Recursive Definitions and Structural Induction

Niloufar Shafiei

Recursion

Sometimes it is difficult to define an object explicitly.

It may be easy to define this object in terms of itself.

This process is called **recursion**.

Recursion

We can use recursion to define sequences, functions, and sets.

Example:

$a_n=2^n$ for $n = 0,1,2,\ldots$

$1,2,4,8,16,32,\ldots$

After giving the first term, each term of the sequence can be defined from the previous term.

$a_1=1$ $\qquad a_{n+1} = 2a_n$

Recursion

When a sequence is defined recursively, mathematical induction can be used to prove results about the sequence.

Let P(k) be proposition about a_k.

Basis step:

- Verify P(1).

Inductive step:

- Show $\forall k \geq 1 \ (P(k) \rightarrow P(k+1))$.

Recursively defined functions

Assume f is a function with the set of nonnegative integers as its domain

We use two steps to define f.

Basis step:

- Specify the value of f(0).

Recursive step:

- Give a rule for f(x) using f(y) where $0 \le y < x$.

Such a definition is called a **recursive** or **inductive definition**.

Example

Suppose

- $f(0) = 3$
- $f(n+1) = 2f(n)+2, \forall n \geq 0.$

Find f(1), f(2) and f(3).

Solution:

$f(1) =$

$$2f(0) + 2 = 2(3) + 2 = 8$$

$f(2) =$

$$2f(1) + 2 = 2(8) + 2 = 18$$

$f(3) =$

$$2f(2) + 2 = 2(18) + 2 = 38$$

Example

Give an inductive definition of the factorial function $F(n) = n!$.

Solution:

- Basis step: (Find $F(0)$.)

  $F(0)=1$

- Recursive step: (Find a recursive formula for $F(n+1)$.)

  $F(n+1) = (n+1)\ F(n)$

- What is the value of $F(5)$?

  $F(5) = 5F(4)$

  $= 5 . 4F(3)$

  $= 5 . 4 . 3F(2)$

  $= 5 . 4 . 3 . 2F(1)$

  $= 5 . 4 . 3 . 2 . 1F(0)$

  $= 5 . 4 . 3 . 2 . 1 . 1 = 120$

Recursive functions

Recursively defined functions should be **well defined.**

It means for every positive integer, the value of the function at this integer is determined in an unambiguous way.

Example

Assume a is a nonzero real number and n is a nonnegative integer.

Give a recursive definition of a^n.

Solution:

- Basis step: (Find F(0).)

  $F(0) = a^0 = 1$

- Recursive step: (Find a recursive formula for F(n+1).)

  $F(n+1) = a \cdot a^n = a \cdot F(n)$

Example

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

Solution:

- Basis step: (Find F(0).)

$$F(0) = \sum_{K=0}^{0} a_k = a_0$$

- Recursive step: (Find a recursive formula for F(n+1).)

$$F(n+1) = F(n) + a_{n+1}$$

Recursive functions

In some recursive functions,

- The values of the function at the first k positive integers are specified
- A rule is given to determine the value of the function at larger integer from its values at some of the preceding k integers.

Example:

$f(0) = 2$ and $f(1) = 3$

$f(n+2) = 2f(n) + f(n+1) + 5, \ \forall \ n \geq 0.$

Fibonacci numbers

The Fibonacci numbers, f_0, f_1, f_2, …, are defined by the equations

- $f_0 = 0$
- $f_1 = 1$
- $f_n = f_{n-1} + f_{n-2}$

for $n = 2,3,4,\ldots$.

Example

Find the Fibonacci number f_4.

Solution:

$f_4 = f_3 + f_2$

$f_2 = f_0 + f_1 = 0 + 1 = 1$

$f_3 = f_1 + f_2 = 1 + 1 = 2$

$f_4 = f_3 + f_2 = 1 + 2 = 3$

Example

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

Proof by strong induction:

- Find P(n)

  P(n) is $f_n > \alpha^{n-2}$.

- Basis step: (Verify P(3) and P(4) are true.)

