-k}$

**Proof:** How many subsets of size $k$? $\binom{n}{k}$

How many subsets of size $k$?
Choose a subset of size $n - k$
and what’s left out is a subset of size $k$.
Choosing a subset of size $k$ is same
Combinatorial Proofs.

Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)

Proof: How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Choose a subset of size \( n - k \)
    and what’s left out is a subset of size \( k \).
Choosing a subset of size \( k \) is same
    as choosing \( n - k \) elements to not take.
Theorem: \( \binom{n}{k} = \binom{n}{n-k} \)

Proof: How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Choose a subset of size \( n-k \)
and what’s left out is a subset of size \( k \).
Choosing a subset of size \( k \) is same
as choosing \( n-k \) elements to not take.
\( \Rightarrow \binom{n}{n-k} \) subsets of size \( k \).
**Theorem:** \( \binom{n}{k} = \binom{n}{n-k} \)

**Proof:** How many subsets of size \( k \)? \( \binom{n}{k} \)

How many subsets of size \( k \)?
Choose a subset of size \( n-k \)
   and what’s left out is a subset of size \( k \).
Choosing a subset of size \( k \) is same
   as choosing \( n-k \) elements to not take.
\[ \Rightarrow \binom{n}{n-k} \] subsets of size \( k \).
Pascal’s Triangle

Row $n$: coefficients of $(1 + x)^n$.

Foil (4 terms): $2^n$ terms: choose 1 or $x$ from each term $(1 + x)$.

Simplify: collect all terms corresponding to $x^k$.

Coefficient of $x^k$: choose $k$ terms with $x$ in product. $(0 \ 0) \ (1 \ 0) \ (1 \ 1) \ (2 \ 0) \ (2 \ 1) \ (2 \ 2) \ (3 \ 0) \ (3 \ 1) \ (3 \ 2) \ (3 \ 3)$

Pascal’s rule $\Rightarrow (n+1 \ k) = (n \ k) + (n \ k-1)$.
Pascal’s Triangle

\[
\begin{array}{cccc}
0 & \\
1 & 1 & \\
\end{array}
\]
Pascal’s Triangle

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
\end{array}
\]

\[\text{Row } n \text{ coefficients of } (1 + x)^n = \left(1 + x\right)\left(1 + x\right) \cdots \left(1 + x\right).\]

\text{Foil (4 terms)}

\[2^n \text{ terms: choose 1 or } x \text{ from each term } (1 + x).\]

\[\text{Simplify: collect all terms corresponding to } x^k.\]

\[\text{Coefficient of } x^k \left(\binom{n}{k}\right): \text{choose } k \text{ terms with } x \text{ in product.}\]

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
\end{array}
\]

\[\text{Pascal’s rule } = \Rightarrow \left(\binom{n+1}{k}\right) = \left(\binom{n}{k}\right) + \left(\binom{n}{k-1}\right).\]
Pascal’s Triangle

0

1 1

1 2 1

1 3 3 1

Row $n$: coefficients of $(1 + x)^n$ = $(1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms) on steroids: $2^n$ terms: choose 1 or $x$ from each term $(1 + x)$.

Simplify: collect all terms corresponding to $x^k$.

Coefficient of $x^k$ \((n\choose k}\): choose $k$ terms with $x$ in product.

\[
\begin{array}{cccccc}
& & & & & \\
& & & & 1 & \\
& & & 1 & & 1 \\
& & 1 & & 2 & & 1 \\
& 1 & & 3 & & 3 & & 1 \\
\end{array}
\]
Pascal’s Triangle

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
\end{array}
\]

Row \(n\): coefficients of \((1 + x)^n\) = \((1 + x)(1 + x) \cdots (1 + x)\).

Foil (4 terms): 2\(^n\) terms: choose 1 or \(x\) from each term \((1 + x)(1 + x) \cdots (1 + x)\).

Simplify: collect all terms corresponding to \(x^k\).

Coefficient of \(x^k\): \(\binom{n}{k}\) terms with \(x\) in product.

Pascal’s rule \(\implies \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}\).
Pascal’s Triangle

0

1 1
1 2 1
1 3 3 1
1 4 6 4 1

Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms): $2^n$ terms: choose 1 or $x$ from each term $(1 + x)$.

Simplify: collect all terms corresponding to $x^k$.

Coefficient of $x^k$: choose $k$ terms with $x$ in product.

\[
\begin{array}{ccccccc}
 0 & 1 & 1 & 0 & 1 & 3 & 3 & 1 \\
0 & 1 & 2 & 0 & 1 & 3 & 3 & 1 \\
0 & 1 & 3 & 0 & 1 & 3 & 3 & 1 \\
0 & 1 & 4 & 0 & 1 & 3 & 3 & 1 \\
\end{array}
\]

Pascal’s rule $\Rightarrow (n+1)^k = n^k + n^{k-1}$. 
Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.
Pascal’s Triangle

\[
\begin{array}{ccccccc}
0 & & & & & & \\
1 & 1 & & & & & \\
1 & 2 & 1 & & & & \\
1 & 3 & 3 & 1 & & & \\
1 & 4 & 6 & 4 & 1 & & \\
\end{array}
\]

Row \( n \): coefficients of \((1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)\).

Foil (4 terms)
Pascal’s Triangle

Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms) on steroids:
Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms) on steroids:

$2^n$ terms:
Pascal’s Triangle

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
\end{array}
\]

Row \( n \): coefficients of \( (1 + x)^n = (1 + x)(1 + x) \cdots (1 + x) \).

Foil (4 terms) on steroids:
\( 2^n \) terms: choose 1 or \( x \) from each term \( (1 + x) \).
Pascal’s Triangle

Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms) on steroids:
- $2^n$ terms: choose 1 or $x$ from each term $(1 + x)$.
Pascal’s Triangle

Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms) on steroids:

$2^n$ terms: choose 1 or $x$ from each term $(1 + x)$.

Simplify: collect all terms corresponding to $x^k$.
Pascal’s Triangle

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
\end{array}
\]

Row \( n \): coefficients of \((1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)\).

Foil (4 terms) on steroids:

- 2\(^n\) terms: choose 1 or \(x\) from each term \((1 + x)\).

Simplify: collect all terms corresponding to \(x^k\).

Coefficient of \(x^k \binom{n}{k} \): choose \(k\) terms with \(x\) in product.
Pascal’s Triangle

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1
\end{array}
\]

Row \(n\): coefficients of \((1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)\).

Foil (4 terms) on steroids:

2\(^n\) terms: choose 1 or \(x\) from each term \((1 + x)\).

Simplify: collect all terms corresponding to \(x^k\).

Coefficient of \(x^k\) \(\binom{n}{k}\): choose \(k\) terms with \(x\) in product.

\[
\begin{array}{c}
\binom{0}{0} \\
\binom{1}{0} \quad \binom{1}{1}
\end{array}
\]
Pascal’s Triangle

\[
\begin{array}{cccccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
\end{array}
\]

Row \( n \): coefficients of \((1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)\).

Foil (4 terms) on steroids:

\( 2^n \) terms: choose 1 or \( x \) from each term \((1 + x)\).

Simplify: collect all terms corresponding to \( x^k \).

Coefficient of \( x^k \) \( \binom{n}{k} \): choose \( k \) terms with \( x \) in product.
Pascal’s Triangle

Row $n$: coefficients of $(1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)$.

Foil (4 terms) on steroids:

$2^n$ terms: choose 1 or $x$ from each term $(1 + x)$.

Simplify: collect all terms corresponding to $x^k$.

Coefficient of $x^k \binom{n}{k}$: choose $k$ terms with $x$ in product.
Pascal’s Triangle

\[
\begin{array}{cccc}
0 \\
1 & 1 \\
1 & 2 & 1 \\
1 & 3 & 3 & 1 \\
1 & 4 & 6 & 4 & 1 \\
\end{array}
\]

Row \( n \): coefficients of \((1 + x)^n = (1 + x)(1 + x) \cdots (1 + x)\).

Foil (4 terms) on steroids:
\( 2^n \) terms: choose 1 or \( x \) from each term \((1 + x)\).

Simplify: collect all terms corresponding to \( x^k \).
Coefficient of \( x^k \) \( \binom{n}{k} \): choose \( k \) terms with \( x \) in product.

Pascal’s rule \( \implies \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)?
Combinatorial Proofs.

Theorem: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

Proof: How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).
Theorem: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

Proof: How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
**Combinatorial Proofs.**

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
Chose first element,
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
  Chose first element, need \( k-1 \) more from remaining \( n \) elements.
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?

How many contain the first element?

Chose first element, need \( k - 1 \) more from remaining \( n \) elements.

\[ \implies \binom{n}{k-1} \]
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
Chose first element, need \( k-1 \) more from remaining \( n \) elements.
\( \implies \binom{n}{k-1} \)

Sum Rule: size of union of disjoint sets of objects.
Without and with first element \( \rightarrow \) disjoint.
So, \( \binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k} \).
Theorem: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

Proof: How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
   Chose first element, need \( k-1 \) more from remaining \( n \) elements.
\[ \implies \binom{n}{k-1} \]
How many don’t contain the first element?

Sum Rule: size of union of disjoint sets of objects.
Without and with first element \( \rightarrow \) disjoint.

So, \( \binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k} \).
Theorem: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

Proof: How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?

How many contain the first element?
- Chose first element, need \( k-1 \) more from remaining \( n \) elements.
  \[ \Rightarrow \binom{n}{k-1} \]

How many don’t contain the first element?
- Need to choose \( k \) elements from remaining \( n \) elts.
Theorem: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

Proof: How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
Chose first element, need \( k - 1 \) more from remaining \( n \) elements.
\( \implies \binom{n}{k-1} \)

How many don’t contain the first element?
Need to choose \( k \) elements from remaining \( n \) elts.
\( \implies \binom{n}{k} \)
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)? How many contain the first element?

Chose first element, need \( k-1 \) more from remaining \( n \) elements. 
\( \implies \binom{n}{k-1} \)

How many don’t contain the first element? Need to choose \( k \) elements from remaining \( n \) elts. 
\( \implies \binom{n}{k} \)
Combinatorial Proofs.

Theorem: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

Proof: How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
  Chose first element, need \( k-1 \) more from remaining \( n \) elements.
  \( \implies \binom{n}{k-1} \)

How many don’t contain the first element?
  Need to choose \( k \) elements from remaining \( n \) elts.
  \( \implies \binom{n}{k} \)

Sum Rule: size of union of disjoint sets of objects.
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?

How many contain the first element?

Chose first element, need \( k-1 \) more from remaining \( n \) elements.

\[ \implies \binom{n}{k-1} \]

How many don’t contain the first element?

Need to choose \( k \) elements from remaining \( n \) elts.

\[ \implies \binom{n}{k} \]

**Sum Rule:** size of union of disjoint sets of objects.

Without and with first element \( \rightarrow \) disjoint.
Combinatorial Proofs.

**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1 \)? \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1 \)?
How many contain the first element?
Chose first element, need \( k-1 \) more from remaining \( n \) elements.
\[ \implies \binom{n}{k-1} \]

How many don’t contain the first element?
Need to choose \( k \) elements from remaining \( n \) elts.
\[ \implies \binom{n}{k} \]

**Sum Rule:** size of union of disjoint sets of objects.
Without and with first element \( \rightarrow \) disjoint.

So, \( \binom{n}{k-1} + \binom{n}{k} \)
**Theorem:** \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).

**Proof:** How many size \( k \) subsets of \( n+1? \) \( \binom{n+1}{k} \).

How many size \( k \) subsets of \( n+1? \)
How many contain the first element?
Chose first element, need \( k - 1 \) more from remaining \( n \) elements.
\[ \implies \binom{n}{k-1} \]

How many don’t contain the first element?
Need to choose \( k \) elements from remaining \( n \) elts.
\[ \implies \binom{n}{k} \]

**Sum Rule:** size of union of disjoint sets of objects.
Without and with first element \( \rightarrow \) disjoint.

So, \( \binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k} \).
Theorem: \( \binom{n}{k} = \binom{n-1}{k-1} + \cdots + \binom{k-1}{k-1} \).
Combinatorial Proof.

**Theorem:** \( \binom{n}{k} = \binom{n-1}{k-1} + \cdots + \binom{k-1}{k-1} \).

**Proof:** Consider size \( k \) subset where \( i \) is the first element chosen.
Combinatorial Proof.

**Theorem:** \( \binom{n}{k} = \binom{n-1}{k-1} + \cdots + \binom{k-1}{k-1} \).

**Proof:** Consider size \( k \) subset where \( i \) is the first element chosen.

\[
\{1, \ldots, i, \ldots, n\}
\]

Must choose \( k - 1 \) elements from \( n - i \) remaining elements.
Theorem: \( \binom{n}{k} = \binom{n-1}{k-1} + \cdots + \binom{k-1}{k-1} \).

Proof: Consider size \( k \) subset where \( i \) is the first element chosen.

\( \{1, \ldots, i, \ldots, n\} \)

Must choose \( k - 1 \) elements from \( n - i \) remaining elements.

\( \implies \binom{n-i}{k-1} \) such subsets.
Theorem: \( \binom{n}{k} = \binom{n-1}{k-1} + \cdots + \binom{k-1}{k-1} \).

Proof: Consider size \( k \) subset where \( i \) is the first element chosen.

\[ \{1, \ldots, i, \ldots, n\} \]

Must choose \( k - 1 \) elements from \( n - i \) remaining elements.

\[ \Rightarrow \binom{n-i}{k-1} \text{ such subsets.} \]

Add them up to get the total number of subsets of size \( k \).
Theorem: \( \binom{n}{k} = \binom{n-1}{k-1} + \cdots + \binom{k-1}{k-1} \).

Proof: Consider size \( k \) subset where \( i \) is the first element chosen.

\[
\{1, \ldots, i, \ldots, n\}
\]

Must choose \( k - 1 \) elements from \( n - i \) remaining elements.

\[ \Rightarrow \binom{n-i}{k-1} \text{ such subsets.} \]

Add them up to get the total number of subsets of size \( k \) which is also \( \binom{n+1}{k} \).
Binomial Theorem: \( x = 1 \)

**Theorem:** \( 2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0} \)
Binomial Theorem: $x = 1$

**Theorem:** $2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0}$

**Proof:** How many subsets of $\{1, \ldots, n\}$?
Binomial Theorem: $x = 1$

**Theorem:** $2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0}$

**Proof:** How many subsets of $\{1, \ldots, n\}$?
Construct a subset with sequence of $n$ choices:
Binomial Theorem: $x = 1$

**Theorem:** $2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0}$

**Proof:** How many subsets of $\{1, \ldots, n\}$?
Construct a subset with sequence of $n$ choices:
   - element $i$ is in or is not in the subset: 2 poss.
Binomial Theorem: $x = 1$

Theorem: $2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0}$

Proof: How many subsets of $\{1, \ldots, n\}$?
Construct a subset with sequence of $n$ choices:
  element $i$ is in or is not in the subset: 2 poss.
First rule of counting: $2 \times 2 \cdots \times 2 = 2^n$ subsets.
Binomial Theorem: \( x = 1 \)

**Theorem:** \( 2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0} \)

**Proof:** How many subsets of \( \{1, \ldots, n\} \)?
Construct a subset with sequence of \( n \) choices:
- element \( i \) **is in** or **is not** in the subset: 2 poss.
First rule of counting: \( 2 \times 2 \cdots \times 2 = 2^n \) subsets.

How many subsets of \( \{1, \ldots, n\} \)?
Theorem: \(2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0}\)

Proof: How many subsets of \(\{1, \ldots, n\}\)?
Construct a subset with sequence of \(n\) choices:
- element \(i\) is in or is not in the subset: 2 poss.
First rule of counting: \(2 \times 2 \cdots \times 2 = 2^n\) subsets.

How many subsets of \(\{1, \ldots, n\}\)?
- \(\binom{n}{i}\) ways to choose \(i\) elts of \(\{1, \ldots, n\}\).
Binomial Theorem: $x = 1$

**Theorem:** $2^n = \binom{n}{n} + \binom{n}{n-1} + \cdots + \binom{n}{0}$

**Proof:** How many subsets of \{1,\ldots, n\}? 
Construct a subset with sequence of $n$ choices: 
  - element $i$ is in or is not in the subset: 2 poss. 
First rule of counting: $2 \times 2 \cdots \times 2 = 2^n$ subsets.

How many subsets of \{1,\ldots, n\}? 
  - $\binom{n}{i}$ ways to choose $i$ elts of \{1,\ldots, n\}. 
Sum over $i$ to get total number of subsets.
Theorem: $2^n = \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n}$

Proof: How many subsets of $\{1, \ldots, n\}$?

Construct a subset with sequence of $n$ choices:
- element $i$ is in or is not in the subset: 2 poss.

First rule of counting: $2 \times 2 \cdots \times 2 = 2^n$ subsets.

How many subsets of $\{1, \ldots, n\}$?
- $\binom{n}{i}$ ways to choose $i$ elts of $\{1, \ldots, n\}$.

Sum over $i$ to get total number of subsets..which is also $2^n$.  □
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets
   by adding number of subsets of size 1, 2, 3, ...
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets by adding number of subsets of size 1, 2, 3,\ldots

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets by adding number of subsets of size 1, 2, 3,\ldots

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**
For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$. 
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets by adding number of subsets of size 1, 2, 3,...

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**
For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$.
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets
   by adding number of subsets of size 1, 2, 3, . . .

Also reasoned about subsets that contained
   or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**
For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$. 

```
In T. \implies |T|
```
**Simple Inclusion/Exclusion**

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|

Used to reason about all subsets by adding number of subsets of size 1, 2, 3, \ldots

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**

For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|.$
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets by adding number of subsets of size 1, 2, 3,\ldots

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**
For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$.

Elements in $S \cap T$ are counted twice.
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets
by adding number of subsets of size 1, 2, 3,...

Also reasoned about subsets that contained
or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**
For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$. 

Elements in $S \cap T$ are counted twice.
Subtract. $\implies -|S \cap T|$
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets
by adding number of subsets of size 1, 2, 3, …

Also reasoned about subsets that contained
or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:**
For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$.

Elements in $S \cap T$ are counted twice.
Subtract. $\implies -|S \cap T|$
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets by adding number of subsets of size 1, 2, 3, ...
Simple Inclusion/Exclusion

Sum Rule: For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets
  by adding number of subsets of size 1, 2, 3, \ldots

Also reasoned about subsets that contained
  or didn’t contain an element. (E.g., first element, first $i$ elements.)
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets by adding number of subsets of size 1, 2, 3,...

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,
$|S \cup T| = |S| + |T| - |S \cap T|$. 

Example: How many 10-digit phone numbers have 7 as their first or second digit?

- $S =$ phone numbers with 7 as first digit.
  $|S| = 10^9$
- $T =$ phone numbers with 7 as second digit.
  $|T| = 10^9$
- $S \cap T =$ phone numbers with 7 as first and second digit.
  $|S \cap T| = 10^8$

Answer: $|S| + |T| - |S \cap T| = 10^9 + 10^9 - 10^8$. 
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets
  by adding number of subsets of size 1, 2, 3,\ldots

Also reasoned about subsets that contained
  or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,

$$|S \cup T| = |S| + |T| - |S \cap T|.$$  

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets
   by adding number of subsets of size 1, 2, 3, …

Also reasoned about subsets that contained
   or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,
$|S \cup T| = |S| + |T| - |S \cap T|$.

