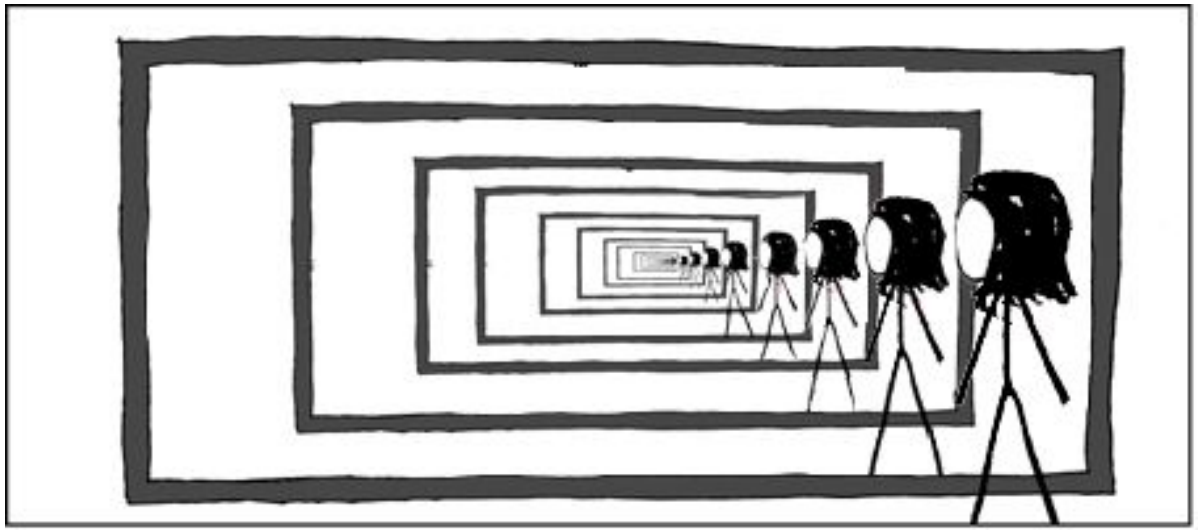
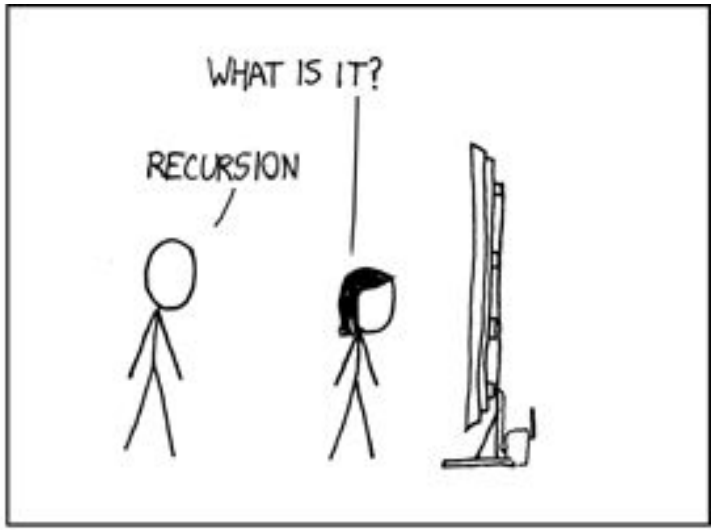




Structural Induction

CSE 311: Foundations of Computing I
Lecture 16

Structural Induction Template

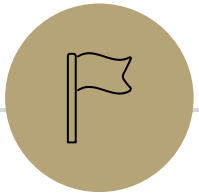
1. Define $\mathrm{P}()$. Claim that $\mathrm{P}(s)$ holds for all $s \in S$. State your proof is by structural induction.

2. Base Case: Show $\mathrm{P}(b_1), \ldots, \mathrm{P}(b_n)$ holds for each basis step $b_1, \ldots, b_n$ in S.

3. Inductive Hypothesis: Suppose $\mathrm{P}(x_1), \ldots, \mathrm{P}(x_m)$ for all values listed in the recursive rules.

4. Inductive Step: Show $\mathrm{P}()$ holds for the "new element" given by the recursive step. **You will need a separate step for every rule.**

5. Conclusion: Conclude that $\mathrm{P}(s)$ holds for all $s \in S$ by structural induction.



Structural Induction

On Strings

String Terminology

Σ is the alphabet, i.e. the set of all letters you can use in strings.

For example: $\Sigma = \{0,1\}$ or $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \ldots, \mathrm{z}, \_\}$

Σ^* is the set of all strings you can build from the letters in the alphabet.

