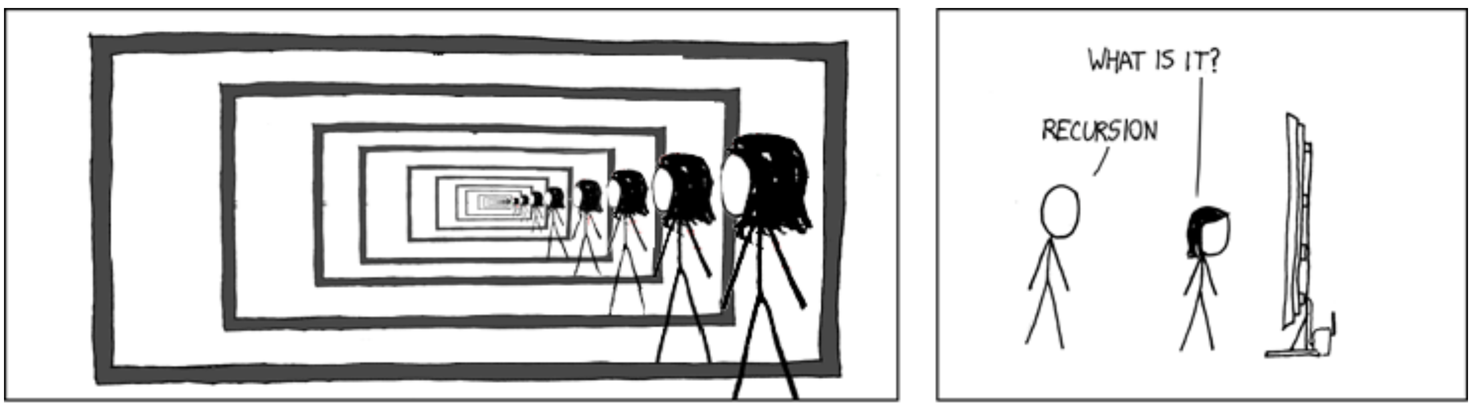
CSE 311: Foundations of Computing

Topic 8: Recursive Data & Functions



Recursive definitions of functions

- $0! = 1$; $(n+1)! = (n+1) \cdot n!$ **for all** $n \geq 0$**.**
- $F(0) = 0$; $F(n+1) = F(n) + 1$ **for all** $n \geq 0$**.**
- $G(0) = 1$; $G(n+1) = 2 \cdot G(n)$ **for all** $n \geq 0$**.**
- $H(0) = 1$; $H(n+1) = 2^{H(n)}$ **for all** $n \geq 0$**.**

Prove $n! \leq n^n$ for all $n \geq 1$

1. Let P(n) be "$n! \leq n^n$". We will show that P(n) is true for all integers $n \geq 1$ by induction.
2. Base Case (n=1): $1! = 1 \cdot 0! = 1 \cdot 1 = 1 = 1^1$ so P(1) is true.
3. Inductive Hypothesis: Suppose that P(k) is true for some arbitrary integer $k \geq 1$. I.e., suppose $k! \leq k^k$.

Prove $n! \leq n^n$ for all $n \geq 1$

1. Let $P(n)$ be "$n! \leq n^n$". We will show that $P(n)$ is true for all integers $n \geq 1$ by induction.
2. Base Case $(n=1)$: $1!=1\cdot 0!=1\cdot 1=1=1^1$ so $P(1)$ is true.
3. Inductive Hypothesis: Suppose that $P(k)$ is true for some arbitrary integer $k \geq 1$. I.e., suppose $k! \leq k^k$.
4. Inductive Step:

   Goal: Show $P(k+1)$, i.e. show $(k+1)! \leq (k+1)^{k+1}$

   $$\begin{aligned} (k+1)! &= (k+1)\cdot k! && \text{by definition of !} \\ &\leq (k+1)\cdot k^k && \text{by the IH} \\ &\leq (k+1)\cdot (k+1)^k && \text{since } k \geq 0 \\ &= (k+1)^{k+1} \end{aligned}$$

   Therefore $P(k+1)$ is true.
5. Thus $P(n)$ is true for all $n \geq 1$, by induction.

More Recursive Definitions

Suppose that $h: \mathbb{N} \to \mathbb{R}$.

Then we have familiar summation notation:

$\sum_{i=0}^{0} h(i) = h(0)$

$\sum_{i=0}^{n+1} h(i) = h(n+1) + \sum_{i=0}^{n} h(i)$ **for** $n \geq 0$

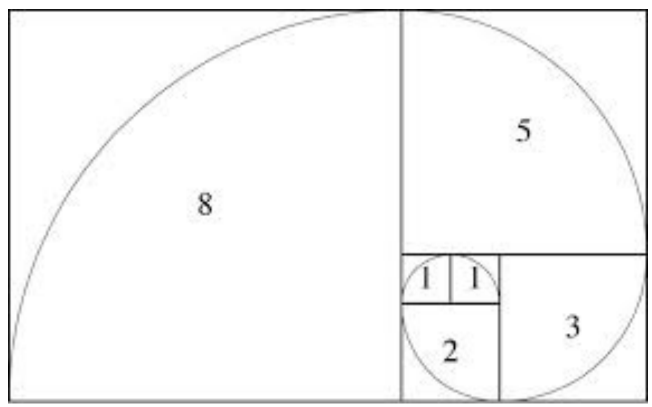
There is also product notation:

$\prod_{i=0}^{0} h(i) = h(0)$

$\prod_{i=0}^{n+1} h(i) = h(n+1) \cdot \prod_{i=0}^{n} h(i)$ **for** $n \geq 0$

Fibonacci Numbers

$$f_0 = 0$$
$$f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \quad \textbf{for all } n \geq 2$$



Fibonacci Numbers

$$f_0 = 0$$
$$f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \quad \textbf{for all } n \geq 2$$

Tamás Görbe
@TamasGorbe

A Mathematician's Way* of Converting Miles to Kilometers

$$\begin{aligned} 3 \text{ mi} &\approx 5 \text{ km} \\ 5 \text{ mi} &\approx 8 \text{ km} \\ 8 \text{ mi} &\approx 13 \text{ km} \end{aligned}$$

$$f_n \text{ mi} \approx f_{n+1} \text{ km}$$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.

$f_0 = 0 \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. **Let** P(n) **be "**$f_n < 2^n$**". We prove that** P(n) **is true for all integers** $n \geq 0$ **by strong induction.**
2. **Base Case:** $f_0=0 < 1= 2^0$ **so** P(0) **is true.**

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$
$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ for all $\boldsymbol{n \geq 2}$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.
2. Base Case: $f_0 = 0 < 1 = 2^0$ so P(0) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 0$, we have $f_j < 2^j$ for every integer j from 0 to k.

