Lecture 5a

Part C

Binary Trees
Definition, Terminology, Properties
(continued)

BT Properties: Relating #s of Ext. and Int. Nodes

Given a *binary tree* that is:

- ***nonempty*** and ***proper***
- with n_I ***internal nodes*** and n_E ***external nodes***

We can then expect that: $\boxed{n_E = n_I + 1}$

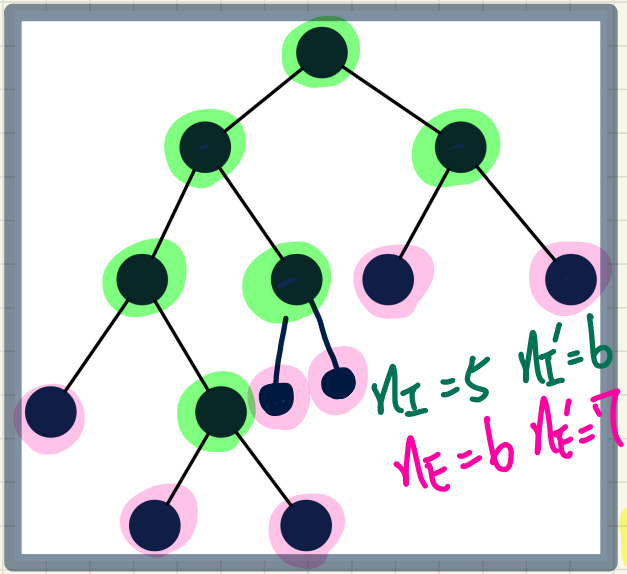
Induction on Size of Proper BT

$* \quad n_E' = n_E + 1$

$= \{\text{ind. hypoth.}\}$

$(n_I + 1) + 1$

$= n_I' + 1$ ∎



Base Case: A proper BT with size 1

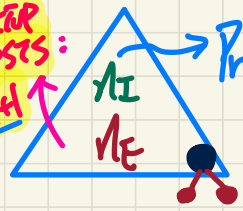
no internal node

⇒ "proper" property is satisfied

(without violation witness)

