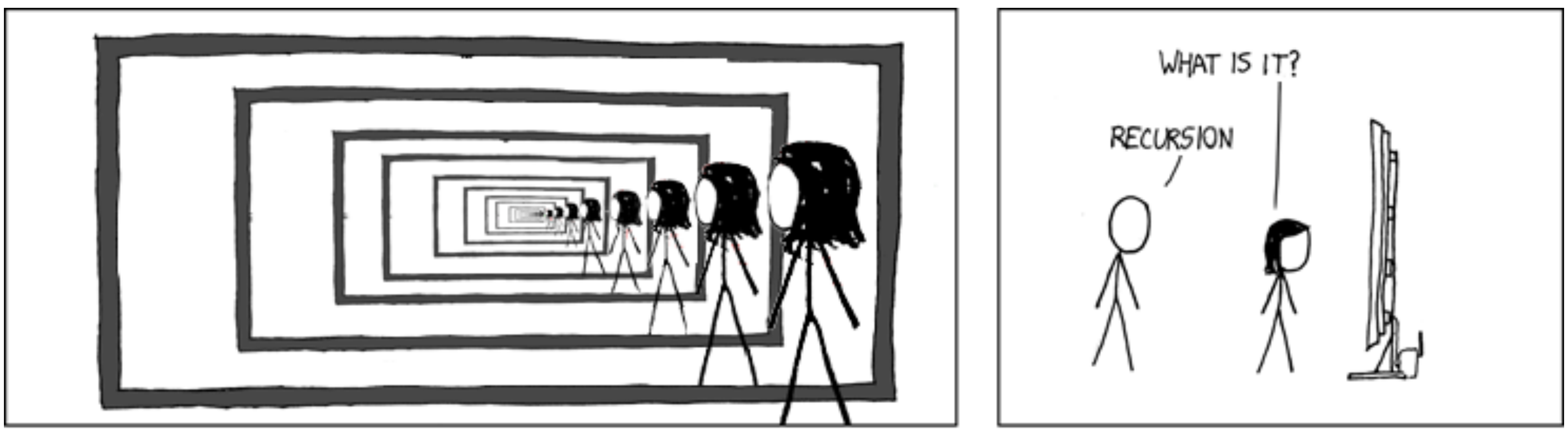
CSE 311: Foundations of Computing

Lecture 17: Recursively Defined Sets & Structural Induction



Midterm

- Wednesday in class
- Covers material up to end of ordinary induction
- Closed book, closed notes
  - will include reference sheets that seem helpful
- No calculators
  - arithmetic is intended to be straightforward

Midterm

- 5 problems covering:
  - Logic / English translation
  - Boolean circuits, algebra, and normal forms
  - Solving modular equations
  - Induction
  - Set theory
  - English proofs

Last time: 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.

Last time: 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** $n \geq 1$**.**

<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.** $1 \leq j \leq k$

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 ≥ b>0 **takes** n **steps** ⟶ a ≥ f_{n+1}" **for all** n ≥ 1**.**

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

Induction Hypothesis: **Suppose that for some integer** k ≥ 1, P(j) **is true for all integers** j **s.t.** 1 ≤ j ≤ k

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

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

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

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 $k \geq 1$, $P(j)$ **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** $r_k \geq f_k$.

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: Goal: 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 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.** ■

Recursive Definitions: Data

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 $x+2 \in S$

Recursive Definition of Sets

Recursive definition of set S

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

We need the exclusion rule because otherwise $S=\mathbb{N}$ **would satisfy the other two parts. However, we won't always write it down on these slides.**

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

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 w/ no chars)
  - **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** $x1 \in S$

Functions on Recursively Defined Sets (on Σ^*)

Length:

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

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

$\varepsilon^R = \varepsilon$

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

Number of c’**s in a string:**

$\#_c(\varepsilon) = 0$

$\#_c(wc) = \#_c(w) + 1$ for $w \in \Sigma^*$

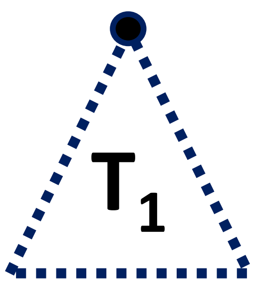
$\#_c(wa) = \#_c(w)$ for $w \in \Sigma^*$, $a \in \Sigma$, $a \neq c$