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$

$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ for all $\boldsymbol{n \geq 2}$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.
2. Base Case: $f_0=0 < 1= 2^0$ so P(0) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 0$, we have $f_j < 2^j$ for every integer j from 0 to k.
4. Inductive Step: Goal: Show P(k+1); that is, $f_{k+1} < 2^{k+1}$

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$
$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ for all $\boldsymbol{n \geq 2}$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.
2. Base Case: $f_0=0 < 1= 2^0$ so P(0) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 0$, we have $f_j < 2^j$ for every integer j from 0 to k.
4. Inductive Step: Goal: Show P(k+1); that is, $f_{k+1} < 2^{k+1}$

   <u>Case k+1 = 1</u>:

   <u>Case k+1 ≥ 2</u>:

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.
2. Base Case: $f_0=0 < 1= 2^0$ so P(0) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 0$, we have $f_j < 2^j$ for every integer j from 0 to k.
4. Inductive Step: Goal: Show P(k+1); that is, $f_{k+1} < 2^{k+1}$

   Case k+1 = 1: Then $f_1 = 1 < 2 = 2^1$ so P(k+1) is true here.

   Case k+1 ≥ 2:

$f_0 = 0 \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.
2. Base Case: $f_0=0 < 1= 2^0$ so P(0) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 0$, we have $f_j < 2^j$ for every integer j from 0 to k.
4. Inductive Step: $\boxed{\text{Goal: Show } P(k+1)\text{; that is, } f_{k+1} < 2^{k+1}}$

   Case k+1 = 1: Then $f_1 = 1 < 2 = 2^1$ so P(k+1) is true here.

   Case k+1 ≥ 2: Then $f_{k+1} = f_k + f_{k-1}$ by definition

   $< 2^k + 2^{k-1}$ by the IH since $k-1 \geq 0$

   $< 2^k + 2^k = 2 \cdot 2^k$

   $= 2^{k+1}$

   so P(k+1) is true in this case.

   These are the only cases so P(k+1) follows.

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$

$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ for all $\boldsymbol{n \geq 2}$

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. Let P(n) be "$f_n < 2^n$". We prove that P(n) is true for all integers $n \geq 0$ by strong induction.
2. Base Case: $f_0=0 < 1= 2^0$ so P(0) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 0$, we have $f_j < 2^j$ for every integer j from 0 to k.
4. Inductive Step: Goal: Show P(k+1); that is, $f_{k+1} < 2^{k+1}$

   Case k+1 = 1: Then $f_1 = 1 < 2 = 2^1$ so P(k+1) is true here.

   Case k+1 ≥ 2: Then $f_{k+1} = f_k + f_{k-1}$ by definition

   $< 2^k + 2^{k-1}$ by the IH since $k-1 \geq 0$

   $< 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$

   so P(k+1) is true in this case.

   These are the only cases so P(k+1) follows.

5. Therefore by strong induction,

   $f_n < 2^n$ for all integers $n \geq 0$.

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$

$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ for all $\boldsymbol{n \geq 2}$

Inductive Proofs with Multiple Base Cases

1. **"Let $P(n)$ be... . We will show that $P(n)$ is true for all integers $n \geq b$ by induction."**
2. **"Base Cases:" Prove** $\boxed{P(b), P(b+1), \ldots, P(c)}$
3. **"Inductive Hypothesis:**

   Assume $P(k)$ is true for an arbitrary integer $\boxed{k \geq c}$"
4. **"Inductive Step:" Prove that $P(k+1)$ is true:**

   Use the goal to figure out what you need.

   Make sure you are using I.H. and point out where you are using it. (Don't assume $P(k+1)$!!)
5. **"Conclusion: $P(n)$ is true for all integers $n \geq b$"**

Inductive Proofs With Multiple Base Cases

1. "Let $P(n)$ be… . We will show that $P(n)$ is true for all integers $n \geq b$ by *strong* induction."
2. "Base Cases:" Prove $\boxed{P(b), P(b+1), \ldots, P(c)}$
3. "Inductive Hypothesis:

   Assume that for some arbitrary integer $\boxed{k \geq c}$,

   $P(j)$ is true for every integer j from b to k"
4. "Inductive Step:" Prove that $P(k+1)$ is true:

   Use the goal to figure out what you need.

   Make sure you are using I.H. (that $P(b), \ldots, P(k)$ are true) and point out where you are using it.
   (Don't assume $P(k+1)$!!)
5. "Conclusion: $P(n)$ is true for all integers $n \geq b$"

Bounding Fibonacci I: $f_n < 2^n$ for all $n \geq 0$

1. **Let** P(n) **be "**$f_n < 2^n$**". We prove that** P(n) **is true for all integers** $n \geq 0$ **by strong induction.**
2. **Base Cases:** $f_0 = 0 < 1 = 2^0$ **so** P(0) **is true.**
   $f_1 = 1 < 2 = 2^1$ **so** P(1) **is true.**
3. **Inductive Hypothesis: Assume that for some arbitrary integer** $k \geq 1$, **we have** $f_j < 2^j$ **for every integer** j **from** 0 **to** k.
4. **Inductive Step:** **Goal: Show** P(k+1); **that is,** $f_{k+1} < 2^{k+1}$

   We have $f_{k+1} = f_k + f_{k-1}$ **by definition since** $k+1 \geq 2$

   $< 2^k + 2^{k-1}$ **by the IH since** $k-1 \geq 0$

   $< 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$

   so P(k+1) **is true.**
5. **Therefore, by strong induction,** $f_n < 2^n$ **for all integers** $n \geq 0$.

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci II: $f_n \geq 2^{n/2 - 1}$ for all $n \geq 2$

1. Let P(n) be "$f_n \geq 2^{n/2 - 1}$". We prove that P(n) is true for all integers $n \geq 2$ by strong induction.

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci II: $f_n \geq 2^{n/2 - 1}$ for all $n \geq 2$

1. **Let** P(n) **be "**$f_n \geq 2^{n/2 - 1}$**". We prove that** P(n) **is true for all integers** $n \geq 2$ **by strong induction.**
2. **Base Case:** $f_2 = f_1 + f_0 = 1$ and $2^{2/2 - 1} = 2^0 = 1$ **so** P(2) **is true.**

$$f_0 = 0 \quad f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \text{ for all } n \geq 2$$

Bounding Fibonacci II: $f_n \geq 2^{n/2-1}$ for all $n \geq 2$

1. Let P(n) be "$f_n \geq 2^{n/2 -1}$". We prove that P(n) is true for all integers $n \geq 2$ by strong induction.
2. Base Case: $f_2 = f_1 + f_0 = 1$ and $2^{2/2-1} = 2^0 = 1$ so P(2) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 2$, P(j) is true for every integer j from 2 to k.

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$
$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ for all $\boldsymbol{n \geq 2}$

Bounding Fibonacci II: $f_n \geq 2^{n/2 - 1}$ for all $n \geq 2$

1. **Let** P(n) **be "**$f_n \geq 2^{n/2 - 1}$**". We prove that** P(n) **is true for all integers** $n \geq 2$ **by strong induction.**
2. **Base Case:** $f_2 = f_1 + f_0 = 1$ and $2^{2/2-1} = 2^0 = 1$ **so** P(2) **is true.**
3. **Inductive Hypothesis: Assume that for some arbitrary integer** $k \geq 2$, P(j) **is true for every integer** j **from** 2 **to** k.
4. **Inductive Step:** $\boxed{\textbf{Goal: Show } P(k+1)\textbf{; that is, } f_{k+1} \geq 2^{(k+1)/2 - 1}}$

$$f_0 = 0 \quad f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2} \text{ for all } n \geq 2$$

Bounding Fibonacci II: $f_n \geq 2^{n/2\,-\,1}$ for all $n \geq 2$

1. Let P(n) be "$f_n \geq 2^{n/2 - 1}$". We prove that P(n) is true for all integers $n \geq 2$ by strong induction.
2. Base Case: $f_2 = f_1 + f_0 = 1$ and $2^{2/2 - 1} = 2^0 = 1$ so P(2) is true.
3. Inductive Hypothesis: Assume that for some arbitrary integer $k \geq 2$, P(j) is true for every integer j from 2 to k.
4. Inductive Step: Goal: Show P(k+1); that is, $f_{k+1} \geq 2^{(k+1)/2 - 1}$

No need for cases for the definition here:

$f_{k+1} = f_k + f_{k-1}$ since $k+1 \geq 2$

Now just want to apply the IH to get P(k) and P(k-1)

Problem: Though we can get P(k) since $k \geq 2$,

k-1 may only be 1 so we can't conclude P(k-1)

Solution: Separate cases for when k-1=1 (or k+1=3).

