Warm up:
What is the following recursively-defined set?

Basis Step: $4 \in S, \; 5 \in S$

Recursive Step: If $x \in S$ and $y \in S$ then $x - y \in S$
Strings

$\varepsilon$ is “the empty string”

The string with 0 characters – “” in Java (not null!)

$\Sigma^*$:

- Basis: $\varepsilon \in \Sigma^*$.
- Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$

$wa$ means the string of $w$ with the character $a$ appended.

You’ll also see $w \cdot a$ (a · to mean “concatenate” i.e. + in Java)
Functions on Strings
Since strings are defined recursively, most functions on strings are as well.

Length:
\[ \text{len}(\varepsilon) = 0; \]
\[ \text{len}(wa) = \text{len}(w) + 1 \text{ for } w \in \Sigma^*, a \in \Sigma \]

Reversal:
\[ \varepsilon^R = \varepsilon; \]
\[ (wa)^R = aw^R \text{ for } w \in \Sigma^*, a \in \Sigma \]

Concatenation
\[ x \cdot \varepsilon = x \text{ for all } x \in \Sigma^*; \]
\[ x \cdot (wa) = (x \cdot w)a \text{ for } w \in \Sigma^*, a \in \Sigma \]

Number of c's in a string
\[ \#_c(\varepsilon) = 0 \]
\[ \#_c(wa) = \#_c(w) + 1 \text{ for } w \in \Sigma^*; \]
\[ \#_c(wa) = \#_c(w) \text{ for } w \in \Sigma^*, a \in \Sigma \setminus \{c\}. \]
1. Define $P()$ Show that $P(x)$ holds for all $x \in S$. State your proof is by structural induction.

2. Base Case: Show $P(x)$
[Do that for every base cases $x$ in $S$.]

Let $y$ be an arbitrary element of $S$ not covered by the base cases. By the exclusion rule, $y = \text{<recursive rules>}$

3. Inductive Hypothesis: Suppose $P(x)$
[Do that for every $x$ listed as in $S$ in the recursive rules.]

4. Inductive Step: Show $P()$ holds for $y$.
[You will need a separate case/step for every recursive rule.]

5. Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.
Claim for all $x, y \in \Sigma^*$ \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \). 

Let $P(y)$ be “for all $x \in \Sigma^*$ \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \).”

Notice the strangeness of this $P(y)$ there is a “for all $x$“ inside the definition of $P(y)$.

That means we’ll have to introduce an arbitrary $x$ as part of the base case and the inductive step!
Claim for all $x, y \in \Sigma^*$ \(\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)\).

Define Let $P(y)$ be “for all $x \in \Sigma^*$ \(\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)\).”

We prove $P(y)$ for all $y \in \Sigma^*$ by structural induction.

Base Case: $P(\varepsilon)$ holds for all $x$.

Inductive Hypothesis:

Inductive Step:

$\Sigma^*$: Basis: $\varepsilon \in \Sigma^*$.

Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$. 
Claim for all $x, y \in \Sigma^*$, $\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$.

Define $P(y)$ to be "for all $x \in \Sigma^*$, $\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$." We prove $P(y)$ for all $y \in \Sigma^*$ by structural induction.

Base Case: Let $x$ be an arbitrary string, $\text{len}(x \cdot \varepsilon) = \text{len}(x) = \text{len}(x) + 0 = \text{len}(x) + \text{len}(\varepsilon)$.

Let $y$ be an arbitrary string not covered by the base case. By the exclusion rule, $y = wa$ for a string $w$ and character $a$.

Inductive Hypothesis: Suppose $P(w)$.

Inductive Step: Let $x$ be an arbitrary string.

Therefore, $\text{len}(xwa) = \text{len}(x) + \text{len}(wa)$.

$\Sigma^*$: Basis: $\varepsilon \in \Sigma^*$.

Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$. 

\[ \text{len}(x \cdot y) = \text{len}(x \cdot w) \]
\[ = \frac{\text{len}(x \cdot w) + 1}{[\text{def of len}]} \]
\[ = \text{len}(x) + \text{len}(w) + 1 \quad [\text{def of len}] \]
\[ = \text{len}(x) + \text{len}(w) \quad [\text{def of len}] \]
\[ = \text{len}(x) + \text{len}(y). \]
Claim for all $x, y \in \Sigma^*$ \(\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)\).

Define Let $P(y)$ be “for all $x \in \Sigma^*$ \(\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)\).”

We prove $P(y)$ for all $y \in \Sigma^*$ by structural induction.