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?
$S =$ phone numbers with 7 as first digit.
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets
by adding number of subsets of size 1, 2, 3, \ldots

Also reasoned about subsets that contained
or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,
$|S \cup T| = |S| + |T| - |S \cap T|.$

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?

$S =$ phone numbers with 7 as first digit. $|S| = 10^9$
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets
   by adding number of subsets of size 1, 2, 3,…

Also reasoned about subsets that contained
   or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,
$|S \cup T| = |S| + |T| - |S \cap T|.$

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?

$S =$ phone numbers with 7 as first digit. $|S| = 10^9$

$T =$ phone numbers with 7 as second digit.
Sum Rule: For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets
   by adding number of subsets of size 1, 2, 3,…

Also reasoned about subsets that contained
   or didn’t contain an element. (E.g., first element, first $i$ elements.)

Inclusion/Exclusion Rule: For any $S$ and $T$, $|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

$S$ = phone numbers with 7 as first digit. $|S| = 10^9$

$T$ = phone numbers with 7 as second digit. $|T| = 10^9$. 
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$

Used to reason about all subsets
  by adding number of subsets of size 1, 2, 3, ...

Also reasoned about subsets that contained
  or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,
$|S \cup T| = |S| + |T| - |S \cap T|.$

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?

$S =$ phone numbers with 7 as first digit. $|S| = 10^9$

$T =$ phone numbers with 7 as second digit. $|T| = 10^9.$

$S \cap T =$ phone numbers with 7 as first and second digit.
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, $|S \cup T| = |S| + |T|$
Used to reason about all subsets
   by adding number of subsets of size 1, 2, 3,\ldots

Also reasoned about subsets that contained
or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$,
$|S \cup T| = |S| + |T| - |S \cap T|$.

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?

$S =$ phone numbers with 7 as first digit. $|S| = 10^9$

$T =$ phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T =$ phone numbers with 7 as first and second digit. $|S \cap T| = 10^8$. 
Simple Inclusion/Exclusion

**Sum Rule:** For disjoint sets $S$ and $T$, \(|S \cup T| = |S| + |T|\)

Used to reason about all subsets by adding number of subsets of size 1, 2, 3,\ldots

Also reasoned about subsets that contained or didn’t contain an element. (E.g., first element, first $i$ elements.)

**Inclusion/Exclusion Rule:** For any $S$ and $T$, 
\(|S \cup T| = |S| + |T| - |S \cap T|\).

**Example:** How many 10-digit phone numbers have 7 as their first or second digit?

$S =$ phone numbers with 7 as first digit. $|S| = 10^9$

$T =$ phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T =$ phone numbers with 7 as first and second digit. $|S \cap T| = 10^8$.

Answer: $|S| + |T| - |S \cap T| = 10^9 + 10^9 - 10^8$. 
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Inclusion/Exclusion

$$|A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|.$$  

Idea: For $n = 3$ how many times is an element counted? 
Consider $x \in A_i \cap A_j$. 

Formulaically:
$x$ is in intersection of three sets.
3 for terms of form $|A_i|$, $(3^2)$ for terms of form $|A_i \cap A_j|$, $(3^3)$ for terms of form $|A_i \cap A_j \cap A_k|$. 
Total: $(3^1) - (3^2) + (3^3) = 1$. 
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
Inclusion/Exclusion

$|A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|.$

Idea: For $n = 3$ how many times is an element counted?

Consider $x \in A_i \cap A_j$.

$x$ counted once for $|A_i|$ and once for $|A_j|$.

$x$ subtracted from count once for $|A_i \cap A_j|$.
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
- \( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
- \( x \) subtracted from count once for \( |A_i \cap A_j| \).
Total: \( 2 - 1 = 1 \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?

Consider \( x \in A_i \cap A_j \).

\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted from count once for \( |A_i \cap A_j| \).

Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted from count once for \( |A_i \cap A_j| \).
Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
\( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted once from count once for \( |A_i \cap A_j| \).
Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
\( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
\( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?

Consider \( x \in A_i \cap A_j \).

\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted from count once for \( |A_i \cap A_j| \).

Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)

\( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
\( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
\( x \) added once in \( |A_1 \cap A_2 \cap A_3| \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?

Consider \( x \in A_i \cap A_j \).

- \( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
- \( x \) subtracted from count once for \( |A_i \cap A_j| \).

Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)

- \( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
- \( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
- \( x \) added once in \( |A_1 \cap A_2 \cap A_3| \).

Total: \( 3 - 3 + 1 = 1 \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \]
\[ \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted from count once for \( |A_i \cap A_j| \).
Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
\( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
\( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
\( x \) added once in \( |A_1 \cap A_2 \cap A_3| \).
Total: \( 3 - 3 + 1 = 1 \).

Formulaically: \( x \) is in intersection of three sets.
\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted from count once for \( |A_i \cap A_j| \).
Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
\( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
\( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
\( x \) added once in \( |A_1 \cap A_2 \cap A_3| \).
Total: \( 3 - 3 + 1 = 1 \).

Formulaically: \( x \) is in intersection of three sets.
\( 3 \) for terms of form \( |A_i| \), \( \binom{3}{2} \) for terms of form \( |A_i \cap A_j| \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
- \( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
- \( x \) subtracted from count once for \( |A_i \cap A_j| \).
Total: \( 2 - 1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
- \( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
- \( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
- \( x \) added once in \( |A_1 \cap A_2 \cap A_3| \).
Total: \( 3 - 3 + 1 = 1 \).

Formulaically: \( x \) is in intersection of three sets.
- \( 3 \) for terms of form \( |A_i| \), \( \binom{3}{2} \) for terms of form \( |A_i \cap A_j| \).
- \( \binom{3}{3} \) for terms of form \( |A_i \cap A_j \cap A_k| \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]

Idea: For \( n = 3 \) how many times is an element counted?
Consider \( x \in A_i \cap A_j \).
\( x \) counted once for \( |A_i| \) and once for \( |A_j| \).
\( x \) subtracted from count once for \( |A_i \cap A_j| \).
Total: \( 2 -1 = 1 \).

Consider \( x \in A_1 \cap A_2 \cap A_3 \)
\( x \) counted once in each term: \( |A_1|, |A_2|, |A_3| \).
\( x \) subtracted once in terms: \( |A_1 \cap A_3|, |A_1 \cap A_2|, |A_2 \cap A_3| \).
\( x \) added once in \( |A_1 \cap A_2 \cap A_3| \).
Total: \( 3 - 3 + 1 = 1 \).

Formulaically: \( x \) is in intersection of three sets.
3 for terms of form \( |A_i| \), \( \binom{3}{2} \) for terms of form \( |A_i \cap A_j| \).
\( \binom{3}{3} \) for terms of form \( |A_i \cap A_j \cap A_k| \).
Total: \( \binom{3}{1} - \binom{3}{2} + \binom{3}{3} = 1 \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]

Idea: how many times is each element counted?

Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]

Idea: how many times is each element counted?

Element \(x\) in \(m\) sets: \(x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m}\).

Counted \((\text{m choose i})\) times in \(i\)th summation.
Inclusion/Exclusion

$$|A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|.$$  

Idea: how many times is each element counted?

Element $x$ in $m$ sets: $x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m}$.

Counted $\binom{m}{i}$ times in $i$th summation.

Total: $\binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m}$. 
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?

Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m}. \)

Counted \( \binom{m}{i} \) times in \( i \)th summation.

Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m}. \)
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots \cap A_n|. \]

Idea: how many times is each element counted?
Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).
  Counted \( \binom{m}{i} \) times in \( i \)th summation.
  Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:
\[ (x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m. \]
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \]
\[ \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?
Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).

Counted \( \binom{m}{i} \) times in \( i \)th summation.
Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:
\( (x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m \).
Proof: \( m \) factors in product: \( (x + y)(x + y) \cdots (x + y) \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?

Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).

Counted \( \binom{m}{i} \) times in \( i \)th summation.

Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:

\( (x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m. \)

Proof: \( m \) factors in product: \( (x + y)(x + y) \cdots (x + y) \).

Get a term \( x^{m-i} y^i \) by choosing \( i \) factors to use for \( y \).
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?

Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).

Counted \( \binom{m}{i} \) times in \( i \)th summation.

Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:

\( (x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m \).

Proof: \( m \) factors in product: \( (x + y)(x + y) \cdots (x + y) \).

Get a term \( x^{m-i} y^i \) by choosing \( i \) factors to use for \( y \).

are \( \binom{m}{i} \) ways to choose factors where \( y \) is provided.
Inclusion/Exclusion

\(|A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|.

Idea: how many times is each element counted?
  Element \(x\) in \(m\) sets: \(x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m}\).
  Counted \(\binom{m}{i}\) times in \(i\)th summation.
  Total: \((\binom{m}{1}) - (\binom{m}{2}) + (\binom{m}{3}) \cdots + (-1)^{m-1} (\binom{m}{m}).

Binomial Theorem:
\((x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m.\)
  Proof: \(m\) factors in product: \((x + y)(x + y) \cdots (x + y)\).
  Get a term \(x^{m-i} y^i\) by choosing \(i\) factors to use for \(y\).
  are \(\binom{m}{i}\) ways to choose factors where \(y\) is provided.
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| - \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?

Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).

Counted \( \binom{m}{i} \) times in \( i \)th summation.

Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:

\[ (x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m. \]

Proof: \( m \) factors in product: \((x + y)(x + y)\cdots(x + y)\).

Get a term \( x^{m-i} y^i \) by choosing \( i \) factors to use for \( y \).

are \( \binom{m}{i} \) ways to choose factors where \( y \) is provided.

For \( x = 1, y = -1 \),

\[ \Rightarrow 1 = \binom{m}{0} - \binom{m}{1} + \binom{m}{2} - \cdots + (-1)^m \binom{m}{m}. \]
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?
Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).
Counted \( \binom{m}{i} \) times in \( i \)th summation.
Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:
\( (x+y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m. \)
Proof: \( m \) factors in product: \( (x+y)(x+y)\cdots(x+y) \).
Get a term \( x^{m-i} y^i \) by choosing \( i \) factors to use for \( y \).
are \( \binom{m}{i} \) ways to choose factors where \( y \) is provided.

For \( x = 1, y = -1 \),
\[ 0 = (1 - 1)^m = \binom{m}{0} - \binom{m}{1} + \binom{m}{2} \cdots + (-1)^m \binom{m}{m} \]
Inclusion/Exclusion

\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?

Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).

Counted \( \binom{m}{i} \) times in \( i \)th summation.

Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:

\[ (x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots \binom{m}{m} y^m. \]

Proof: \( m \) factors in product: \( (x + y)(x + y) \cdots (x + y) \).

Get a term \( x^{m-i} y^i \) by choosing \( i \) factors to use for \( y \).

are \( \binom{m}{i} \) ways to choose factors where \( y \) is provided.

For \( x = 1, y = -1 \),

\[ 0 = (1 - 1)^m = \binom{m}{0} - \binom{m}{1} + \binom{m}{2} \cdots + (-1)^m \binom{m}{m} \]

\[ \implies 1 = \binom{m}{0} = \binom{m}{1} - \binom{m}{2} \cdots + (-1)^{m-1} \binom{m}{m}. \]
\[ |A_1 \cup \cdots \cup A_n| = \sum_i |A_i| - \sum_{i,j} |A_i \cap A_j| + \sum_{i,j,k} |A_i \cap A_j \cap A_k| \cdots (-1)^n |A_1 \cap \cdots A_n|. \]

Idea: how many times is each element counted?
- Element \( x \) in \( m \) sets: \( x \in A_{i_1} \cap A_{i_2} \cdots \cap A_{i_m} \).
- Counted \( \binom{m}{i} \) times in \( i \)th summation.
- Total: \( \binom{m}{1} - \binom{m}{2} + \binom{m}{3} \cdots + (-1)^{m-1} \binom{m}{m} \).

Binomial Theorem:
\[(x + y)^m = \binom{m}{0} x^m + \binom{m}{1} x^{m-1} y + \binom{m}{2} x^{m-2} y^2 + \cdots + \binom{m}{m} y^m.\]

Proof: \( m \) factors in product: \((x + y)(x + y) \cdots (x + y)\).
- Get a term \( x^{m-i} y^i \) by choosing \( i \) factors to use for \( y \).
- are \( \binom{m}{i} \) ways to choose factors where \( y \) is provided.

For \( x = 1, y = -1 \),
\[
0 = (1 - 1)^m = \binom{m}{0} - \binom{m}{1} + \binom{m}{2} \cdots + (-1)^m \binom{m}{m}
\]
\[
\implies 1 = \binom{m}{0} = \binom{m}{1} - \binom{m}{2} \cdots + (-1)^{m-1} \binom{m}{m}.
\]

Each element counted once!
Summary.

First Rule of counting:
Summary.

First Rule of counting: Objects from a sequence of choices:
Summary.

First Rule of counting: Objects from a sequence of choices:

\[ n_i \text{ possibilities for } i\text{th choice :} \]
Summary.

First Rule of counting: Objects from a sequence of choices: 

\( n_i \) possibilities for \( i \)th choice : \( n_1 \times n_2 \times \cdots \times n_k \) objects.
Summary.

First Rule of counting: Objects from a sequence of choices: 
\[ n_i \text{ possibilities for } i\text{'th choice} : n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting:
Summary.

First Rule of counting: Objects from a sequence of choices: 
$n_i$ possibilities for $i$th choice: $n_1 \times n_2 \times \cdots \times n_k$ objects.

Second Rule of counting: If order does not matter.
Summary.

First Rule of counting: Objects from a sequence of choices: 
\[ n_i \text{ possibilities for } i\text{th choice : } n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter. 
Count with order:

---

RHS: Number of subsets of \( n+1 \) items size \( k \).
LHS: \( (n \choose k-1) + (n \choose k) \).
Disjoint – so add!
Summary.

First Rule of counting: Objects from a sequence of choices: $n_i$ possibilities for $i$th choice: $n_1 \times n_2 \times \cdots \times n_k$ objects.

Second Rule of counting: If order does not matter. Count with order: Divide number of orderings.
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i\text{th choice} : n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).
Summary.

First Rule of counting: Objects from a sequence of choices:

\( n_1 \) possibilities for \( i \)th choice : \( n_1 \times n_2 \times \cdots \times n_k \) objects.

Second Rule of counting: If order does not matter.

Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars:
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i\text{th choice : } n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Summary.

First Rule of counting: Objects from a sequence of choices:  
\( n_i \) possibilities for \( i \)th choice: \( n_1 \times n_2 \times \cdots \times n_k \) objects.

Second Rule of counting: If order does not matter.  
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).  
Order doesn’t matter:
Summary.

First Rule of counting: Objects from a sequence of choices: 
\( n_i \) possibilities for \( i \)th choice : \( n_1 \times n_2 \times \cdots \times n_k \) objects.

Second Rule of counting: If order does not matter. 
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \). 
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).
Summary.

First Rule of counting: Objects from a sequence of choices:

\[ n_i \text{ possibilities for } i\text{th choice} : n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.

Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).

Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Summary.

First Rule of counting: Objects from a sequence of choices:

\[ n_i \text{ possibilities for } i\text{th choice : } n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.

Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).

Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.

Add number of each subtract intersection of sets.
First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i\text{th choice} : n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersection of sets.
Sum Rule: If disjoint just add.
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i\text{th choice : } n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersection of sets.
Sum Rule: If disjoint just add.
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i^{th} \text{ choice : } n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersection of sets.
Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.
Summary.

First Rule of counting: Objects from a sequence of choices:
\( n_i \) possibilities for \( i \)th choice: \( n_1 \times n_2 \times \cdots \times n_k \) objects.

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersection of sets.

Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.
Pascal’s Triangle Example: \( \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \).
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i\text{th choice } : n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersect of sets.
Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.
Pascal’s Triangle Example: \( \binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1} \).
RHS: Number of subsets of \( n+1 \) items size \( k \).
Summary.

First Rule of counting: Objects from a sequence of choices:

\[ n_i \text{ possibilities for } i\text{th choice} : n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.

Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).

Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.

Add number of each subtract intersection of sets.

Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.

Pascal’s Triangle Example: \( \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \).

RHS: Number of subsets of \( n+1 \) items size \( k \).

LHS: \( \binom{n}{k-1} \) counts subsets of \( n+1 \) items with first item.
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \] possibilities for \( i \)th choice : \( n_1 \times n_2 \times \cdots \times n_k \) objects.

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersection of sets.
Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.
Pascal’s Triangle Example: \( \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \).
RHS: Number of subsets of \( n+1 \) items size \( k \).
LHS: \( \binom{n}{k-1} \) counts subsets of \( n+1 \) items with first item.
\( \binom{n}{k} \) counts subsets of \( n+1 \) items without first item.
Summary.

First Rule of counting: Objects from a sequence of choices:
\[ n_i \text{ possibilities for } i\text{th choice : } n_1 \times n_2 \times \cdots \times n_k \text{ objects.} \]

Second Rule of counting: If order does not matter.
Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
Add number of each subtract intersection of sets.

Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.
Pascal’s Triangle Example: \( \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \).
RHS: Number of subsets of \( n+1 \) items size \( k \).
LHS: \( \binom{n}{k-1} \) counts subsets of \( n+1 \) items with first item.
\( \binom{n}{k} \) counts subsets of \( n+1 \) items without first item.
Disjoint
Summary.

First Rule of counting: Objects from a sequence of choices:
   \( n_i \) possibilities for \( i \)th choice : \( n_1 \times n_2 \times \cdots \times n_k \) objects.

Second Rule of counting: If order does not matter.
   Count with order: Divide number of orderings. Typically: \( \binom{n}{k} \).

Stars and Bars: Sample \( k \) objects with replacement from \( n \).
   Order doesn’t matter: Typically: \( \binom{n+k-1}{n-1} = \binom{n+k-1}{k} \).

Inclusion/Exclusion: two sets of objects.
   Add number of each subtract intersection of sets.
Sum Rule: If disjoint just add.

Combinatorial Proofs: Identity from counting same in two ways.
Pascal’s Triangle Example: \( \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \).
   RHS: Number of subsets of \( n+1 \) items size \( k \).
   LHS: \( \binom{n}{k-1} \) counts subsets of \( n+1 \) items with first item.
   \( \binom{n}{k} \) counts subsets of \( n+1 \) items without first item.
   Disjoint – so add!
Midterm Review

Now...
First there was logic...

A statement is true or false.
First there was logic...

A statement is true or false.
Statements?

\[ 3 = 4 - 1 \]

Statement!

\[ 3 = 5 \]

Statement!

Not a statement!

\[ n = 3 \]

Not a statement...

but a predicate.

Predicate: Statement with free variable(s).

Example:

\[ x = 3 \]

Given a value for \( x \), becomes a statement.

Predicate?

\[ n > 3 \]

Predicate:

\[ P(n) \]

\[ x = y \]

Predicate:

\[ P(x, y) \]

No.

An expression, not a statement.

Quantifiers:

\[ \forall x \ P(x) \]

For every \( x \), \( P(x) \) is true.

\[ \exists x \ P(x) \]

There exists an \( x \), where \( P(x) \) is true.

\[ \forall n \in \mathbb{N}, \ n^2 \geq n \]

\[ \forall x \in \mathbb{R} \left( \exists y \in \mathbb{R} \ y > x \right) \]
First there was logic...