For example: If $\Sigma = \{0,1\}$ then $01001 \in \Sigma^*$. If $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \ldots, \mathrm{z}, \_\}$, then $\mathrm{i\_love\_induction} \in \Sigma^*$

ε is the empty string

Analogous to “” in Java

Recursive definition of Strings

Σ is the alphabet
Σ^* is the set of all strings
ε is the empty string

The set of all strings Σ^* can be defined recursively (using Σ, ε):

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

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

wa here means the string w with the character a appended on to it

Proof

$\text{len}(\varepsilon) = 0$
$\text{len}(wa) = \text{len}(w) + 1$

$\varepsilon^R = \varepsilon$
$(wa)^R = aw^R$

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

1. Let $P(s)$ be $\text{len}(s^R) = \text{len}(s)$. We prove $P(s)$ for all strings s by structural induction.

2. Base Case(s): $(s = \varepsilon)$. LHS: Since $\varepsilon^R = \varepsilon$, $\text{len}(\varepsilon^R) = \text{len}(\varepsilon) = 0$. RHS: $\text{len}(\varepsilon) = 0$. Since $0 = 0$, the base case holds.

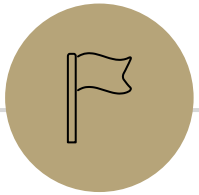
3. Inductive Hypothesis: Suppose $P(w)$ for some arbitrary string w. Then $\text{len}(w^R) = \text{len}(w)$

4. Inductive Step: **Goal:** $\text{len}\big((wa)^R\big) = \text{len}(wa)$

Let a be an arbitrary character. Observe:

$$\begin{aligned}
\text{len}\big((wa)^R\big) &= \text{len}(aw^R) && \text{By definition of reverse}\\
&= \text{len}(w^R) + 1 && \text{By definition of length}\\
&= \text{len}(w) + 1 && \text{By IH}\\
&= \text{len}(wa) && \text{By definition of length}
\end{aligned}$$

5. Conclusion: Thus $\text{P}(s)$ holds for all strings s by structural induction.



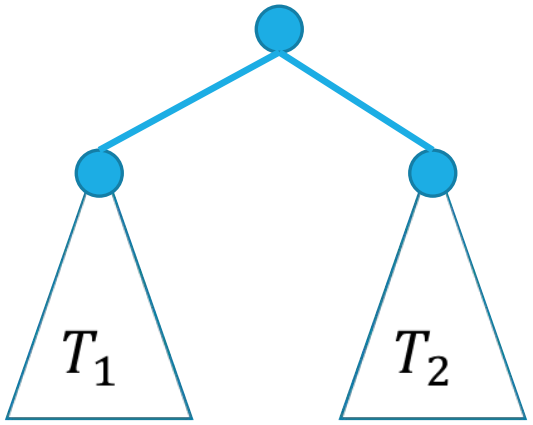
Trees!

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.

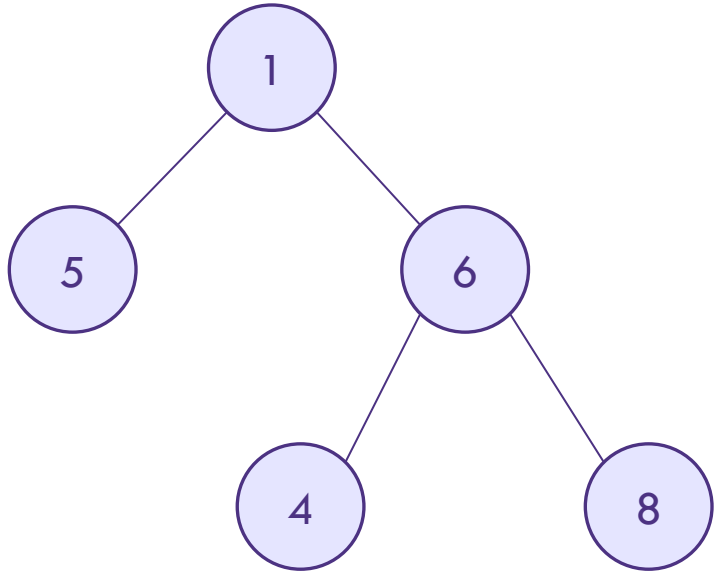


Exercise

Basis: $\text{null} \in \text{Tree}$

Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Write out the following tree using our notation for trees.