$f_0 = 0 \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci II: $f_n \geq 2^{n/2-1}$ for all $n \geq 2$

1. **Let** P(n) **be "**$f_n \geq 2^{n/2-1}$**". We prove that** P(n) **is true for all integers** $n \geq 2$ **by strong induction.**
2. **Base Cases:** $f_2 = f_1 + f_0 = 1$ and $2^{2/2-1} = 2^0 = 1$ **so** P(2) **holds**
   $f_3 = f_2 + f_1 = 2 \geq 2^{1/2} = 2^{3/2-1}$ **so** P(3) **holds**
3. **Inductive Hypothesis: Assume that for some arbitrary integer** $k \geq 3$, P(j) **is true for every integer** j **from** 2 **to** k**.**
4. **Inductive Step:** **Goal: Show** P(k+1)**; that is,** $f_{k+1} \geq 2^{(k+1)/2-1}$

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Bounding Fibonacci II: $f_n \geq 2^{n/2-1}$ for all $n \geq 2$

1. **Let** P(n) **be "**$f_n \geq 2^{n/2-1}$**".** **We prove that** P(n) **is true for all integers** $n \geq 2$ **by strong induction.**
2. **Base Cases:** $f_2 = f_1 + f_0 = 1$ and $2^{2/2-1} = 2^0 = 1$ **so** P(2) **holds**
   $f_3 = f_2 + f_1 = 2 \geq 2^{1/2} = 2^{3/2-1} = 2^{(k+1)/2-1}$ **so** P(3) **holds**
3. **Inductive Hypothesis: Assume that for some arbitrary integer** $k \geq 3$, P(j) **is true for every integer** j **from** 2 **to** k.
4. **Inductive Step:** $\boxed{\textbf{Goal: Show } P(k+1)\textbf{; that is, } f_{k+1} \geq 2^{(k+1)/2-1}}$

   We have $f_{k+1} = f_k + f_{k-1}$ **by definition since** $k+1 \geq 2$

   $\geq 2^{k/2-1} + 2^{(k-1)/2-1}$ **by the IH since** $k-1 \geq 2$

   $\geq 2^{(k-1)/2-1} + 2^{(k-1)/2-1} = 2^{(k-1)/2} = 2^{(k+1)/2-1}$

   so P(k+1) **is true.**
5. **Therefore by strong induction,** $f_n \geq 2^{n/2-1}$ **for all integers** $n \geq 2$.

$f_0 = 0 \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ for all $n \geq 2$

Running time of Euclid's algorithm

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

Running time of Euclid's algorithm

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

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

We already proved that $f_n \geq 2^{n/2 - 1}$ so $f_{n+1} \geq 2^{(n-1)/2}$

Therefore: if Euclid's Algorithm takes n steps
for $\gcd(a, b)$ with $a \geq b > 0$
then $a \geq 2^{(n-1)/2}$

so $(n-1)/2 \leq \log_2 a$ or $n \leq 1 + 2\log_2 a$
i.e., # of steps ≤ 1 + twice the # of bits in a.

Running time of Euclid's algorithm

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

An informal way to get the idea: Consider an n **step** gcd **calculation starting with** $r_{n+1}=a$ **and** $r_n=b$:

$$
\begin{aligned}
r_{n+1} &= q_n r_n + r_{n-1} \\
r_n &= q_{n-1} r_{n-1} + r_{n-2} \\
&\dots \\
r_3 &= q_2 r_2 + r_1 \\
r_2 &= q_1 r_1
\end{aligned}
$$

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

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.

Running time of Euclid's algorithm

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

We go by strong induction on n**.**

Let P(n) **be** "gcd(a,b) **with** $a \geq b>0$ **takes** n **steps** $\rightarrow a \geq f_{n+1}$" **for all** $n \geq 1$**.**

Base Case: n=1 **Suppose Euclid's Algorithm with** $a \geq b > 0$ **takes** 1 **step.**
By assumption, $a \geq b \geq 1 = f_2$ **so** P(1) **holds.**

Induction Hypothesis: **Suppose that for some integer** $k \geq 1$, P(j) **is true for all integers** j **s.t.** $1 \leq j \leq k$

Running time of Euclid's algorithm

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

We go by strong induction on n.

Let P(n) **be** "gcd(a,b) **with** $a \geq b>0$ **takes** n **steps** $\rightarrow a \geq f_{n+1}$" **for all** $n \geq 1$.

<u>**Base Case**</u>: n=1 **Suppose Euclid's Algorithm with** $a \geq b > 0$ **takes** 1 **step.**
By assumption, $a \geq b \geq 1 = f_2$ **so** P(1) **holds.**

<u>**Induction Hypothesis**</u>: **Suppose that for some integer** $k \geq 1$, P(j) **is true for all integers** j **s.t.** $1 \leq j \leq k$

<u>**Inductive Step**</u>: **We want to show:** **if** gcd(a,b) **with** $a \geq b > 0$ **takes** k+1 **steps, then** $a \geq f_{k+2}$.

Running time of Euclid's algorithm

<u>Induction Hypothesis</u>: **Suppose that for some integer** $k \geq 1$, $P(j)$ **is true for all integers** j **s.t.** $1 \leq j \leq k$

<u>Inductive Step</u>: **Goal: if** $\gcd(a,b)$ **with** $a \geq b > 0$ **takes** $k+1$ **steps, then** $a \geq f_{k+2}$.

Now if $k+1=2$, **then Euclid's algorithm on** a **and** b **can be written as**

$$a = q_2 b + r_1$$
$$b = q_1 r_1$$

and $r_1 > 0$.

Also, since $a \geq b > 0$, **we must have** $q_2 \geq 1$ and $b \geq 1$.

So $a = q_2 b + r_1 \geq b + r_1 \geq 1+1 = 2 = f_3 = f_{k+2}$ as required.

Running time of Euclid's algorithm

Induction Hypothesis: **Suppose that for some integer** $k \geq 1$, $P(j)$ **is true for all integers** j **s.t.** $1 \leq j \leq k$

Inductive Step: $\boxed{\textbf{Goal: if } \gcd(a,b) \textbf{ with } a \geq b>0 \textbf{ takes } k+1 \textbf{ steps, then } a \geq f_{k+2}.}$

Next suppose that $k+1 \geq 3$ **so for the first 3 steps of Euclid's algorithm on** a **and** b **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.**

Running time of Euclid's algorithm

<u>Induction Hypothesis</u>: Suppose that for some integer $k \ge 1$, $P(j)$ **is true for all integers** j **s.t.** $1 \le j \le k$

<u>Inductive Step</u>: **Goal: if** gcd(a,b) **with** $a \ge b > 0$ **takes** $k+1$ **steps, then** $a \ge f_{k+2}$.

