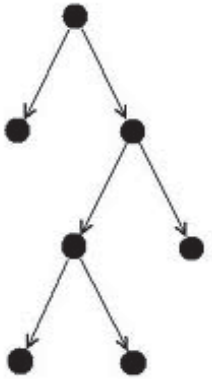
# CompSci 102
# Discrete Math for Computer Science



March 13, 2012

Prof. Rodger

Slides modified from Rosen

## Announcements

- Read for next time Chap. 6.1-6.2
- Recitation on Friday
- Homework 5 out

## Strong Induction

- *Strong Induction*: To prove that $P(n)$ is true for all positive integers $n$, where $P(n)$ is a propositional function, complete two steps:
  - *Basis Step*: Verify that the proposition $P(1)$ is true.
  - *Inductive Step*: Show the conditional statement $[P(1) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ holds for all positive integers $k$.

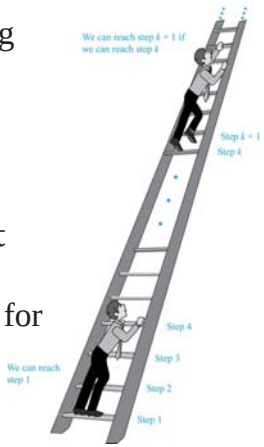
Strong Induction is sometimes called the *second principle of mathematical induction* or *complete induction*.

## Strong Induction and the Infinite Ladder

Strong induction tells us that we can reach all rungs if:
1. We can reach the first rung of the ladder.
2. For every integer $k$, if we can reach the first $k$ rungs, then we can reach the $(k+1)$st rung.

To conclude that we can reach every rung by strong induction:
• BASIS STEP: $P(1)$ holds
• INDUCTIVE STEP: Assume $P(1) \wedge P(2) \wedge \cdots \wedge P(k)$ holds for an arbitrary integer $k$, and show that $P(k+1)$ must also hold.

We will have then shown by strong induction that for every positive integer $n$, $P(n)$ holds, i.e., we can reach the $n$th rung of the ladder.



# Proof using Strong Induction

**Example**: Suppose we can reach the first and second rungs of an infinite ladder, and we know that if we can reach a rung, then we can reach two rungs higher. Prove that we can reach every rung.

(Try this with mathematical induction.)

**Solution**: Prove the result using strong induction.

- BASIS STEP: We can reach the first step.
- INDUCTIVE STEP: The inductive hypothesis is that we can reach the first $k$ rungs, for any $k \geq 2$.
-

◄

# Which Form of Induction Should Be Used?

- Can always use strong induction instead of mathematical induction.
- (if it is simpler use mathematical induction).
- The principles of mathematical induction, strong induction, and the well-ordering property are all equivalent. (*Exercises* 41-43)
- Sometimes it is clear how to proceed using one of the three methods, but not the other two.

Completion of the proof of the Fundamental Theorem of Arithmetic

**Example**: Show that if $n$ is an integer greater than 1, then $n$ can be written as the product of primes.

**Solution:** Let $P(n)$ be the proposition that $n$ can be written as a product of primes.

- BASIS STEP: $P(2)$ is true since 2 itself is prime.
- INDUCTIVE STEP: The inductive hypothesis is $P(j)$ is true for all integers $j$ with $2 \leq j \leq k$.
- To show that $P(k + 1)$ must be true under this assumption, two cases need to be considered:

◄

# Proof using Strong Induction

**Example**: Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

**Solution**: Let $P(n)$ be the proposition that postage of $n$ cents can be formed using 4-cent and 5-cent stamps.

- BASIS STEP: $P(12)$, $P(13)$, $P(14)$, and $P(15)$ hold.
  - $P(12)$ uses
  - $P(13)$ uses
  - $P(14)$ uses
  - $P(15)$ uses
- INDUCTIVE STEP: The inductive hypothesis states that $P(j)$ holds for $12 \leq j \leq k$, where $k \geq 15$. Assuming the inductive hypothesis, it can be shown that $P(k + 1)$ holds.
- Using the inductive hypothesis,
  To form postage of $k + 1$ cents,

Hence, $P(n)$ holds for all $n \geq 12$.

◄

# Proof of Same Example using Mathematical Induction

**Example**: Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

**Solution**: Let $P(n)$ be the proposition that postage of $n$ cents can be formed using 4-cent and 5-cent stamps.

- BASIS STEP: Postage of 12 cents can be formed using
- INDUCTIVE STEP: The inductive hypothesis $P(k)$ for any positive integer $k$ is that postage of $k$ cents can be formed using 4-cent and 5-cent stamps. To show P($k$ + 1) where $k \geq 12$ , we consider two cases:
  - If at least one 4-cent stamp has been used,
  - Otherwise, no 4-cent stamp have been used

◀

Hence, $P(n)$ holds for all $n \geq 12$.

