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 $x - y \in S$

$\{\} - 4.$

$5 \quad 4$

$S = \{\ldots, -1, 0, 1, 2, 3, 4, 5, 6, 7, \ldots\}$

Structural Induction and Regular Expressions

CSE 311 Winter 2024

Lecture 17

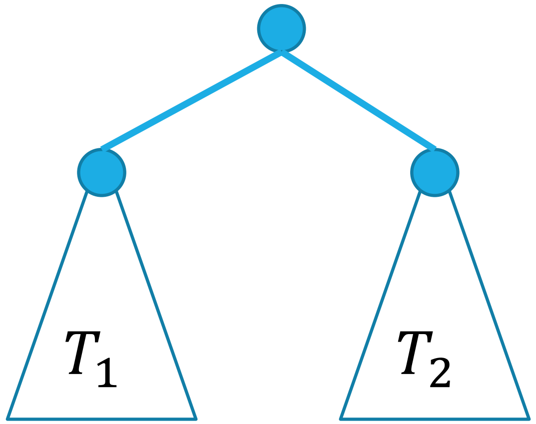
Trees!

More Structural Sets

Binary Trees are another common source of structural induction.

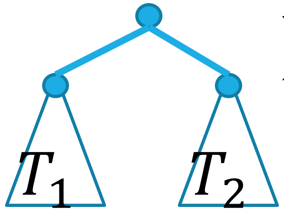
Basis: A single node is a rooted binary tree. 

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.



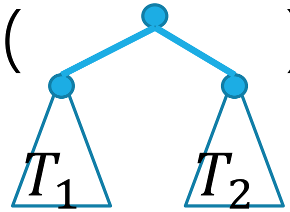
Functions on Binary Trees

size(●)=1

size(

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

height(●) = 0

height(

) = 1+max(height(T_1),height(T_2))

Claim

We want to show that trees of a certain height can't have too many nodes. Specifically our claim is this:

For all trees T, size(T) $\leq 2^{height(T)+1} - 1$

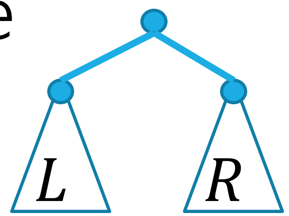
Take a moment to absorb this formula, then we'll do induction!

Structural Induction on Binary Trees

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

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

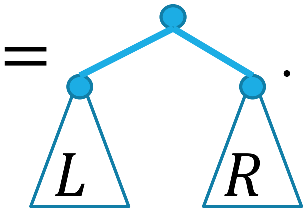
Inductive Hypothesis: Suppose $\mathrm{P}(L)$ and $\mathrm{P}(R)$ hold for arbitrary trees L, R. Let T be the tree



Inductive step: Figure out, (1) what we must show (2) a formula for height and a formula for size of T.

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.

$T =$  .

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

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

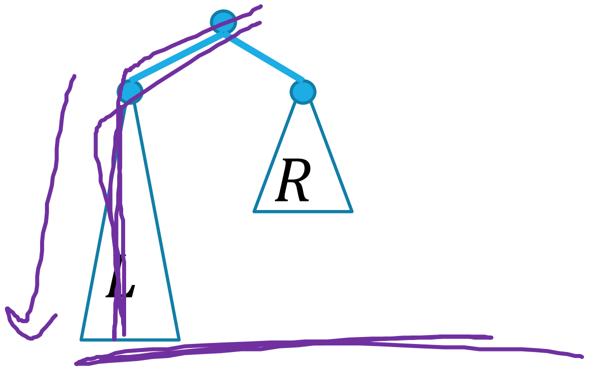
$\text{size}(T) = 1 + \text{size}(L) + \text{size}(R)$

$\leq \cancel{1} + 2^{h(L)+1} \cancel{-1} + 2^{h(R)+1} - 1$

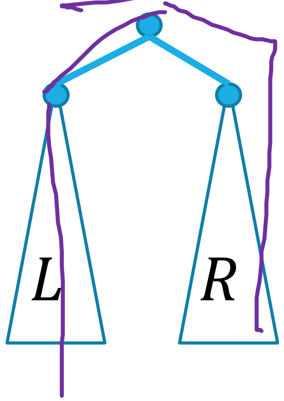
So $P(T)$ holds, and we have $P(T)$ for all binary trees T by the principle of induction.

How do heights compare?

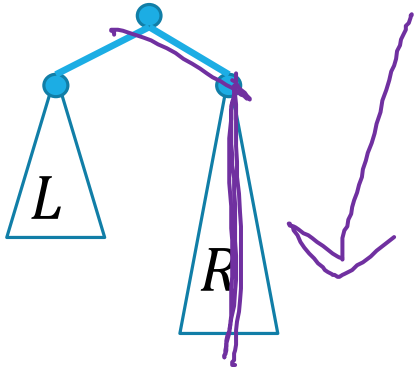
If L is taller than R?