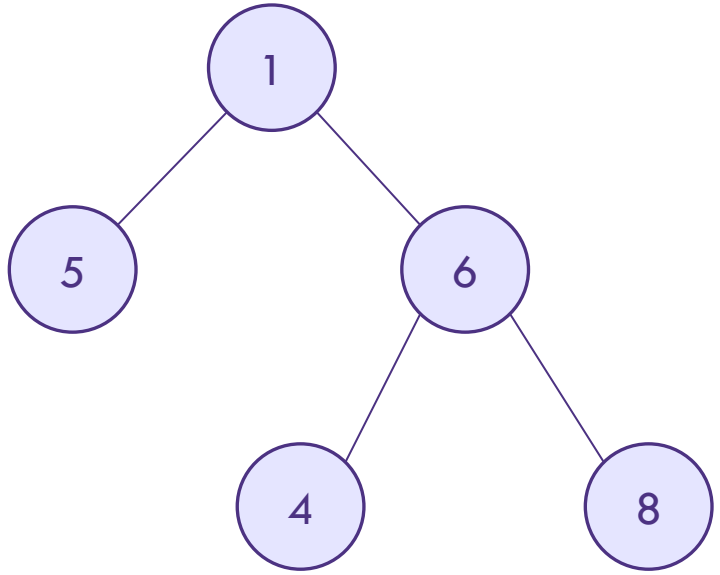


Exercise

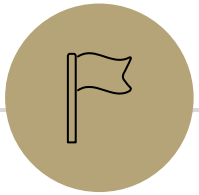
Basis: $\text{null} \in \text{Tree}$

Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Write out the following tree using our notation for trees.



$$\Big((\text{null}, 5, \text{null}), 1, \big((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})\big)\Big)$$



Claim 1

Functions on Binary Trees

Basis: $\text{null} \in \text{Tree}$

Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

To prove interesting facts about trees, we need functions on trees.

Size:

$$\text{size}(\text{null}) = 0$$

$$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$$

Height:

$$\text{height}(\text{null}) = -1$$

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

Functions on Binary Trees

$$\text{size}(\text{null}) = 0$$
$$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$$

What's the size the tree $((\text{null}, 5, \text{null}), 1, ((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})))$?

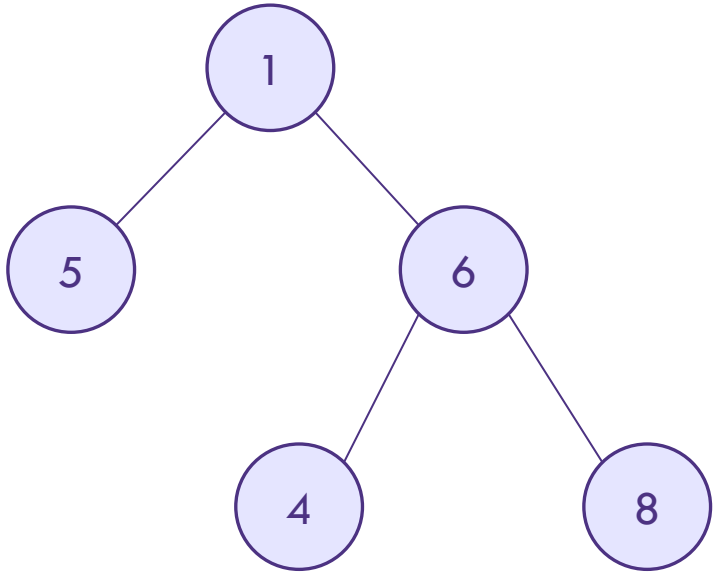
$$\text{size}\Big((\text{null}, 5, \text{null}), 1, \big((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})\big)\Big)$$

$$= 1 + \text{size}\big((\text{null}, 5, \text{null})\big) + \text{size}\big((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})\big)$$

$$= 1 + 1 + \text{size}(\text{null}) + \text{size}(\text{null}) + 1 + \text{size}(\text{null}, 4, \text{null}) + \text{size}(\text{null}, 8, \text{null})$$

$$= 3 + 1 + \text{size}(\text{null}) + \text{size}(\text{null}) + 1 + \text{size}(\text{null}) + \text{size}(\text{null})$$

$$= 5$$



Functions on Binary Trees

$$\text{height}(\text{null}) = -1$$
$$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$$

What's the height of the tree $((\text{null}, 5, \text{null}), 1, ((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})))$?

$$\text{height}\Big((\text{null}, 5, \text{null}), 1, \big((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})\big)\Big)$$

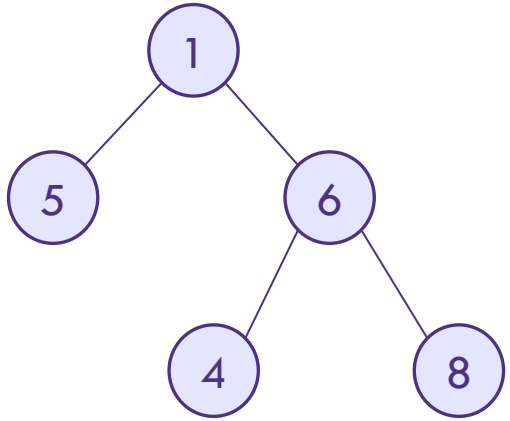
$$= 1 + \max\Big(\text{height}(\text{null}, 5, \text{null}), \text{height}\big((\text{null}, 4, \text{null}), 6, (\text{null}, 8, \text{null})\big)\Big)$$