Next suppose that $k+1 \ge 3$ **so for the first 3 steps of Euclid's algorithm on** a **and** b **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. Note that this means that the** $\gcd(b, r_k)$ **takes** k **steps and** $\gcd(r_k, r_{k-1})$ **takes** $k-1$ **steps.**

So since $k, k-1 \ge 1$, **by the IH we have** $b \ge f_{k+1}$ **and** $r_k \ge f_k$.

Running time of Euclid's algorithm

<u>Induction Hypothesis</u>: Suppose that for some integer $k \geq 1$, $P(j)$ **is true for all integers** j **s.t.** $1 \leq j \leq k$

<u>Inductive Step</u>: $\boxed{\textbf{Goal: if } \gcd(a,b) \textbf{ with } a \geq b>0 \textbf{ takes } k+1 \textbf{ steps, then } a \geq f_{k+2}.}$

Next suppose that $k+1 \geq 3$ **so for the first 3 steps of Euclid's algorithm on** a **and** b **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. Note that this means that the** $\gcd(b, r_k)$ **takes** k **steps and** $\gcd(r_k, r_{k-1})$ **takes** $k-1$ **steps.**

So since $k, k-1 \geq 1$, **by the IH we have** $b \geq f_{k+1}$ **and** $r_k \geq f_k$.

Also, since $a \geq b$, **we must have** $q_{k+1} \geq 1$.

So $a = q_{k+1}b + r_k \geq b + r_k \geq f_{k+1} + f_k = f_{k+2}$ **as required.** ■

Last time: Recursive definitions of functions

- $0! = 1;\ \ (n+1)! = (n+1) \cdot n!$ **for all** $n \geq 0$**.**

- $F(0) = 0;\ \ F(n+1) = F(n) + 1$ **for all** $n \geq 0$**.**

- $G(0) = 1;\ \ G(n+1) = 2 \cdot G(n)$ **for all** $n \geq 0$**.**

- $H(0) = 1;\ \ H(n+1) = 2^{H(n)}$ **for all** $n \geq 0$**.**

Last time: Recursive definitions of functions

- Recursive functions allow general computation
  - saw examples not expressible with simple expressions
- So far, we have considered only simple data
  - inputs and outputs were just integers
- We need general data as well…
  - these will also be described *recursively*
  - will allow us to describe data of real programs

    e.g., strings, lists, trees, expressions, propositions, …
- We'll start simple: sets of numbers

Recursive Definitions of Sets (Data)

Natural numbers

Basis: $0 \in S$

Recursive: If $x \in S$, then $x+1 \in S$

Even numbers

Basis: $0 \in S$

Recursive: If $x \in S$, then $x+2 \in S$

Recursive Definition of Sets

Recursive definition of set S

- **Basis Step:** $0 \in S$
- **Recursive Step: If** $x \in S$**, then** $x + 2 \in S$

The only elements in S **are those that follow from the basis step and a finite number of recursive steps**

Recursive Definitions of Sets

Natural numbers

Basis: $0 \in S$

Recursive: If $x \in S$, then $x+1 \in S$

Even numbers

Basis: $0 \in S$

Recursive: If $x \in S$, then $x+2 \in S$

Powers of 3:

Basis: $1 \in S$

Recursive: If $x \in S$, then $3x \in S$.

Basis: $(0, 0) \in S$, $(1, 1) \in S$

Recursive: If $(n-1, x) \in S$ and $(n, y) \in S$, then $(n+1, x + y) \in S$. ?

Recursive Definitions of Sets

Natural numbers

Basis: $0 \in S$

Recursive: If $x \in S$, then $x+1 \in S$

Even numbers

Basis: $0 \in S$

Recursive: If $x \in S$, then $x+2 \in S$

Powers of 3:

Basis: $1 \in S$

Recursive: If $x \in S$, then $3x \in S$.

Basis: $(0, 0) \in S$, $(1, 1) \in S$

Recursive: If $(n-1, x) \in S$ and $(n, y) \in S$, then $(n+1, x + y) \in S$.

Fibonacci numbers

Last time: Recursive definitions of functions

- Before, we considered only simple data
  - inputs and outputs were just integers
- Proved facts about those functions with **induction**
  - $n! \leq n^n$
  - $f_n < 2^n$ and $f_n \geq 2^{n/2-1}$
- How do we prove facts about functions that work with more complex (recursively defined) data?
  - we need a more sophisticated form of induction

Structural Induction

How to prove $\forall\, x \in S, P(x)$ is true:

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the ***Basis step***

Inductive Hypothesis: Assume that P is true for some arbitrary values of ***each*** of the existing named elements mentioned in the ***Recursive step***

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the ***Recursive step*** using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\, x \in S, P(x)$

Structural Induction

Basis: $0 \in S$
Recursive: If $x \in S$, then $x+2 \in S$

How to prove $\forall\, x \in S, P(x)$ is true:

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

Inductive Hypothesis: Assume that P is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\, x \in S, P(x)$

Structural Induction vs. Ordinary Induction

Ordinary induction is a special case of structural induction:

Recursive definition of $\mathbb{N}$

Basis: $0 \in \mathbb{N}$

Recursive step: If $k \in \mathbb{N}$ then $k + 1 \in \mathbb{N}$

Structural induction follows from ordinary induction:

Define $Q(n)$ to be "for all $x \in S$ that can be constructed in at most n recursive steps, $P(x)$ is true."

Using Structural Induction

- Let S be given by…
  - **Basis:** $6 \in S$; $15 \in S$
  - **Recursive:** if $x, y \in S$ then $x + y \in S$.

Claim: Every element of S is divisible by 3.

Claim: Every element of S is divisible by 3.

1. Let P(x) be "3|x". We prove that P(x) is true for all x ∈ S by structural induction.

Basis: $6 \in S$; $15 \in S$

Recursive: if $x, y \in S$, then $x + y \in S$

Claim: Every element of S is divisible by 3.

1. Let P(x) be "3|x". We prove that P(x) is true for all x ∈ S by structural induction.
2. Base Case: 3|6 and 3|15 so P(6) and P(15) are true

Basis: $6 \in S$; $15 \in S$

Recursive: if $x, y \in S$, then $x + y \in S$

Claim: Every element of S is divisible by 3.

1. **Let** P(x) **be** "3|x". **We prove that** P(x) **is true for all** $x \in S$ **by structural induction.**
2. **Base Case:** 3|6 **and** 3|15 **so** P(6) **and** P(15) **are true**
3. **Inductive Hypothesis: Suppose that** P(x) **and** P(y) **are true for some arbitrary** $x,y \in S$
4. **Inductive Step:** **Goal: Show** P(x+y)

Basis: $6 \in S$; $15 \in S$

Recursive: if $x, y \in S$, then $x + y \in S$

Claim: Every element of S is divisible by 3.