A statement is true or false.
Statements?
3 = 4 − 1 ?
First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!
A statement is true or false.

Statements?
3 = 4 − 1 ? Statement!
3 = 5 ?
First there was logic...

A statement is true or false.
Statements?
  $3 = 4 - 1$ ? Statement!
  $3 = 5$ ? Statement!
First there was logic...

A statement is true or false.
Statements?
  $3 = 4 - 1 \ ?$ Statement!
  $3 = 5 \ ?$ Statement!
  $3 \ ?$

First there was logic...

A statement is true or false.

Statements?

3 = 4 - 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!
First there was logic...

A statement is true or false.
Statements?

3 = 4 − 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!
n = 3 ?
First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$ ? Statement!
$3 = 5$ ? Statement!
$3$ ? Not a statement!
$n = 3$ ? Not a statement...
First there was logic...

A statement is true or false.
Statements?
  \( 3 = 4 - 1 \) ? Statement!
  \( 3 = 5 \) ? Statement!
  \( 3 \) ? Not a statement!
  \( n = 3 \) ? Not a statement...but a predicate.
First there was logic...

A statement is true or false.
Statements?
\[ 3 = 4 - 1 \ ? \text{ Statement!} \]
\[ 3 = 5 \ ? \text{ Statement!} \]
\[ 3 \ ? \text{ Not a statement!} \]
\[ n = 3 \ ? \text{ Not a statement...but a predicate.} \]

**Predicate**: Statement with free variable(s).
First there was logic...

A statement is true or false.

Statements?

3 = 4 – 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!

n = 3 ? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: x = 3
First there was logic...

A statement is true or false.
Statements?
  $3 = 4 - 1 \ ?$ Statement!
  $3 = 5 \ ?$ Statement!
  $3 \ ?$ Not a statement!
  $n = 3 \ ?$ Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).
Example: $x = 3$
  Given a value for $x$, becomes a statement.
First there was logic...

A statement is true or false.

Statements?
  \[3 = 4 - 1\] ? Statement!
  \[3 = 5\] ? Statement!
  \[3\] ? Not a statement!
  \[n = 3\] ? Not a statement...but a predicate.

Predicate: **Statement with free variable(s).**

Example: \[x = 3\]
  Given a value for \(x\), becomes a statement.

Predicate?
First there was logic...

A statement is true or false.

Statements?

3 = 4 − 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!

n = 3 ? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: \( x = 3 \)

Given a value for \( x \), becomes a statement.

Predicate?

\( n > 3 \) ?
A statement is true or false.

Statements?
3 = 4 − 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!
n = 3 ? Not a statement...but a predicate.

Predicate: Statement with free variable(s).
Example: \( x = 3 \)
Given a value for \( x \), becomes a statement.

Predicate?
n > 3 ? Predicate: \( P(n) \)!
First there was logic...

A statement is true or false.

Statements?

3 = 4 − 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!

n = 3 ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).

Example: \( x = 3 \)
Given a value for \( x \), becomes a statement.

Predicate?

\( n > 3 \) ? Predicate: \( P(n)! \)

\( x = y \)?
First there was logic...

A statement is true or false.

Statements?

\[ 3 = 4 - 1 \]
Statement!

\[ 3 = 5 \]
Statement!

\[ 3 \]
Not a statement!

\[ n = 3 \]
Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: \( x = 3 \)

Given a value for \( x \), becomes a statement.

Predicate?

\[ n > 3 \]
Predicate: \( P(n) \)!

\[ x = y \]
Predicate: \( P(x, y) \)!
First there was logic...

A statement is true or false.
Statements?
  \(3 = 4 - 1\) ? Statement!
  \(3 = 5\) ? Statement!
  \(3\) ? Not a statement!
  \(n = 3\) ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).
  Example: \(x = 3\)
  Given a value for \(x\), becomes a statement.

Predicate?
  \(n > 3\) ? Predicate: \(P(n)\)!
  \(x = y\)? Predicate: \(P(x, y)\)!
  \(x + y\)?
First there was logic...

A statement is true or false.
Statements?
  \(3 = 4 - 1\) ? Statement!
  \(3 = 5\) ? Statement!
  \(3\) ? Not a statement!
  \(n = 3\) ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).
  Example: \(x = 3\)
    Given a value for \(x\), becomes a statement.

Predicate?
  \(n > 3\) ? Predicate: \(P(n)\)!
  \(x = y\)? Predicate: \(P(x, y)\)!
  \(x + y\)? No.
First there was logic...

A statement is true or false.

Statements?
3 = 4 − 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!
n = 3 ? Not a statement...but a predicate.

Predicate: Statement with free variable(s).
Example: \( x = 3 \)
Given a value for \( x \), becomes a statement.

Predicate?
n > 3 ? Predicate: \( P(n) \)!
x = y? Predicate: \( P(x, y) \)!
x + y? No. An expression, not a statement.
First there was logic...

A statement is true or false.

Statements?

- $3 = 4 - 1$ ? Statement!
- $3 = 5$ ? Statement!
- $3$ ? Not a statement!
- $n = 3$ ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).

Example: $x = 3$

- Given a value for $x$, becomes a statement.

Predicate?

- $n > 3$ ? Predicate: $P(n)$!
- $x = y$? Predicate: $P(x, y)$!
- $x + y$? No. An expression, not a statement.

**Quantifiers:**
A statement is true or false.
Statements?
   $3 = 4 - 1$ ? Statement!
   $3 = 5$ ? Statement!
   $3$ ? Not a statement!
   $n = 3$ ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).
   Example: $x = 3$
      Given a value for $x$, becomes a statement.

Predicate?
   $n > 3$ ? Predicate: $P(n)$!
   $x = y$? Predicate: $P(x, y)$!
   $x + y$? No. An expression, not a statement.

**Quantifiers:**
   $(\forall x)\ P(x)$.
A statement is true or false.

Statements?

3 = 4 − 1 ? Statement!
3 = 5 ? Statement!
3 ? Not a statement!

n = 3 ? Not a statement... but a predicate.

Predicate: Statement with free variable(s).

Example: x = 3
Given a value for x, becomes a statement.

Predicate?

n > 3 ? Predicate: P(n)!
x = y? Predicate: P(x, y)!
x + y? No. An expression, not a statement.

Quantifiers:

(∀x) P(x). For every x, P(x) is true.
First there was logic...

A statement is true or false.
Statements?
  $3 = 4 - 1$? Statement!
  $3 = 5$? Statement!
  $3$? Not a statement!
  $n = 3$? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).
Example: $x = 3$
  Given a value for $x$, becomes a statement.

Predicate?
  $n > 3$? Predicate: $P(n)$!
  $x = y$? Predicate: $P(x, y)$!
  $x + y$? No. An expression, not a statement.

**Quantifiers:**
  $(\forall x) P(x)$. For every $x$, $P(x)$ is true.
  $(\exists x) P(x)$. 
First there was logic...

A statement is true or false.

Statements?
  3 = 4 − 1 ? Statement!
  3 = 5 ? Statement!
  3 ? Not a statement!
  $n = 3$ ? Not a statement...but a predicate.

**Predicate**: Statement with free variable(s).

Example: $x = 3$
  Given a value for $x$, becomes a statement.

Predicate?
  $n > 3$ ? Predicate: $P(n)$!
  $x = y$ ? Predicate: $P(x, y)$!
  $x + y$? No. An expression, not a statement.

**Quantifiers**:
  $(\forall x) \ P(x)$. For every $x$, $P(x)$ is true.
  $(\exists x) \ P(x)$. There exists an $x$, where $P(x)$ is true.
A statement is true or false.

Statements?

$3 = 4 - 1$ ? Statement!

$3 = 5$ ? Statement!

$3$ ? Not a statement!

$n = 3$ ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).

Example: $x = 3$

Given a value for $x$, becomes a statement.

Predicate?

$n > 3$ ? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a statement.

**Quantifiers:**

$(\forall x) P(x)$. For every $x$, $P(x)$ is true.

$(\exists x) P(x)$. There exists an $x$, where $P(x)$ is true.

$(\forall n \in N), n^2 \geq n$. 
First there was logic...

A statement is true or false. Statements?

\[ 3 = 4 - 1 \ ? \text{ Statement!} \]
\[ 3 = 5 \ ? \text{ Statement!} \]
\[ 3 \ ? \text{ Not a statement!} \]
\[ n = 3 \ ? \text{ Not a statement... but a predicate.} \]

**Predicate:** Statement with free variable(s).

Example: \( x = 3 \)

Given a value for \( x \), becomes a statement.

Predicate?

\[ n > 3 \ ? \text{ Predicate: } P(n)! \]
\[ x = y? \text{ Predicate: } P(x, y)! \]
\[ x + y? \text{ No. An expression, not a statement.} \]

**Quantifiers:**

\( \forall x \) \( P(x) \).

For every \( x \), \( P(x) \) is true.

\( \exists x \) \( P(x) \).

There exists an \( x \), where \( P(x) \) is true.

\( \forall n \in \mathbb{N} \), \( n^2 \geq n \).

\( \forall x \in \mathbb{R} \)(\( \exists y \in \mathbb{R} \)) \( y > x \).
A statement is true or false.

Statements?
- $3 = 4 - 1$ ? Statement!
- $3 = 5$ ? Statement!
- $3$ ? Not a statement!
- $n = 3$ ? Not a statement...but a predicate.

**Predicate:** Statement with free variable(s).

Example: $x = 3$
- Given a value for $x$, becomes a statement.

Predicate?
- $n > 3$ ? Predicate: $P(n)$!
- $x = y$? Predicate: $P(x, y)$!
- $x + y$? No. An expression, not a statement.

**Quantifiers:**
- $(\forall x) P(x)$. For every $x$, $P(x)$ is true.
- $(\exists x) P(x)$. There exists an $x$, where $P(x)$ is true.

$(\forall n \in N), n^2 \geq n.$
$(\forall x \in R)(\exists y \in R)y > x$. 
Connecting Statements

\[ A \land B, \ A \lor B, \ \neg A. \]
Connecting Statements

\[ A \land B, \ A \lor B, \ \neg A. \]

You got this!
Connecting Statements

\[ A \land B, A \lor B, \neg A. \]

You got this!

Propositional Expressions and Logical Equivalence
Connecting Statements

\[ A \land B, \, A \lor B, \, \neg A. \]

You got this!

Propositional Expressions and Logical Equivalence

\[(A \implies B) \equiv (\neg A \lor B)\]
Connecting Statements

$A \land B$, $A \lor B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \lor B)$

$\neg(A \lor B) \equiv (\neg A \land \neg B)$
Connecting Statements

\[ A \land B, \ A \lor B, \ \neg A. \]

You got this!

Propositional Expressions and Logical Equivalence

\[ (A \implies B) \equiv (\neg A \lor B) \]
\[ \neg(A \lor B) \equiv (\neg A \land \neg B) \]
Connecting Statements

$A \land B$, $A \lor B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \lor B)$

$\neg(A \lor B) \equiv (\neg A \land \neg B)$

Proofs: truth table or manipulation of known formulas.
$A \land B$, $A \lor B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$\quad (A \implies B) \equiv (\neg A \lor B)$
$\quad \neg(A \lor B) \equiv (\neg A \land \neg B)$

Proofs: truth table or manipulation of known formulas.

$\quad (\forall x)(P(x) \land Q(x)) \equiv (\forall x)P(x) \land (\forall x)Q(x)$
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$ $\implies a^2 = 4k^2$. What is even? $a^2 = 2(2k^2)$.

Integers closed under multiplication! $a^2$ is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.

Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P \neg P \implies false$ $\neg P \implies R \land \neg R$.

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist.

Example: rogue couple does not exist.
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even?

Contrapositive: \( P \implies Q \)

Example: \( a \) is odd \( \implies a \) is odd.

Contrapositive: \( a^2 \) is even.

Contradiction: \( P \implies \neg P \)

\( \neg P \implies R \land \neg R \)

Useful for prove something does not exist:

Example: rational representation of \( \sqrt{2} \) does not exist.

Example: finite set of primes does not exist.

Example: rogue couple does not exist.
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$. 

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.

Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P \neg P \implies \text{false}$

$\neg P \implies R \land \neg R$.

Useful for prove something does not exist:
Example: rational representation of $\sqrt{2}$ does not exist.
Example: finite set of primes does not exist.
Example: rogue couple does not exist.
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)

\( a^2 = 4k^2 \).

What is even?

\( a^2 = 2(2k^2) \)
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.

Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P \neg P \implies \text{false}$

$\neg P \implies R \land \neg R$.

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist.

Example: rogue couple does not exist.
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)
\[ a^2 = 4k^2. \]
What is even?
\[ a^2 = 2(2k^2) \]

Integers closed under multiplication!
\( a^2 \) is even.
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

$a^2$ is even.
..and then proofs...

Direct: \( P \implies Q \)
Example: \( a \) is even \( \implies \) \( a^2 \) is even.
   Approach: What is even? \( a = 2k \)
   \( a^2 = 4k^2 \).
   What is even?
   \( a^2 = 2(2k^2) \)
   Integers closed under multiplication!
   \( a^2 \) is even.

Contrapositive: \( P \implies Q \)
..and then proofs...

**Direct:** $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

$a^2$ is even.

**Contrapositive:** $P \implies Q$ or $\neg Q \implies \neg P$. 
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

$a^2$ is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)

\[
a^2 = 4k^2.
\]

What is even?

\[
a^2 = 2(2k^2)
\]

Integers closed under multiplication!

\( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).

Example: \( a^2 \) is odd \( \implies a \) is odd.

\( \text{Contrapositive: } a \text{ is even } \implies a^2 \text{ is even.} \)
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)
\[ a^2 = 4k^2. \]

What is even?
\[ a^2 = 2(2k^2) \]

Integers closed under multiplication!
\( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).

Example: \( a^2 \) is odd \( \implies a \) is odd.

Contrapositive: \( a \) is even \( \implies a^2 \) is even.
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

$a^2$ is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.

Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P$
..and then proofs...

Direct: \( P \implies Q \)
Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)
\[
a^2 = 4k^2.
\]
What is even?
\[
a^2 = 2(2k^2)
\]
Integers closed under multiplication!
\( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).
Example: \( a^2 \) is odd \( \implies \) \( a \) is odd.
Contrapositive: \( a \) is even \( \implies a^2 \) is even.

Contradiction: \( P \)
\[
\neg P \implies \text{false}
\]
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

$a^2$ is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.

Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P$

$\neg P \implies \text{false}$

$\neg P \implies R \land \neg R$
..and then proofs...

Direct: $P \implies Q$
   
   Example: $a$ is even $\implies a^2$ is even.

   Approach: What is even? $a = 2k$
   
   $a^2 = 4k^2$.

   What is even?
   
   $a^2 = 2(2k^2)$

   Integers closed under multiplication!

   $a^2$ is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

   Example: $a^2$ is odd $\implies a$ is odd.

   Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P$

   $\neg P \implies \text{false}$

   $\neg P \implies R \land \neg R$

   Useful for prove something does not exist:
..and then proofs...

Direct: \( P \implies Q \)
- Example: \( a \) is even \( \implies a^2 \) is even.
  - Approach: What is even? \( a = 2k \)
    \[ a^2 = 4k^2. \]
  - What is even?
    \[ a^2 = 2(2k^2) \]
  - Integers closed under multiplication!
    \( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).
- Example: \( a^2 \) is odd \( \implies a \) is odd.
  - Contrapositive: \( a \) is even \( \implies a^2 \) is even.

Contradiction: \( P \)
- \( \neg P \implies \text{false} \)
- \( \neg P \implies R \land \neg R \)

Useful for prove something does not exist:
- Example: rational representation of \( \sqrt{2} \)
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)

\[ a^2 = 4k^2. \]

What is even?

\[ a^2 = 2(2k^2) \]

Integers closed under multiplication!

\( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).

Example: \( a^2 \) is odd \( \implies a \) is odd.

Contrapositive: \( a \) is even \( \implies a^2 \) is even.

Contradiction: \( P \)

\( \neg P \implies \text{false} \)

\( \neg P \implies R \land \neg R \)

Useful for prove something does not exist:

Example: rational representation of \( \sqrt{2} \) does not exist.
..and then proofs...

Direct: $P \implies Q$

Example: $a$ is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

$a^2$ is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: $a^2$ is odd $\implies a$ is odd.

Contrapositive: $a$ is even $\implies a^2$ is even.

Contradiction: $P$

$\neg P \implies \text{false}$
$\neg P \implies R \land \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes
..and then proofs...

Direct: \( P \implies Q \)

Example: \( a \) is even \( \implies a^2 \) is even.

Approach: What is even? \( a = 2k \)
\[ a^2 = 4k^2. \]

What is even?
\[ a^2 = 2(2k^2) \]
Integers closed under multiplication!
\( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).

Example: \( a^2 \) is odd \( \implies a \) is odd.

Contrapositive: \( a \) is even \( \implies a^2 \) is even.

Contradiction: \( P \)

\[ \neg P \implies \text{false} \]
\[ \neg P \implies R \land \neg R \]

Useful for prove something does not exist:
Example: rational representation of \( \sqrt{2} \) does not exist.
Example: finite set of primes does not exist.
..and then proofs...

Direct: \( P \implies Q \)
   Example: \( a \) is even \( \implies a^2 \) is even.
   Approach: What is even? \( a = 2k \)
               \( a^2 = 4k^2 \).
   What is even?
               \( a^2 = 2(2k^2) \)
   Integers closed under multiplication!
   \( a^2 \) is even.

Contrapositive: \( P \implies Q \) or \( \neg Q \implies \neg P \).
   Example: \( a^2 \) is odd \( \implies a \) is odd.
   Contrapositive: \( a \) is even \( \implies a^2 \) is even.

Contradiction: \( P \)
               \( \neg P \implies \text{false} \)
               \( \neg P \implies R \land \neg R \)

Useful for prove something does not exist:
   Example: rational representation of \( \sqrt{2} \) does not exist.
   Example: finite set of primes does not exist.
   Example: rogue couple does not exist.
...jumping forward..

Contradiction in induction:
...jumping forward..

Contradiction in induction:
contradict place where induction step doesn’t hold.
...jumping forward..

Contradiction in induction:
contradict place where induction step doesn’t hold.

Well Ordering Principle.
...jumping forward..

Contradiction in induction:
contradict place where induction step doesn’t hold.

Well Ordering Principle.
Stable Marriage:
...jumping forward...

Contradiction in induction:
contradict place where induction step doesn’t hold.

Well Ordering Principle.
Stable Marriage:
first day where candidate gets worse job on string.
...jumping forward..

Contradiction in induction:
contradict place where induction step doesn’t hold.

Well Ordering Principle.
Stable Marriage:
first day where candidate gets worse job on string.
first day where any job is rejected by optimal candidate.
...jumping forward..

Contradiction in induction:
   contradict place where induction step doesn’t hold.

Well Ordering Principle.
Stable Marriage:
   first day where candidate gets worse job on string.
   first day where any job is rejected by optimal candidate.
Do not exist.
Contradiction in induction:
   contradict place where induction step doesn’t hold.

Well Ordering Principle.
Stable Marriage:
   first day where candidate gets worse job on string.
   first day where any job is rejected by optimal candidate.
Do not exist.
...jumping forward..

Contradiction in induction:
contradict place where induction step doesn’t hold.

