Discrete Mathematics

Review - Chapter 5, Induction and recursion

Part 2

Reference book: Discrete Mathematics and Its Applications 8th ed. (Rosen, 2019)

Recursively Defined Functions[1]

A *recursive* or *inductive definition* of a function

- 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.

Examples

- $f(0) = 3, f(n+1) = 2f(n) + 3$
- $\sum_{k=0}^{n+1} a_k = \left(\sum_{k=0}^{n} a_k\right) + a_{n+1}$.
- Fibonacci Numbers: $f_n = f_{n-1} + f_{n-2}$, $f_0 = 0, f_1 = 1$

Recursively Defined Sets and Structures 1

Recursive definitions of sets:

- 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.

Examples

- Subset of Integers S = {3,6,9,12,15, …}
  - BASIS STEP: $3 \in S$
  - RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in S.
- The Natural numbers N = {0, 1, 2, 3, 4, …}
  - BASIS STEP: $0 \in N$.
  - RECURSIVE STEP: If n is in N, then $n + 1$ is in N.

Recursively Defined Sets and Structures 1

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 $wx \in \Sigma^*$.

String Concatenation · *string concatenation operator*

- 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 $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

Length of a String

- BASIS STEP: $l(\lambda) = 0$.
- RECURSIVE STEP: $l(wx) = l(w)+1$ if $w \in \Sigma^*$ and $x \in \Sigma$.

Recursively Defined Sets and Structures 1

Balanced Parentheses

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

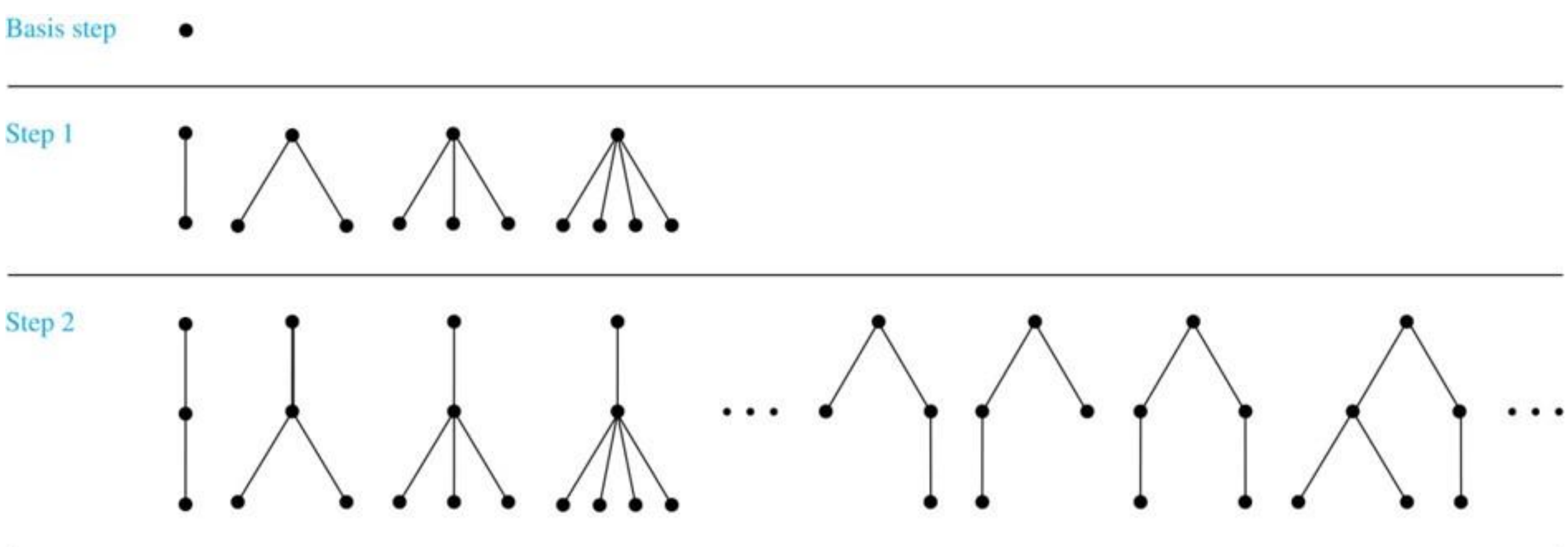
The set of *well-formed formulae* in propositional logic involving **T**, **F**, propositional variables, and operators from the set $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$

- 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.