Base Case: Let $x$ be an arbitrary string, \(\text{len}(x \cdot \epsilon) = \text{len}(x) = \text{len}(x) + 0 = \text{len}(x) + \text{len}(\epsilon)\)

Let $y$ be an arbitrary string not covered by the base case. By the exclusion rule, $y = wa$ for a string $w$ and character $a$.

Inductive Hypothesis: Suppose $P(w)$

Inductive Step: Let $x$ be an arbitrary string.

\[
\text{len}(xy) = \text{len}(xwa) = \text{len}(xw) + 1 \text{ (by definition of len)}
\]

\[
= \text{len}(x) + \text{len}(w) + 1 \text{ (by IH)}
\]

\[
= \text{len}(x) + \text{len}(wa) \text{ (by definition of len)}
\]

Therefore, \(\text{len}(xy) = \text{len}(x) + \text{len}(y)\), as required.

We conclude that $P(y)$ holds for all string $y$ by the principle of induction. Unwrapping the definition of $y$, we get $
\forall x \forall y \in \Sigma^* \text{ len}(xy) = \text{len}(x) + \text{len}(y)$, as required.

$\Sigma^*$: Basis: $\epsilon \in \Sigma^*$.

Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$.
More Structural Sets

Binary Trees are another common source of structural induction.

Basis: A single node is a rooted binary tree.

Recursive Step: If $T_1$ and $T_2$ are rooted binary trees with roots $r_1$ and $r_2$, then a tree rooted at a new node, with children $r_1, r_2$ is a binary tree.
Functions on Binary Trees

size(●) = 1

size(T) = size(T_1) + size(T_2) + 1

height(●) = 0

height(T) = 1 + max(height(T_1), height(T_2))
Structural Induction on Binary Trees

Let $P(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $P(T)$ for all binary trees $T$ by structural induction.

Base Case: Let $T = \text{ }$. size($T$) = 1 and height($T$) = 0, so $\text{size}(T) = 1 \leq 2 - 1 = 2^{0+1} - 1 = 2^{\text{height}(T)+1} - 1$.

Let $T$ be an arbitrary tree not covered by the base case. By the exclusion rule, $T = \text{ }$. for trees $L, R$.

Inductive Hypothesis: Suppose $P(L)$ and $P(R)$.
Let $P(T)$ be "size($T$) $\leq 2^{\text{height}(T)} + 1 - 1$". We show $P(T)$ for all binary trees $T$ by structural induction.

$T = \text{ }$.

height($T$) = $1 + \max\{\text{height}(L), \text{height}(R)\}$

size($T$) = $1 + \text{size}(L) + \text{size}(R)$

So $P(T)$ holds, and we have $P(T)$ for all binary trees $T$ by the principle of induction.
Structural Induction on Binary Trees (cont.)

Let $P(T)$ be "size($T$) $\leq 2^{height(T)}+1 - 1". We show $P(T)$ for all binary trees $T$ by structural induction.

Let $T = \begin{tikzpicture}
\node (T) at (0,0) {$T$};
\node (L) at (-1,-2) {$L$};
\node (R) at (1,-2) {$R$};
\draw (T) -- (L);
\draw (T) -- (R);
\end{tikzpicture}
$

height($T$) = 1 + \max\{height(L), height(R)\}

size($T$) = 1 + size(L) + size(R)

size($T$) $\leq 1 + 2^{\text{height}(L)+1} - 1 + 2^{\text{height}(R)+1} - 1$ (by IH)

$\leq 2^{\text{height}(L)+1} + 2^{\text{height}(R)+1} - 1$ (cancel 1's)

$\leq 2^{\text{height}(T)} + 2^{\text{height}(T)} - 1 = 2^{\text{height}(T)+1} - 1$ ($T$ taller than subtrees)

So $P(T)$ holds, and we have $P(T)$ for all binary trees $T$ by the principle of induction.
What does the inductive step look like?

Here’s a recursively-defined set:

**Basis:** $0 \in T$ and $5 \in T$

**Recursive:** If $x, y \in T$ then $x + y \in T$ and $x - y \in T$.

Let $P(x)$ be "5|\(x\)"

What does the inductive step look like?

Well there’s two recursive rules, so we have two things to show
Just the IS (you still need the other steps)

Let \( t \) be an arbitrary element of \( T \) not covered by the base case. By the exclusion rule \( t = x + y \) or \( t = x - y \) for \( x, y \in T \).

Inductive hypothesis: Suppose \( P(x) \) and \( P(y) \) hold.