Well Ordering Principle.
Stable Marriage:
first day where candidate gets worse job on string.
first day where any job is rejected by optimal candidate.
Do not exist.
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n + 1)) \equiv (\forall n \in N) P(n). \]
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n)). \]

**Thm:** For all \( n \geq 1 \), \( 8 | 3^{2n} - 1 \).
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n)). \]

**Thm:** For all \( n \geq 1, \) \( 8 \mid 3^{2n} - 1. \)

Induction on \( n. \)
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in N) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 | 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 | 3^2 - 1 \).
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n + 1)) \equiv (\forall n \in \mathbb{N}) P(n)). \]

**Thm:** For all \( n \geq 1, 8|3^{2n} - 1. \)

Induction on \( n \).

Base: \( 8|3^2 - 1. \)
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n)) \]  

**Thm:** For all \( n \geq 1 \), \( 8 \mid 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 \mid 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 \mid 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 \mid 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).

Induction Step: Prove \( P(n+1) \)
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n)). \]

**Thm:** For all \( n \geq 1, 8 \mid 3^{2n} - 1. \)

Induction on \( n. \)

Base: \( 8 \mid 3^2 - 1. \)

Induction Hypothesis: Assume \( P(n): \) True for some \( n. \)

Induction Step: Prove \( P(n+1) \)

\[ 3^{2n+2} - 1 = \]
...and then induction...

\[ P(0) \land ((\forall n)P(n) \implies P(n + 1)) \equiv (\forall n \in \mathbb{N}) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 \mid 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 \mid 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).

Induction Step: Prove \( P(n + 1) \)

\[ 3^{2n+2} - 1 = 9(3^{2n}) - 1 \]
...and then induction...

\[
P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n).
\]

**Thm:** For all \( n \geq 1 \), \( 8 | 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 | 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).
\[
(3^{2n} - 1 = 8d)
\]

Induction Step: Prove \( P(n+1) \)
\[
3^{2n+2} - 1 = 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)}
\]
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 \mid 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 \mid 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).

\( (3^{2n} - 1 = 8d) \)

Induction Step: Prove \( P(n+1) \)

\[ 3^{2n+2} - 1 = 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \]
\[ = 9(8d + 1) - 1 \]
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 | 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 | 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).
\( 3^{2n} - 1 = 8d \)

Induction Step: Prove \( P(n+1) \)

\[
3^{2n+2} - 1 = 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)}
\]
\[
= 9(8d + 1) - 1
\]
\[
= 72d + 8
\]
\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 | 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 | 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).

\[ 3^{2n} - 1 = 8d \]

Induction Step: Prove \( P(n+1) \)

\[
3^{2n+2} - 1 = 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\
= 9(8d + 1) - 1 \\
= 72d + 8 \\
= 8(9d + 1)
\]
...and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in N) P(n). \]

**Thm:** For all \( n \geq 1, 8|3^{2n} - 1. \)

Induction on \( n. \)

Base: \( 8|3^2 - 1. \)

Induction Hypothesis: Assume \( P(n): \) True for some \( n. \)

\( 3^{2n} - 1 = 8d \)

Induction Step: Prove \( P(n+1) \)

\[ 3^{2n+2} - 1 = 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \]
\[ = 9(8d + 1) - 1 \]
\[ = 72d + 8 \]
\[ = 8(9d + 1) \]

Divisible by 8.
and then induction...

\[ P(0) \land ((\forall n)(P(n) \implies P(n+1)) \equiv (\forall n \in \mathbb{N}) P(n). \]

**Thm:** For all \( n \geq 1 \), \( 8 \mid 3^{2n} - 1 \).

Induction on \( n \).

Base: \( 8 \mid 3^2 - 1 \).

Induction Hypothesis: Assume \( P(n) \): True for some \( n \).
\( (3^{2n} - 1 = 8d) \)

Induction Step: Prove \( P(n+1) \)

\[
3^{2n+2} - 1 = 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)}
\]
\[ = 9(8d + 1) - 1 \]
\[ = 72d + 8 \]
\[ = 8(9d + 1) \]

Divisible by 8.
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.
Stable Matching: a study in definitions and WOP.

- $n$-jobs, $n$-candidate.

Each entity has completely ordered preference list.
Stable Matching: a study in definitions and WOP.

\( n \)-jobs, \( n \)-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Stable Matching: a study in definitions and WOP.

\( n \)-jobs, \( n \)-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs \((m_i, w_j)\) containing all entities *exactly* once.
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
- Set of pairs $(m_i, w_j)$ containing all entities *exactly* once.
- How many pairs?
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs $(m_i, w_j)$ containing all entities *exactly* once.
How many pairs? $n$. 
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list that contains every entity of opposite type.

**Pairing.**
Set of pairs $(m_i, w_j)$ containing all entities exactly once.
How many pairs? $n$.
Entities in pair are **partners** in pairing.
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs $(m_i, w_j)$ containing all entities exactly once.
How many pairs? $n$.
Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
A $m_j$ and $w_k$ who like each other more than their partners
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs $(m_i, w_j)$ containing all entities exactly once.
How many pairs? $n$.
Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
A $m_j$ and $w_k$ who like each other more than their partners
Stable Matching: a study in definitions and WOP.

n-jobs, n-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
- Set of pairs \((m_i, w_j)\) containing all entities *exactly* once.
  - How many pairs? \(n\).
  - Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
- A \(m_j\) and \(w_k\) who like each other more than their partners

**Stable Pairing.**
Stable Matching: a study in definitions and WOP.

$n$-jobs, $n$-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs $(m_i, w_j)$ containing all entities exactly once.
How many pairs? $n$.
Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
A $m_j$ and $w_k$ who like each other more than their partners

**Stable Pairing.**
Pairing with no rogue couples.
Stable Matching: a study in definitions and WOP.

\( n \)-jobs, \( n \)-candidate.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs \((m_i, w_j)\) containing all entities *exactly* once.

How many pairs? \( n \).

Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
A \( m_j \) and \( w_k \) who like each other more than their partners

**Stable Pairing.**
Pairing with no rogue couples.

Does stable pairing exist?
Stable Matching: a study in definitions and WOP.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs \((m_i, w_j)\) containing all entities *exactly* once.
How many pairs? \(n\).
Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
A \(m_j\) and \(w_k\) who like each other more than their partners

**Stable Pairing.**
Pairing with no rogue couples.

Does stable pairing exist?
Stable Matching: a study in definitions and WOP.

Each entity has completely ordered preference list contains every entity of opposite type.

**Pairing.**
Set of pairs \((m_i, w_j)\) containing all entities exactly once.
How many pairs? \(n\).
Entities in pair are **partners** in pairing.

**Rogue Couple in a pairing.**
A \(m_j\) and \(w_k\) who like each other more than their partners

**Stable Pairing.**
Pairing with no rogue couples.

Does stable pairing exist?
No, for roommates problem.
Job Propose or reject Matching Algorithm:

Each Day:
- All jobs propose to favorite non-rejecting candidate.
- Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
- Job crosses off candidate who rejected him.
- Candidate's current proposer is "on string."

"Propose and Reject."
- Either jobs propose or candidate.
- But not both.

Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
- Every day, if job on string for candidate, \( \Rightarrow \) any future job on string is better.

Stability:
- No rogue couple.
- Rogue couple \( (M,W) \) \( \Rightarrow \) \( M \) proposed to \( W \) \( \Rightarrow \) \( W \) ended up with someone she liked better than \( M \).

Not rogue couple!
Job Propose or reject Matching Algorithm:

Each Day:

Useful Algorithmic Definitions:

- Job crosses off candidate who rejected him.
- Candidate's current proposer is "on string."
- "Propose and Reject." Either jobs propose or candidate. But not both.
- Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:

Every day, if job on string for candidate, \( \Rightarrow \) any future job on string is better.

Stability:

No rogue couple.

rogue couple (M,W) \( \Rightarrow \) M proposed to W \( \Rightarrow \) W ended up with someone she liked better than M.

Not rogue couple!
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.
Job Propose or reject Matching Algorithm:

Each Day:
  All jobs propose to favorite non-rejecting candidate.
  Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”
TMA.

Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.”
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate.
Job Propose or reject Matching Algorithm:

Each Day:
  All jobs propose to favorite non-rejecting candidate.
  Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
  Job crosses off candidate who rejected him.
  Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate. Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him. Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
Traditional propose and reject where jobs propose.
Job Propose or reject Matching Algorithm:

Each Day:

- All jobs propose to favorite non-rejecting candidate.
- Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:

- Job **crosses off** candidate who rejected him.
- Candidate’s current proposer is “**on string**.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.

- Traditional propose and reject where jobs propose.
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
   Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
Every day, if job on string for candidate,
Job Propose or reject Matching Algorithm:

Each Day:
- All jobs propose to favorite non-rejecting candidate.
- Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
- Job crosses off candidate who rejected him.
- Candidate’s current proposer is “on string.”

“Propose and Reject.”: Either jobs propose or candidate. But not both.
- Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
- Every day, if job on string for candidate,
  \[ \implies \text{any future job on string is better.} \]
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
   Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
Every day, if job on string for candidate,
   any future job on string is better.

Stability:
TMA.

Job Propose or reject Matching Algorithm:

Each Day:
- All jobs propose to favorite non-rejecting candidate.
- Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
- Job crosses off candidate who rejected him.
- Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
- Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
- Every day, if job on string for candidate,
  ➔ any future job on string is better.

Stability: No rogue couple.
TMA.

Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
Every day, if job on string for candidate,
⇒ any future job on string is better.

Stability: No rogue couple.
rogue couple (M,W)
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job crosses off candidate who rejected him.
Candidate’s current proposer is “on string.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
Every day, if job on string for candidate,
\[ \implies \text{any future job on string is better.} \]

Stability: No rogue couple.
rogue couple (M,W)
\[ \implies \text{M proposed to W} \]
Job Propose or reject Matching Algorithm:

Each Day:
- All jobs propose to favorite non-rejecting candidate.
- Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
- Job crosses off candidate who rejected him.
- Candidate’s current proposer is “on string.”

“Propose and Reject.”: Either jobs propose or candidate. But not both.
- Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
- Every day, if job on string for candidate,
  - any future job on string is better.

Stability: No rogue couple.
- rogue couple (M,W)
  - M proposed to W
  - W ended up with someone she liked better than M.
Job Propose or reject Matching Algorithm:

Each Day:
All jobs propose to favorite non-rejecting candidate.
Every candidate rejects all but best job who proposes.

Useful Algorithmic Definitions:
Job **crosses off** candidate who rejected him.
Candidate’s current proposer is “**on string**.”

“Propose and Reject.” : Either jobs propose or candidate. But not both.
   Traditional propose and reject where jobs propose.

Key Property: Improvement Lemma:
Every day, if job on string for candidate,
   \[ \implies \] any future job on string is better.

Stability: No rogue couple.
rogue couple \((M,W)\)
   \[ \implies \] M proposed to W
   \[ \implies \] W ended up with someone she liked better than \(M\).
Not rogue couple!
Optimality/Pessimal

Optimal partner if best partner in any *stable* pairing.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing. Not necessarily first in list.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing. Not necessarily first in list. Possibly no stable pairing with that partner.
Optimality/Pessimal

Optimal partner if best partner in any **stable** pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.
Optimality/Pessimal

Optimal partner if best partner in any **stable** pairing.
   Not necessarily first in list.
   Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, S.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, S.
First job $M$ to lose optimal partner.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, $S$.
First job $M$ to lose optimal partner.
Better partner $W$ for $M$. 
Optimality/Pessimal

Optimal partner if best partner in any **stable** pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, S.
*First job* $M$ *to lose optimal partner.*
Better partner $W$ for $M$.
Different stable pairing $T$. 
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, S.
First job $M$ to lose optimal partner.
Better partner $W$ for $M$.
Different stable pairing $T$.
TMA: $M$ asked $W$ first!
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
   Not necessarily first in list.
   Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, S.
   First job $M$ to lose optimal partner.
   Better partner $W$ for $M$.
   Different stable pairing $T$.
   TMA: $M$ asked $W$ first!
   There is $M'$ who bumps $M$ in TMA.
Optimality/Pessimal

Optimal partner if best partner in any *stable* pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, $S$.
First job $M$ to lose optimal partner.
Better partner $W$ for $M$.
Different stable pairing $T$.
TMA: $M$ asked $W$ first!
There is $M'$ who bumps $M$ in TMA.
$W$ prefers $M'$.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing. Not necessarily first in list. Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

Thm: TMA produces male optimal pairing, S. First job $M$ to lose optimal partner. Better partner $W$ for $M$. Different stable pairing $T$. TMA: $M$ asked $W$ first! There is $M'$ who bumps $M$ in TMA. $W$ prefers $M'$. $M'$ likes $W$ at least as much as optimal partner.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, $S$.
First job $M$ to lose optimal partner.
Better partner $W$ for $M$.
Different stable pairing $T$.
TMA: $M$ asked $W$ first!
There is $M'$ who bumps $M$ in TMA.
$W$ prefers $M'$.
$M'$ likes $W$ at least as much as optimal partner.
Since $M'$ was not the first to be bumped.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, $S$.
First job $M$ to lose optimal partner.
Better partner $W$ for $M$.
Different stable pairing $T$.
TMA: $M$ asked $W$ first!
There is $M'$ who bumps $M$ in TMA.
$W$ prefers $M'$.
$M'$ likes $W$ at least as much as optimal partner.
Since $M'$ was not the first to be bumped.
$M'$ and $W$ is rogue couple in $T$. 
Optimality/Pessimal

Optimal partner if best partner in any stable pairing. Not necessarily first in list. Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, $S$.

First job $M$ to lose optimal partner.
Better partner $W$ for $M$.
Different stable pairing $T$.
TMA: $M$ asked $W$ first!
There is $M'$ who bumps $M$ in TMA.
$W$ prefers $M'$.
$M'$ likes $W$ at least as much as optimal partner.
Since $M'$ was not the first to be bumped.
$M'$ and $W$ is rogue couple in $T$. 
Optimality/Pessimal

Optimal partner if best partner in any **stable** pairing.
   Not necessarily first in list.
   Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, \( S \).
   First job \( M \) to lose optimal partner.
   Better partner \( W \) for \( M \).
   Different stable pairing \( T \).
   TMA: \( M \) asked \( W \) first!
   There is \( M' \) who bumps \( M \) in TMA.
   \( W \) prefers \( M' \).
   \( M' \) likes \( W \) at least as much as optimal partner.
      Since \( M' \) was not the first to be bumped.
   \( M' \) and \( W \) is rogue couple in \( T \).

**Thm:** candidate pessimal.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, S.
First job $M$ to lose optimal partner.
Better partner $W$ for $M$.
Different stable pairing $T$.
TMA: $M$ asked $W$ first!
There is $M'$ who bumps $M$ in TMA.
$W$ prefers $M'$.
$M'$ likes $W$ at least as much as optimal partner.
   Since $M'$ was not the first to be bumped.
$M'$ and $W$ is rogue couple in $T$.

**Thm:** candidate pessimal.
Job optimal $\implies$ Candidate pessimal.
Optimality/Pessimal

Optimal partner if best partner in any stable pairing. Not necessarily first in list. Possibly no stable pairing with that partner.

Job-optimal pairing is pairing where every job gets optimal partner.

**Thm:** TMA produces male optimal pairing, $S$.

First job $M$ to lose optimal partner.

Better partner $W$ for $M$.

Different stable pairing $T$.

TMA: $M$ asked $W$ first!

There is $M'$ who bumps $M$ in TMA.

$W$ prefers $M'$.

$M'$ likes $W$ at least as much as optimal partner.

Since $M'$ was not the first to be bumped.

$M'$ and $W$ is rogue couple in $T$.

**Thm:** candidate pessimal.

Job optimal $\implies$ Candidate pessimal.

Candidate optimal $\implies$ Job pessimal.
...Graphs...

\[ G = (V, E) \]

- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.
  - Directed: ordered pair of vertices.
  - Adjacent, Incident, Degree.
  - In-degree, Out-degree.

**Thm:**
Sum of degrees is 2 \( |E| \).

- Edge is incident to 2 vertices.
- Degree of vertices is total incidences.

- Pair of Vertices are Connected:
  - If there is a path between them.

- Connected Component: maximal set of connected vertices.
- Connected Graph: one connected component.
Graphs

\[ G = (V, E) \]

\( V \) - set of vertices.
Graphs

\[ G = (V, E) \]
- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.
...Graphs...

\[ G = (V, E) \]
\[ V - \text{set of vertices.} \]
\[ E \subseteq V \times V - \text{set of edges.} \]
...Graphs...

\[ G = (V, E) \]
- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.

Directed: ordered pair of vertices.

Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.
\[ G = (V, E) \]
- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Theorem:
The sum of degrees is equal to twice the number of edges:
\[ |E| = \sum \text{degree} \]

Edge is incident to 2 vertices.

Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.
\[ G = (V, E) \]
\[ V \] - set of vertices.
\[ E \subseteq V \times V \] - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

Thm: 
\[ \text{Sum of degrees is } 2 |E| \].

Edge is incident to 2 vertices.

Pair of Vertices are Connected: If there is a path between them.

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.
...Graphs...

\[ G = (V, E) \]
\[ V - \text{set of vertices.} \]
\[ E \subseteq V \times V - \text{set of edges.} \]

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

In-degree, Out-degree.

Thm:

\[ \text{Sum of degrees is } 2 |E|. \]

Edge is incident to 2 vertices.

Pair of Vertices are Connected:

If there is a path between them.

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.
...Graphs...

\[ G = (V, E) \]
\[ V - \text{set of vertices.} \]
\[ E \subseteq V \times V - \text{set of edges.} \]

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).
Graphs

\[ G = (V, E) \]
\[ V \text{ - set of vertices.} \]
\[ E \subseteq V \times V \text{ - set of edges.} \]

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).

Edge is incident to 2 vertices.
Graphs

\[ G = (V, E) \]
\[ V - \text{set of vertices.} \]
\[ E \subseteq V \times V - \text{set of edges.} \]

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).
Edge is incident to 2 vertices.
Degree of vertices is total incidences.
$G = (V, E)$
$V$ - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.
Adjacent, Incident, Degree.
In-degree, Out-degree.

**Thm:** Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.
Graphs

\[ G = (V, E) \]
- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
- In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).
- Edge is incident to 2 vertices.
- Degree of vertices is total incidences.

Pair of Vertices are Connected:
Graphs

\( G = (V, E) \)
- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.

Directed: ordered pair of vertices.
- Adjacent, Incident, Degree.
  - In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).
- Edge is incident to 2 vertices.
- Degree of vertices is total incidences.

Pair of Vertices are Connected:
- If there is a path between them.
Graphs

\[ G = (V, E) \]
\[ V \] - set of vertices.
\[ E \subseteq V \times V \] - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).

Edge is incident to 2 vertices.

Degree of vertices is total incidences.

Pair of Vertices are Connected:

If there is a path between them.
Graphs...

\[ G = (V, E) \]
\[ V \] - set of vertices.
\[ E \subseteq V \times V \] - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
  In-degree, Out-degree.

**Thm:** Sum of degrees is \(2|E|\).
  Edge is incident to 2 vertices.
  Degree of vertices is total incidences.

Pair of Vertices are Connected:
  If there is a path between them.

Connected Component: maximal set of connected vertices.
Graphs...

$G = (V, E)$
- $V$ - set of vertices.
- $E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

**Thm:** Sum of degrees is $2|E|$.
- Edge is incident to 2 vertices.
- Degree of vertices is total incidences.

Pair of Vertices are Connected:
- If there is a path between them.