  $f_3 > \alpha^1$

  $2 > (1 + \sqrt{5}) / 2$

  $f_4 > \alpha^2$

  $3 > (1 + \sqrt{5})^2 / 4$

Example

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

Proof by strong induction:

- Inductive step: (Show $\forall k\ ([P(3) \wedge P(4) \wedge \ldots \wedge P(k)] \rightarrow P(k+1))$ is true.)
  - Inductive hypothesis:
    $f_j > \alpha^{j-2}$ when $3 \leq j \leq k$.
  - Show $\forall\ k \geq 4$ $P(k+1)$ is true. (Show $f_{k+1} > \alpha^{k-1}$ is true.)
    - Let $k \geq 4$.

    $f_{k+1} = f_k + f_{k-1}$

    By induction hypothesis, $f_k > \alpha^{k-2}$ and $f_{k-1} > \alpha^{k-3}$.

    $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha \cdot \alpha^{k-3} + \alpha^{k-3} = (\alpha+1) \cdot \alpha^{k-3}$

    $= \alpha^2 \cdot \alpha^{k-3} = \alpha^{k-1}$

We showed $P(k+1)$ is true, so by strong induction $f_n > \alpha^{n-2}$ is true.

Recursively defined sets and structures

Assume S is a set.

We use two steps to define the elements of S.

Basis step:

- Specify an initial collection of elements.

Recursive step:

- Give a rule for forming new elements from those already known to be in S.

Example

Consider $S \subseteq \mathbf{Z}$ defined by

Basis step: (Specify initial elements.)

$3 \in S$

Recursive step: (Give a rule using existing elements)

If $x \in S$ and $y \in S$, then $x+y \in S$.

$3 \in S$

$3 + 3 = 6 \in S$

$6 + 3 = 9 \in S$

$6 + 6 = 12 \in S$

…

Example

Show that the set S defined in previous slide, 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.
- We want to show that A=S
- Part 1: (Show $A \subseteq S$ using mathematical induction.)
  - Show $\forall x \, (x \in A \rightarrow x \in S)$.
  - Define P(n).
    P(n) is "$3n \in S$".
  - Basis step: (Show P(1).)
    P(1) is "$3 \in S$".
    By recursive definition of S, $3 \in S$, so P(1) is true.

Example

Show that the set S defined in previous slide, is the set of all positive integers that are multiples of 3.

Solution:

- Part 1: (Show $A \subseteq S$ using mathematical induction.)
  - Inductive step: (Show $\forall k \geq 1\ P(k) \rightarrow P(k+1)$.)
    - Define inductive hypothesis:

      P(k) is "$3k \in S$".
    - Show $\forall k \geq 1\ P(k+1)$ is true.

      P(k+1) is "$3(k+1) \in S$".

      $3(k+1) = 3k + 3$

    By recursive definition of S, since $3 \in S$ and $3k \in S$, $(3k+3) \in S$.

  By mathematical induction, $\forall n \geq 1\ 3n \in S$.

Example

Show that the set S defined in previous slide, is the set of all positive integers that are multiples of 3.

Solution:

- Part 2: (Show $S \subseteq A$.)
  - Show $\forall x\ (x \in S \rightarrow x \in A)$
    - By basis step, $3 \in S$.

      Since 3 is positive multiple of by 3, $3 \in A$.
    - By recursive step, If $x \in S$ and $y \in S$, then $x+y \in S$.
      - Show If $x \in A$ and $y \in A$, then $x+y \in A$.
      - Assume $x \in A$ and $y \in A$, so x and y are positive multiples of 3.
      - So, x+y is positive multiple of 3 and $x+y \in A$.

So, S=A.

Set of strings

Finite sequences of form $a_1, a_2, \ldots, a_n$ are called strings.

The **set Σ^* of strings** over the alphabet Σ can be defined by

Basic step: $\lambda \in \Sigma^*$

(λ is the empty string containing no symbols.)

Recursive step:

If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma$.