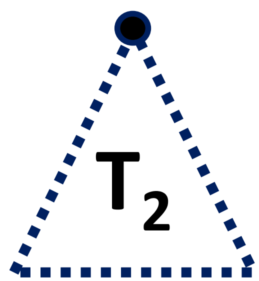
Rooted Binary Trees

- **Basis:** • is a rooted binary tree

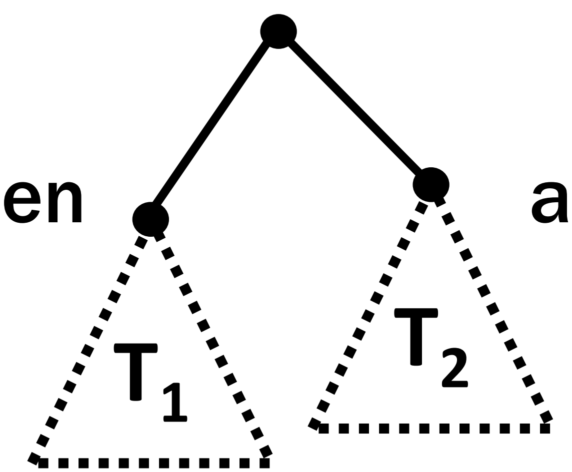
Rooted Binary Trees

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

If 

and 

are rooted binary trees,

then 

also is a rooted binary tree.

Rooted Binary Trees in Java

```
public static class BinaryTree {
  static BinaryTree LEAF = ...;
  public BinaryTree(
      BinaryTree T1, BinaryTree T2) {
    ...
  }
}
```

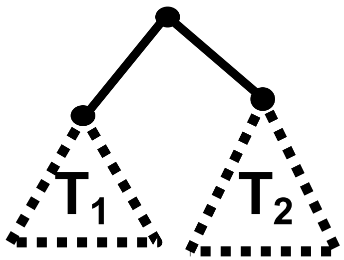
Recursively-defined Sets translate natural into Java classes

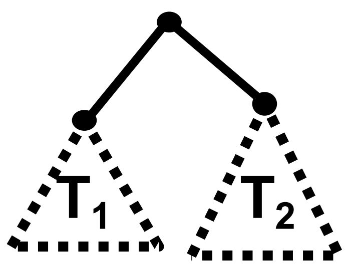
Create a binary tree with

`BinaryTree.LEAF` **or**

`new BinaryTree(T1, T2)`

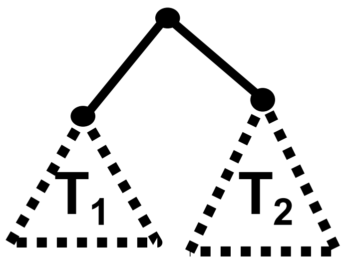
Defining Functions on Rooted Binary Trees

- size(•) = 1
- size (

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

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

Functions on Rooted Binary Trees in Java

- size(•) = 1
- size (

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

```
public int size(BinaryTree T) {
  if (T == BinaryTree.LEAF) {
    return 1;
  } else {
    return 1 + size(T.left()) + size(T.right());
  }
}
```

Recursive Functions translate natural into Java functions

Recursive Sets translate natural into Java classes

Structural Induction

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

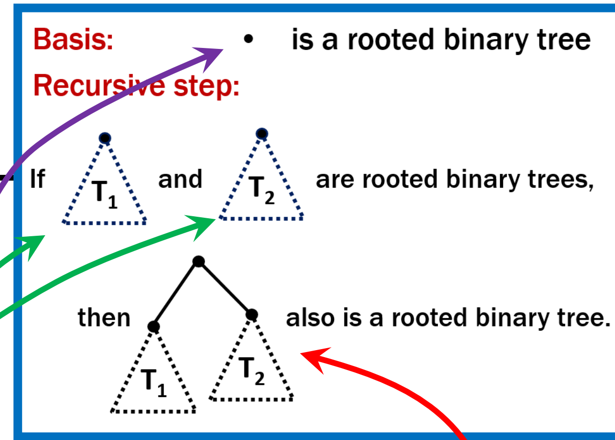
Base Case: Show that $P(u)$ is true for all specific elements u of S mentioned in the *Basis step*

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



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