Recursive Definitions and Structural Induction

Section 5.3

Section Summary

- Recursively Defined Functions
- Recursively Defined Sets and Structures
- Structural Induction

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 $a_0, a_1, \ldots$ where $f(i) = a_i$.

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:

- $f(1) = 2 \cdot f(0) + 3 = 2 \cdot 3 + 3 = 9$
- $f(2) = 2 \cdot f(1) + 3 = 2 \cdot 9 + 3 = 21$
- $f(3) = 2 \cdot f(2) + 3 = 2 \cdot 21 + 3 = 45$
- $f(4) = 2 \cdot f(3) + 3 = 2 \cdot 45 + 3 = 93$

Recursively Defined Functions

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

Solution:

$f(0) = 1$

$f(n + 1) = (n + 1) \cdot f(n)$

Recursively Defined Functions

Example: Give a recursive definition of $\sum_{k=0}^{n} a_k$.

Solution: The first part of the definition is

$$\sum_{k=0}^{0} a_k = a_0.$$

The second part is

$$\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^{n} a_k\right) + a_{n+1}.$$

Fibonacci
(1170- 1250)

Fibonacci Numbers

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., $f_{k+1} > \alpha^{k-1}$.

continued →

Fibonacci Numbers

- INDUCTIVE STEP: Assume that $P(j)$ holds, i.e., $f_j > \alpha^{j-2}$ for all integers j with $3 \le j \le k$, where $k \ge 4$. Show that $P(k+1)$ holds, i.e., $f_{k+1} > \alpha^{k-1}$.

$$\begin{aligned}
f_{k+1} &= f_k + f_{k-1} && \text{(by definition)} \\
&> \alpha^{k-2} + \alpha^{k-3} && \text{(by induction hypothesis)} \\
&= \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} && \\
&= (\alpha+1) \cdot \alpha^{k-3} && \text{(factor out } \alpha^{k-3}\text{)} \\
&= \alpha^2 \cdot \alpha^{k-3} && (\alpha^2 = \alpha+1\text{, from last slide)} \\
&= \alpha^{k-1}
\end{aligned}$$

- Hence, $P(k+1)$ is true. ◀

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.

- The ***exclusion rule*** 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 this is true, even if 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 S of the set of Integers :

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 $3 + 6 = 9$, then $3 + 9 = 12$, then $3 + 12 = 15$, 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 $1 + 1 = 2$, then $2 + 1 = 3$, then $3 + 1 = 4$, etc.

Strings

Definition: The set Σ^* of *strings* over the alphabet Σ:

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

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

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

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

- 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 Σ be a set of symbols and Σ* be the set of strings formed from the symbols in Σ. We can define the concatenation of two strings, denoted by ·, 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 $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

- Often $w_1 \cdot w_2$ is written as $w_1 w_2$.
- **Example:** 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:

BASIS STEP: $l(\lambda) = 0$;

RECURSIVE STEP: $l(wx) = l(w) + 1$ 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: () ∈ P