Example:

$\Sigma=\{0,1\}$

$\lambda \in \Sigma^*$

$\lambda 0 = 0 \in \Sigma^*$	$\lambda 1 = 1 \in \Sigma^*$
$01 \in \Sigma^*$	$11 \in \Sigma^*$
$010 \in \Sigma^*$	$110 \in \Sigma^*$

Concatenation

Let Σ be a set of symbols and Σ^* be a set of strings formed from symbols in Σ.

The concatenation of two strings, denoted by ., recursively as follows.

Basic step:

If $w \in \Sigma^*$, then $w.\lambda = w$.

Recursive step:

If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w_1 .(w_2x) = (w_1 . w_2) x$.

Example:

w_1 = abc w_2 = def

$w_1 . w_2 = w_1w_2$ = abcdef

Example

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

Solution:

Basis step:

$$l(\lambda) = 0$$

Recursive step:

$$l(wx) =$$

$$l(w) + 1, \text{ where } w \in \Sigma^* \text{ and } x \in \Sigma.$$

Example:

$l(ab) = l(a) + 1$

$= l(\lambda) + 1 + 1$

$= 0 + 1 + 1 = 2$

Structural induction

Instead of mathematical induction to prove a result about a recursively defined sets, we can used more convenient form of induction known as **structural induction**.

Structural induction

Assume we have recursive definition for the set S.

Let $n \in S$.

Show P(n) is true using **structural induction**:

Basis step:

- Assume j is an element specified in the basis step of the definition.
- Show $\forall j\, P(j)$ is true.

Recursive step:

- Let x be a new element constructed in the recursive step of the definition.
- Assume $k_1, k_2, \ldots, k_m$ are elements used to construct an element x in the recursive step of the definition.
- Show $\forall\, k_1, k_2, \ldots, k_m\; ((P(k_1) \wedge P(k_2) \wedge \ldots \wedge P(k_m)) \rightarrow P(x))$.

Example

Use structural induction, to prove that $l(xy) = l(x)+l(y)$, where $x \in \Sigma^*$ and $y \in \Sigma^*$.

Proof by structural induction:

- Define $P(n)$.

  $P(n)$ is $l(xn) = l(x)+l(n)$ whenever $x \in \Sigma^*$.

- Basis step: ($P(j)$ is true, if j is specified in basis step of the definition.)
  - Show $P(\lambda)$ is true.
  - $P(\lambda)$ is $l(\lambda x) = l(\lambda) + l(x)$.
  - Since $\lambda x = x$, $l(\lambda x) = l(x)$

    $= l(x) + 0 = l(x) + l(\lambda)$
  - So, $P(\lambda)$ is true.

Example

Use structural induction, to prove that $l(xy) = l(x)+l(y)$, where $x \in \Sigma^*$ and $y \in \Sigma^*$.

Proof by structural induction:

- Inductive step: $(P(y) \rightarrow P(ya)$ where $a \in \Sigma)$
  - Inductive hypothesis: $(P(y))$
    - $l(xy) = l(x) + l(y)$
  - Show that $P(ya)$ is true.
    - Show $l(xya) = l(x) + l(ya)$
    - By recursive definition, $l(xya) = l(xy) + 1$.
    - By inductive hypothesis, $l(xya)= l(x) + l(y) + 1$.
    - By recursive definition $(l(ya)= l(y) + 1)$, $l(xya)= l(x) + l(ya)$.
    - So, $P(ya)$ is true.

Example

Well-formed formulae for compound propositions:

Basis step: T(true), F(false) and p, where p is a propositional variable, are well-formed.

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

Example:

$p \vee F$ is well-formed.

$p \neg \rightarrow q$ is not well-formed.

Example

Well-formed formulae for operations:

Basis step: x, where x is a numeral or variable, is well-formed.

Recursive step: If F and E are well-formed formulae, then (E+F), (E-F), (E*F), (E/F) and (E↑F) are well-formed formulae.

(* denotes multiplication and ↑ denotes exponentiation.)