# Well-Ordering Property

- *Well-ordering property*: Every nonempty set of nonnegative integers has a least element.
- The well-ordering property is one of the axioms of the positive integers listed in Appendix 1.
- The well-ordering property can be used directly in proofs, as the next example illustrates.
- The well-ordering property can be generalized.
  - **Definition:** A set is *well ordered if every subset has a least element.*
    - **N** is well ordered under $\leq$.
    - The set of finite strings over an alphabet using lexicographic ordering is well ordered.
  - We will see a generalization of induction to sets other than the integers.

# Well-Ordering Property

**Example**: Use the well-ordering property to prove the division algorithm, which states that if $a$ is an integer and $d$ is a positive integer, then there are unique integers $q$ and $r$ with $0 \leq r < d$, such that $a = dq + r$.

**Solution**: Let $S$ be the set of nonnegative integers of the form $a - dq$, where $q$ is an integer. The set is nonempty since $-dq$ can be made as large as needed.

- By the well-ordering property, S has a least element $r = a - dq_0$. The integer $r$ is nonnegative. It also must be the case that $r < d$. If it were not, then there would be a smaller nonnegative element in S, namely, $a - d(q_0 + 1) = a - dq_0 - d = r - d > 0$.
- Therefore, there are integers $q$ and $r$ with $0 \leq r < d$.

(*uniqueness of q and r is Exercise* 37) ◀

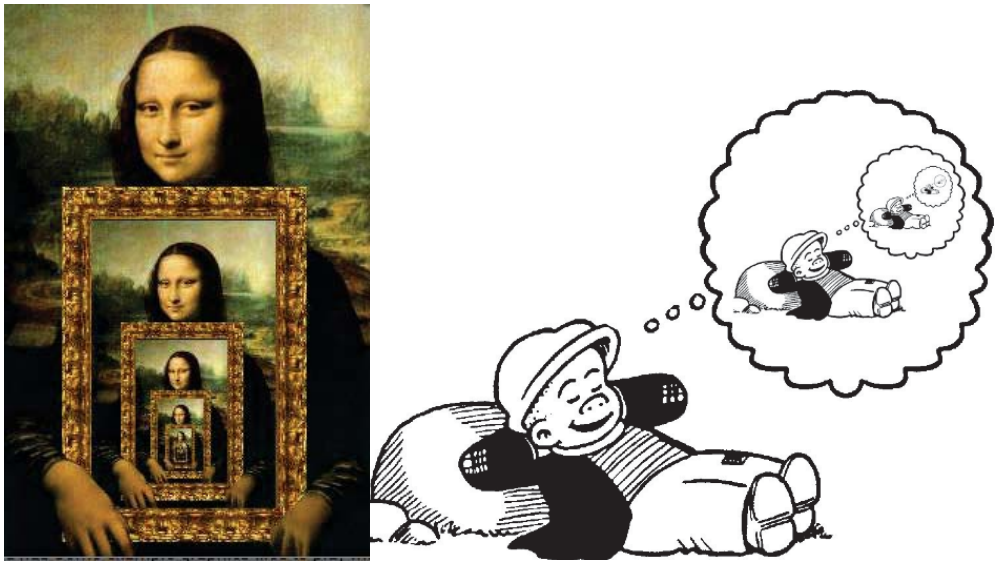
# Sec 5.3 - Recursively Defined Functions

**Definition**: A *recursive* or *inductive definition* of a function consists of two steps.

- BASIS STEP: Specify the value of the function at zero.
- RECURSIVE STEP: Give a rule for finding its value at an integer from its values at smaller integers.

- A function $f(n)$ is the same as a sequence $a_0$, $a_1$, … , where $a_i$, where $f(i) = a_i$. This was done using recurrence relations in Section 2.4.

# Recursively Defined Pictures



# Recursively Defined Functions

**Example**: Suppose $f$ is defined by:

$f(0) = 3$,

$f(n + 1) = 2f(n) + 3$

Find $f(1)$, $f(2)$, $f(3)$, $f(4)$

**Solution**:

- $f(1) =$
- $f(2) =$
- $f(3) =$
- $f(4) =$

**Example:** Give a recursive definition of the factorial function $n!$:

**Solution**:

# Recursively Defined Functions

**Example**: Give a recursive definition of:

$$\sum_{k=0}^{n} a_k.$$

**Solution**: The first part of the definition is

The second part is

# Fibonacci Numbers

Fibonacci
(1170- 1250)

**Example** : The Fibonacci numbers are defined as follows:

$f_0 = 0$

$f_1 = 1$

$f_n = f_{n-1} + f_{n-2}$

Find $f_2, f_3, f_4, f_5$.