1. **Let** P(x) **be** "3|x". **We prove that** P(x) **is true for all** x ∈ S **by structural induction.**
2. **Base Case:** 3|6 **and** 3|15 **so** P(6) **and** P(15) **are true**
3. **Inductive Hypothesis: Suppose that** P(x) **and** P(y) **are true for some arbitrary** x,y ∈ S
4. **Inductive Step:** **Goal: Show** P(x+y)

   Since P(x) **is true,** 3|x **and so** x=3m **for some integer** m **and since** P(y) **is true,** 3|y **and so** y=3n **for some integer** n. **Therefore** x+y=3m+3n=3(m+n) **and thus** 3|(x+y).

   Hence P(x+y) **is true.**

Basis: $6 \in S$; $15 \in S$

Recursive: if $x, y \in S$, then $x + y \in S$

Claim: Every element of S is divisible by 3.

1. **Let** P(x) **be** "3|x". **We prove that** P(x) **is true for all** x ∈ S **by structural induction.**
2. **Base Case:** 3|6 **and** 3|15 **so** P(6) **and** P(15) **are true**
3. **Inductive Hypothesis: Suppose that** P(x) **and** P(y) **are true for some arbitrary** x,y ∈ S
4. **Inductive Step:** **Goal: Show** P(x+y)

   Since P(x) **is true,** 3|x **and so** x=3m **for some integer** m **and since** P(y) **is true,** 3|y **and so** y=3n **for some integer** n.
   Therefore x+y=3m+3n=3(m+n) **and thus** 3|(x+y).

   Hence P(x+y) **is true.**
5. **Therefore by induction** 3|x **for all** x ∈ S.

Basis: $6 \in S$; $15 \in S$

Recursive: if $x, y \in S$, then $x + y \in S$

Using Structural Induction

- Let T be given by…
  - **Basis:** $6 \in T$; $15 \in T$
  - **Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

- Two base cases and two *recursive* cases

Claim: Every element of T is also in S.

Claim: Every element of T is an element of S

1. Let P(x) be "x ∈ S". We prove that P(x) is true for all x ∈ T by structural induction.

Basis: $6 \in S$; $15 \in S$	**Basis:** $6 \in T$; $15 \in T$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

Claim: Every element of T is an element of S

1. **Let** P(x) **be** "x ∈ S". **We prove that** P(x) **is true for all** x ∈ T **by structural induction.**
2. **Base Case:** 6 ∈ S **and** 15 ∈ S **so** P(6) **and** P(15) **are true**

Basis: $6 \in S$; $15 \in S$	**Basis:** $6 \in T$; $15 \in T$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

Claim: Every element of T is an element of S

1. Let P(x) be "x ∈ S". We prove that P(x) is true for all x ∈ T by structural induction.
2. Base Case: 6 ∈ S and 15 ∈ S so P(6) and P(15) are true
3. Inductive Hypothesis: Suppose that P(x) is true for some arbitrary x ∈ T

Basis: $6 \in S$; $15 \in S$	**Basis:** $6 \in T$; $15 \in T$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

Claim: Every element of T is an element of S

1. **Let** P(x) **be** "x ∈ S". **We prove that** P(x) **is true for all** x ∈ T **by structural induction.**
2. **Base Case:** 6 ∈ S **and** 15 ∈ S **so** P(6) **and** P(15) **are true**
3. **Inductive Hypothesis: Suppose that** P(x) **is true for some arbitrary** x ∈ T
4. **Inductive Step: Goal: Show** P(x+6) **and** P(x+15)

Basis: $6 \in S$; $15 \in S$	**Basis:** $6 \in T$; $15 \in T$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

Claim: Every element of T is an element of S

1. Let P(x) be "x ∈ S". We prove that P(x) is true for all x ∈ T by structural induction.
2. Base Case: 6 ∈ S and 15 ∈ S so P(6) and P(15) are true
3. Inductive Hypothesis: Suppose that P(x) is true for some arbitrary x ∈ T
4. Inductive Step: **Goal: Show P(x+6) and P(x+15)**

   Since P(x) holds, we have x ∈ S. From the recursive step of S, we can see that x + 6 ∈ S, so P(x+6) is true, and we can see that x + 15 ∈ S, so P(x+15) is true.

Basis: $6 \in S$; $15 \in S$	**Basis:** $6 \in T$; $15 \in T$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

Claim: Every element of T is an element of S

1. Let P(x) be "x ∈ S". We prove that P(x) is true for all x ∈ T by structural induction.
2. Base Case: 6 ∈ S and 15 ∈ S so P(6) and P(15) are true
3. Inductive Hypothesis: Suppose that P(x) is true for some arbitrary x ∈ T
4. Inductive Step: Goal: Show P(x+6) and P(x+15)

   Since P(x) holds, we have x ∈ S. From the recursive step of S, we can see that x + 6 ∈ S, so P(x+6) is true, and we can see that x + 15 ∈ S, so P(x+15) is true.
5. Therefore P(x) for all x ∈ T by induction.

Basis: $6 \in S$; $15 \in S$	**Basis:** $6 \in T$; $15 \in T$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in T$, then $x + 6 \in T$ and $x + 15 \in T$

Last time: Recursive Definitions

- Recursively defined functions and sets are our mathematical models of code and the data it uses
  - any recursively defined set can be translated into a Java class
  - any recursively defined function can be translated into a Java function
    - some (but not all) can be written more cleanly as loops

- Can now do proofs about CS-specific objects

Lists of Integers

- **Basis:** $\text{nil} \in$ **List**
- **Recursive step:**

  if $\mathrm{L} \in$ **List and** $\mathrm{a} \in \mathbb{Z}$,

  then $\mathrm{a} :: \mathrm{L} \in$ **List**

Examples:

– nil	
– $1 :: \text{nil}$	1
– $1 :: 2 :: \text{nil}$	$1 \to 2$
– $1 :: 2 :: 3 :: \text{nil}$	$1 \to 2 \to 3$

Functions on Lists

Length:

$\text{len}(\text{nil}) := 0$

$\text{len}(a :: L) := \text{len}(L) + 1$ **for any** L ∈ **List and** a ∈ $\mathbb{Z}$

Concatenation:

$\text{concat}(\text{nil}, R) :=$ R **for any** R ∈ **List**

$\text{concat}(a :: L, R) := a :: \text{concat}(L, R)$ **for any** L, R ∈ **List and any** a ∈ $\mathbb{Z}$

Structural Induction

Basis: $\mathrm{nil} \in$ **List**

Recursive step:

if $\mathrm{L} \in$ **List and** $\mathrm{a} \in \mathbb{Z}$**,**

then $\mathrm{a} :: \mathrm{L} \in$ **List**

How to prove $\forall\, x \in S, P(x)$ **is true:**

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

Inductive Hypothesis: Assume that P is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\, x \in S, P(x)$

Claim: $\text{concat}(\text{L}, \text{nil}) = \text{L}$ **for all** $\text{L} \in$ **List**

Claim: $\mathrm{concat}(\mathrm{L}, \mathrm{nil}) = \mathrm{L}$ **for all** $\mathrm{L} \in$ **List**

Let $\mathsf{P(L)}$ **be** "$\mathrm{concat}(\mathrm{L}, \mathrm{nil}) = \mathrm{L}$" .