Example:

3 + (5-x) is well-formed.

3 * + x is not well-formed.

Example

Show that well-formed formulae for compound propositions contains an equal number of left and right parentheses.

Proof by structural induction:

- Define P(x)

  P(x) is "well-formed compound proposition x contains an equal number of left and right parentheses"
- Basis step: (P(j) is true, if j is specified in basis step of the definition.)
  - T, F and propositional variable p is constructed in the basis step of the definition.
  - Since they do not have any parentheses, P(T), P(F) and P(p) are true.

Example

Proof by structural induction:

- Recursive step:
  - Assume p and q are well-formed formulae.
  - Let l_p be the number of left parentheses in p.
  - Let r_p be the number of right parentheses in p.
  - Let l_q be the number of left parentheses in q.
  - Let r_q be the number of right parentheses in q.
  - Assume $l_p = r_p$ and $l_q = r_q$.
  - We need to show that $(\neg p)$, $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$ and $(p \leftrightarrow q)$ also contains an equal number of left and right parentheses.

Example

Proof by structural induction:

- Recursive step:
  - The number of left parentheses in $(\neg p)$ is l_p+1 and the number of right parentheses in $(\neg p)$ is r_p+1.
  - Since $l_p= r_p$, $l_p+1= r_p+1$ and $(\neg p)$ contains an equal number of left and right parentheses.
  - The number of left parentheses in other compund propositions is $l_p+ l_q+ 1$ and the number of right parentheses in $(\neg p)$ is $r_p+ r_q+ 1$.
  - Since $l_p= r_p$ and $l_q= r_q$, $l_p+ l_q+ 1 = r_p+ r_q+ 1$ and other compound propositions contain an equal number of left and right parentheses.

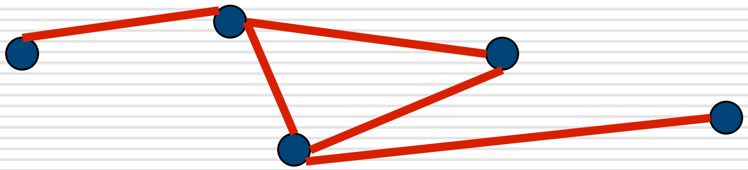
So, by structural induction, the statement is true.

Structural induction

Structural induction is really just a version of (strong) induction.

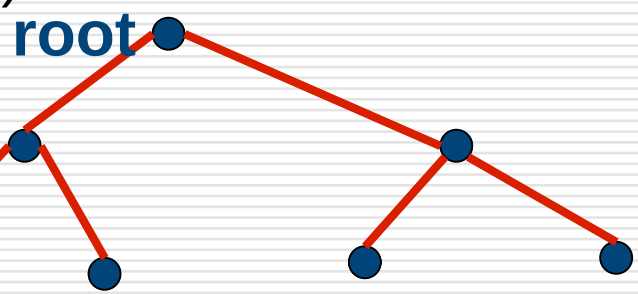
Tree

A graph is made up vertices and edges connecting some pairs of vertices.



A tree is a special type of a graph.

A rooted tree consists of a set of vertices a distinguished vertex called root and edges connecting these vertices. (A tree has no cycle.)



Rooted tree

The set of **rooted trees** can be defined recursively by these steps:

Basis step:

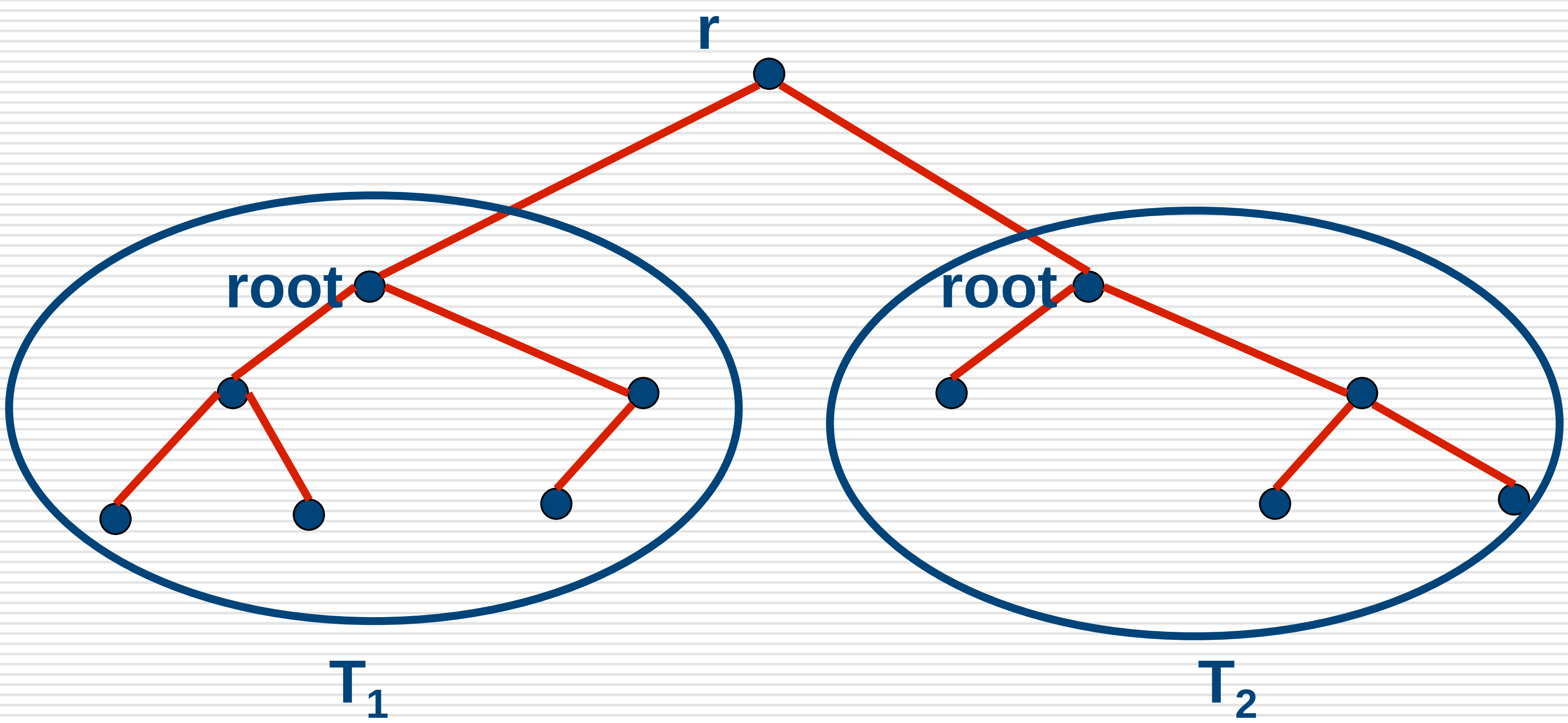
A single vertex r is a rooted tree.

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 tree $T_1, T_2, \ldots, T_n$, and adding an edge from r to each of the vertices $r_1, r_2, \ldots, r_n$, is also a rooted tree.

