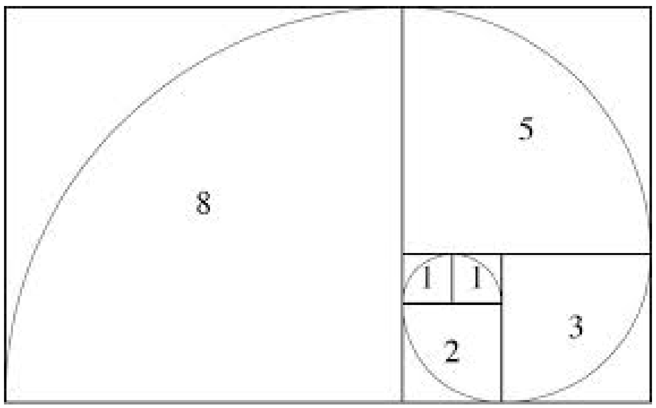
CSE 311: Foundations of Computing

Lecture 16: Recursion & Strong Induction
Applications: Fibonacci & Euclid



Avg / Median

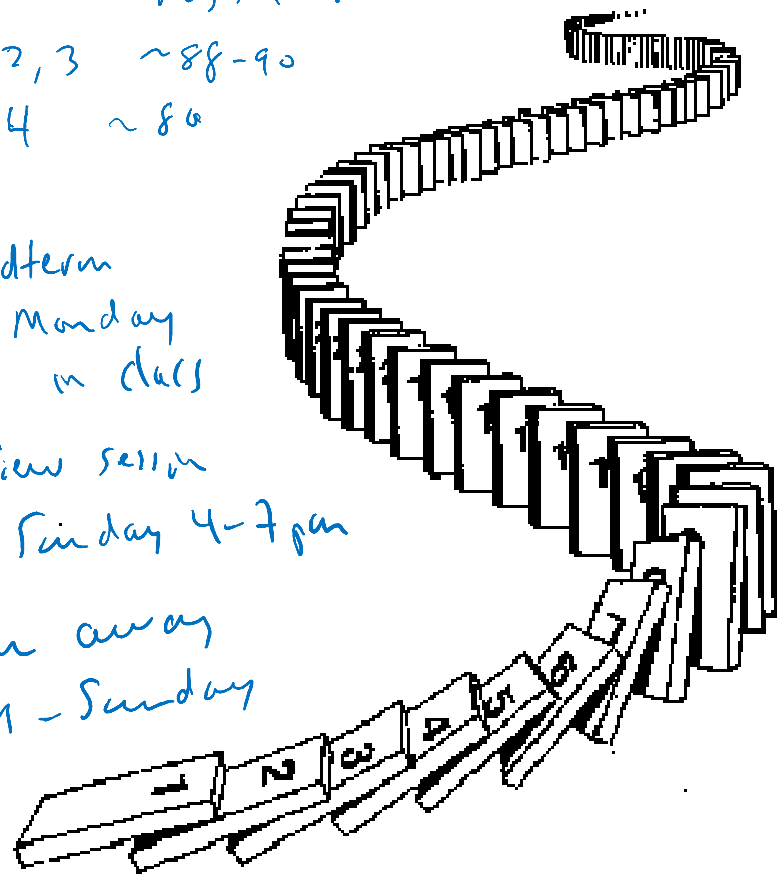
1, 2, 3 ~88-90

4 ~80

Midterm
Monday
in class

Review session
Sunday 4-7pm

I am away
Thurs - Sunday



More Recursive Definitions

$\sum_{i=1}^{k} h(i) = 0$ $k=0$

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** $n \geq 0$

$k=0$

$\prod_{i=1}^{k} h(i) = \underline{1}$

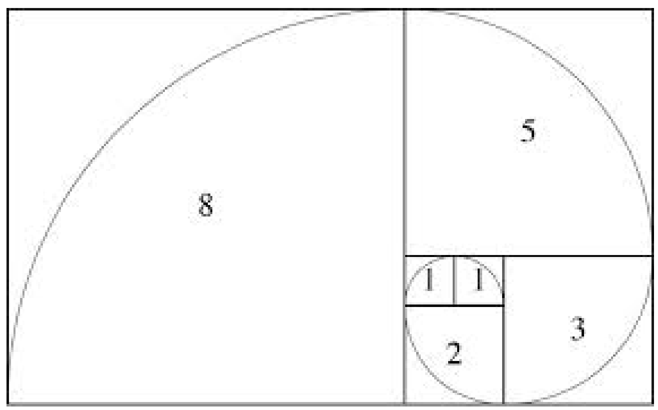
There is also product notation:

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

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

Fibonacci Numbers

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



Strong Inductive Proofs In 5 Easy Steps

1. "Let $P(n)$ be... . We will show that $P(n)$ is true for all integers $n \geq b$ by *strong* induction."
2. "Base Case:" Prove $P(b)$
3. "Inductive Hypothesis:

   Assume that for some arbitrary integer $k \geq b$,

   $P(j)$ *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 $P(b), \dots, P(k)$ 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 "$f_n < 2^n$" for all $n \geq 0$. We will prove that $P(n)$ is true for all $n \geq 0$.