- $f_2 = f_1 + f_0 = 1 + 0 = 1$
- $f_3 = f_2 + f_1 = 1 + 1 = 2$
- $f_4 = f_3 + f_2 = 2 + 1 = 3$
- $f_5 = f_4 + f_3 = 3 + 2 = 5$

# Fibonacci Numbers

**Example 4**:
Show that whenever $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = (1+\sqrt{5})/2$.

**Solution**: Let $P(n)$ be the statement $f_n > \alpha^{n-2}$. Use strong induction to show that $P(n)$ is true whenever $n \geq 3$.

- BASIS STEP: $P(3)$ holds since $\alpha < 2 = f_3$
  $P(4)$ holds since $\alpha^2 = (3+\sqrt{5})/2 < 3 = f_4$.
- INDUCTIVE STEP: Assume that $P(j)$ holds, i.e., $f_j > \alpha^{j-2}$ for all integers $j$ with
  $3 \leq j \leq k$, where $k \geq 4$. Show that $P(k+1)$ holds, i.e., $f_{k+1} > \alpha^{k-1}$.
  - Since $\alpha^2 = \alpha + 1$ (because $\alpha$ is a solution of $x^2 - x - 1 = 0$),
  - By the inductive hypothesis, because $k \geq 4$ we have
  - Therefore, it follows that
  - Hence, $P(k+1)$ is true.

# Lamé's Theorem

Gabriel Lamé
(1795-1870)

**Lamé's Theorem**: Let $a$ and $b$ be positive integers with $a \geq b$. Then the number of divisions used by the Euclidian algorithm to find gcd($a$,$b$) is less than or equal to five times the number of decimal digits in $b$.

**Proof**: *in book.*

- As a consequence of Lamé's Theorem, $O(\log b)$ divisions are used by the Euclidian algorithm to find gcd($a$,$b$) whenever $a > b$.

Lamé's Theorem was the first result in computational complexity

# Recursively Defined Sets and Structures

*Recursive definitions* of sets have two parts:
- The *basis step* specifies an initial collection of elements.
- The *recursive step* gives the rules for forming new elements in the set from those already known to be in the set.

- Sometimes the recursive definition has an *exclusion rule*, which specifies that the set contains nothing other than those elements specified in the basis step and generated by applications of the rules in the recursive step.
- We will always assume that the exclusion rule holds, even if it is not explicitly mentioned.
- We will later develop a form of induction, called *structural induction*, to prove results about recursively defined sets.

# Recursively Defined Sets and Structures

**Example** : Subset of Integers $S$:
BASIS STEP: $3 \in S$.
RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in $S$.

- Initially 3 is in $S$, then 3 + 3 = 6, then 3 + 6 = 9, etc.

**Example**: The natural numbers **N**.
BASIS STEP: $0 \in \mathbf{N}$.
RECURSIVE STEP: If $n$ is in **N**, then $n + 1$ is in **N**.

- Initially 0 is in $S$, then 0 + 1 = 1, then 1 + 1 = 2, etc.

# Strings

**Definition**: The set $\Sigma^*$ of *strings* over the alphabet $\Sigma$:

BASIS STEP: $\lambda \in \Sigma^*$ ($\lambda$ is the empty string)

RECURSIVE STEP: If $w$ is in $\Sigma^*$ and $x$ is in $\Sigma$, then $wx \in \Sigma^*$.

**Example**: If $\Sigma = \{0,1\}$, the strings in in $\Sigma^*$ are the set of all bit strings, $\lambda$,0,1, 00,01,10, 11, etc.

**Example**: If $\Sigma = \{a,b\}$, show that *aab* is in $\Sigma^*$.

- Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, $a \in \Sigma^*$.
- Since $a \in \Sigma^*$ and $a \in \Sigma$, $aa \in \Sigma^*$.
- Since $aa \in \Sigma^*$ and $b \in \Sigma$, $aab \in \Sigma^*$.

# String Concatenation

**Definition**: Two strings can be combined via the operation of *concatenation*. Let $\Sigma$ be a set of symbols and $\Sigma^*$ be the set of strings formed from the symbols in $\Sigma$. We can define the concatenation of two strings, denoted by $\cdot$, recursively as follows.

BASIS STEP: If $w \in \Sigma^*$, then $w \cdot \lambda = w$.