$$= 1 + \max\Big(1 + \max\big(\text{height}(\text{null}), \text{height}(\text{null})\big), 1 + \max\big(\text{height}(\text{null}, 4, \text{null}), \text{height}(\text{null}, 8, \text{null})\big)\Big)$$

$$= 1 + \max\Big(1 - 1, 1 + \max\big(1 + \max\big(\text{height}(\text{null}), \text{height}(\text{null})\big), 1 + \max\big(\text{height}(\text{null}), \text{height}(\text{null})\big)\big)\Big)$$

$$= 1 + \max\big(0, 1 + \max(1 - 1, 1 - 1)\big)$$

$$= 1 + \max\big(0, 1 + \max(0, 0)\big) = 1 + \max(0, 1 + 0) = 1 + 1 = 2$$

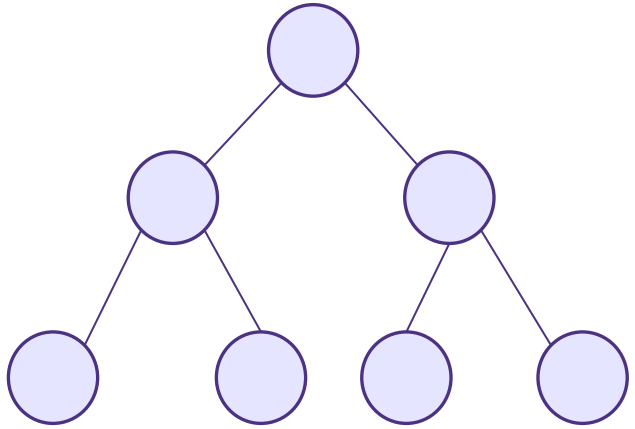


Claim 1

$$\text{height}(\text{null}) = -1$$
$$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$$

$$\text{size}(\text{null}) = 0$$
$$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$$

Claim 1: For every binary tree, $\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$.



$\text{height}(T) = 2$
$\text{size}(T) = 7$
$7 \leq 7 = 8 - 1 = 2^3 - 1 = 2^{2+1} - 1$

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be

We show $\text{P}(T)$ for all binary trees T by structural induction.

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

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

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $\text{P}(T)$ for all binary trees T by structural induction.
2. Base Case: Consider the null tree. Then $\text{height}(\text{null}) = -1$ and $\text{size}(\text{null}) = 0$. Since $0 \leq 0 = 1 - 1 = 2^0 - 1 = 2^{-1+1} - 1$, the base case holds.

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $\text{P}(T)$ for all binary trees T by structural induction.
2. Base Case: Consider the null tree. Then $\text{height}(\text{null}) = -1$ and $\text{size}(\text{null}) = 0$. Since $0 \leq 0 = 1 - 1 = 2^0 - 1 = 2^{-1+1} - 1$, the base case holds.
3. IH: Suppose $\text{P}(L)$ and $\text{P}(R)$ hold for arbitrary binary trees L, R. Then $\text{size}(L) \leq 2^{\text{height}(L)+1} - 1$ and $\text{size}(R) \leq 2^{\text{height}(R)+1} - 1$.
4. IS: Let $a \in \mathbb{Z}$ be arbitrary. Consider the binary tree (L, a, R):

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $\text{P}(T)$ for all binary trees T by structural induction.
2. Base Case: Consider the null tree. Then $\text{height}(\text{null}) = -1$ and $\text{size}(\text{null}) = 0$. Since $0 \leq 0 = 1 - 1 = 2^0 - 1 = 2^{-1+1} - 1$, the base case holds.
3. IH: Suppose $\text{P}(L)$ and $\text{P}(R)$ hold for arbitrary binary trees L, R. Then $\text{size}(L) \leq 2^{\text{height}(L)+1} - 1$ and $\text{size}(R) \leq 2^{\text{height}(R)+1} - 1$.
4. IS: Let $a \in \mathbb{Z}$ be arbitrary. Consider the binary tree (L, a, R):
   $\text{size}((L, a, R)) = 1 + \text{size}(L) + \text{size}(R)$ Definition of Size