2. Base Case: ~~when~~ $n = 0$: $2^0 = 1 > 0 = f_0$ so $P(0)$ is true.

$\cancel{f_0 < 2^0 \quad 0 < 1}$

Since $0 < 1$ we have $f_0 < 2^0$ and ~~are~~.

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ 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.**

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ 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** $k \geq 0$, P(j) **is true 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 $k \geq 0$, P(j) is true for every integer j from 0 to k.
4. Inductive Step: $\boxed{\text{Goal: Show P(k+1); that is, } f_{k+1} < 2^{k+1}}$

<u>Case k+1 = 1</u>: $f_{k+1} = f_1 = 1 \quad 2^{k+1} = 2^1 = 2$ so $f_{k+1} < 2^{k+1}$ ✓

<u>Case k+1 ≥ 2</u>: $f_{k+1} = f_k + f_{k-1}$ since $k+1 \geq 2$

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

$< 2^k + 2^k$

$= 2(2^k) = 2^{k+1}$ ✓

$f_0 = 0 \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ 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** $k \geq 0$, P(j) **is true 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>**: Then** $f_1 = 1 < 2 = 2^1$ **so** P(k+1) **is true here.**

   <u>Case k+1 ≥ 2</u>**: 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** $\boldsymbol{n} \geq \mathbf{2}$

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

$\boldsymbol{f_0 = 0} \quad \boldsymbol{f_1 = 1}$
$\boldsymbol{f_n = f_{n-1} + f_{n-2}}$ 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:** $f_2 = f_1 + \underset{\wedge}{f_0} \overset{= 1+0}{=} 1$ and $2^{2/2-1} = 2^0 = 1$ **so** P(2) **is true.**

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ 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 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: $\boxed{\text{Goal: Show } P(k+1)\text{; that is, } f_{k+1} \geq 2^{(k+1)/2-1}}$

   k = 2

   <u>Case k+1 = 3</u>:

   k ≥ 3

   <u>Case k+1 ≥ 4</u>:

$f_0 = 0 \quad f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ 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 ≥ 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 ≥ 2, P(j) **is true for every integer** j **from** 2 **to** k**.**
4. **Inductive Step:** $\boxed{\textbf{Goal: Show } P(k+1)\textbf{; that is, } f_{k+1} \geq 2^{(k+1)/2-1}}$

   <u>Case k+1 = 3</u>: **Then** $f_{k+1} = f_3 = f_2 + f_1 = 2 \geq 2^{1/2} = 2^{3/2-1} = 2^{(k+1)/2-1}$

   <u>Case k+1 ≥ 4</u>:

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

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

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

   $= 2^{\frac{k-1}{2}} = 2^{\frac{k+1}{2}-1}$ ✓

$f_0 = 0 \quad f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$ 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 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:** $\boxed{\textbf{Goal: Show } P(k+1)\textbf{; that is, } f_{k+1} \geq 2^{(k+1)/2-1}}$

   <u>Case k+1 = 3</u>: **Then** $f_{k+1} = f_3 = f_2 + f_1 = 2 \geq 2^{1/2} = 2^{3/2-1} = 2^{(k+1)/2-1}$

   <u>Case k+1 ≥ 4</u>: $f_{k+1} = f_k + f_{k-1}$ **by definition**

   $\geq 2^{k/2-1} + 2^{(k-1)/2-1}$ **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 in both cases.**

Actually

$f_n = \frac{1}{\sqrt{5}}\left(\left(\frac{\sqrt{5}+1}{2}\right)^n - \left(\frac{\sqrt{5}-1}{2}\right)^n\right)$

5. **Therefore by strong induction,** $f_n \geq 2^{n/2-1}$ **for all integers** $n \geq 0$.

$f_0 = 0 \quad f_1 = 1$

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

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, $a \geq f_{n+1}$.**

r_{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:

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

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

$r_1 \geq 1$

$q_n \geq 1$ $\forall n$ $q_n \geq 1$

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 $n \geq 1$.