If L, R same height?

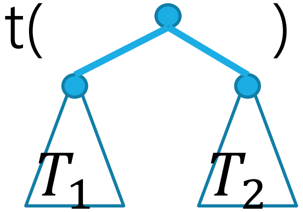


If R is taller than L?



$\text{height}(L) \le \max(h(L), h(R))$

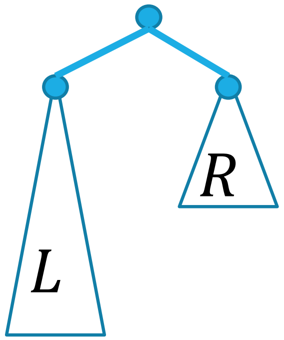
height(●) = 0

height() =

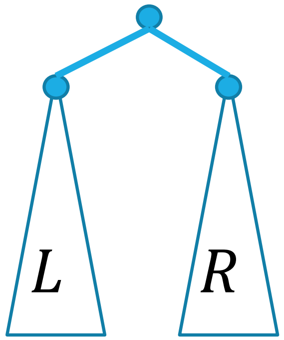
$1+\max(\text{height}(T_1),\text{height}(T_2))$

How do heights compare?

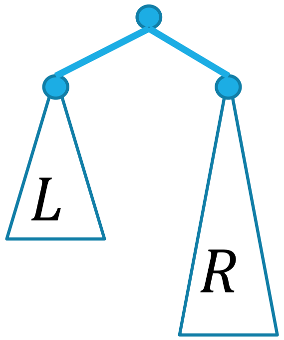
If L is taller than R?



If L, R same height?



If R is taller than L?



height(T) =height(L) + 1
height(T) >height(R) + 1

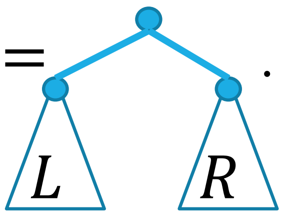
height(T) =height(L) + 1
height(T) =height(R) + 1

height(T) >height(L) + 1
height(T) =height(R) + 1

In all cases: height(T)≥height(L)+1, height(T)≥height(R)+1

Structural Induction on Binary Trees (cont.)

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

$T =$ .

$P(L)$: size$(L) \le 2^{height(L)+1} - 1$

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

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

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

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

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

So $P(T)$ holds, and we have $P(T)$ for all binary trees T by the principle of induction.

Structural Induction Template

1. Define $P()$ State that you will show $P(x)$ holds for all $x \in S$ and that your proof is by structural induction.

2. Base Case: Show $P(b)$
[Do that for every b in the basis step of defining S]

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

4. Inductive Step: Show $P()$ holds for the "new elements."
[You will need a separate step for every element created by the recursive rules].

5. Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.

Structural Induction on Strings

Strings

$\Sigma \to$ chars

ε is "the empty string"

The string with 0 characters – `""  in Java` (not `null!`)

Σ^*:

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 $\cdot$ to mean "concatenate" i.e. + in Java)

Functions on Strings

Since strings are defined recursively, most functions on strings are as well.

Length:

len(ε)=0;

len(wa)=len(w)+1 for $w \in \Sigma^*$, $a \in \Sigma$

Reversal:

$\varepsilon^R = \varepsilon$;
$(wa)^R = aw^R$ for $w \in \Sigma^*$, $a \in \Sigma$

Concatenation

$x \cdot \varepsilon = x$ for all $x \in \Sigma^*$;
$x \cdot (wa) = (x \cdot w)a$ for $w \in \Sigma^*, a \in \Sigma$

Number of c's in a string

$\#_c(\varepsilon) = 0$
$\#_c(wc) = \#_c(w) + 1$ for $w \in \Sigma^*$;
$\#_c(wa) = \#_c(w)$ for $w \in \Sigma^*, a \in \Sigma \setminus \{c\}$.

Claim ~~for all~~ $\forall x \forall y$ $x, y \in \Sigma^*$ len(x·y)=len(x) + len(y).

Let $P(y)$ be "for all $x \in \Sigma^*$ len(x·y)=len(x) + len(y). "

"ab"

concat

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

$P(x,y) = \text{len}(x \ldots)$

That means we'll have to introduce an arbitrary x as part of the base case and the inductive step!

Claim for all $x, y \in \Sigma^*$ len(x·y)=len(x) + len(y).

$P(\varepsilon)$

Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

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

Base Case:

Inductive Hypothesis

Inductive Step:

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

$P(\varepsilon)$ $\forall x \in \Sigma^*$ $(\text{len}(x \cdot \varepsilon) = \text{len}(x) + \text{len}(\varepsilon))$

Claim for all $x, y \in \Sigma^*$ len(x·y)=len(x) + len(y).

Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

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

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

Inductive Hypothesis: Suppose $P(w)$ for an arbitrary string w.

Inductive Step:

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

Claim for all $x, y \in \Sigma^*$ len(x·y)=len(x) + len(y).

Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

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

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

Inductive Hypothesis: Suppose $P(w)$ for an arbitrary string w.

Inductive Step: Let $y = wa$ for an arbitrary $a \in \Sigma$. We show $P(y)$. Let x be an arbitrary string.

...

Therefore, len(xy)=len(x) + len(y), as required.

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

Claim for all $x, y \in \Sigma^*$ len(x·y)=len(x) + len(y).

Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

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

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

Inductive Hypothesis: Suppose $P(w)$ for an arbitrary string w.

Inductive Step: Let $y = wa$ for an arbitrary $a \in \Sigma$. We show $P(y)$. Let x be an arbitrary string.

len(xy)=len(xwa) =len(xw)+1 (by definition of len)

=len(x) + len(w) + 1 (by IH)

=len(x) + len(wa) (by definition of len)

Therefore, len(xy)=len(x) + len(y), as required.

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

Why all those arbitraries?

Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

$P(\varepsilon)$ is a for-all statement, introduce arbitrary variable to show for-all.

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

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

Needs to be arbitrary because it's in the IH (induction wouldn't show "all strings" otherwise)

Inductive Hypothesis: Suppose $P(w)$ for an arbitrary string w.

Inductive Step: Let $y = wa$ for an arbitrary $a \in \Sigma$. We show $P(y)$. Let x be an arbitrary string.

Recursive rule says "every $a \in \Sigma$" so we need to argue for every a.

len(xy)=len(xwa) =len(xw)+1 (by definition of len)

=len(x) + len(w) + 1 (by IH)

=len(x) + len(wa) (by definition of len)

$P(y)$ is a for-all statement, introduce arbitrary variable to show for-all.

Therefore, len(xy)=len(x) + len(y), as required.

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

A few last comments

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: $\mathrm{t} = x + y$

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: $\mathrm{t} = x - y$

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

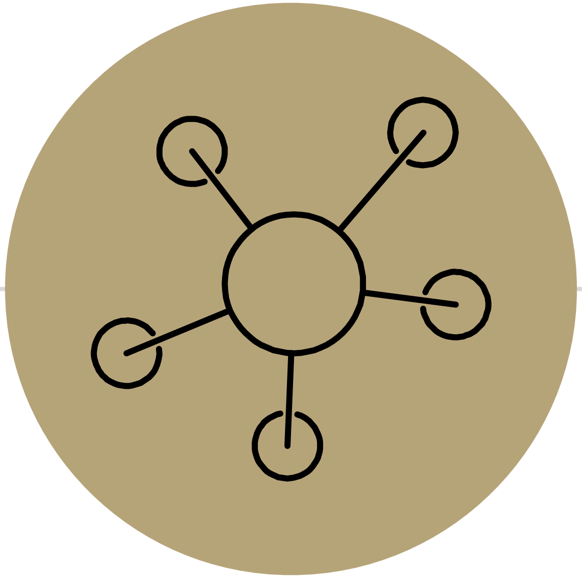
You won't do structural induction.

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.



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.

First up: regular expressions, context free grammars, automata – 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.

The definitions for Friday

Regular Expressions

I have a giant text document. And I want to find all the email addresses inside. What does an email address look like?

[some letters and numbers] @ [more letters] . [com, net, or edu]

We want to ctrl-f for a **pattern of strings** rather than a single string

Languages

A set of strings is called a **language.**

Σ^* is a language

“the set of all binary strings of even length” is a language.

“the set of all palindromes” is a language.

“the set of all English words” is a language.

“the set of all strings matching a given **pattern**” is a language.

Regular Expressions

Basis:

ε is a regular expression. The empty string itself matches the pattern (and nothing else does).

$\emptyset$ is a regular expression. No strings match this pattern.

a is a regular expression, for any $a \in \Sigma$ (i.e. any character). The character itself matching this pattern.

Recursive

If A, B are regular expressions then $(A \cup B)$ is a regular expression

matched by any string that matches A or that matches B [or both]).

If A, B are regular expressions then AB is a regular expression.

matched by any string x such that $x = yz$, y matches A and z matches B.

If A is a regular expression, then A^* is a regular expression.

matched by any string that can be divided into 0 or more strings that match A.

Regular Expressions

$(a \cup bc)$

$0(0 \cup 1)1$

0^*

$(0 \cup 1)^*$

Extra Practice

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 $n \geq 2$