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $\text{P}(T)$ for all binary trees T by structural induction.
2. Base Case: Consider the null tree. Then $\text{height}(\text{null}) = -1$ and $\text{size}(\text{null}) = 0$. Since $0 \leq 0 = 1 - 1 = 2^0 - 1 = 2^{-1+1} - 1$, the base case holds.
3. IH: Suppose $\text{P}(L)$ and $\text{P}(R)$ hold for arbitrary binary trees L, R. Then $\text{size}(L) \leq 2^{\text{height}(L)+1} - 1$ and $\text{size}(R) \leq 2^{\text{height}(R)+1} - 1$.
4. IS: Let $a \in \mathbb{Z}$ be arbitrary. Consider the binary tree (L, a, R):

$$\begin{aligned} \text{size}((L, a, R)) &= 1 + \text{size}(L) + \text{size}(R) && \text{Definition of Size} \\ &\leq 1 + 2^{\text{height}(L)+1} - 1 + 2^{\text{height}(R)+1} - 1 && \text{By the IH} \\ &= 2^{\text{height}(L)+1} + 2^{\text{height}(R)+1} - 1 && \text{Algebra} \end{aligned}$$

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $\text{P}(T)$ for all binary trees T by structural induction.
2. Base Case: Consider the null tree. Then $\text{height}(\text{null}) = -1$ and $\text{size}(\text{null}) = 0$. Since $0 \leq 0 = 1 - 1 = 2^0 - 1 = 2^{-1+1} - 1$, the base case holds.
3. IH: Suppose $\text{P}(L)$ and $\text{P}(R)$ hold for arbitrary binary trees L, R. Then $\text{size}(L) \leq 2^{\text{height}(L)+1} - 1$ and $\text{size}(R) \leq 2^{\text{height}(R)+1} - 1$.
4. IS: Let $a \in \mathbb{Z}$ be arbitrary. Consider the binary tree (L, a, R):

$$\begin{aligned}
\text{size}((L, a, R)) &= 1 + \text{size}(L) + \text{size}(R) && \text{Definition of Size} \\
&\leq 1 + 2^{\text{height}(L)+1} - 1 + 2^{\text{height}(R)+1} - 1 && \text{By the IH} \\
&= 2^{\text{height}(L)+1} + 2^{\text{height}(R)+1} - 1 && \text{Algebra} \\
&\leq 2^{\max(\text{height}(L),\text{height}(R))+1} + 2^{\max(\text{height}(L),\text{height}(R))+1} - 1 && \text{Property of max} \\
&= 2 \cdot 2^{\max(\text{height}(L),\text{height}(R))+1} - 1 && \text{Algebra}
\end{aligned}$$

$\text{size}(\text{null}) = 0$
$\text{size}\big((L, a, R)\big) = 1 + \text{size}(L) + \text{size}(R)$

$\text{height}(\text{null}) = -1$
$\text{height}\big((L, a, R)\big) = 1 + \max\big(\text{height}(L), \text{height}(R)\big)$

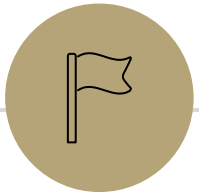
Basis: $\text{null} \in \text{Tree}$
Recursive: If $L, R \in \text{Tree}$, and $a \in \mathbb{Z}$ then $(L, a, R) \in \text{Tree}$

Claim 1 Proof

1. Let $\text{P}(T)$ be "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". We show $\text{P}(T)$ for all binary trees T by structural induction.
2. Base Case: Consider the null tree. Then $\text{height}(\text{null}) = -1$ and $\text{size}(\text{null}) = 0$. Since $0 \leq 0 = 1 - 1 = 2^0 - 1 = 2^{-1+1} - 1$, the base case holds.
3. IH: Suppose $\text{P}(L)$ and $\text{P}(R)$ hold for arbitrary binary trees L, R. Then $\text{size}(L) \leq 2^{\text{height}(L)+1} - 1$ and $\text{size}(R) \leq 2^{\text{height}(R)+1} - 1$.
4. IS: Let $a \in \mathbb{Z}$ be arbitrary. Consider the binary tree (L, a, R):

$$
\begin{aligned}
\text{size}((L, a, R)) &= 1 + \text{size}(L) + \text{size}(R) && \text{Definition of Size} \\
&\leq 1 + 2^{\text{height}(L)+1} - 1 + 2^{\text{height}(R)+1} - 1 && \text{By the IH} \\
&= 2^{\text{height}(L)+1} + 2^{\text{height}(R)+1} - 1 && \text{Algebra} \\
&\leq 2^{\max(\text{height}(L),\text{height}(R))+1} + 2^{\max(\text{height}(L),\text{height}(R))+1} - 1 && \text{Property of max} \\
&= 2 \cdot 2^{\max(\text{height}(L),\text{height}(R))+1} - 1 && \text{Algebra} \\
&= 2 \cdot 2^{\text{height}((L,a,R))} - 1 && \text{Definition of Height} \\
&= 2^{\text{height}((L,a,R))+1} - 1 && \text{Algebra}
\end{aligned}
$$

5. Thus $\text{P}(T)$ holds for all trees T by structural induction.



Claim 2

Functions on Lists

Basis: $[\,] \in \text{List}$

Recursive: If $\text{L} \in \text{List}$ and $a \in \mathbb{Z}$ then $a :: \text{L} \in \text{List}$

To prove interesting facts about lists, we need functions on lists.