We will prove $\mathsf{P(L)}$ **for all** $\mathsf{L} \in$ **List by structural induction.**

Claim: $\text{concat}(\text{L}, \text{nil}) = \text{L}$ **for all** $\text{L} \in$ **List**

Let P(L) **be** "$\text{concat}(\text{L}, \text{nil}) = \text{L}$" .
We will prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **By the definition of concat, we can see that** $\text{concat}(\text{nil}, \text{nil}) = \text{nil}$, **which is** P(nil)**.**

Claim: $\text{concat}(L, \text{nil}) = L$ **for all** $L \in$ **List**

Let P(L) **be** "$\text{concat}(L, \text{nil}) = L$".
We will prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **By the definition of concat, we can see that** $\text{concat}(\text{nil}, \text{nil}) = \text{nil}$, **which is** $P(\text{nil})$**.**

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** L ∈ List, **i.e.,** $\text{concat}(L, \text{nil}) = L$.

Inductive Step: **Goal: Show that** $P(a :: L)$ **is true for any** $a \in \mathbb{Z}$

Claim: $\mathrm{concat}(\mathrm{L}, \mathrm{nil}) = \mathrm{L}$ **for all** $\mathrm{L} \in$ **List**

Let P(L) **be** "$\mathrm{concat}(\mathrm{L}, \mathrm{nil}) = \mathrm{L}$".
We will prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **By the definition of concat, we can see that** $\mathrm{concat}(\mathrm{nil}, \mathrm{nil}) = \mathrm{nil}$, **which is** P(nil).

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** L ∈ List, **i.e.,** $\mathrm{concat}(\mathrm{L}, \mathrm{nil}) = \mathrm{L}$.

Inductive Step: **Goal: Show that** $\mathrm{P}(\mathrm{a} :: \mathrm{L})$ **is true for any** $\mathrm{a} \in \mathbb{Z}$.

Let $\mathrm{a} \in \mathbb{Z}$ **be arbitrary. We can calculate as follows**

$\mathrm{concat}(\mathrm{a} :: \mathrm{L}, \mathrm{nil})$	$= \mathrm{a} :: \mathrm{concat}(\mathrm{L}, \mathrm{nil}))$	**def of** concat
	$= \mathrm{a} :: \mathrm{L}$	**IH**

which is $\mathrm{P}(\mathrm{a} :: \mathrm{L})$.

By induction, we have shown the claim holds for all L ∈ **List**.

Claim: $\mathrm{len}(\mathrm{concat}(\mathrm{L}, \mathrm{R})) = \mathrm{len}(\mathrm{L}) + \mathrm{len}(\mathrm{R})$ **for all** $\mathrm{L}, \mathrm{R} \in$ **List**

Claim: $\mathrm{len}(\mathrm{concat}(L, R)) = \mathrm{len}(L) + \mathrm{len}(R)$ **for all** $L, R \in$ **List**

Let $P(L)$ **be** "$\mathrm{len}(\mathrm{concat}(L, R)) = \mathrm{len}(L) + \mathrm{len}(R)$ **for all** $R \in$ **List**".
We prove $P(L)$ **for all** $L \in$ **List by structural induction.**

Claim: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L, R \in$ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**" .
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil)**: Let** R ∈ **List be arbitrary. Then,**

Length:

$\text{len}(\text{nil}) := 0$
$\text{len}(a :: L) := \text{len}(L) + 1$

Concatenation:

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

Claim: $\text{len}(\text{concat}(\text{L}, \text{R})) = \text{len}(\text{L}) + \text{len}(\text{R})$ **for all** L, R ∈ **List**

Let P(L) **be** "$\text{len}(\text{concat}(\text{L}, \text{R})) = \text{len}(\text{L}) + \text{len}(\text{R})$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **List be arbitrary. Then,**

$$
\begin{aligned}
\text{len}(\text{concat}(\text{nil}, \text{R})) &= \text{len}(\text{R}) && \textbf{def of concat} \\
&= 0 + \text{len}(\text{R}) \\
&= \text{len}(\text{nil}) + \text{len}(\text{R}) && \textbf{def of len}
\end{aligned}
$$

Since R **was arbitrary,** P(nil) **holds.**

Claim: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** L, R ∈ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **List be arbitrary. Then,** $\text{len}(\text{concat}(\text{nil}, R))$
$= \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R)$, **showing** P(nil).

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** L ∈ List, **i.e.,** $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**.

Claim: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** L, R ∈ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **List be arbitrary. Then,** $\text{len}(\text{concat}(\text{nil}, R))$ $= \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R)$, **showing** P(nil).

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** L ∈ List, **i.e.,** $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**.

Inductive Step: **Goal: Show that** P(a :: L) **is true for any** $a \in \mathbb{Z}$**.**

Claim: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L, R \in$ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **List be arbitrary. Then,** $\text{len}(\text{concat}(\text{nil}, R))$
$= \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R)$, **showing** P(nil).

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** L ∈ List, **i.e.,** $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**.

Inductive Step: $\boxed{\textbf{Goal: Show that } P(a :: L) \textbf{ is true for any } a \in \mathbb{Z}.}$

Let $a \in \mathbb{Z}$ **and** R ∈ **List be arbitrary. Then,**

Length:

$\text{len}(\text{nil}) := 0$
$\text{len}(a :: L) := \text{len}(L) + 1$

Concatenation:

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

Claim: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L, R \in$ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $R \in$ **List**".
We prove P(L) **for all** $L \in$ **List by structural induction.**

Base Case (nil): **Let** $R \in$ **List be arbitrary. Then,** $\text{len}(\text{concat}(\text{nil}, R))$ $= \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R)$, **showing** P(nil).

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** $L \in$ List, **i.e.,** $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $R \in$ **List**.

Inductive Step: **Goal: Show that** $P(a :: L)$ **is true for any** $a \in \mathbb{Z}$.

Let $a \in \mathbb{Z}$ **and** $R \in$ **List be arbitrary. Then, we can calculate**

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

Since R **was arbitrary, we have shown** $P(a :: L)$.

By induction, we have shown the claim holds for all $L \in$ **List**.

Claim: $\mathrm{len}(\mathrm{concat}(L, R)) = \mathrm{len}(L) + \mathrm{len}(R)$ **for all** $L \in$ **List**

Alternative Strategy:

- **Do the direct proof *outside* the induction!**

Let R **be an arbitrary list.**

> **Prove** P(L) **by structural induction,**
> **where** P(L) **is** "$\mathrm{len}(\mathrm{concat}(L, R)) = \mathrm{len}(L) + \mathrm{len}(R)$"

Since R **was arbitrary, we have proven the claim.**

Claim: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L \in$ **List**

Let R **be an arbitrary list. We continue by induction.**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$". **We will prove** P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **We have** $\text{len}(\text{concat}(\text{nil}, R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R)$, **showing** P(nil).

Inductive Hypothesis: Assume that P(L) **is true for some arbitrary** L ∈ List, **i.e.**, $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$.