RECURSIVE STEP: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w \cdot (w_2 x) = (w_1 \cdot w_2)x$.

- Often $w_1 \cdot w_2$ is written as $w_1 w_2$.
- If $w_1 = abra$ and $w_2 = cadabra$, the concatenation $w_1 w_2 = abracadabra$.

# Length of a String

**Example**: Give a recursive definition of $l(w)$, the length of the string $w$.

**Solution**: The length of a string can be recursively defined by:

$l(w) = 0$;

$l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$.

# Balanced Parentheses

**Example**: Give a recursive definition of the set of balanced parentheses $P$.

**Solution**:

BASIS STEP: $() \in P$

RECURSIVE STEP: If $w \in P$, then $() w \in P$, $(w) \in P$ and $w () \in P$.

- Show that (() ()) is in $P$.
- Why is ))(() not in $P$?

# Well-Formed Formulae in Propositional Logic

**Definition**: The set of *well-formed formulae* in propositional logic involving **T**, **F**, propositional variables, and operators from the set $\{\neg,\wedge,\vee,\rightarrow,\leftrightarrow\}$.

BASIS STEP: **T**,**F**, and *s*, where *s* is a propositional variable, are well-formed formulae.

RECURSIVE STEP: If *E* and *F* are well formed formulae, then $(\neg E)$, $(E \wedge F)$, $(E \vee F)$, $(E \rightarrow F)$, $(E \leftrightarrow F)$, are well-formed formulae.

**Examples**: $((p \vee q) \rightarrow (q \wedge \mathbf{F}))$ is a well-formed formula.

$pq \wedge$ is not a well formed formula.

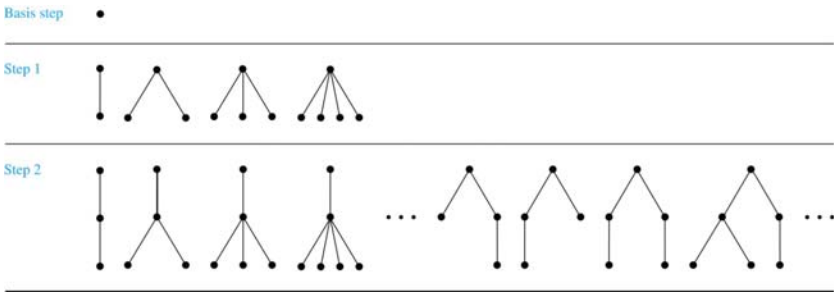
# Rooted Trees

**Definition**: The set of *rooted trees*, where a rooted tree consists of a set of vertices containing a distinguished vertex called the *root*, and edges connecting these vertices, can be defined recursively by these steps:

BASIS STEP: A single vertex *r* is a rooted tree.

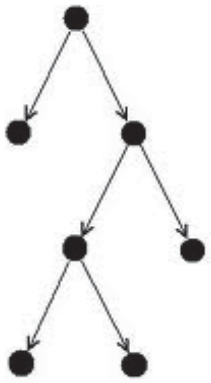
RECURSIVE STEP: Suppose that $T_1, T_2, \ldots, T_n$ are disjoint rooted trees with roots $r_1, r_2, \ldots, r_n$, respectively. Then the graph formed by starting with a root *r*, which is not in any of the rooted trees $T_1, T_2, \ldots, T_n$, and adding an edge from *r* to each of the vertices $r_1, r_2, \ldots, r_n$, is also a rooted tree.

# Building Up Rooted Trees




- Trees are studied extensively in Chapter 11.
- Next we look at a special type of tree, the full binary tree.

# How do you construct this rooted tree recursively?



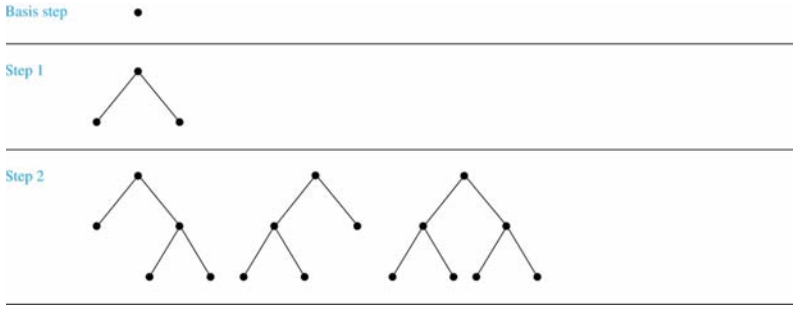
# Full Binary Trees

**Definition:** The set of *full binary trees* can be defined recursively by these steps.

BASIS STEP: There is a full binary tree consisting of only a single vertex $r$.