Length:

$$\text{len}([\,]) = 0$$

$$\text{len}(a :: \text{L}) = 1 + \text{len}(\text{L})$$

Concatenation:

$$\text{concat}([\,], \text{R}) = \text{R}$$

$$\text{concat}(a :: \text{L}, \text{R}) = a :: \text{concat}(\text{L}, \text{R})$$

Claim 2

$$\text{len}([\,]) = 0$$
$$\text{len}(a :: \text{L}) = 1 + \text{len}(\text{L})$$

$$\text{concat}([\,], \text{R}) = \text{R}$$
$$\text{concat}(a :: \text{L}, \text{R}) = a :: \text{concat}(\text{L}, \text{R})$$

Claim 2: For all lists L, R, $\text{len}\big(\text{concat}(\text{L}, \text{R})\big) = \text{len}(\text{L}) + \text{len}(\text{R})$.

How do we prove a nested forall?

Let $\text{P}(\text{L})$ be "$\text{len}\big(\text{concat}(\text{L}, \text{R})\big) = \text{len}(\text{L}) + \text{len}(\text{R})$ for all lists $\text{R} \in \text{List}$".

We prove $\text{P}(\text{L})$ for all lists $\text{L} \in \text{List}$ by structural induction.

Claim 2 Proof

Basis: $[\,] \in \text{List}$
Recursive: If $L \in \text{List}$ and $a \in \mathbb{Z}$ then $a :: L \in \text{List}$

$\text{concat}([\,], R) = R$
$\text{concat}(a :: L, R) = a :: \text{concat}(L, R)$

$\text{len}([\,]) = 0$
$\text{len}(a :: L) = 1 + \text{len}(L)$

1. Let $P(L)$ be " :". We prove $P(L)$ for all lists $L \in \text{List}$ by structural induction.
2. Base Case:
3. IH:
4. IS:
5. Thus $P(L)$ holds for all lists $L \in \text{List}$ by structural induction.

Claim 2 Proof

Basis: $[\,] \in \text{List}$
Recursive: If $L \in \text{List}$ and $a \in \mathbb{Z}$ then $a :: L \in \text{List}$

$\text{concat}([\,], R) = R$
$\text{concat}(a :: L, R) = a :: \text{concat}(L, R)$

$\text{len}([\,]) = 0$
$\text{len}(a :: L) = 1 + \text{len}(L)$

1. Let $P(L)$ be "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ for all lists $R \in \text{List}$". We prove $P(L)$ for all lists $L \in \text{List}$ by structural induction.
2. Base Case:
3. IH:
4. IS:
5. Thus $P(L)$ holds for all lists $L \in \text{List}$ by structural induction.

Claim 2 Proof

Basis: $[\,] \in \text{List}$
Recursive: If $L \in \text{List}$ and $a \in \mathbb{Z}$ then $a :: L \in \text{List}$

$\text{concat}([\,], R) = R$
$\text{concat}(a :: L, R) = a :: \text{concat}(L, R)$

$\text{len}([\,]) = 0$
$\text{len}(a :: L) = 1 + \text{len}(L)$

1. Let $P(L)$ be "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ for all lists $R \in \text{List}$". We prove $P(L)$ for all lists $L \in \text{List}$ by structural induction.
2. **Base Case**: Let $R \in \text{List}$ be arbitrary. Observe that $\text{len}(\text{concat}([\,], R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}([\,]) + \text{len}(R)$, so $P([\,])$ holds.
3. IH:
4. IS:
5. Thus $P(L)$ holds for all lists $L \in \text{List}$ by structural induction.

Claim 2 Proof

Basis: $[\,] \in \text{List}$
Recursive: If $L \in \text{List}$ and $a \in \mathbb{Z}$ then $a :: L \in \text{List}$

$\text{concat}([\,], R) = R$
$\text{concat}(a :: L, R) = a :: \text{concat}(L, R)$

$\text{len}([\,]) = 0$
$\text{len}(a :: L) = 1 + \text{len}(L)$

1. Let $P(L)$ be "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ for all lists $R \in \text{List}$". We prove $P(L)$ for all lists $L \in \text{List}$ by structural induction.
2. **Base Case**: Let $R \in \text{List}$ be arbitrary. Observe that $\text{len}(\text{concat}([\,], R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}([\,]) + \text{len}(R)$, so $P([\,])$ holds.
3. IH: Suppose that $P(L)$ holds for some arbitrary $L \in \text{List}$. That is, for all $R \in \text{List}$, $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$.
4. IS:
5. Thus $P(L)$ holds for all lists $L \in \text{List}$ by structural induction.