Connected Component: maximal set of connected vertices.
\( G = (V, E) \)

- \( V \) - set of vertices.
- \( E \subseteq V \times V \) - set of edges.

**Directed:** ordered pair of vertices.

- Adjacent, Incident, Degree.
  - In-degree, Out-degree.

**Thm:** Sum of degrees is \( 2|E| \).

- Edge is incident to 2 vertices.
- Degree of vertices is total incidences.

**Pair of Vertices are Connected:**
- If there is a path between them.

**Connected Component:** maximal set of connected vertices.

**Connected Graph:** one connected component.
Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

**Algorithm:**
- Take a walk using each edge at most once.
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

**Algorithm:**
- Take a walk using each edge at most once.
- **Property:** return to starting point.
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

**Algorithm:**
- Take a walk using each edge at most once.
- **Property:** return to starting point.
  - Proof Idea: Even degree.
**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

**Algorithm:**
- Take a walk using each edge at most once.

**Property:** return to starting point.
- Proof Idea: Even degree.

Recurse on connected components.
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

**Algorithm:**
- Take a walk using each edge at most once.
- **Property:** return to starting point.
  - Proof Idea: Even degree.

Recurse on connected components.
Put together.
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:
- *Take a walk using each edge at most once.*
- **Property:** return to starting point.
  - Proof Idea: Even degree.

Recurse on connected components.
Put together.
- **Property:** walk visits every component.
Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:
Take a walk using each edge at most once.

Property: return to starting point.
Proof Idea: Even degree.

Recurse on connected components.
Put together.

Property: walk visits every component.
Proof Idea: Original graph connected.
Graph Algorithm: Eulerian Tour

**Thm:** Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

**Algorithm:**
- Take a walk using each edge at most once.
- **Property:** return to starting point.
  - Proof Idea: Even degree.

**Recurse on connected components.**
**Put together.**
- **Property:** walk visits every component.
  - Proof Idea: Original graph connected.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.
Graph Coloring.

Given \( G = (V, E) \), a coloring of a \( G \) assigns colors to vertices \( V \) where for each edge the endpoints have different colors.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.
Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

- Notice that the last one has three colors.
- Fewer colors than the number of vertices.
- Fewer colors than the max degree node.

Interesting things to do.

Algorithm!
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

Notice that the last one, has one three colors.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

Notice that the last one, has one three colors. Fewer colors than number of vertices.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

Notice that the last one, has one three colors. Fewer colors than number of vertices. Fewer colors than max degree node.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

Notice that the last one, has one three colors. Fewer colors than number of vertices. Fewer colors than max degree node.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

Notice that the last one, has one three colors. Fewer colors than number of vertices. Fewer colors than max degree node.

Interesting things to do.
Graph Coloring.

Given $G = (V, E)$, a coloring of a $G$ assigns colors to vertices $V$ where for each edge the endpoints have different colors.

Notice that the last one, has one three colors.
Fewer colors than number of vertices.
Fewer colors than max degree node.

Interesting things to do. Algorithm!
Planar graphs and maps.

Planar graph coloring $\equiv$ map coloring.
Planar graphs and maps.

Planar graph coloring ≡ map coloring.

Four color theorem is about planar graphs!
Six color theorem.

**Theorem:** Every planar graph can be colored with six colors.
Six color theorem.

**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Six color theorem.

**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Recall: \( e \leq 3v - 6 \) for any planar graph where \( v > 2 \).
Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: \( e \leq 3v - 6 \) for any planar graph where \( v > 2 \).
From Euler’s Formula.
Six color theorem.

**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Recall: \( e \leq 3v - 6 \) for any planar graph where \( v > 2 \).
   From Euler’s Formula.
Total degree: \( 2e \)
Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
  From Euler’s Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v}$
Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
   From Euler’s Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v}$
Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
   From Euler’s Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$. 
Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: \( e \leq 3v - 6 \) for any planar graph where \( v > 2 \).

From Euler’s Formula.

Total degree: \( 2e \)
Average degree: \( \leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v} \).

There exists a vertex with degree \( < 6 \).
**Theorem**: Every planar graph can be colored with six colors.

**Proof**:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
   From Euler’s Formula.
Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.
There exists a vertex with degree $< 6$ or at most 5.
Six color theorem.

**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Recall: \( e \leq 3v - 6 \) for any planar graph where \( v > 2 \).

From Euler’s Formula.

Total degree: \( 2e \)
Average degree: \( \leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v} \).

There exists a vertex with degree \( < 6 \) or at most 5.

Remove vertex \( v \) of degree at most 5.
**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
    From Euler’s Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree $< 6$ or at most 5.

    Remove vertex $v$ of degree at most 5.
    Inductively color remaining graph.
**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
    From Euler’s Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree $< 6$ or at most 5.

Remove vertex $v$ of degree at most 5.
Inductively color remaining graph.
Color is available for $v$ since only five neighbors...
Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
   From Euler’s Formula.
Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree $< 6$ or at most 5.

   Remove vertex $v$ of degree at most 5.
   Inductively color remaining graph.
   Color is available for $v$ since only five neighbors... and only five colors are used.
**Theorem:** Every planar graph can be colored with six colors.

**Proof:**
Recall: \( e \leq 3v - 6 \) for any planar graph where \( v > 2 \).

From Euler’s Formula.

Total degree: \( 2e \)
Average degree: \( \leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v} \).

There exists a vertex with degree \(< 6\) or at most 5.

Remove vertex \( v \) of degree at most 5.
Inductively color remaining graph.
Color is available for \( v \) since only five neighbors...
and only five colors are used.

□
Five color theorem: summary.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.
Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.
Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

Proof:
Five color theorem: summary.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

Proof: Again with the degree 5 vertex.
Five color theorem: summary.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

Proof: Again with the degree 5 vertex. Again recurse.
Five color theorem: summary.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

Proof: Again with the degree 5 vertex. Again recurse.
Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

Proof: Again with the degree 5 vertex. Again recurse.

Either switch green.
Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

**Proof:** Again with the degree 5 vertex. Again recurse.

Either switch green.

Or try switching orange.
Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Theorem: Every planar graph can be colored with five colors.

Proof: Again with the degree 5 vertex. Again recurse.

Either switch green.
Or try switching orange.
One will work.
Graph Types: Complete Graph.

- $K_n$, $|V| = n$, every edge present.
- Degree of vertex $|V| - 1$.
- Very connected. Lots of edges: $n(n-1)/2$. 
Graph Types: Complete Graph.

$K_n, |V| = n$
Graph Types: Complete Graph.

$K_n, |V| = n$

every edge present.
Graph Types: Complete Graph.

\[ K_n, |V| = n \]

every edge present.

degree of vertex?

\[ \frac{n(n-1)}{2} \]
Graph Types: Complete Graph.

$K_n, |V| = n$

every edge present.

degree of vertex? $|V| - 1$. 

Very connected.
Lots of edges: 

$n(n-1)/2$. 
Graph Types: Complete Graph.

$K_n$, $|V| = n$

- every edge present.
- degree of vertex? $|V| - 1$.

Very connected.
Graph Types: Complete Graph.

\[ K_n, |V| = n \]

- every edge present.
- degree of vertex? \(|V| - 1\).

Very connected.
Lots of edges:
Graph Types: Complete Graph.

$K_n$, $|V| = n$

- every edge present.
- degree of vertex? $|V| - 1$.

Very connected.
Lots of edges: $n(n - 1)/2$. 
Trees.

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- An acyclic graph where any edge addition creates a cycle.
- To tree or not to tree! Minimally connected, minimum number of edges to connect.

Property: Can remove a single node and break into components of size at most $|V| / 2$. 
Trees.

Definitions:
A connected graph without a cycle.
Trees.

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
Trees.

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
Trees.

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.
Trees.

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.
Trees.

Definitions:

- A connected graph without a cycle.
- A connected graph with \( |V| - 1 \) edges.
- A connected graph where any edge removal disconnects it.
- An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!
Trees.

Definitions:
- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!

Minimally connected, minimum number of edges to connect.
Trees.

Definitions:

- A connected graph without a cycle.
- A connected graph with $|V| - 1$ edges.
- A connected graph where any edge removal disconnects it.
- An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!

Minimally connected, minimum number of edges to connect.

Property:
Trees.

Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!

Minimally connected, minimum number of edges to connect.

Property:
Can remove a single node and break into components of size at most $|V|/2$. 

Hypercube

Hypercubes.
Hypercubes. Really connected.
Hypercubes. Really connected. $|V| \log |V|$ edges!
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
Hypercubes. Really connected. \( |V| \log |V| \) edges!
Also represents bit-strings nicely.

\[
G = (V, E)  \\
|V| = \{0, 1\}^n,
\]
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$

$|V| = \{0, 1\}^n,$

$|E| = \{(x, y) | x \text{ and } y \text{ differ in one bit position.}\}$
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$$G = (V, E)$$

$|V| = \{0, 1\}^n,$

$|E| = \{(x, y) | x \text{ and } y \text{ differ in one bit position.}\}$
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An \( n \)-dimensional hypercube consists of a 0-subcube (1-subcube) which is a \( n - 1 \)-dimensional hypercube with nodes labelled 0x (1x) with the additional edges (0x, 1x).
**Recursive Definition.**

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n - 1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$. 
A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n - 1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x,1x)$. 
A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n - 1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$. 

![Diagram of hypercubes](image_url)
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n - 1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$. 
Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An $n$-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.
Rudrata Cycle: cycle that visits every node.
Rudrata Cycle: cycle that visits every node.
Eulerian?
Rudrata Cycle: cycle that visits every node. Eulerian? If $n$ is even.
Rudrata Cycle: cycle that visits every node.
Eulerian? If $n$ is even.
Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
Hypercube: properties

Rudrata Cycle: cycle that visits every node.
  Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
  Best cut?
Rudrata Cycle: cycle that visits every node.
Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes:
Hypercube: properties

Rudrata Cycle: cycle that visits every node.
   Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
   Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.
Rudrata Cycle: cycle that visits every node.
Eulerian? If \( n \) is even.

Large Cuts: Cutting off \( k \) nodes needs \( \geq k \) edges.
Best cut? Cut apart subcubes: cuts off \( 2^n \) nodes with \( 2^{n-1} \) edges.

FYI:
Rudrata Cycle: cycle that visits every node.
Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.
FYI: Also cuts represent boolean functions.
Hypercube: properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.

FYI: Also cuts represent boolean functions.
Hypercube: properties

Rudrata Cycle: cycle that visits every node.
   Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
   Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Rudrata Cycle: cycle that visits every node. 
   Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges. 
   Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes. 
   Get from 000100 to 101000.
Rudrata Cycle: cycle that visits every node. Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.

$$000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000$$
Hypercube: properties

Rudrata Cycle: cycle that visits every node. Eulerian? If \( n \) is even.

Large Cuts: Cutting off \( k \) nodes needs \( \geq k \) edges.
Best cut? Cut apart subcubes: cuts off \( 2^n \) nodes with \( 2^{n-1} \) edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
\[
000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000
\]
Correct bits in string, moves along path in hypercube!
Rudrata Cycle: cycle that visits every node. Eulerian? If \( n \) is even.

Large Cuts: Cutting off \( k \) nodes needs \( \geq k \) edges. 
Best cut? Cut apart subcubes: cuts off \( 2^n \) nodes with \( 2^{n-1} \) edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes. 
Get from 000100 to 101000.
  000100 \( \rightarrow \) 100100 \( \rightarrow \) 101100 \( \rightarrow \) 101000
Correct bits in string, moves along path in hypercube!
Hypercube: properties

Rudrata Cycle: cycle that visits every node. Eulerian? If $n$ is even.

Large Cuts: Cutting off $k$ nodes needs $\geq k$ edges. Best cut? Cut apart subcubes: cuts off $2^n$ nodes with $2^{n-1}$ edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes. Get from 000100 to 101000.

$\begin{align*}
000100 & \rightarrow 100100 \rightarrow 101100 \rightarrow 101000 \\
Correct \ bits \ in \ string, \ moves \ along \ path \ in \ hypercube!
\end{align*}$

Good communication network!
...Modular Arithmetic...

Arithmetic modulo $m$.
Elements of equivalence classes of integers.
...Modular Arithmetic...

Arithmetic modulo $m$.
Elements of equivalence classes of integers.
$\{0, \ldots, m - 1\}$
Arithmetic modulo $m$. Elements of equivalence classes of integers.

\{0, \ldots, m - 1\}

and integer $i \equiv a \pmod{m}$
Modular Arithmetic

Arithmetic modulo \( m \).

Elements of equivalence classes of integers.
\{0,\ldots,m-1\}

and integer \( i \equiv a \pmod{m} \)

if \( i = a + km \) for integer \( k \).
Arithmetic modulo $m$.
Elements of equivalence classes of integers.
$\{0, \ldots, m - 1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer $k$.
or if the remainder of $i$ divided by $m$ is $a$. 

Can do calculations by taking remainders at the beginning,
in the middle or at the end.

\[
\begin{align*}
58 + 32 &= 90 = 6 \pmod{7} \\
58 + 32 &= 2 + 4 = 6 \pmod{7} \\
58 + 32 &= 2 + (-3) = -1 = 6 \pmod{7}
\end{align*}
\]

Negative numbers work the way you are used to.

\[
\begin{align*}
-3 &= 0 \pmod{7} \\
-3 &= 7 \pmod{7} \\
-3 &= 4 \pmod{7}
\end{align*}
\]

Additive inverses are intuitively negative numbers.
Arithmetic modulo $m$.

Elements of equivalence classes of integers.

$$\{0, \ldots, m-1\}$$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer $k$.

or if the remainder of $i$ divided by $m$ is $a$. 

Can do calculations by taking remainders at the beginning, in the middle or at the end.

$$58 + 32 = 90 = 6 \pmod{7}$$

$$58 + 32 = 2 + 4 = 6 \pmod{7}$$

$$58 + 32 = 2 + (-3) = -1 = 6 \pmod{7}$$

Negative numbers work the way you are used to.

$$-3 = 0 \pmod{7}$$

$$-3 = 7 \pmod{7}$$

$$-3 = 4 \pmod{7}$$

Additive inverses are intuitively negative numbers.
Arithmetic modulo $m$.

Elements of equivalence classes of integers.

$$\{0, \ldots, m-1\}$$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer $k$.

or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders at the beginning,
Arithmetic modulo $m$.

Elements of equivalence classes of integers.

$$\{0, \ldots, m-1\}$$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer $k$.

or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders at the beginning, in the middle.
Arithmetic modulo \( m \).

Elements of equivalence classes of integers. \([0, \ldots, m - 1]\)
and integer \( i \equiv a \pmod{m} \)
if \( i = a + km \) for integer \( k \).
or if the remainder of \( i \) divided by \( m \) is \( a \).

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.
Arithmetic modulo $m$.
Elements of equivalence classes of integers.
\[ \{0, \ldots, m - 1\} \]
and integer \( i \equiv a \pmod{m} \)
if \( i = a + km \) for integer \( k \).
or if the remainder of \( i \) divided by \( m \) is \( a \).

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

\[ 58 + 32 = 90 = 6 \pmod{7} \]
...Modular Arithmetic...

Arithmetic modulo $m$.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer $k$.

or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders
at the beginning,
in the middle
 or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
Arithmetic modulo $m$.

Elements of equivalence classes of integers.

\{0, \ldots, m - 1\}

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer $k$.

or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders

at the beginning,

in the middle

or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + (-3) = -1 = 6 \pmod{7}$
Arithmetic modulo \( m \).
Elements of equivalence classes of integers.
\[ \{0, \ldots, m-1\} \]
and integer \( i \equiv a \pmod{m} \)
if \( i = a + km \) for integer \( k \).
or if the remainder of \( i \) divided by \( m \) is \( a \).

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

\[
58 + 32 = 90 = 6 \pmod{7} \\
58 + 32 = 2 + 4 = 6 \pmod{7} \\
58 + 32 = 2 + -3 = -1 = 6 \pmod{7}
\]

Negative numbers work the way you are used to.
\[
-3 = 0 \\
-3 + 3 = 7 \\
-3 + 4 = 4 \pmod{7}
\]

Additive inverses are intuitively negative numbers.
Arithmetic modulo $m$.
Elements of equivalence classes of integers.
\{0, \ldots, m - 1\}
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer $k$.
  or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders
at the beginning,
in the middle
  or at the end.

\begin{align*}
58 + 32 &= 90 = 6 \pmod{7} \\
58 + 32 &= 2 + 4 = 6 \pmod{7} \\
58 + 32 &= 2 + -3 = -1 = 6 \pmod{7}
\end{align*}

Negative numbers work the way you are used to.
Arithmetic modulo \( m \).
Elements of equivalence classes of integers.
\[ \{0, \ldots, m-1\} \]
and integer \( i \equiv a \pmod{m} \)
if \( i = a + km \) for integer \( k \).

or if the remainder of \( i \) divided by \( m \) is \( a \).

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

\[
\begin{align*}
58 + 32 &= 90 = 6 \pmod{7} \\
58 + 32 &= 2 + 4 = 6 \pmod{7} \\
58 + 32 &= 2 + (-3) = -1 = 6 \pmod{7}
\end{align*}
\]

Negative numbers work the way you are used to.
\[
-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}
\]
Arithmetic modulo $m$.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer $k$.
or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + (-3) = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$
Arithmetic modulo $m$.

Elements of equivalence classes of integers. 
\{0, \ldots, m - 1\}

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer $k$.

or if the remainder of $i$ divided by $m$ is $a$.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

\[
\begin{align*}
58 + 32 &= 90 = 6 \pmod{7} \\
58 + 32 &= 2 + 4 = 6 \pmod{7} \\
58 + 32 &= 2 + -3 = -1 = 6 \pmod{7}
\end{align*}
\]

Negative numbers work the way you are used to.

\[
\begin{align*}
-3 &= 0 - 3 = 7 - 3 = 4 \pmod{7}
\end{align*}
\]

Additive inverses are intuitively negative numbers.
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$?

Inverse Unique? Yes.

Proof:

$a$ and $b$ inverses of $x \pmod{n}$

$ax = bx = 1 \pmod{n}$

$axb = bxb = b \pmod{n}$

$a = b \pmod{n}$.

$3^{-1} \pmod{6}$?

No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$

$\{3, 6, 3, 6, 3\}$

See, ... no inverse!
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7} = 5$

Inverse Unique? Yes.

Proof:

Let $a$ and $b$ be inverses of $x \pmod{n}$.

$ax = bx = 1 \pmod{n}$

Then:

$axb = bxb = b \pmod{n}$

So $a = b \pmod{n}$.

Specifically:

$3^{-1} \pmod{6}$

Is $\{3 \cdot 1, 3 \cdot 2, 3 \cdot 3, 3 \cdot 4, 3 \cdot 5\} = \{3, 6, 3, 6, 3\}$

See, ... no inverse!
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7} \equiv 5$
$5^{-1} \pmod{7} \not\equiv$
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7} \equiv 5$
$5^{-1} \pmod{7} \equiv 3$

Inverse Unique?
Yes. Proof:
a and b inverses of x (mod n)
$ax = bx = 1 \pmod{n}$
$axb = bxb = b \pmod{n}$
$a = b \pmod{n}$.