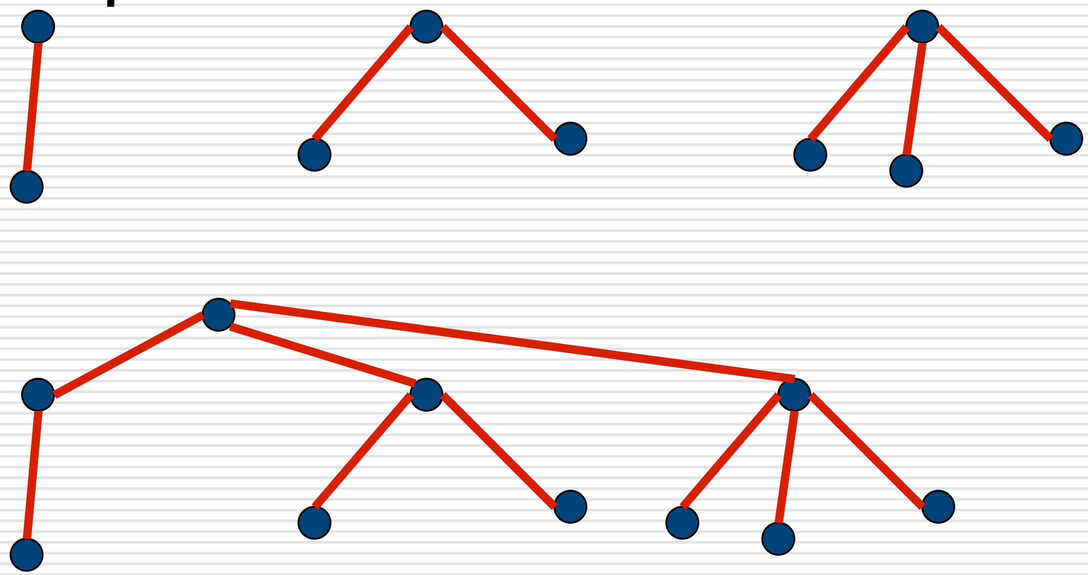
Rooted tree



Rooted tree

Basis step:

Inductive step:



Extended binary trees

The set of **extended binary trees** can be defined recursively by these steps:

Basis step:

The empty set is an extended binary tree.

Recursive step:

Assume T_1 and T_2 are disjoint extended binary trees.

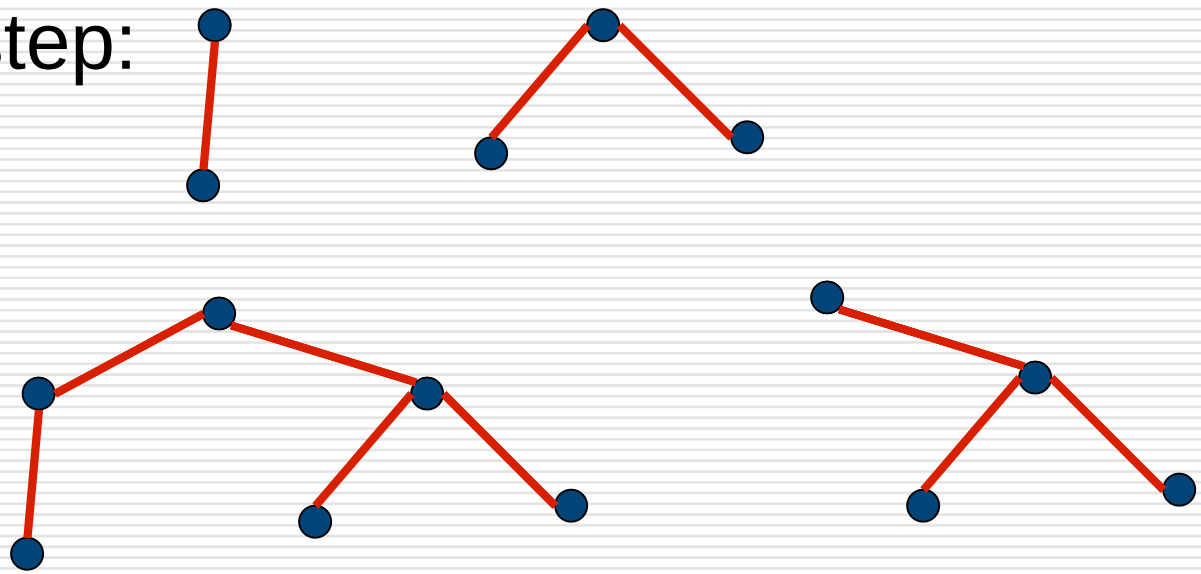
Then, there is an extended binary tree, denoted $T_1 \cdot T_2$, consisting of a root r together with edges connecting the roots of left subtree T_1 and the right subtree T_2 when these trees are nonempty.