Claim 2 Proof

Basis: $[\,] \in \text{List}$
Recursive: If $L \in \text{List}$ and $a \in \mathbb{Z}$ then $a :: L \in \text{List}$

$\text{concat}([\,], R) = R$
$\text{concat}(a :: L, R) = a :: \text{concat}(L, R)$

$\text{len}([\,]) = 0$
$\text{len}(a :: L) = 1 + \text{len}(L)$

1. Let $P(L)$ be "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ for all lists $R \in \text{List}$". We prove $P(L)$ for all lists $L \in \text{List}$ by structural induction.
2. Base Case: Let $R \in \text{List}$ be arbitrary. Observe that $\text{len}(\text{concat}([\,], R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}([\,]) + \text{len}(R)$, so $P([\,])$ holds.
3. IH: Suppose that $P(L)$ holds for some arbitrary $L \in \text{List}$. That is, for all $R \in \text{List}$, $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$.
4. IS: Let $R \in \text{List}$ be arbitrary. Then observe that:

$$\begin{aligned} \text{len}(\text{concat}(a :: L, R)) &= \text{len}(a :: \text{concat}(L, R)) && \text{Def of concat} \\ &= 1 + \text{len}(\text{concat}(L, R)) && \text{Def of length} \\ &= 1 + \text{len}(L) + \text{len}(R) && \text{IH} \\ &= \text{len}(a :: L) + \text{len}(R) && \text{Def of length} \end{aligned}$$

5. Thus $P(L)$ holds for all lists $L \in \text{List}$ by structural induction.

Runtime of Euclid's Algorithm

Theorem: Suppose that Euclid's Algorithm takes n steps for gcd(a,b) with $a \geq b \geq 0$. Then $a \geq f_{n+2}$ where f_n is the nth Fibonacci number

Exercise: Prove that $f_n \geq 2^{\frac{n}{2}-1}$ for all integers n ≥ 2 by strong induction.

Runtime of Euclid's Algorithm

Theorem: Suppose that Euclid's Algorithm takes n steps for gcd(a,b) with $a \geq b \geq 0$. Then $a \geq f_{n+2}$ where f_n is the nth Fibonacci number

Why does this help us bound the running time of Euclid's Algorithm?

Left as an exercise to prove that $f_n \geq 2^{\frac{n}{2}-1}$, so $f_{n+1} \geq 2^{\frac{n-1}{2}}$

Therefore: if Euclid's Algorithm takes n steps, then $a \geq f_{n+2}$,

so $\frac{n-1}{2} \leq \log_2 a$ or $n \leq 1 + 2\log_2 a$

i.e # of steps $\leq 1 +$ twice the # of bits in a

Runtime of Euclid's Algorithm

Theorem: Suppose that Euclid's Algorithm takes n steps for gcd(a,b) with $a \geq b \geq 0$. Then $a \geq f_{n+2}$ where f_n is the nth Fibonacci number

Intuition: Consider an n step gcd calculation starting with r_{n+1}=a and r_n=b:

$r_{n+1} = q_n r_n + r_{n-1}$

$r_n = q_{n-1} r_{n-1} + r_{n-2}$

...

$r_3 = q_2 r_2 + r_1$

$r_2 = q_1 r_1$

For all k ≥ 2, $r_{k-1} = r_{k+1} \bmod r_k$

Now $r_1 \geq 1$ and each q_k must be ≥ 1. If we replace all the q_K's by 1 and replace r_1 by 1 , we can only reduce the r_k's. After that reduction, $r_k = f_k$ for every k.

Theorem: **Suppose that Euclid's Algorithm takes n steps for $\gcd(a, b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.**

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Base Case:

/e

Theorem: **Suppose that Euclid's Algorithm takes n steps for $\gcd(a, b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.**

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Base Case:

n=1 Suppose Euclid's Algorithm with $a \geq b > 0$ takes 1 step. By assumption, we have that $a \geq b > 1 = f_2$ thus P(1) holds.

Theorem: Suppose that Euclid's Algorithm takes n steps for $\gcd(a,b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Base Case:

n=1 Suppose Euclid's Algorithm with $a \geq b > 0$ takes 1 step. By assumption, we have that $a \geq b > 1 = f_2$ thus P(1) holds.

n = 2 If the Euclid's algorithm takes 2 steps, then we have $a = q_2 b + r_1$ and $b = q_1 r_1$ and $r_1 > 0$. Since $a \geq b > 0$ we must have $q_2 \geq 1$ and b ≥ 1. So $a = q_2 b + r_1 \geq b + r_1 \geq 2 = f_3$ thus P(2) holds.