$3^{-1} \pmod{6}$?
No, no, no....
{$3(1), 3(2), 3(3), 3(4), 3(5)$}
{$3, 6, 3, 6, 3$}

See, ... no inverse!
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7} \equiv 5$

$5^{-1} \pmod{7} \equiv 3$

Inverse Unique?
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7} \equiv 5$

$5^{-1} \pmod{7} \equiv 3$

Inverse Unique? Yes.
Modular Arithmetic and multiplicative inverses.

\[ 3^{-1} \pmod{7} \equiv 5 \]
\[ 5^{-1} \pmod{7} \equiv 3 \]

Inverse Unique? Yes.

Proof: \( a \) and \( b \) inverses of \( x \pmod{n} \)
Modular Arithmetic and multiplicative inverses.

$$3^{-1} \pmod{7} \text{? } 5$$

$$5^{-1} \pmod{7} \text{? } 3$$

Inverse Unique? Yes.

Proof: $a$ and $b$ inverses of $x \pmod{n}$

$$ax = bx = 1 \pmod{n}$$
Modular Arithmetic and multiplicative inverses.

\[ 3^{-1} \pmod{7}? \ 5 \]
\[ 5^{-1} \pmod{7}? \ 3 \]

Inverse Unique? Yes.

Proof: \( a \) and \( b \) inverses of \( x \pmod n \)

\[ ax = bx = 1 \pmod n \]

\[ axb = bxb = b \pmod n \]
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}?$ 5
$5^{-1} \pmod{7}?$ 3

Inverse Unique? Yes.
Proof: $a$ and $b$ inverses of $x \pmod{n}$

$\begin{align*}
ax &= bx = 1 \pmod{n} \\
axb &= bxb = b \pmod{n} \\
a &= b \pmod{n}.
\end{align*}$
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique? Yes.
Proof: $a$ and $b$ inverses of $x \pmod{n}$

$ax = bx = 1 \pmod{n}$
$AXB = BXB = B \pmod{n}$
$a = b \pmod{n}$.

$3^{-1} \pmod{6}$?
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique? Yes.
Proof: $a$ and $b$ inverses of $x \pmod{n}$
$ax = bx = 1 \pmod{n}$
$axb = bxb = b \pmod{n}$
$a = b \pmod{n}$.

$3^{-1} \pmod{6}$? No, no, no....
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique? Yes.

Proof: $a$ and $b$ inverses of $x \pmod{n}$

$ax = bx = 1 \pmod{n}$

$axb = bxb = b \pmod{n}$

$a = b \pmod{n}$.

$3^{-1} \pmod{6}$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5

$5^{-1} \pmod{7}$? 3

Inverse Unique? Yes.
Proof: $a$ and $b$ inverses of $x \pmod{n}$

\[
\begin{align*}
ax &= bx = 1 \pmod{n} \\
axb &= bxb = b \pmod{n} \\
a &= b \pmod{n}.
\end{align*}
\]

$3^{-1} \pmod{6}$? No, no, no....

\[
\begin{align*}
\{3(1), 3(2), 3(3), 3(4), 3(5)\} \\
\{3, 6, 3, 6, 3\}
\end{align*}
\]
Modular Arithmetic and multiplicative inverses.

\[ 3^{-1} \pmod{7} \text{? } 5 \]
\[ 5^{-1} \pmod{7} \text{? } 3 \]

Inverse Unique? Yes.
Proof: \( a \) and \( b \) inverses of \( x \pmod{n} \)
\[ ax = bx = 1 \pmod{n} \]
\[ axb = bxb = b \pmod{n} \]
\[ a = b \pmod{n}. \]

\[ 3^{-1} \pmod{6} \text{? } \text{No, no, no...} \]
\[ \{3(1), 3(2), 3(3), 3(4), 3(5)\} \]
\[ \{3, 6, 3, 6, 3\} \]
Modular Arithmetic and multiplicative inverses.

3\(^{-1}\) (mod 7)? 5
5\(^{-1}\) (mod 7)? 3

Inverse Unique? Yes.
Proof: a and b inverses of x (mod n)
\[ ax = bx = 1 \text{ (mod } n) \]
\[ axb = bxb = b \text{ (mod } n) \]
\[ a = b \text{ (mod } n). \]

3\(^{-1}\) (mod 6)? No, no, no....
\{3(1), 3(2), 3(3), 3(4), 3(5)\}
\{3, 6, 3, 6, 3\}
See,
Modular Arithmetic and multiplicative inverses.

\[ 3^{-1} \pmod{7}? \ 5 \]
\[ 5^{-1} \pmod{7}? \ 3 \]

Inverse Unique? Yes.

Proof: \( a \) and \( b \) inverses of \( x \) (mod \( n \))

\[ \begin{align*}
    ax &= bx = 1 \pmod{n} \\
    axb &= bxb = b \pmod{n} \\
    a &= b \pmod{n}.
\end{align*} \]

\[ 3^{-1} \pmod{6}? \text{ No, no, no...} \]

\[ \{3(1), 3(2), 3(3), 3(4), 3(5)\} \]
\[ \{3, 6, 3, 6, 3\} \]

See,... no inverse!
Modular Arithmetic Inverses and GCD

\( x \) has inverse modulo \( m \) if and only if \( \gcd(x, m) = 1 \).
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Finding $gcd$.

$gcd(x, y) = gcd(y, x - y) = gcd(y, x \mod y)$.

Give recursive Algorithm!

Base Case?

$gcd(x, 0) = x$.

Extended-gcd($x, y$) returns $(d, a, b)$

$d = gcd(x, y)$ and $d = ax + by$.

Multiplicative inverse of $(x, m)$.

$egcd(x, m) = (1, a, b)$

$a$ is inverse!

$1 = ax + bm = ax \mod m$.

Idea: $egcd$. $gcd$ produces 1 by adding and subtracting multiples of $x$ and $y$.
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m - 1)x\} are distinct modulo $m$ if and only if $gcd(x, m) = 1$. 

Finding gcd.

$gcd(x, y) = gcd(y, x - y)$.

Give recursive Algorithm!

Base Case?

$gcd(x, 0) = x$.

\text{egcd}(x, y) \text{ returns } (d, a, b)$

$d = gcd(x, y)$ and $d = ax + by$.

Multiplicative inverse of $(x, m)$.

$\text{egcd}(x, m) = (1, a, b)$

$a$ is inverse!

$1 = ax + bm = ax (mod \ m)$. 

Idea: egcd. 

gcd produces 1 by adding and subtracting multiples of $x$ and $y$.
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
{$0x, \ldots, (m-1)x$} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.

Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y)$
Modular Arithmetic Inverses and GCD

\( x \) has inverse modulo \( m \) if and only if \( \gcd(x, m) = 1 \).

Group structures more generally.

Proof Idea:
\( \{0x, \ldots, (m-1)x\} \) are distinct modulo \( m \) if and only if \( \gcd(x, m) = 1 \).

Finding gcd.
\[
\gcd(x, y) = \gcd(y, x - y) = \gcd(y, x \mod y).
\]
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots , (m - 1)x\}$ are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
\[gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod \ y)).\]

Give recursive Algorithm!
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m-1)x\} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
\[ gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod \ y)) \]

Give recursive Algorithm! Base Case?
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $\gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
{$0x, \ldots, (m-1)x$} are distinct modulo $m$ if and only if $\gcd(x, m) = 1$.

Finding gcd.

$\gcd(x, y) = \gcd(y, x - y) = \gcd(y, x \mod y)$.

Give recursive Algorithm! Base Case? $\gcd(x, 0) = x$. 

Extended-gcd($x, y$) returns ($d, a, b$) $d = \gcd(x, y)$ and $d = ax + by$. 

Multiplicative inverse of $(x, m)$.

egcd($x, m$) = ($1, a, b$) $a$ is inverse!

$1 = ax + bm = ax (\mod m)$. 

Idea: egcd. $\gcd$ produces 1 by adding and subtracting multiples of $x$ and $y$. 
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $\gcd(x, m) = 1$.

Group structures more generally.

Proof Idea: 
\{0x, \ldots, (m - 1)x\} are distinct modulo $m$ if and only if $\gcd(x, m) = 1$.

Finding gcd.
\[ \gcd(x, y) = \gcd(y, x - y) = \gcd(y, x \pmod{y}). \]

Give recursive Algorithm! Base Case? $\gcd(x, 0) = x$.

Extended-gcd($x, y$)
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
{$0x, \ldots, (m-1)x$} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \mod y))$.

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-gcd($x, y$) returns ($d, a, b$)
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m-1)x\} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
\[ gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod\ y)) . \]

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-gcd($x, y$) returns $(d, a, b)$
\[ d = gcd(x, y) \]
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod \ y))$.

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-gcd($x, y$) returns $(d, a, b)$
$\quad d = gcd(x, y)$ and $d = ax + by$
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
{$0x, \ldots, (m-1)x$} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding $gcd$.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod \ y))$.

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-$gcd(x, y)$ returns $(d, a, b)$
$d = gcd(x, y)$ and $d = ax + by$

Multiplicative inverse of $(x, m)$. 
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $\text{gcd}(x, m) = 1$.

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m-1)x\} are distinct modulo $m$ if and only if $\text{gcd}(x, m) = 1$.

Finding gcd.
\[
\text{gcd}(x, y) = \text{gcd}(y, x - y) = \text{gcd}(y, x \pmod{y}).
\]

Give recursive Algorithm! Base Case? $\text{gcd}(x, 0) = x$.

Extended-gcd$(x, y)$ returns $(d, a, b)$
\[
d = \text{gcd}(x, y) \text{ and } d = ax + by
\]

Multiplicative inverse of $(x, m)$.
\[
\text{egcd}(x, m) = (1, a, b)
\]
Modular Arithmetic Inverses and GCD

\( x \) has inverse modulo \( m \) if and only if \( \gcd(x, m) = 1 \).

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m - 1)x\} are distinct modulo \( m \) if and only if \( \gcd(x, m) = 1 \).

Finding \( \gcd \).
\[
\gcd(x, y) = \gcd(y, x - y) = \gcd(y, x \pmod{y}).
\]

Give recursive Algorithm! Base Case? \( \gcd(x, 0) = x \).

Extended-\( \gcd(x, y) \) returns \( (d, a, b) \)
\[
d = \gcd(x, y) \text{ and } d = ax + by
\]

Multiplicative inverse of \( (x, m) \).
\[
\text{egcd}(x, m) = (1, a, b)
\]
\( a \) is inverse!
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod \ y))$.

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-gcd$(x, y)$ returns $(d, a, b)$
$d = gcd(x, y)$ and $d = ax + by$

Multiplicative inverse of $(x, m)$.
$egcd(x, m) = (1, a, b)$
$a$ is inverse! $1 = ax + bm$
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m - 1)x\} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
\[
gcd(x, y) = gcd(y, x - y) = gcd(y, x \mod y)\]

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-gcd$(x, y)$ returns $(d, a, b)$
\[
d = gcd(x, y) \text{ and } d = ax + by\]

Multiplicative inverse of $(x, m)$.
\[
egcd(x, m) = (1, a, b)\]
\[
a \text{ is inverse! } 1 = ax + bm = ax \mod m.\]
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $gcd(x, m) = 1$.

Group structures more generally.

Proof Idea:
{$0x, \ldots, (m - 1)x$} are distinct modulo $m$ if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

Extended-gcd$(x, y)$ returns $(d, a, b)$
$\quad d = gcd(x, y)$ and $d = ax + by$

Multiplicative inverse of $(x, m)$.
$\quad$egcd$(x, m) = (1, a, b)$
$\quad$ $a$ is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $\text{gcd}(x, m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m - 1)x\}$ are distinct modulo $m$ if and only if $\text{gcd}(x, m) = 1$.

Finding gcd.
$\text{gcd}(x, y) = \text{gcd}(y, x - y) = \text{gcd}(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $\text{gcd}(x, 0) = x$.

Extended-gcd($x, y$) returns $(d, a, b)$

$d = \text{gcd}(x, y)$ and $d = ax + by$

Multiplicative inverse of $(x, m)$.
$\text{egcd}(x, m) = (1, a, b)$

$a$ is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
$\text{gcd}$ produces 1
Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if \( gcd(x, m) = 1 \).

Group structures more generally.

Proof Idea:
\{0, \ldots, (m-1)x\} are distinct modulo m if and only if \( gcd(x, m) = 1 \).

Finding gcd.
\( gcd(x, y) = gcd(y, x - y) = gcd(y, x \ (mod \ y)) \).

Give recursive Algorithm! Base Case? \( gcd(x, 0) = x \).

Extended-gcd\( (x, y) \) returns \((d, a, b)\)
\( d = gcd(x, y) \) and \( d = ax + by \)

Multiplicative inverse of \((x, m)\).
\( egcd(x, m) = (1, a, b) \)
\( a \) is inverse! \( 1 = ax + bm = ax \ (mod \ m) \).

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of \( x \) and \( y \)
Modular Arithmetic Inverses and GCD

$x$ has inverse modulo $m$ if and only if $\text{gcd}(x, m) = 1$.

Group structures more generally.

Proof Idea:
\{0x, \ldots, (m-1)x\} are distinct modulo $m$ if and only if $\text{gcd}(x, m) = 1$.

Finding $\text{gcd}$.

$\text{gcd}(x, y) = \text{gcd}(y, x - y) = \text{gcd}(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $\text{gcd}(x, 0) = x$.

Extended-$\text{gcd}(x, y)$ returns $(d, a, b)$

$d = \text{gcd}(x, y)$ and $d = ax + by$

Multiplicative inverse of $(x, m)$.

$\text{egcd}(x, m) = (1, a, b)$

$a$ is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: $\text{egcd}$.

gcd produces 1

by adding and subtracting multiples of $x$ and $y$
Hand calculation: egcd.

Extended GCD: egcd(7, 60) = 1.
Hand calculation: $\text{egcd}$.

Extended GCD: $\text{egcd}(7, 60) = 1$.

$\text{egcd}(7, 60)$. 
Hand calculation: egcd.

Extended GCD: $\text{egcd}(7, 60) = 1$.

$\text{egcd}(7, 60)$.

$$7(0) + 60(1) = 60$$
Hand calculation: egcd.

Extended GCD: \( \text{egcd}(7, 60) = 1 \).
\( \text{egcd}(7, 60) \).

\[
7(0) + 60(1) = 60 \\
7(1) + 60(0) = 7
\]
Hand calculation: egcd.

Extended GCD: \( \text{egcd}(7, 60) = 1 \).

\[ \text{egcd}(7,60). \]

\[ 7(0) + 60(1) = 60 \]
\[ 7(1) + 60(0) = 7 \]
\[ 7(-8) + 60(1) = 4 \]
Hand calculation: egcd.

Extended GCD: egcd(7, 60) = 1.

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3
\end{align*}
\]
Hand calculation: egcd.

Extended GCD: \( \text{egcd}(7, 60) = 1 \).
\( \text{egcd}(7, 60) \).

\[
\begin{align*}
7(0) + 60(1) & = 60 \\
7(1) + 60(0) & = 7 \\
7(-8) + 60(1) & = 4 \\
7(9) + 60(-1) & = 3 \\
7(-17) + 60(2) & = 1
\end{align*}
\]
Hand calculation: egcd.

Extended GCD: \( \text{egcd}(7, 60) = 1. \)
\( \text{egcd}(7, 60). \)

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]
Hand calculation: egcd.

Extended GCD: \( \text{egcd}(7, 60) = 1 \).
\[ \text{egcd}(7, 60). \]

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]

Confirm:
Hand calculation: egcd.

Extended GCD: \( \text{egcd}(7, 60) = 1 \).

\[ \text{egcd}(7, 60). \]

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]

Confirm: \(-119 + 120 = 1\)
Hand calculation: egcd.

Extended GCD: $\text{egcd}(7, 60) = 1$.

$\text{egcd}(7, 60)$.

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1 \\
\end{align*}
\]

Confirm: $-119 + 120 = 1$

$d = e^{-1} = -17 = 43 = (\text{mod } 60)$
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \, (\text{mod } p)$,
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),
\[
a^{p-1} \equiv 1 \pmod{p}.
\]
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$, 
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{ a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p} \}$. 

Since multiplication is commutative, 
\[ a \cdot (p-1) \cdot \ldots \cdot 1 \equiv 1 \cdot 2 \cdot \ldots \cdot (p-1) \pmod{p}, \]

Each of $2, \ldots, (p-1)$ has an inverse modulo $p$, multiply by inverses to get... 
\[ a \cdot (p-1) \equiv 1 \pmod{p}. \]
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.
$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$. Each of $2, \ldots, (p-1)$ has an inverse modulo $p$, multiply by inverses to get $a^{p-1} \equiv 1 \pmod{p}$. 
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),

\[
a^{p-1} \equiv 1 \pmod{p}.
\]

**Proof:** Consider \( T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p - 1) \pmod{p}\} \).

\( T \) is range of function \( f(x) = ax \pmod{p} \) for set \( S = \{1, \ldots, p - 1\} \).

Invertible function:
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

$T$ is range of function $f(x) = ax \pmod{p}$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
Fermat from Bijection.

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
$$a^{p-1} \equiv 1 \pmod{p}.$$  

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.
$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{ a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p} \}$.

$T$ is range of function $f(x) = ax \pmod{p}$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$. 

Since multiplication is commutative.

$\prod_{i=1}^{p-1} a^i \equiv \prod_{i=1}^{p-1} i \pmod{p}$,
Since multiplication is commutative.

$a^{p-1} (1 \cdot 2 \cdots (p-1)) \equiv (1 \cdot 2 \cdots (p-1)) \pmod{p}$.
Each of $2, \ldots, (p-1)$ has an inverse modulo $p$,
multiply by inverses to get...

$a^{p-1} \equiv 1 \pmod{p}$. 


Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$, $a^{p-1} \equiv 1 \pmod{p}$.

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

$T$ is range of function $f(x) = ax \pmod{p}$ for set $S = \{1, \ldots, p-1\}$.

Invertible function: one-to-one.

$T \subseteq S$ since $0 \not\in T$.

$p$ is prime.
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{ a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p} \}$.

$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$. Invertible function: one-to-one.

$T \subseteq S$ since $0 \not\in T$.

$p$ is prime.

$\implies T = S$. 
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \neq 0 \pmod{p}$,

\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.

Invertible function: one-to-one.

$T \subseteq S$ since $0 \notin T$.

$p$ is prime.

\[ \Rightarrow \quad T = S. \]

Product of elts of $T = \text{Product of elts of } S$. 

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

$T$ is range of function $f(x) = ax \pmod{p}$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
$p$ is prime.
$\implies T = S$.
Product of elts of $T = \text{Product of elts of } S$.
\[ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}, \]
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),
\[
a^{p-1} \equiv 1 \pmod{p}.
\]

**Proof:** Consider \( T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\} \).

\( T \) is range of function \( f(x) = ax \pmod{p} \) for set \( S = \{1, \ldots, p-1\} \).

Invertible function: one-to-one.

\( T \subseteq S \) since \( 0 \not\in T \).

\( p \) is prime.

\( \implies T = S \).

Product of els of \( T = \) Product of els of \( S \).

\[
(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},
\]

Since multiplication is commutative.
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$  

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.  

$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.  
Invertible function: one-to-one.  
$T \subseteq S$ since $0 \not\in T$.  
$p$ is prime.  
$\implies T = S$.  
Product of elts of $T = $ Product of elts of $S$.  

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$  
Since multiplication is commutative.  