Rooted Trees

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

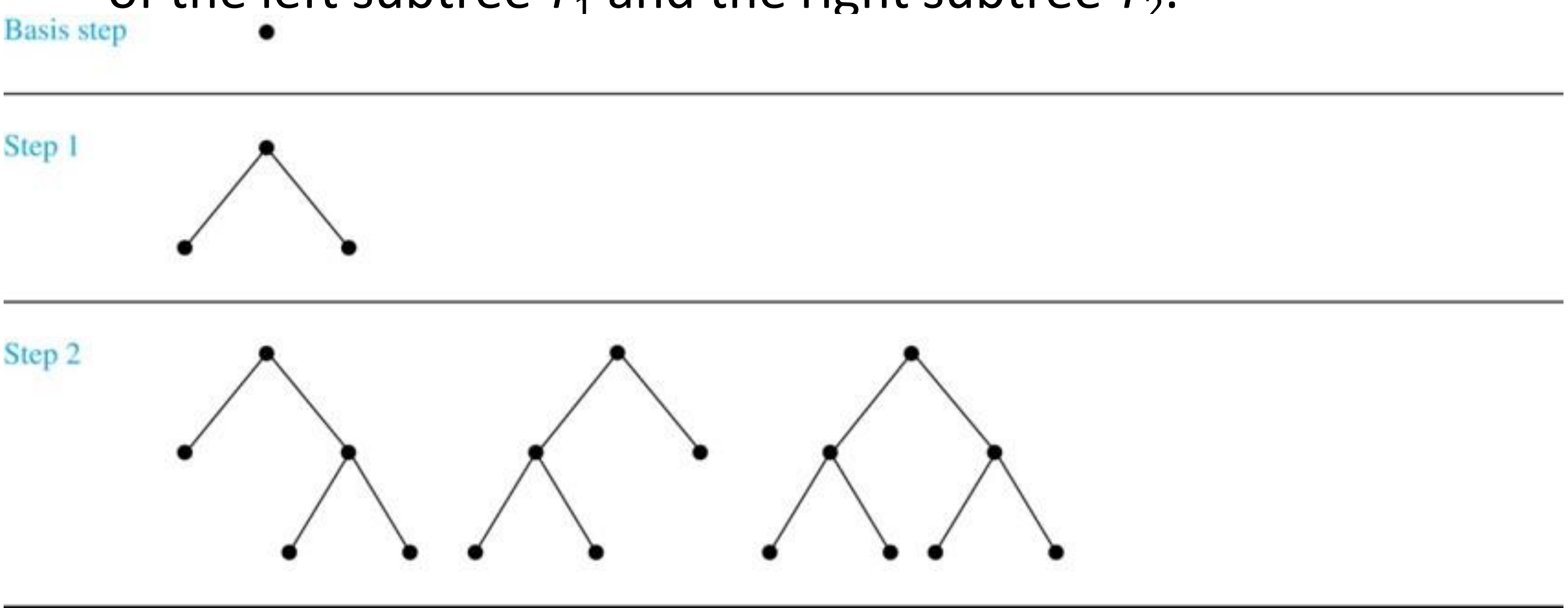
RECURSIVE STEP: Suppose that $T_1, T_2, \ldots, T_n$ are disjoint rooted trees with roots $r_1, r_2, \ldots, r_n$, respectively. Then the graph formed by starting with a root r, which is not in any of the rooted trees $T_1, T_2, \ldots, T_n$, and adding an edge from r to each of the vertices $r_1, r_2,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



Full Binary Trees [1]

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.



Structural Induction

structural induction : To prove a property of the elements of a recursively defined set.

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, then the result holds for these new elements. *ex: P(k) -> P(k+1)*

Examples (*full binary trees T_1, T_2 and $T=T_1 \cdot T_2$*)

- Height $h(T) = 1 + \max\big(h(T_1), h(T_2)\big)$.
- Nodes $n(T) = 1 + n(T_1) + n(T_2)$.
- If T is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.
  - BASIS STEP: It holds for T consisting only of a root
  - RECURSIVE STEP: Assume $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$ show $n(T) \leq 2^{h(T)+1} - 1$. for $T = T_1 \cdot T_2$

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

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

Exercise

Inductive definition (= recursive definition)

Provide the inductive definition for the following,

1) The set of odd numbers

2) The set of powers of 3

Exercise

Structural Induction

Prove that $\mathrm{n}(T) \geq 2h(T) + 1$ for a full binary tree T.