Extended binary trees

The root of an extended binary tree is connected to at most two subtrees.

Basis step: $\emptyset$

Inductive step:



Full binary trees

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

Basis step:

There is a full binary tree consisting only of a single vertex r.

Recursive step:

Assume T_1 and T_2 are disjoint full binary trees.

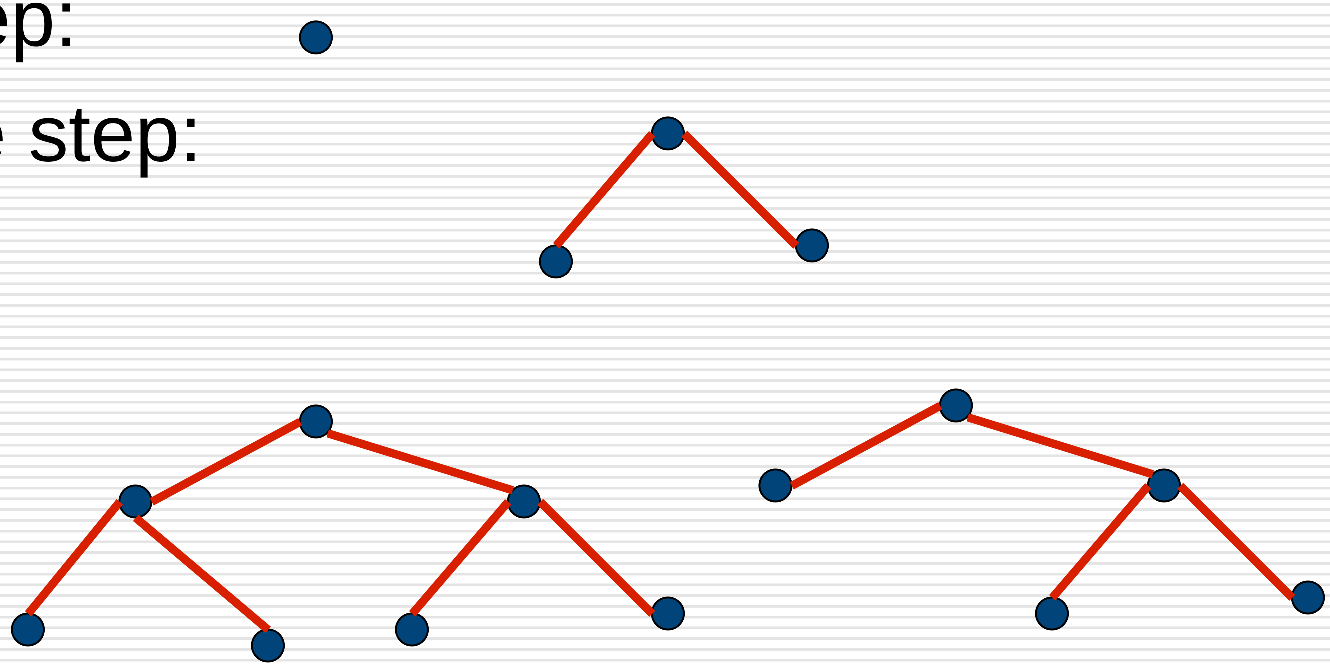
Then, there is a full binary tree, denoted $T_1 \cdot T_2$, consisting of a root r together with edges connecting the root to each of the roots of the left subtree T_1 and the right subtree T_2.

Full binary tree

The root of a full binary tree is connected to exactly two subtrees.

Basis step:

Inductive step:



Height of full binary trees

We define height h(T) of a full binary tree T recursively.