Inductive Step: Let $a \in \mathbb{Z}$ **be arbitrary. We can prove** $P(a :: L)$ **since**

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

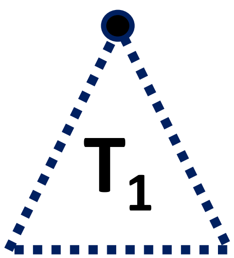
By induction, we have shown the claim holds for all L ∈ **List**.
Since R **was arbitrary, we have proven the claim.**

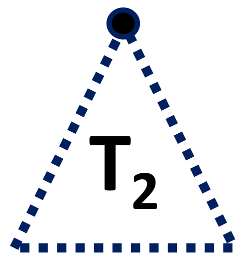
Rooted Binary Trees

- Basis: • is a rooted binary tree

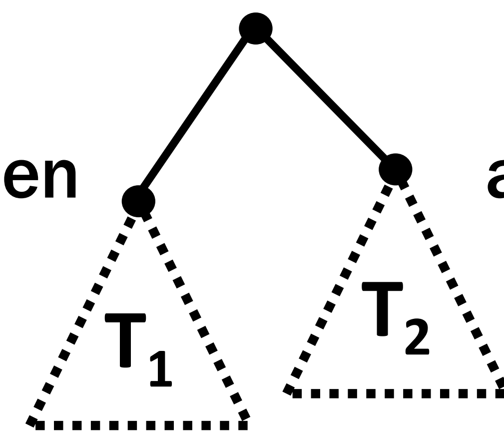
Rooted Binary Trees

- Basis: • is a rooted binary tree
- Recursive step:

If 

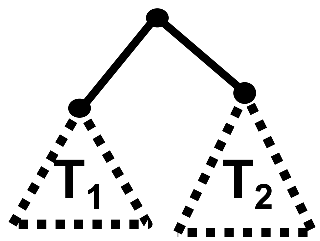
and 

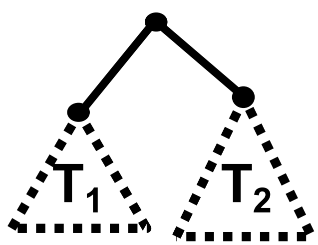
are rooted binary trees,

then 

also is a rooted binary tree.

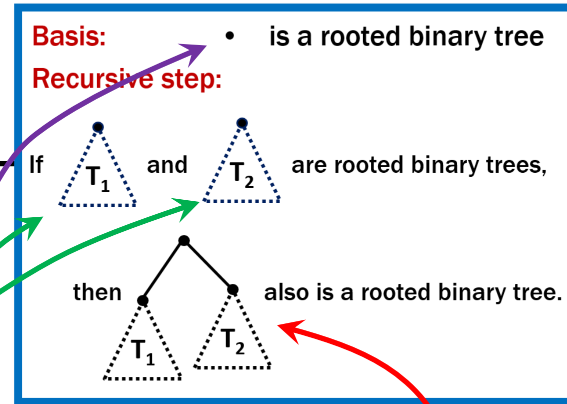
Defining Functions on Rooted Binary Trees

- size(•) ::= 1
- size (

) ::= 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)
- height(•) ::= 0
- height (

) ::= 1 + max{height($\mathbf{T_1}$), height($\mathbf{T_2}$)}

Last time: Structural Induction



How to prove $\forall\ x \in S, P(x)$ is true:

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

Inductive Hypothesis: Assume that P is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\ x \in S, P(x)$

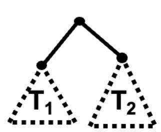
Claim: **For every rooted binary tree T,** size(**T**) $\leq 2^{\text{height}(\mathbf{T}) + 1} - 1$

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

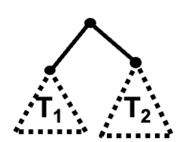
---

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

size(•) ::= 1

size () ::= 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)

height(•) ::= 0

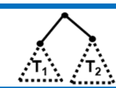
height () ::= 1 + max{height($\mathbf{T_1}$), height($\mathbf{T_2}$)}

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

1. **Let** P(**T**) **be** "size(T) $\leq 2^{height(T)+1}-1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **is true.**

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

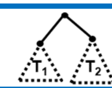
---

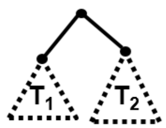
1. **Let** P(**T**) **be** "size(T) ≤ $2^{\text{height}(T)+1}-1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **is true.**
3. **Inductive Hypothesis: Suppose that** P(T_1) **and** P(T_2) **are true for some rooted binary trees** T_1 **and** T_2**, i.e.,** size(T_k) ≤ $2^{\text{height}(T_k) + 1} - 1$ **for** k=1,2
4. **Inductive Step:** **Goal: Prove** P(

).

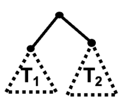
Claim: For every rooted binary tree T, size(T) $\leq 2^{\text{height}(\mathbf{T})+1} - 1$

1. **Let** P(**T**) **be** "size(T) $\leq 2^{\text{height}(T)+1} - 1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **is true.**
3. **Inductive Hypothesis: Suppose that** P($\mathbf{T_1}$) **and** P($\mathbf{T_2}$) **are true for some rooted binary trees** $\mathbf{T_1}$ **and** $\mathbf{T_2}$**, i.e.,** size($\mathbf{T}_k$) $\leq 2^{\text{height}(\mathbf{T}_k)+1} - 1$ **for** k=1,2
4. **Inductive Step:**

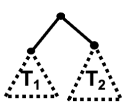
Goal: Prove P().

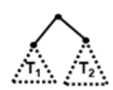
size()

size(•) ::= 1

size () ::= 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)

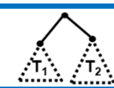
height(•) ::= 0

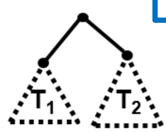
height () ::= 1 + max{height($\mathbf{T_1}$), height($\mathbf{T_2}$)}

$\leq 2^{\text{height(}}$  $^{)+1} - 1$

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

---

1. **Let** P(**T**) **be** "size(T) $\leq 2^{\text{height}(T)+1} - 1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **is true.**
3. **Inductive Hypothesis: Suppose that** P(**T_1**) **and** P(**T_2**) **are true for some rooted binary trees T_1 and T_2, i.e.,** size(**T_k**) $\leq 2^{\text{height}(T_k)+1} - 1$ **for** k=1,2
4. **Inductive Step:** **Goal: Prove** P().

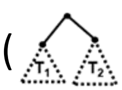
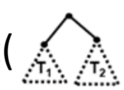
By def, size() $= 1+\text{size}(\mathbf{T_1})+\text{size}(\mathbf{T_2})$

$\leq 1+2^{\text{height}(\mathbf{T_1})+1}-1+2^{\text{height}(\mathbf{T_2})+1}-1$

by IH for $\mathbf{T_1}$ and $\mathbf{T_2}$

$\leq 2^{\text{height}(\mathbf{T_1})+1}+2^{\text{height}(\mathbf{T_2})+1}-1$

$\leq 2(2^{\max(\text{height}(\mathbf{T_1}),\text{height}(\mathbf{T_2}))+1})-1$

$\leq 2(2^{\text{height}($  $)}) - 1 \leq 2^{\text{height}($  $)+1} - 1$

which is what we wanted to show.