<u>Base Case</u>: $n=1$ If Euclid's Algorithm on a, b with $a \geq b > 0$ takes 1 step, then $a = q_1 b$ for some q_1 and $a \geq b \geq 1 = f_2$ and $P(1)$ 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>: We want to show: if $\gcd(a,b)$ **with** $a \geq b>0$ **takes** $k+1$ **steps, then** $a \geq f_{k+2}.$

$k=1$

Now if $k+1 = 2$**, the two steps of Euclid's algorithm on** a **and** b **are given by** $\gcd(a,b)=\gcd(b,c)=\gcd(c,0)=c$ **where**

$$a = q_2 b + c$$

$$b = q_1 c$$

and $c = a \bmod b > 0$

~~**By definition of mod we have** $b > c \geq 0$~~

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

So $a = q_2 b + c \geq b + c \geq 1+1 = 2 = f_3 = f_{k+2}$ 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: We want to show: if $\gcd(a,b)$ **with** $a \geq b>0$ **takes** $k+1$ **steps, then** $a \geq f_{k+2.}$

Next suppose that $k+1 \geq 3$ **so for the first three steps of Euclid's algorithm on** a **and** b **we have** $\gcd(a,b)=\gcd(b,c)=\gcd(c,d)$ **where**

$a = q_{k+1}b + c$

$b = q_k\ c + d$

$c = q_{k-1}d + e$ $\quad$ ($c = a \bmod b$, $d = b \bmod c$, $e = c \bmod d$ and $d>0$)

By definition of mod we have $b > c > d>0$, $\gcd(b,c)$ **takes** k **steps and** $\gcd(c,d)$ **takes** $k-1 \geq 1$ **steps, so by the IH we have** $b \geq f_{k+1}$ **and** $c \geq f_k$.

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

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

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, $a \geq f_{n+1}$.

Informal Recap: 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, $a \geq f_{n+1}$.

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 $\leq$ twice the # of bits in a.

Recursive Definition of Sets

Recursive Definition

- Basis Step: $0 \in S$
- Recursive Step: If $x \in S$, then $x + 2 \in S$
- Exclusion Rule: Every element in S follows from basis steps and a finite number of recursive steps.

Recursive Definitions of Sets

Basis: $6 \in S$, $15 \in S$
Recursive: If $x, y \in S$, then $x+y \in S$

Basis: $[1, 1, 0] \in S$, $[0, 1, 1] \in S$
Recursive: If $[x, y, z] \in S$, then $[\alpha x, \alpha y, \alpha z] \in S$
If $[x_1, y_1, z_1] \in S$ and $[x_2, y_2, z_2] \in S$, then
$[x_1 + x_2, y_1 + y_2, z_1 + z_2] \in S$.

Powers of 3:

Recursive Definitions of Sets

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

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

Basis: $[1, 1, 0] \in S$, $[0, 1, 1] \in S$

Recursive: If $[x, y, z] \in S$, then $[\alpha x, \alpha y, \alpha z] \in S$

If $[x_1, y_1, z_1] \in S$ and $[x_2, y_2, z_2] \in S$, then

$[x_1 + x_2, y_1 + y_2, z_1 + z_2] \in S$.

Powers of 3:

Basis: $1 \in S$

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

Recursive Definitions of Sets: General Form

Recursive definition

- *Basis step:* Some specific elements are in S
- *Recursive step:* Given some existing named elements in S some new objects constructed from these named elements are also in S.
- *Exclusion rule*: Every element in S follows from basis steps and a finite number of recursive steps

Strings

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

- The set Σ^* of *strings* over the alphabet Σ is defined by
  - **Basis:** $\varepsilon \in \Sigma$ (ε is the empty string)
  - **Recursive:** if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$

Palindromes

Palindromes are strings that are the same backwards and forwards

Basis:

ε is a palindrome and any $a \in \Sigma$ is a palindrome

Recursive step:

If p is a palindrome then apa is a palindrome for every $a \in \Sigma$

All Binary Strings with no 1's before 0's

All Binary Strings with no 1's before 0's

Basis:

$\varepsilon \in S$

Recursive:

If $x \in S$, **then** $0x \in S$

If $x \in S$, **then** $x1 \in S$

Function Definitions on Recursively Defined Sets

Length:

$\text{len}(\mathcal{E}) = 0$

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

Reversal:

$\mathcal{E}^R = \mathcal{E}$

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

Concatenation:

$x \bullet \mathcal{E} = x$ for $x \in \Sigma^*$

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