Case 1: \( t = x + y \)

By IH 5|\( x \) and 5|\( y \) so \( 5a = x \) and \( 5b = y \) for integers \( a, b \).

Adding, we get \( x + y = 5a + 5b = 5(a + b) \). Since \( a, b \) are integers, so is \( a + b \), and \( P(x + y) \), i.e. \( P(t) \), holds.

Case 2: \( t = x - y \)

By IH 5|\( x \) and 5|\( y \) so \( 5a = x \) and \( 5b = y \) for integers \( a, b \).

Subtracting, we get \( x - y = 5a - 5b = 5(a - b) \). Since \( a, b \) are integers, so is \( a - b \), and \( P(x - y) \), i.e., \( P(t) \), holds.

In all cases, we have \( P(t) \). By the principle of induction, \( P(x) \) holds for all \( x \in T \).
If you don’t have a recursively-defined set

You won’t do structural induction.
You can do weak or strong induction though.
For example, Let $P(n)$ be “for all elements of $S$ of “size” $n$ <something> is true”
To prove “for all $x \in S$ of size $n$...” you need to start with “let $x$ be an arbitrary element of size $k + 1$ in your IS.
You CAN’T start with size $k$ and “build up” to an arbitrary element of size $k + 1$ it isn’t arbitrary.
Induction: Hats!

You have $n$ people in a line ($n \geq 2$). Each of them wears either a purple hat or a gold hat. The person at the front of the line wears a purple hat. The person at the back of the line wears a gold hat.

Show that for every arrangement of the line satisfying the rule above, there is a person with a purple hat next to someone with a gold hat.

Yes, this is kinda obvious. I promise this is good induction practice.

Yes, you could argue this by contradiction. I promise this is good induction practice.
Induction: Hats!

Define \( P(n) \) to be “in every line of \( n \) people with gold and purple hats, with a purple hat at one end and a gold hat at the other, there is a person with a purple hat next to someone with a gold hat”

We show \( P(n) \) for all integers \( n \geq 2 \) by induction on \( n \).

Base Case: \( n = 2 \)
Inductive Hypothesis:
Inductive Step:

By the principle of induction, we have \( P(n) \) for all \( n \geq 2 \)
Induction: Hats!

Define $P(n)$ to be “in every line of $n$ people with gold and purple hats, with a purple hat at one end and a gold hat at the other, there is a person with a purple hat next to someone with a gold hat”

We show $P(n)$ for all integers $n \geq 2$ by induction on $n$.

Base Case: $n = 2$ The line must be just a person with a purple hat and a person with a gold hat, who are next to each other.

Inductive Hypothesis: Suppose $P(k)$ holds for an arbitrary $k \geq 2$.

Inductive Step: Consider an arbitrary line with $k + 1$ people in purple and gold hats, with a gold hat at one end and a purple hat at the other.

Target: there is someone in a purple hat next to someone in a gold hat.

By the principle of induction, we have $P(n)$ for all $n \geq 2$
Induction: Hats!

Define $P(n)$ to be “in every line of $n$ people with gold and purple hats, with a purple hat at one end and a gold hat at the other, there is a person with a purple hat next to someone with a gold hat”

We show $P(n)$ for all integers $n \geq 2$ by induction on $n$.

Base Case: $n = 2$ The line must be just a person with a purple hat and a person with a gold hat, who are next to each other.

Inductive Hypothesis: Suppose $P(k)$ holds for an arbitrary $k \geq 2$.

Inductive Step: Consider an arbitrary line with $k + 1$ people in purple and gold hats, with a gold hat at one end and a purple hat at the other.

Case 1: There is someone with a purple hat next to the person in the gold hat at one end. Then those people are the required adjacent opposite hats.

Case 2: There is a person with a gold hat next to the person in the gold hat at the end. Then the line from the second person to the end is length $k$, has a gold hat at one end and a purple hat at the other. Applying the inductive hypothesis, there is an adjacent, opposite-hat wearing pair.

In either case we have $P(k + 1)$.

By the principle of induction, we have $P(n)$ for all $n \geq 2$
Part 3 of the course!
Course Outline

Symbolic Logic (training wheels)
- Just make arguments in mechanical ways.

Set Theory/Number Theory (bike in your backyard)

Models of computation (biking in your neighborhood)
- Still make and communicate rigorous arguments
- But now with objects you haven’t used before.
  - A first taste of how we can argue rigorously about computers.

Next week: regular expressions and context free grammars – understand these “simpler computers”

Soon: what these simple computers can do

Then: what simple computers can’t do.

Last week: A problem our computers cannot solve.