5. **So, the** P(T) **is true for all rooted binary trees by structural induction.**

Strings

- An *alphabet* Σ is any finite set of characters

- The set Σ^* of *strings* over the alphabet Σ
  - example: {0,1}* is the set of *binary strings*

    0, 1, 00, 01, 10, 11, 000, 001, … and “”

- Σ^* is defined recursively by
  - Basis: $\varepsilon \in \Sigma^*$ (ε is the empty string, i.e., “”)
  - Recursive: if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$

Last time: Structural Induction

How to prove $\forall\, x \in S, P(x)$ is true:

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

Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

Inductive Hypothesis: Assume that P is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

Inductive Step: Prove that $P(w)$ holds for each of the new elements w constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

Conclude that $\forall\, x \in S, P(x)$

Functions on Recursively Defined Sets (on Σ^*)

Length:

$\text{len}(\varepsilon) ::= 0$

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

Concatenation:

$x \bullet \varepsilon ::= x$ for $x \in \Sigma^*$

$x \bullet wa ::= (x \bullet w)a$ for $x \in \Sigma^*$, $a \in \Sigma$

Reversal:

$\varepsilon^R ::= \varepsilon$

$(wa)^R ::= \varepsilon a \bullet w^R$ 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 $w \in \Sigma^*$, $a \in \Sigma$, $a \neq c$

separate cases for
c vs $a \neq c$

Claim: $\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ **for all** $x, y \in \Sigma^*$

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

Base Case ($y = \varepsilon$): **Let** $x \in \Sigma^*$ **be arbitrary. Then,** $\text{len}(x \bullet \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon)$ **since** $\text{len}(\varepsilon)=0$. **Since** x **was arbitrary,** $P(\varepsilon)$ **holds.**

Inductive Hypothesis: Assume that $P(w)$ **is true for some arbitrary** $w \in \Sigma^*$**, i.e.,** $\text{len}(x \bullet w) = \text{len}(x) + \text{len}(w)$ for all x

Claim: $\mathrm{len}(x \bullet y) = \mathrm{len}(x) + \mathrm{len}(y)$ **for all** $x,y \in \Sigma^*$

Does this look familiar?

Let $P(y)$ **be** "$\mathrm{len}(x \bullet y) = \mathrm{len}(x) + \mathrm{len}(y)$ for all $x \in$
We prove $P(y)$ **for all** $y \in \Sigma^*$ **by structural indu**

Base Case ($y = \varepsilon$): **Let** $x \in \Sigma^*$ **be arbitrary. Then,** $\mathrm{len}(x \bullet \varepsilon) = \mathrm{len}(x) = \mathrm{len}(x) + \mathrm{len}(\varepsilon)$ **since** $\mathrm{len}(\varepsilon)=0$. **Since** x **was arbitrary,** $P(\varepsilon)$ **holds.**

Inductive Hypothesis: Assume that $P(w)$ **is true for some arbitrary** $w \in \Sigma^*$**, i.e.,** $\mathrm{len}(x \bullet w) = \mathrm{len}(x) + \mathrm{len}(w)$ for all x

Inductive Step: **Goal: Show that** $P(wa)$ **is true for every** $a \in \Sigma$

Let $a \in \Sigma$ **and** $x \in \Sigma^*$. **Then**

$$\begin{aligned}\mathrm{len}(x \bullet wa) &= \mathrm{len}((x \bullet w)a) && \textbf{by def of } \bullet \\ &= \mathrm{len}(x \bullet w)+1 && \textbf{by def of len} \\ &= \mathrm{len}(x)+\mathrm{len}(w)+1 && \textbf{by I.H.} \\ &= \mathrm{len}(x)+\mathrm{len}(wa) && \textbf{by def of len}\end{aligned}$$

Therefore, $\mathrm{len}(x \bullet wa)= \mathrm{len}(x)+\mathrm{len}(wa)$ **for all** $x \in \Sigma^*$**, so** $P(wa)$ **is true.**

So, by induction $\mathrm{len}(x \bullet y) = \mathrm{len}(x) + \mathrm{len}(y)$ **for all** $x,y \in \Sigma^*$

Lists versus Strings

- Our strings are basically lists
  except that we draw them backward

[1, 2, 3]	1 :: 2 :: 3 :: nil	1 → 2 → 3
“abc”	εabc	a ⇠ b ⇠ c

  – would be represented the same way in memory
  – but we think of head as the right-most not left-most

Claim: $\mathrm{len}(x^R) = \mathrm{len}(x)$ for all $x \in \Sigma^*$

Let $P(x)$ **be** "$\mathrm{len}(x^R) = \mathrm{len}(x)$".

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

Length:

$\mathrm{len}(\varepsilon) ::= 0$

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

Reversal:

$\varepsilon^R ::= \varepsilon$

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

Claim: len(x^R) = len(x) **for all** x $\in \Sigma^*$

Let P(x) **be** "len(x^R) = len(x)".

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

Base Case (x = ε): **Then,** len(ε^R) = len(ε) **by def of string reverse.**

Claim: len(x^R) = len(x) for all x $\in \Sigma^*$

Let P(x) **be** "len(x^R) = len(x)".

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

Base Case (x = ε): **Then,** len(ε^R) = len(ε) **by def of string reverse.**

Inductive Hypothesis: Assume that P(w) **is true for some arbitrary** w $\in \Sigma^*$**, i.e.,** len(w^R) = len(w).

Inductive Step: $\boxed{\textbf{Goal: Show that } \mathrm{len}((wa)^R) = \mathrm{len}(wa) \textbf{ for every } a}$

Length:
len(ε) ::= 0
len(wa) ::= len(w) + 1 for w $\in \Sigma^*$, a $\in \Sigma$

Reversal:
ε^R ::= ε
$(wa)^R$ ::= εa • w^R for w $\in \Sigma^*$, a $\in \Sigma$

Claim: $\text{len}(x^R) = \text{len}(x)$ for all $x \in \Sigma^*$

Let $P(x)$ **be** "$\text{len}(x^R) = \text{len}(x)$".

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

Base Case ($x = \varepsilon$): **Then,** $\text{len}(\varepsilon^R) = \text{len}(\varepsilon)$ **by def of string reverse.**

Inductive Hypothesis: Assume that $P(w)$ **is true for some arbitrary** $w \in \Sigma^*$, **i.e.,** $\text{len}(w^R) = \text{len}(w)$.

Inductive Step: **Goal: Show that** $\text{len}((wa)^R) = \text{len}(wa)$ **for every** a

Let $a \in \Sigma$. **Then,**

$\text{len}((wa)^R)$	$= \text{len}(\varepsilon a \bullet w^R)$	**def of reverse**
	$= \text{len}(\varepsilon a) + \text{len}(w^R)$	**by previous result**
	$= \text{len}(\varepsilon a) + \text{len}(w)$	**IH**
	$= 1 + \text{len}(w)$	**def of len (twice)**
	$= \text{len}(wa)$	**def of len**

Therefore, $\text{len}((wa)^R) = \text{len}(wa)$, **so** $P(wa)$ **is true for every** $a \in \Sigma$.

So, we have shown $\text{len}(x^R) = \text{len}(x)$ **for all** $x \in \Sigma^*$ **by induction.**

More Theorems

Structural induction is the tool used to prove many more interesting theorems

- General associativity follows from our one rule
  - likewise for generalized De Morgan's laws

- Okay to substitute y for x everywhere in a modular equation when we know that $x \equiv_m y$

- More coming shortly…