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}.$$


**Fermat from Bijection.**

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.

Invertible function: one-to-one.

$T \subseteq S$ since $0 \notin T$.

$p$ is prime.

$\implies T = S$.

Product of els of $T = \text{Product of els of } S$.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo $p$, 

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

Proof: Consider $T = \{ a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p} \}$.

$T$ is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1 \}$.
Invertible function: one-to-one.

$T \subseteq S$ since $0 \not\in T$.

$p$ is prime.

$\implies T = S$.

Product of elts of $T = \text{Product of elts of } S$.

\[ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}, \]

Since multiplication is commutative.

\[ a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}. \]

Each of $2, \ldots (p-1)$ has an inverse modulo $p$,
multiply by inverses to get...
**Fermat from Bijection.**

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{ a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p} \}$.

$T$ is range of function $f(x) = ax \mod(p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.

$T \subseteq S$ since $0 \not\in T$.
$p$ is prime.
\[ \implies T = S. \]

Product of elts of $T = \text{Product of elts of } S$.
\[ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}, \]
Since multiplication is commutative.
\[ a^{(p-1)}(1 \cdots (p-1)) \equiv (1\cdots (p-1)) \pmod{p}. \]

Each of $2, \ldots, (p-1)$ has an inverse modulo $p$,
multiply by inverses to get...
\[ a^{(p-1)} \equiv 1 \pmod{p}. \]
Fermat from Bijection.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

**Proof:** Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

$T$ is range of function $f(x) = ax \pmod{p}$ for set $S = \{1, \ldots, p-1\}$.

Invertible function: one-to-one.

$T \subseteq S$ since $0 \not\in T$.

$p$ is prime.

$\implies T = S$.

Product of elts of $T = \text{Product of elts of } S$.

\[(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},\]

Since multiplication is commutative.

\[ a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}. \]

Each of $2, \ldots (p-1)$ has an inverse modulo $p$,
multiply by inverses to get...

\[ a^{(p-1)} \equiv 1 \pmod{p}. \]
RSA

RSA:

\[ N = p \times q \] with \( \gcd(e, (p-1)(q-1)) = 1 \).

\[ d = e^{-1} \pmod{(p-1)(q-1)} \].

**Theorem:**

\[ x^{ed} = x \pmod{N} \]

**Proof:**

\[ x^{ed} - x \] is divisible by \( p \) and \( q \) \( \Rightarrow \) theorem!

\[ x^{ed} - x = x^k(p-1)(q-1) + 1 - x = x((x^k(q-1))p - 1 - 1) \]

If \( x \) is divisible by \( p \), the product is.

Otherwise \( (x^k(q-1))p - 1 \) divisible by \( p \).

= \( \Rightarrow (x^k(q-1))p - 1 \) \( \pmod{p} \) by Fermat.

Similarly for \( q \).
RSA:

$$N = p, q$$
RSA

RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
RSA:

\( N = p, q \)
\( e \) with \( \gcd(e, (p - 1)(q - 1)) = 1. \)
\( d = e^{-1} \pmod{(p - 1)(q - 1)}. \)
RSA:

$N = p, q$

$e$ with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

**Theorem:** $x^{ed} = x \pmod{N}$
RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p-1)(q-1)) = 1. \]
\[ d = e^{-1} \pmod{(p-1)(q-1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**
RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
\[ d = e^{-1} \mod (p - 1)(q - 1). \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**
\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!
RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**
\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!
\( x^{ed} - x \)
RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**

\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!

\[ x^{ed} - x = x^{k(p-1)(q-1)+1} - x \]
RSA

RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**

\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!

\[ x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1) \]
RSA

RSA:
\[ N = p, q \]
with
\[ e \text{ with } \gcd(e, (p-1)(q-1)) = 1. \]
\[ d = e^{-1} \pmod{(p-1)(q-1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**
\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!
\[ x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1) \]
If \( x \) is divisible by \( p \), the product is.
RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**
\[ x^{ed} - x \text{ is divisible by } p \text{ and } q \implies \text{theorem!} \]
\[ x^{ed} - x = x^{k(p - 1)(q - 1) + 1} - x = x((x^{k(q - 1)})^{p - 1} - 1) \]

If \( x \) is divisible by \( p \), the product is.
Otherwise \((x^{k(q - 1)})^{p - 1} = 1 \pmod{p}\) by Fermat.
RSA:

\[ N = p, q \]

\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]

\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:**

\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!

\[ x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1) \]

If \( x \) is divisible by \( p \), the product is.

Otherwise \( (x^{k(q-1)})^{p-1} = 1 \pmod{p} \) by Fermat.

\( \implies (x^{k(q-1)})^{p-1} - 1 \) divisible by \( p \).
RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p-1)(q-1)) = 1. \]
\[ d = e^{-1} \pmod{(p-1)(q-1)}. \]

**Theorem:** \( x^{ed} = x \pmod{N} \)

**Proof:** \( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!

\[ x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1) \]

If \( x \) is divisible by \( p \), the product is.
Otherwise \( (x^{k(q-1)})^{p-1} = 1 \pmod{p} \) by Fermat.

\[ \implies (x^{k(q-1)})^{p-1} - 1 \text{ divisible by } p. \]

Similarly for \( q \).
RSA

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) = 1. \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

**Theorem:** \( x^{ed} = x \pmod N \)

**Proof:**
\( x^{ed} - x \) is divisible by \( p \) and \( q \) \( \implies \) theorem!
\[ x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1) \]
If \( x \) is divisible by \( p \), the product is.
Otherwise \( (x^{k(q-1)})^{p-1} = 1 \pmod p \) by Fermat.
\[ \implies (x^{k(q-1)})^{p-1} - 1 \] divisible by \( p \).

Similarly for \( q \).
RSA, Public Key, and Signatures.

RSA:

\[ N = p \cdot q \]

with \( \gcd(e, (p-1)(q-1)) = 1 \).

\[ d = e^{-1} \pmod{(p-1)(q-1)} \]

Public Key Cryptography:

\[ D(E(m, K), k) = m^e d \bmod N = m. \]

Signature scheme:

\[ S(C) = D(C). \]

Announce \( (C, S(C)) \)

Verify: Check \( C = E(C) \).

\[ E(D(C, K), K) = C^d e = C \pmod N \]
RSA, Public Key, and Signatures.

RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p-1)(q-1)) \]
\[ d = e^{-1} \pmod{(p-1)(q-1)} \]

Public Key Cryptography:

\[ D(E(m, K), k) = (m^e)^d \equiv m \pmod{N} \]

Signature scheme:

\[ S(C) = D(C) \]

Announce \( (C, S(C)) \):

Verify: Check \( C = E(C) \).

\[ E(D(C, K), K) = (C^d)^e \equiv C \pmod{N} \]
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)) \]
\[ d = e^{-1} \mod (p - 1)(q - 1) \]

Public Key Cryptography:
RSA, Public Key, and Signatures.

RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p-1)(q-1)). \]
\[ d = e^{-1} \pmod{(p-1)(q-1)}. \]

Public Key Cryptography:

\[ D(E(m, K), k) = (m^e)^d \pmod{N} = m. \]
RSA, Public Key, and Signatures.

RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \mod (p - 1)(q - 1). \]

Public Key Cryptography:

\[ D(E(m, K), k) = (m^e)^d \mod N = m. \]

Signature scheme:
RSA, Public Key, and Signatures.

RSA:

\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

Public Key Cryptography:

\[ D(E(m, K), k) = (m^e)^d \pmod{N} = m. \]

Signature scheme:

\[ S(C) = D(C). \]
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

Public Key Cryptography:
\[ D(E(m, K), k) = (m^e)^d \pmod{N} = m. \]

Signature scheme:
\[ S(C) = D(C). \]
Announce \((C, S(C))\)
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

Public Key Cryptography:
\[ D(E(m, K), k) = (m^e)^d \mod N = m. \]

Signature scheme:
\[ S(C) = D(C). \]
Announce \((C, S(C))\)
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

Public Key Cryptography:
\[ D(E(m, K), k) = (m^e)^d \pmod{N} = m. \]

Signature scheme:
\[ S(C) = D(C). \]
Announce \((C, S(C))\)
Verify: Check \(C = E(C).\)
RSA, Public Key, and Signatures.

RSA:
\[ N = p, q \]
\[ e \text{ with } \gcd(e, (p - 1)(q - 1)). \]
\[ d = e^{-1} \pmod{(p - 1)(q - 1)}. \]

Public Key Cryptography:
\[ D(E(m, K), k) = (m^e)^d \pmod{N} = m. \]

Signature scheme:
\[ S(C) = D(C). \]
Announce \((C, S(C))\)

Verify: Check \(C = E(C).\)
\[ E(D(C, k), K) = (C^d)^e = C \pmod{N} \]
Modular Arithmetic in a minute.

Euclid's Alg:
\[ \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \]
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \(a\), \(b\) where \(ax + by = \text{gcd}(x, y)\).
Idea: compute \(a\), \(b\) recursively (euclid), or iteratively.

Inverse: \(ax + by = ax = \text{gcd}(x, y) \mod y\).
If \(\text{gcd}(x, y) = 1\), we have \(ax = 1 \mod y\) → \(a = x^{-1} \mod y\).

Chinese Remainder Theorem: If \(\text{gcd}(n, m) = 1\),\( x = a \mod n\), \(x = b \mod m\) unique sol.
Proof: Find \(u = 1 \mod n\), \(u = 0 \mod m\), and \(v = 0 \mod n\), \(v = 1 \mod m\).
Then:
\[ x = au + bv = a \mod n \]
\[ u = m(m - 1) \mod n \]
works!

Fermat: Prime \(p\), \(a^{p-1} = 1 \mod p\).
Proof Idea: \(f(x) = a^x \mod p\): bijection on \(S = \{1, \ldots, p-1\}\).
Product of elts == for range/domain:
\(a^{p-1}\) factor in range.
Modular Arithmetic in a minute.

Euclid’s Alg: $\text{gcd}(x, y) = \text{gcd}(y, x \mod y)$
Modular Arithmetic in a minute.

Euclid’s Alg: \( \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \)

Fast cuz value drops by a factor of two every two recursive calls.
Modular Arithmetic in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.
Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.

Idea: compute $a, b$ recursively (euclid), or iteratively.
Modular Arithmetic in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.

Idea: compute $a, b$ recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \mod y$. 

Chinese Remainder Theorem:

If $gcd(x, y) = 1$, we have $ax = 1 \mod y → a = x - 1 \mod y$.

Fermat: Prime $p$, $a^p - 1 = 1 \mod p$.

Proof Idea: $f(x) = a^x (\mod p)$: bijection on $S = \{1, \ldots, p-1\}$.

Product of elts == for range/domain: $a^{p-1}$ factor in range.
Modular Arithmetic in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
   If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
Modular Arithmetic in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
   Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
   Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
   If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
   \( \rightarrow a = x^{-1} \mod y \).
Modular Arithmetic in a minute.

Euclid’s Alg: \( \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \)

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \text{gcd}(x, y) \).

Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse: \( ax + by = ax = \text{gcd}(x, y) \mod y \).

If \( \text{gcd}(x, y) = 1 \), we have \( ax = 1 \mod y \)

\( \rightarrow a = x^{-1} \mod y \).
Modular Arithmetic in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
   If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
   \( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
   If \( \gcd(n, m) = 1 \), \( x = a \mod n, x = b \mod m \) unique sol.
Modular Arithmetic in a minute.

Euclid's Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
   Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
   Inverse: \( ax + by = ax = \gcd(x, y) \mod y. \)
   If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
   \( \rightarrow a = x^{-1} \mod y. \)

Chinese Remainder Theorem:
   If \( \gcd(n, m) = 1, x = a \mod n, x = b \mod m \) unique sol.
   Proof: Find \( u = 1 \mod n, u = 0 \mod m, \)
Modular Arithmetic in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
   Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
   If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
   \( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
   If \( \gcd(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.
   Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \),
       and \( v = 0 \mod n \), \( v = 1 \mod m \).
Modular Arithmetic in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
   Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.
   Idea: compute $a, b$ recursively (euclid), or iteratively.
   Inverse: $ax + by = ax = gcd(x, y) \mod y$.
      If $gcd(x, y) = 1$, we have $ax = 1 \mod y$
       $\rightarrow a = x^{-1}\mod y$.

Chinese Remainder Theorem:
   If $gcd(n, m) = 1$, $x = a \mod n, x = b \mod m$ unique sol.

      Proof: Find $u = 1 \mod n, u = 0 \mod m$, 
            and $v = 0 \mod n, v = 1 \mod m$.
      Then: $x = au + bv = a \mod n$...
Modular Arithmetic in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
\( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
If \( \gcd(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.

Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \),
and \( v = 0 \mod n \), \( v = 1 \mod m \).
Then: \( x = au + bv = a \mod n \)...
\( u = m(m^{-1} \mod n) \mod n \) works!
Modular Arithmetic in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
- Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
- Idea: compute \( a, b \) recursively (euclid), or iteratively.
- Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
  - If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
    \( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
- If \( \gcd(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.
- Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \),
  and \( v = 0 \mod n \), \( v = 1 \mod m \).
  Then: \( x = au + bv = a \mod n \)...
  \( u = m(m^{-1} \mod n)) \mod n \) works!

Fermat: Prime \( p \), \( a^{p-1} = 1 \mod p \).
Modular Arithmetic in a minute.

Euclid’s Alg: \( \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \text{gcd}(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \text{gcd}(x, y) \mod y \).
   If \( \text{gcd}(x, y) = 1 \), we have \( ax = 1 \mod y \)
   \( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
   If \( \text{gcd}(n, m) = 1 \), \( x = a \mod n, x = b \mod m \) unique sol.
   Proof: Find \( u = 1 \mod n, u = 0 \mod m \),
   and \( v = 0 \mod n, v = 1 \mod m \).
   Then: \( x = au + bv = a \mod n \)...
   \( u = m(m^{-1} \mod n) \mod n \) works!

Fermat: Prime \( p \), \( a^{p-1} = 1 \mod p \).
   Proof Idea: \( f(x) = a(x) \mod p \): bijection on \( S = \{1, \ldots, p-1\} \).
Modular Arithmetic in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.
   Idea: compute $a, b$ recursively (euclid), or iteratively.
Invert: $ax + by = ax = gcd(x, y) \mod y$.
   If $gcd(x, y) = 1$, we have $ax = 1 \mod y$
   $\rightarrow a = x^{-1} \mod y$.

Chinese Remainder Theorem:
   If $gcd(n, m) = 1$, $x = a \mod n, x = b \mod m$ unique sol.
   Proof: Find $u = 1 \mod n, u = 0 \mod m$,
      and $v = 0 \mod n, v = 1 \mod m$.
   Then: $x = au + bv = a \mod n$...
      $u = m(m^{-1} \mod n)) \mod n$ works!

Fermat: Prime $p$, $a^{p-1} = 1 \mod p$.
   Proof Idea: $f(x) = a(x) \mod p$: bijection on $S = \{1, \ldots, p-1\}$.
   Product of els $=$ for range/domain: $a^{p-1}$ factor in range.
Modular Arithmetic Fact: Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d + 1$ points.
Lemma 1: $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0:
$P(x) = (x - a)Q(x)$. 

Only $d$ roots.

Lemma 2: $P(x)$ has $d$ roots; $r_1, \ldots, r_d$ then $P(x) = c(x - r_1)(x - r_2)\cdots(x - r_d)$. 

Proof Sketch: By induction.

Induction Step: $P(x) = (x - r_1)Q(x)$ by Lemma 1. $Q(x)$ has smaller degree so use the induction hypothesis.

Implication: $d + 1$ roots $\rightarrow \geq d + 1$ terms $\Rightarrow$ degree is $\geq d + 1$.

Roots fact: Any degree $\leq d$ polynomial has at most $d$ roots.
Lemma 1: $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0: 
$P(x) = (x - a)Q(x)$.

Proof: $P(x) = (x - a)Q(x) + r$. Plugin $a$: $P(a) = r$. 
Lemma 1: \( P(x) \) has root \( a \) iff \( P(x)/(x - a) \) has remainder 0: 
\[ P(x) = (x - a)Q(x). \]

Proof: \( P(x) = (x - a)Q(x) + r. \)
Plugin \( a \): \( P(a) = r. \)
It is a root if and only if \( r = 0. \)
Lemma 1: \( P(x) \) has root \( a \) iff \( P(x)/(x - a) \) has remainder 0: \( P(x) = (x - a)Q(x) \).

Proof: \( P(x) = (x - a)Q(x) + r \).
Plug in \( a \): \( P(a) = r \).
It is a root if and only if \( r = 0 \).
Only $d$ roots.

**Lemma 1:** $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0: $P(x) = (x - a)Q(x)$.

**Proof:** $P(x) = (x - a)Q(x) + r$. Plugin $a$: $P(a) = r$. It is a root if and only if $r = 0$.

**Lemma 2:** $P(x)$ has $d$ roots; $r_1, \ldots, r_d$ then $P(x) = c(x)(x - r_1)(x - r_2) \cdots (x - r_d)$. 
Lemma 1: $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0: $P(x) = (x - a)Q(x)$.

Proof: $P(x) = (x - a)Q(x) + r$.
Plugin $a$: $P(a) = r$.
It is a root if and only if $r = 0$.

Lemma 2: $P(x)$ has $d$ roots; $r_1, \ldots, r_d$ then $P(x) = c(x)(x - r_1)(x - r_2) \cdots (x - r_d)$.
Proof Sketch: By induction.
Only $d$ roots.

Lemma 1: $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0: 
$P(x) = (x - a)Q(x)$.

Proof: $P(x) = (x - a)Q(x) + r$.
Plug in $a$: $P(a) = r$.
It is a root if and only if $r = 0$.

Lemma 2: $P(x)$ has $d$ roots; $r_1, \ldots, r_d$ then 
$P(x) = c(x)(x - r_1)(x - r_2)\cdots(x - r_d)$.

Proof Sketch: By induction.

Induction Step: $P(x) = (x - r_1)Q(x)$ by Lemma 1. $Q(x)$ has smaller 
degree so use the induction hypothesis.
Lemma 1: \( P(x) \) has root \( a \) iff \( P(x)/(x - a) \) has remainder 0: \( P(x) = (x - a)Q(x) \).

Proof: \( P(x) = (x - a)Q(x) + r \).

Plugin \( a \): \( P(a) = r \).

It is a root if and only if \( r = 0 \).

Lemma 2: \( P(x) \) has \( d \) roots; \( r_1, \ldots, r_d \) then \( P(x) = c(x)(x - r_1)(x - r_2)\cdots(x - r_d) \).

Proof Sketch: By induction.

Induction Step: \( P(x) = (x - r_1)Q(x) \) by Lemma 1. \( Q(x) \) has smaller degree so use the induction hypothesis.
Only $d$ roots.

Lemma 1: $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0: $P(x) = (x - a)Q(x)$.

Proof: $P(x) = (x - a)Q(x) + r$.
Plugin $a$: $P(a) = r$.
It is a root if and only if $r = 0$.

Lemma 2: $P(x)$ has $d$ roots; $r_1, \ldots, r_d$ then $P(x) = c(x)(x - r_1)(x - r_2) \cdots (x - r_d)$.

Proof Sketch: By induction.

Induction Step: $P(x) = (x - r_1)Q(x)$ by Lemma 1. $Q(x)$ has smaller degree so use the induction hypothesis.