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

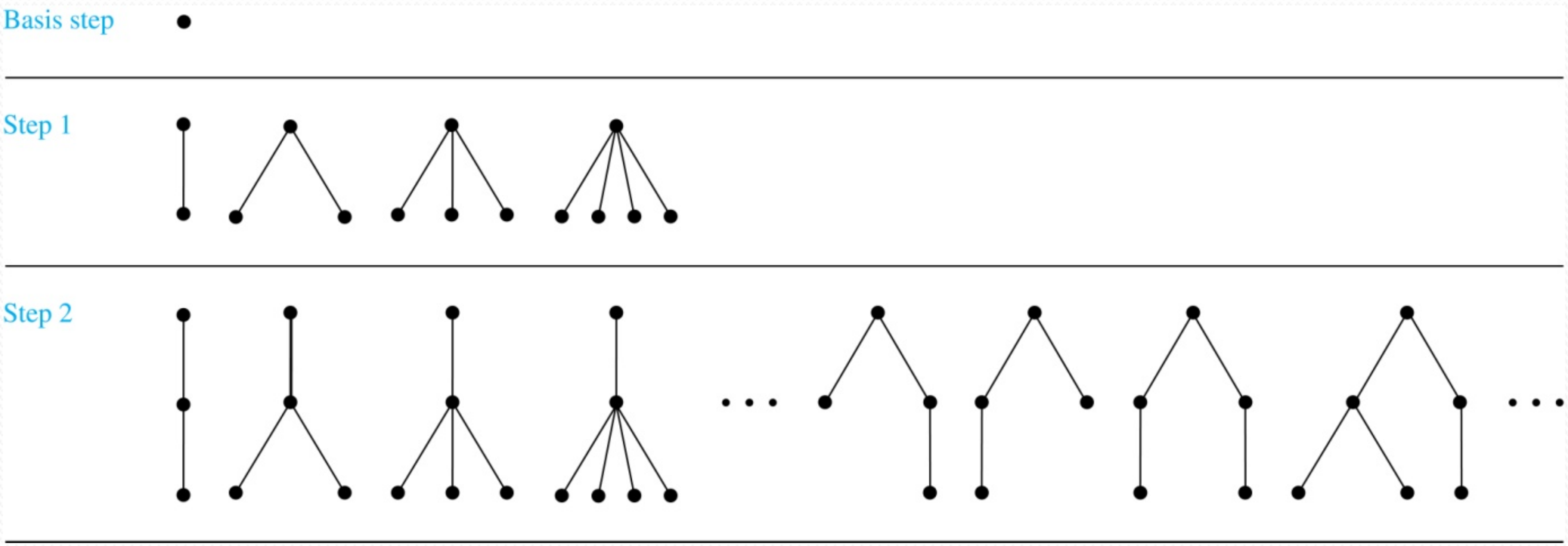
Rooted Trees

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, …,T_n are disjoint rooted trees with roots r_1, r_2,…,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, …,T_n, and adding an edge from r to each of the vertices r_1, r_2,…,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.

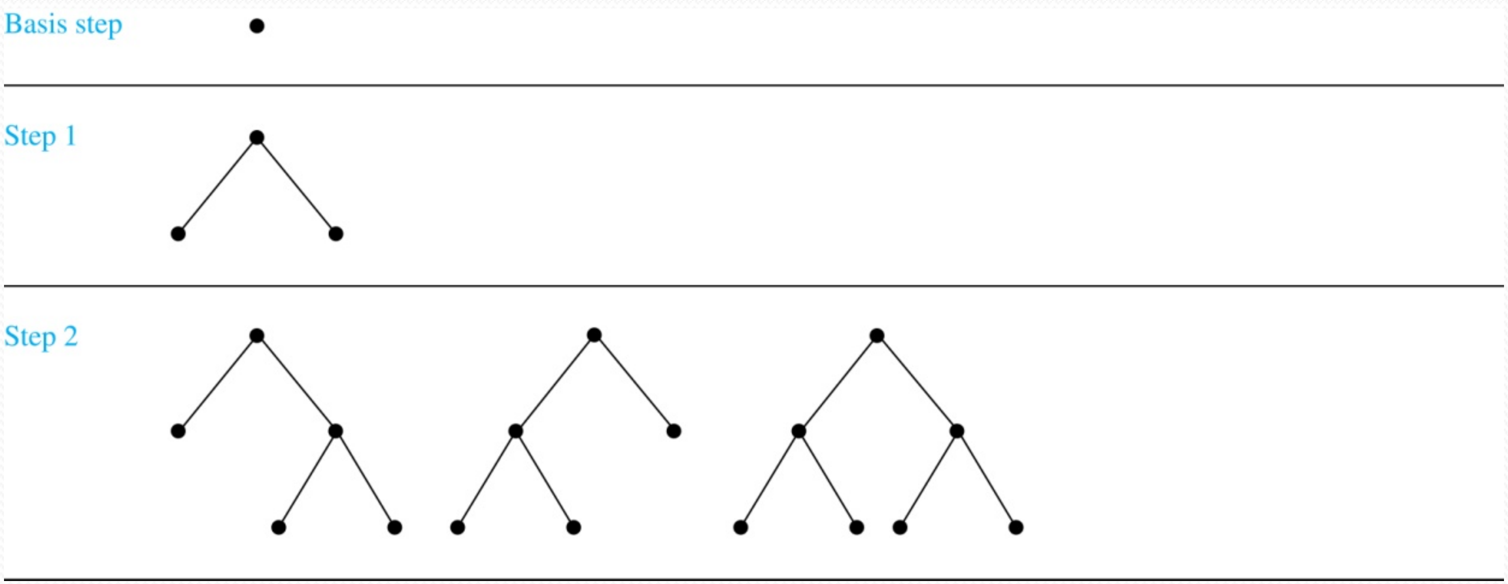
Full Binary Trees

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



Structural Induction

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 $h(T) = 0$.
- 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 $h(T) = 1 + \max(h(T_1), h(T_2))$.

- 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 $n(T) = 1$.
  - 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
$$n(T) = 1 + n(T_1) + n(T_2).$$

Structural Induction and Binary Trees

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

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, $n(T) = 1 \le 2^{0+1} - 1 = 1$.
- RECURSIVE STEP: Assume $n(T_1) \le 2^{h(T_1)+1} - 1$ and also $n(T_2) \le 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees.

$$
\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \textit{(by recursive formula of } n(T)\textit{)} \\
&\le 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) && \textit{(by inductive hypothesis)} \\
&\le 2\cdot\max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 \\
&= 2\cdot 2^{\max(h(T_1),h(T_2))+1} - 1 && (\max(2^x, 2^y) = 2^{\max(x,y)}) \\
&= 2\cdot 2^{h(T)} - 1 && \textit{(by recursive definition of } h(T)\textit{)} \\
&= 2^{h(T)+1} - 1
\end{aligned}
$$