Theorem: Suppose that Euclid's Algorithm takes n steps for $\gcd(a,b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Base Case:

n=1 Suppose Euclid's Algorithm with $a \geq b > 0$ takes 1 step. By assumption, we have that $a \geq b > 1 = f_2$ thus P(1) holds.

n = 2 If the Euclid's algorithm takes 2 steps, then we have $a = q_2 b + r_1$ and $b = q_1 r_1$ and $r_1 > 0$. Since $a \geq b > 0$ we must have $q_2 \geq 1$ and b ≥ 1. So $a = q_2 b + r_1 \geq b + r_1 \geq 2 = f_3$ thus P(2) holds.

Inductive Hypothesis: Suppose that $P(1) \wedge \cdots \wedge P(k)$ hold for some arbitrary $k \geq 2$

Inductive step: Suppose that gcd(a,b) with $a \geq b > 0$ takes k+1 steps.

Theorem: Suppose that Euclid's Algorithm takes n steps for $\gcd(a,b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Inductive Hypothesis: Suppose that $P(1) \wedge \cdots \wedge P(k)$ hold for some arbitrary $k \geq 2$

Inductive step:

Suppose that gcd(a,b) with $a \geq b > 0$ takes k+1 steps.

Theorem: **Suppose that Euclid's Algorithm takes n steps for $\gcd(a, b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.**

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Inductive Hypothesis: Suppose that $P(1) \wedge \cdots \wedge P(k)$ hold for some arbitrary $k \geq 2$

Inductive step:

Suppose that gcd(a,b) with $a \geq b > 0$ takes k+1 steps. We know that $k + 1 \geq 3$ so for the first 3 steps, we have:

$$a = q_{k+1}b + r_k$$
$$b = q_k r_k + r_{k-1}$$
$$r_k = q_{k-1} r_{k-1} + r_{k-2}$$

Theorem: Suppose that Euclid's Algorithm takes n steps for $\gcd(a,b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Inductive Hypothesis: Suppose that $P(1) \wedge \cdots \wedge P(k)$ hold for some arbitrary $k \geq 2$

Inductive step:

Suppose that gcd(a,b) with $a \geq b > 0$ takes k+1 steps. We know that $k + 1 \geq 3$ so for the first 3 steps, we have:

$$a = q_{k+1}b + r_k$$
$$b = q_k r_k + r_{k-1}$$
$$r_k = q_{k-1}\, r_{k-1} + r_{k-2}$$

And there are k-2 more steps after this. This means that the $\gcd(r_k r_{k-1})$ takes k-1 steps. Since $k, k-1 \geq 1$, by the IH we have that $b \geq f_{k-1}$ and $r_k \geq f_k$.

Theorem: **Suppose that Euclid's Algorithm takes n steps for $\gcd(a,b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.**

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Inductive Hypothesis: Suppose that $P(1) \wedge \cdots \wedge P(k)$ hold for some arbitrary $k \geq 2$

Inductive step:

Suppose that gcd(a,b) with $a \geq b > 0$ takes k+1 steps. We know that $k + 1 \geq 3$ so for the first 3 steps, we have:

$$a = q_{k+1}b + r_k$$
$$b = q_k r_k + r_{k-1}$$
$$r_k = q_{k-1}\, r_{k-1} + r_{k-2}$$

And there are k-2 more steps after this. This means that the $\gcd(r_k r_{k-1})$ takes k-1 steps. Since $k, k-1 \geq 1$, by the IH we have that $b \geq f_{k-1}$ and $r_k \geq f_k$.

Since $a \geq b$, we must have that $q_{k+1} \geq 1$.

Theorem: Suppose that Euclid's Algorithm takes n steps for $\gcd(a,b)$ with $a \geq b > 0$. Then, $a \geq f_{n+1}$.

Let P(n) be "gcd(a,b) with a ≥ b>0 takes n steps → a ≥ f_{n+1}". We will prove this for all n ≥ 1 by strong induction.

Inductive Hypothesis: Suppose that $P(1) \wedge \cdots \wedge P(k)$ hold for some arbitrary $k \geq 2$

Inductive step:

Suppose that gcd(a,b) with $a \geq b > 0$ takes k+1 steps. We know that $k + 1 \geq 3$ so for the first 3 steps, we have:

$$a = q_{k+1}b + r_k$$
$$b = q_k r_k + r_{k-1}$$
$$r_k = q_{k-1}\, r_{k-1} + r_{k-2}$$

And there are k-2 more steps after this. This means that the $\gcd(r_k r_{k-1})$ takes k-1 steps. Since $k, k-1 \geq 1$, by the IH we have that $b \geq f_{k-1}$ and $r_k \geq f_k$.

Since $a \geq b$, we must have that $q_{k+1} \geq 1$.

Thus $a = q_{k+1} + r_k \geq b + r_k \geq f_{k+1} + f_k = f_{k+2}$