Implication: $d + 1$ roots $\rightarrow \geq d + 1$ terms $\implies$ degree is $\geq d + 1$. 
Only $d$ roots.

**Lemma 1:** $P(x)$ has root $a$ iff $P(x)/(x - a)$ has remainder 0: $P(x) = (x - a)Q(x)$.

**Proof:** $P(x) = (x - a)Q(x) + r$.
Plugin $a$: $P(a) = r$.
It is a root if and only if $r = 0$.

**Lemma 2:** $P(x)$ has $d$ roots; $r_1, \ldots, r_d$ then $P(x) = c(x)(x - r_1)(x - r_2)\cdots(x - r_d)$.

**Proof Sketch:** By induction.

Induction Step: $P(x) = (x - r_1)Q(x)$ by Lemma 1. $Q(x)$ has smaller degree so use the induction hypothesis.

Implication: $d + 1$ roots $\rightarrow \geq d + 1$ terms $\implies$ degree is $\geq d + 1$.

**Roots fact:** Any degree $\leq d$ polynomial has at most $d$ roots.
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d + 1$ points.
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p), P(x)$, that hits $d + 1$ points.

**Shamir’s $k$ out of $n$ Scheme:**
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d + 1$ points.

**Shamir's $k$ out of $n$ Scheme:**
Secret $s \in \{0, \ldots, p - 1\}$
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree \( \leq d \) over \( GF(p) \), \( P(x) \), that hits \( d + 1 \) points.

**Shamir’s \( k \) out of \( n \) Scheme:**
Secret \( s \in \{0,\ldots,p-1\} \)

1. Choose \( a_0 = s \), and random \( a_1, \ldots, a_{k-1} \).
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d + 1$ points.

**Shamir’s $k$ out of $n$ Scheme:**

Secret $s \in \{0, \ldots, p - 1\}$

1. Choose $a_0 = s$, and random $a_1, \ldots, a_{k-1}$.

2. Let $P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots a_0$ with $a_0 = s$. 
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d + 1$ points.

**Shamir’s $k$ out of $n$ Scheme:**
Secret $s \in \{0, \ldots, p-1\}$

1. Choose $a_0 = s$, and random $a_1, \ldots, a_{k-1}$.
2. Let $P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots a_0$ with $a_0 = s$.
3. Share $i$ is point $(i, P(i) \mod p)$. 

**Robustness:** Any $k$ knows secret.

**Knowing $k$ pts, only one $P(x)$, evaluate $P(0)$.**

**Secrecy:** Any $k-1$ knows nothing.

**Knowing $\leq k-1$ pts, any $P(0)$ is possible.**

**Efficiency:** ???
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree \( \leq d \) over \( GF(p) \), \( P(x) \), that hits \( d + 1 \) points.

**Shamir’s \( k \) out of \( n \) Scheme:**
Secret \( s \in \{0,\ldots,p-1\} \)

1. Choose \( a_0 = s \), and random \( a_1,\ldots,a_{k-1} \).
2. Let \( P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots a_0 \) with \( a_0 = s \).
3. Share \( i \) is point \( (i, P(i) \mod p) \).
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree \( \leq d \) over \( GF(p) \), \( P(x) \), that hits \( d + 1 \) points.

**Shamir’s \( k \) out of \( n \) Scheme:**
Secret \( s \in \{0, \ldots, p-1\} \)

1. Choose \( a_0 = s \), and random \( a_1, \ldots, a_{k-1} \).
2. Let \( P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots a_0 \) with \( a_0 = s \).
3. Share \( i \) is point \((i, P(i) \mod p)\).

**Robustness:** Any \( k \) knows secret.
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d+1$ points.

**Shamir’s $k$ out of $n$ Scheme:**
Secret $s \in \{0, \ldots, p-1\}$

1. Choose $a_0 = s$, and random $a_1, \ldots, a_{k-1}$.

2. Let $P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots a_0$ with $a_0 = s$.

3. Share $i$ is point $(i, P(i) \mod p)$.

**Robustness:** Any $k$ knows secret.
Knowing $k$ pts, only one $P(x)$, evaluate $P(0)$.

**Secrecy:** Any $k-1$ knows nothing.
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree \( \leq d \) over \( GF(p) \), \( P(x) \), that hits \( d + 1 \) points.

**Shamir’s \( k \) out of \( n \) Scheme:**
Secret \( s \in \{0, \ldots, p - 1\} \)

1. Choose \( a_0 = s \), and random \( a_1, \ldots, a_{k-1} \).
2. Let \( P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_0 \) with \( a_0 = s \).
3. Share \( i \) is point \( (i, P(i) \mod p) \).

**Robustness:** Any \( k \) knows secret.
Knowing \( k \) pts, only one \( P(x) \), evaluate \( P(0) \).

**Secrecy:** Any \( k - 1 \) knows nothing.
Knowing \( \leq k - 1 \) pts, any \( P(0) \) is possible.
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree \( \leq d \) over \( GF(p) \), \( P(x) \), that hits \( d + 1 \) points.

**Shamir’s \( k \) out of \( n \) Scheme:**

Secret \( s \in \{0, \ldots, p - 1\} \)

1. Choose \( a_0 = s \), and random \( a_1, \ldots, a_{k-1} \).
2. Let \( P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots + a_0 \) with \( a_0 = s \).
3. Share \( i \) is point \((i, P(i) \mod p)\).

**Roubustness:** Any \( k \) knows secret.
Knowing \( k \) pts, only one \( P(x) \), evaluate \( P(0) \).

**Secrecy:** Any \( k - 1 \) knows nothing.
Knowing \( \leq k - 1 \) pts, any \( P(0) \) is possible.

**Efficiency:** ????
Secret Sharing

**Modular Arithmetic Fact:** Exactly one polynomial degree $\leq d$ over $GF(p)$, $P(x)$, that hits $d+1$ points.

**Shamir’s $k$ out of $n$ Scheme:**

Secret $s \in \{0, \ldots, p-1\}$

1. Choose $a_0 = s$, and random $a_1, \ldots, a_{k-1}$.
2. Let $P(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \cdots a_0$ with $a_0 = s$.
3. Share $i$ is point $(i, P(i) \mod p)$.

**Robustness:** Any $k$ knows secret.
Knowing $k$ pts, only one $P(x)$, evaluate $P(0)$.

**Secrecy:** Any $k-1$ knows nothing.
Knowing $\leq k-1$ pts, any $P(0)$ is possible.

**Efficiency:** ???
Efficiency.

Need to hand out \( n \) shares:

\[ P(1) \ldots P(n) \.

For \( b \)-bit secret, must choose a prime \( p > 2^b \).

Theorem: There is always a prime between \( n \) and \( 2n \).

Chebyshev said it, and I say it again, there is always a prime between \( n \) and \( 2n \).

Working over numbers within 1 bit of secret size.

Minimal!

With \( k \) shares, reconstruct polynomial, \( P(x) \).

With \( k - 1 \) shares, any of \( p \) values possible for \( P(0) \)!

(Within 1 bit of) any \( b \)-bit string possible!

(Within 1 bit of) \( b \)-bits are missing: one \( P(i) \).

Within 1 of optimal number of bits.
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$. 

Working over numbers within 1 bit of secret size. Minimal!

With $k$ shares, reconstruct polynomial, $P(x)$. With $k-1$ shares, any of $p$ values possible for $P(0)$! (Within 1 bit of) any $b$-bit string possible! (Within 1 bit of) $b$-bits are missing: one $P(i)$. Within 1 of optimal number of bits.
Efficiency.

Need \( p > n \) to hand out \( n \) shares: \( P(1) \ldots P(n) \).

For \( b \)-bit secret, must choose a prime \( p > 2^b \).
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$. 
Chebyshev said it,
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.

Chebyshev said it,
And I say it again,
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.

Chebyshev said it,
And I say it again,
There is always a prime...
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.

Chebyshev said it,
And I say it again,
There is always a prime
Between $n$ and $2n$. 
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.
	Chebyshev said it,
	And I say it again,
	There is always a prime
	Between $n$ and $2n$.

Working over numbers *within 1 bit* of secret size.
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem**: There is always a prime between $n$ and $2n$.
Chebyshev said it,
And I say it again,
There is always a prime
Between $n$ and $2n$.

Working over numbers within 1 bit of secret size.
**Minimal!**
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.

For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.
Chebyshev said it,
And I say it again,
There is always a prime
Between $n$ and $2n$.

Working over numbers **within 1 bit** of secret size.

**Minimal!**

With $k$ shares, reconstruct polynomial, $P(x)$. 
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.
For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.
Chebyshev said it,
And I say it again,
There is always a prime
Between $n$ and $2n$.

Working over numbers within 1 bit of secret size.
**Minimal!**

With $k$ shares, reconstruct polynomial, $P(x)$.
With $k - 1$ shares, any of $p$ values possible for $P(0)$!
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.
For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.
Chebyshev said it,
And I say it again,
There is always a prime
Between $n$ and $2n$.

Working over numbers within 1 bit of secret size.
**Minimal!**

With $k$ shares, reconstruct polynomial, $P(x)$.
With $k - 1$ shares, any of $p$ values possible for $P(0)$!
(Within 1 bit of) any $b$-bit string possible!
Efficiency.

Need \( p > n \) to hand out \( n \) shares: \( P(1) \ldots P(n) \).

For \( b \)-bit secret, must choose a prime \( p > 2^b \).

**Theorem:** There is always a prime between \( n \) and \( 2n \).
Chebyshev said it,
And I say it again,
There is always a prime
Between \( n \) and \( 2n \).

Working over numbers within 1 bit of secret size.

**Minimal!**

With \( k \) shares, reconstruct polynomial, \( P(x) \).

With \( k - 1 \) shares, any of \( p \) values possible for \( P(0) \! \)
(Within 1 bit of) any \( b \)-bit string possible!
(Within 1 bit of) \( b \)-bits are missing: one \( P(i) \).
Efficiency.

Need $p > n$ to hand out $n$ shares: $P(1) \ldots P(n)$.
For $b$-bit secret, must choose a prime $p > 2^b$.

**Theorem:** There is always a prime between $n$ and $2n$.
Chebyshev said it,
And I say it again,
There is always a prime
Between $n$ and $2n$.

Working over numbers within 1 bit of secret size.

**Minimal!**

With $k$ shares, reconstruct polynomial, $P(x)$.
With $k - 1$ shares, any of $p$ values possible for $P(0)$!
(Within 1 bit of) any $b$-bit string possible!
(Within 1 bit of) $b$-bits are missing: one $P(i)$.
Within 1 of optimal number of bits.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets?
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode?
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$. 
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree?
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover?
Summary. Error Correction.

Communicate \( n \) packets, with \( k \) erasures.

How many packets? \( n + k \)
How to encode? With polynomial, \( P(x) \).
Of degree? \( n - 1 \)
Recover? Reconstruct \( P(x) \) with any \( n \) points!

Reed-Solomon codes.
Welsh-Berlekamp Decoding.
Perfection!
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$  
How to encode? With polynomial, $P(x)$.  
Of degree? $n - 1$  
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets?
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
Summary. Error Correction.

Communicate \( n \) packets, with \( k \) erasures.

- How many packets? \( n + k \)
- How to encode? With polynomial, \( P(x) \).
- Of degree? \( n - 1 \)
- Recover? Reconstruct \( P(x) \) with any \( n \) points!

Communicate \( n \) packets, with \( k \) errors.

- How many packets? \( n + 2k \)
- Why?
Summary. Error Correction.

Communicate \( n \) packets, with \( k \) erasures.

How many packets? \( n + k \)
How to encode? With polynomial, \( P(x) \).
Of degree? \( n - 1 \)
Recover? Reconstruct \( P(x) \) with any \( n \) points!

Communicate \( n \) packets, with \( k \) errors.

How many packets? \( n + 2k \)
Why?
  \( k \) changes to make diff. messages overlap
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
- Why?
  - $k$ changes to make diff. messages overlap
- How to encode?
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
- Why?
  - $k$ changes to make diff. messages overlap
- How to encode? With polynomial, $P(x)$.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
    $k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree?
Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
   $k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree? $n - 1$. 

Nonlinear equations.
Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$.
Linear Equations.
Polynomial division!

Reed-Solomon codes.
Welsh-Berlekamp Decoding.
Perfection!
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n − 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
$k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree? $n − 1$.
Recover?

Reed-Solomon codes.
Welsh-Berlekamp Decoding.
Perfection!
Communicate \( n \) packets, with \( k \) erasures.

- How many packets? \( n + k \)
- How to encode? With polynomial, \( P(x) \).
- Of degree? \( n - 1 \)
- Recover? Reconstruct \( P(x) \) with any \( n \) points!

Communicate \( n \) packets, with \( k \) errors.

- How many packets? \( n + 2k \)
- Why?
  - \( k \) changes to make diff. messages overlap
- How to encode? With polynomial, \( P(x) \). Of degree? \( n - 1 \).
- Recover?
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
  $k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree? $n - 1$.
Recover?
  Reconstruct error polynomial, $E(X)$, and $P(x)$!
**Summary. Error Correction.**

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
- Why?
  - $k$ changes to make diff. messages overlap
- Recover?
  - Reconstruct error polynomial, $E(X)$, and $P(x)$!
    - *Nonlinear equations.*
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
- Why?
  - $k$ changes to make diff. messages overlap
- Recover?
  - Reconstruct error polynomial, $E(X)$, and $P(x)$!
    - Nonlinear equations.
  - Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$. 
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
   $k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree? $n - 1$.
Recover?
   Reconstruct error polynomial, $E(X)$, and $P(x)$!
   Nonlinear equations.
Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$. Linear Equations.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
  $k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree? $n - 1$.
Recover?
  Reconstruct error polynomial, $E(X)$, and $P(x)$!
    Nonlinear equations.
Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$. Linear Equations.
  Polynomial division!
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
  - Why?
    - $k$ changes to make diff. messages overlap
- Recover?
  - Reconstruct error polynomial, $E(X)$, and $P(x)$!
    - **Nonlinear equations.**
  - Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$. Linear Equations.
  - Polynomial division! $P(x) = Q(x)/E(x)$!
Communicate $n$ packets, with $k$ erasures.

How many packets? $n + k$
How to encode? With polynomial, $P(x)$.
Of degree? $n - 1$
Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

How many packets? $n + 2k$
Why?
    $k$ changes to make diff. messages overlap
How to encode? With polynomial, $P(x)$. Of degree? $n - 1$.
Recover?
    Reconstruct error polynomial, $E(X)$, and $P(x)$!
    Nonlinear equations.
Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$.
    Linear Equations.
Polynomial division! $P(x) = Q(x)/E(x)$!

Reed-Solomon codes.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
- Why?
  - $k$ changes to make diff. messages overlap
- Recover?
  - Reconstruct error polynomial, $E(X)$, and $P(x)$!
    - Nonlinear equations.
  - Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$. Linear Equations.
    - Polynomial division! $P(x) = Q(x)/E(x)$!

Reed-Solomon codes. Welsh-Berlekamp Decoding.
Summary. Error Correction.

Communicate $n$ packets, with $k$ erasures.

- How many packets? $n + k$
- How to encode? With polynomial, $P(x)$.
- Of degree? $n - 1$
- Recover? Reconstruct $P(x)$ with any $n$ points!

Communicate $n$ packets, with $k$ errors.

- How many packets? $n + 2k$
- Why?
  - $k$ changes to make diff. messages overlap
- Recover?
  - Reconstruct error polynomial, $E(X)$, and $P(x)$!
    - Nonlinear equations.
  - Reconstruct $E(x)$ and $Q(x) = E(x)P(x)$. Linear Equations.
  - Polynomial division! $P(x) = Q(x)/E(x)$!

Reed-Solomon codes. Welsh-Berlekamp Decoding. Perfection!
Midterm format

Time: 120 minutes.
Midterm format

Time: 120 minutes.
Some short answers.
Midterm format

Time: 120 minutes.

Some short answers.
   Get at ideas that you learned.
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
Know material well:
Midterm format

Time: 120 minutes.

Some short answers.
  Get at ideas that you learned.
  Know material well: fast,
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.
Midterm format

Time: 120 minutes.

Some short answers.
   Get at ideas that you learned.
   Know material well: fast, correct.
   Know material medium:
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.

Know material well: fast, correct.
Know material medium: slower,
Midterm format

Time: 120 minutes.

Some short answers.
  Get at ideas that you learned.
    Know material well: fast, correct.
    Know material medium: slower, less correct.
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
  Know material well: fast, correct.
  Know material medium: slower, less correct.
  Know material not so well:
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.

Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.
Midterm format

Time: 120 minutes.

Some short answers.
   Get at ideas that you learned.
      Know material well:    fast, correct.
      Know material medium: slower, less correct.
      Know material not so well:  Uh oh.

Some longer questions.
Midterm format

Time: 120 minutes.

Some short answers.
  Get at ideas that you learned.
  Know material well: fast, correct.
  Know material medium: slower, less correct.
  Know material not so well: Uh oh.

Some longer questions.
  Proofs,
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
   Know material well: fast, correct.
   Know material medium: slower, less correct.
   Know material not so well: Uh oh.

Some longer questions.
Proofs, algorithms,
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
  Know material well: fast, correct.
  Know material medium: slower, less correct.
  Know material not so well: Uh oh.

Some longer questions.
Proofs, algorithms, properties.
Midterm format

Time: 120 minutes.

Some short answers.
  Get at ideas that you learned.
    Know material well: fast, correct.
    Know material medium: slower, less correct.
    Know material not so well: Uh oh.

Some longer questions.
  Proofs, algorithms, properties.
  Not so much calculation.
Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.

Some longer questions.
Proofs, algorithms, properties.
Not so much calculation.
Midterm format

Time: 120 minutes.

Some short answers.
  Get at ideas that you learned.
  Know material well: fast, correct.
  Know material medium: slower, less correct.
  Know material not so well: Uh oh.

Some longer questions.
  Proofs, algorithms, properties.
  Not so much calculation.
Wrapup.
Wrapup.

Other issues....
Wrapup.

Other issues....
sp21@eeecs70.org
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.
Wrapup.

Other issues....
  sp21@eecs70.org
  Private message on piazza.

Good Studying!
Wrapup.

Other issues....
    sp21@eecs70.org
    Private message on piazza.

Good Studying!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!
Wrapup.

Other issues...
   sp21@eecs70.org
   Private message on piazza.

Good Studying!!!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!!!!
Wrapup.

Other issues....
   sp21@eecs70.org
   Private message on piazza.

Good Studying!!!!!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!!!!!!
Wrapup.

Other issues....
   sp21@eecs70.org
   Private message on piazza.

Good Studying!!!!!!!!!!
Wrapup.

Other issues....
   sp21@eecs70.org
   Private message on piazza.

Good Studying!!!!!!!!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!!!!!!!!
Wrapup.

Other issues....
   sp21@eecs70.org
   Private message on piazza.

Good Studying!!!!!!!!!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!!!!!!!!!!!
Wrapup.

Other issues....
  sp21@eecs70.org
  Private message on piazza.

Good Studying!!!!!!!!!!!!!
Wrapup.

Other issues....
   sp21@eecs70.org
   Private message on piazza.

Good Studying!!!!!!!!!!!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!!!!!!!!!!!
Wrapup.

Other issues....
sp21@eecs70.org
Private message on piazza.

Good Studying!!!!!!!!!!!!!