Basis step:

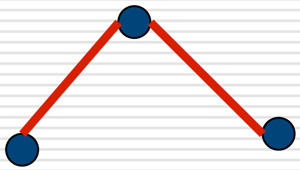
- Assume T is a full binary tree consisting of a single vertex.
- $h(T)=0$

Recursive step:

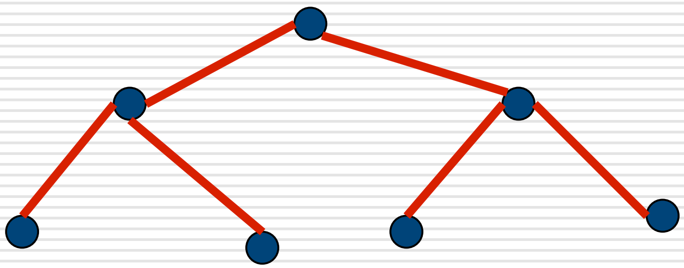
- Assume T_1 and T_2 are full binary trees.
- $h(T_1 \cdot T_2) = 1 + \max(h(T_1), h(T_2))$

Height of full binary trees

h(T) = 0



h(T) = 1+ max(0,0) = 1



h(T) = 1+ max(1,1) = 2

Number of vertices of full binary trees

We define number of vertices $n(T)$ of a full binary tree T recursively.

Basis step:

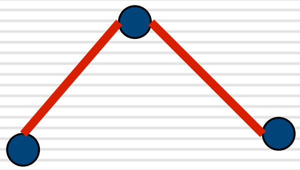
- Assume T is a full binary tree consisting of a single vertex.
- $n(T)=1$

Recursive step:

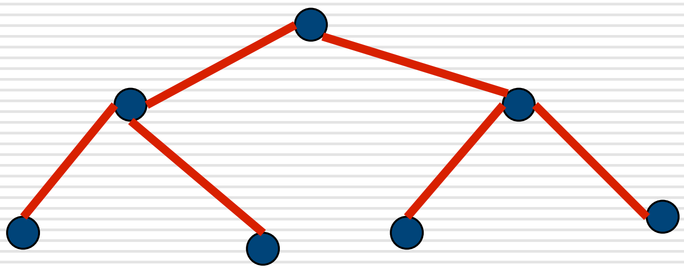
- Assume T_1 and T_2 are full binary trees.
- $n(T_1 \cdot T_2) = 1 + n(T_1) + n(T_2)$

Number of vertices of full binary trees

n(T) = 1



n(T) = 1+ 1 + 1 = 3



n(T) = 1+ 3 + 3 = 7

Structural induction

How to show a result about full binary trees using structural induction?

Basis step:

Show that the result is true for the tree consisting of a single vertex.

Recursive step:

Show that if the result is true for trees T_1 and T_2, then it is true for $T_1 \cdot T_2$, consisting of a root r which has T_1 as its left subtree and T_2 as its right subtree.

Example

Show if T is a full binary tree T, then $n(T) \leq 2^{h(T)+1} - 1$.

Proof by structural induction:

- Basis step:
  - Assume T is a full binary tree consisting of a single vertex.
  - Show $n(T) \leq 2^{h(T)+1} - 1$ is true.

    $1 \leq 2^{0+1} - 1=1$
  - So, it is true for T.
- Inductive step:
  - Assume T_1 and T_2 are full binary trees.
  - Assume $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$ are true.
  - Assume $T= T_1 \cdot T_2$.

Example

Show if T is a full binary tree T, then $n(T) \leq 2^{h(T)+1} - 1$.

Proof by structural induction:

- Inductive step:
  - Show $n(T) \leq 2^{h(T)+1} - 1$ is true.
  - By recursive definition, $n(T) = 1 + n(T_1) + n(T_2)$.
  - By recursive definition, $h(T) = 1 + \max(h(T_1),h(T_2))$.

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

$\leq 1 + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1$ (by inductive hypothesis)

$\leq 2 . \max(2^{h(T_1)+1},2^{h(T_2)+1}) - 1$

$= 2 . 2^{\max(h(T_1),h(T_2)) + 1} - 1$ $(\max(2^x,2^y) = 2^{\max(x,y)})$

$= 2 . 2^{h(T)} - 1$ (by recursive definition)

$= 2^{h(T)+1} - 1$

Recommended exercises

1,3,5,7,9,21,23,25,27,29,33,35,44,57,59