RECURSIVE STEP: If $T_1$ and $T_2$ are disjoint full binary trees, there is a full binary tree, denoted by $T_1 \cdot T_2$, consisting of a root $r$ together with edges connecting the root to each of the roots of the left subtree $T_1$ and the right subtree $T_2$.

# Building Up Full Binary Trees




What can you say about the nonleaf nodes in a full binary tree?

# Induction and Recursively Defined Sets

**Example**: Show that the set S defined by specifying that $3 \in S$ and that if $x \in S$ and $y \in S$, then $x + y$ is in $S$, is the set of all positive integers that are multiples of 3.

**Solution**: Let $A$ be the set of all positive integers divisible by 3. To prove that $A = S$, show that $A$ is a subset of $S$ and $S$ is a subset of $A$.

- $A \subset S$: Let P($n$) be the statement that $3n$ belongs to $S$.
  BASIS STEP:
  INDUCTIVE STEP: Assume $P(k)$ is true.
  By the second part of the recursive definition,

  Hence, $P(k + 1)$ is true.
- $S \subset A$:
  BASIS STEP by the first part of recursive definition, and
  INDUCTIVE STEP: The second part of the recursive definition

  By part (i) of Theorem 1 of Section 4.1, it follows that

# Structural Induction

We used mathematical induction to prove a result about a recursively defined set. Next we study a more direct form induction for proving results about recursively defined sets.

**Definition**: To prove a property of the elements of a recursively defined set, we use *structural induction*.

BASIS STEP: Show that the result holds for all elements specified in the basis step of the recursive definition.

RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

- The validity of structural induction can be shown to follow from the principle of mathematical induction.

# Full Binary Trees

**Definition**: The *height h(T)* of a full binary tree $T$ is defined recursively as follows:

- BASIS STEP: The height of a full binary tree $T$ consisting of only a root $r$ is
- RECURSIVE STEP: If $T_1$ and $T_2$ are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has height

- The number of vertices $n(T)$ of a full binary tree $T$ satisfies the following recursive formula:
  - **BASIS STEP**: The number of vertices of a full binary tree $T$ consisting of only a root $r$ is
  - **RECURSIVE STEP**: If $T_1$ and $T_2$ are full binary trees, then the full binary tree $T = T_1 \cdot T_2$ has the number of vertices

# Structural Induction and Binary Trees

**Theorem**: If $T$ is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

**Proof**: Use structural induction.

- BASIS STEP: The result holds for a full binary tree consisting only of a root, $n(T) = 1$ and $h(T) = 0$.

  Hence,
- RECURSIVE STEP: Assume $n(T_1) \leq 2^{h(T1)+1} - 1$ and also $n(T_2) \leq 2^{h(T2)+1} - 1$ whenever $T_1$ and $T_2$ are full binary trees.

$n(T) = 1 + n(T_1) + n(T_2)$ (*by recursive formula of n(T)*)

# Generalized Induction

- *Generalized induction* is used to prove results about sets other than the integers that have the well-ordering property. (*explored in more detail in Chapter 9*)
- For example, consider an ordering on $\mathbf{N} \times \mathbf{N}$, ordered pairs of nonnegative integers. Specify that $(x_1, y_1)$ is less than or equal to $(x_2, y_2)$ if either $x_1 < x_2$, or $x_1 = x_2$ and $y_1 \leq y_2$ . This is called the *lexicographic ordering*.
- Strings are also commonly ordered by a *lexicographic ordering*.
- The next example uses generalized induction to prove a result about ordered pairs from $\mathbf{N} \times \mathbf{N}$.

# Generalized Induction

**Example**: Suppose that $a_{m,n}$ is defined for $(m,n) \in \mathbf{N} \times \mathbf{N}$ by $a_{0,0} = 0$ and

$$a_{m,n} = \begin{cases} a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\ a_{m,n-1} + n & \text{if } n > 0 \end{cases}$$

Show that $a_{m,n} = m + n(n+1)/2$ is defined for all $(m,n) \in \mathbf{N} \times \mathbf{N}$.

**Solution**: Use generalized induction.

BASIS STEP:

INDUCTIVE STEP: Assume that $a_{m',n'} = m' + n'(n'+1)/2$ whenever $(m',n')$ is less than $(m,n)$ in the lexicographic ordering of $\mathbf{N} \times \mathbf{N}$.

- If $n = 0$, by the inductive hypothesis we can conclude
- If $n > 0$, by the inductive hypothesis we can conclude