Inductive Recursive Case

Inductive hypothesis: $n_E = n_I + 1$



→ Proper BT with size > 1

$n_E' = n_E - 1 + 2 = n_E + 1$

$n_I' = n_I + 1$ *

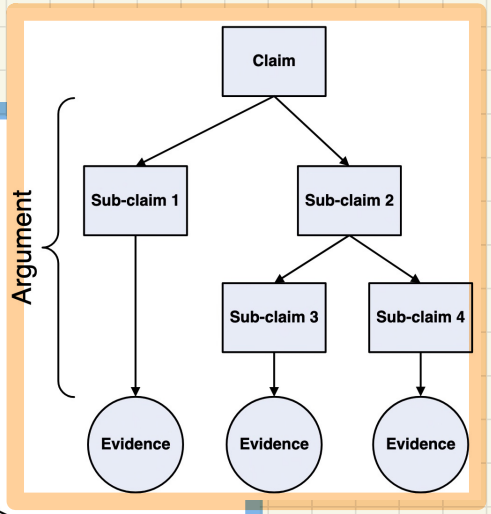
Lecture 5a

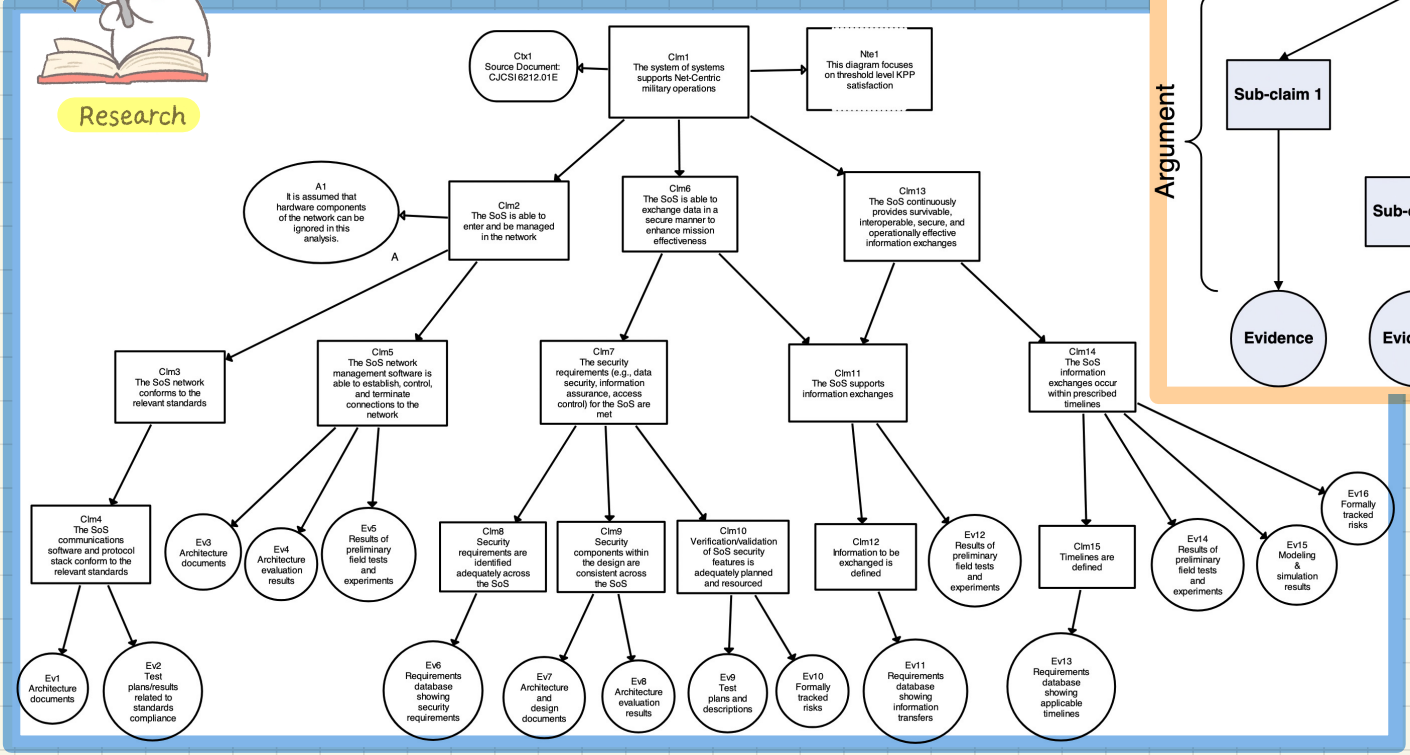
Part D

Binary Trees
Applications

Applications of General Trees: Assurance Cases

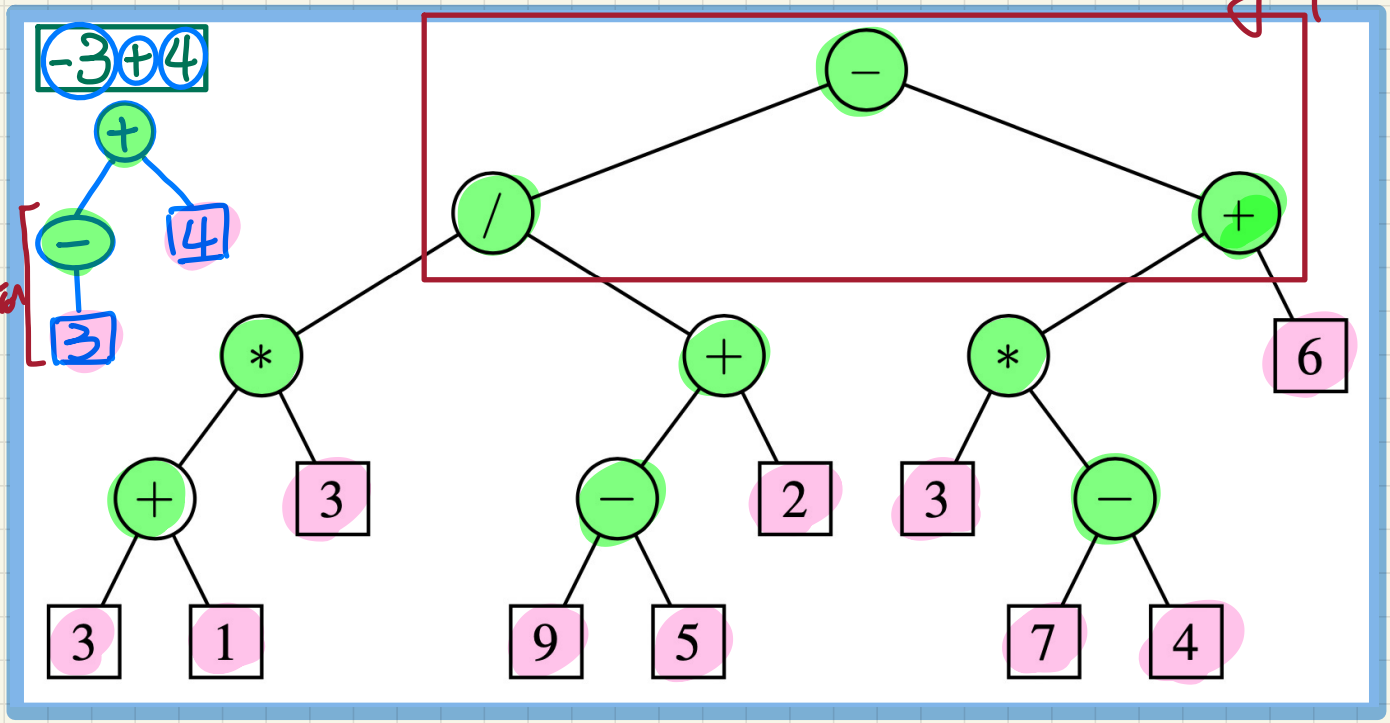
Research on "Assurance Cases" if interested!





Source: https://resources.sei.cmu.edu/asset_files/whitepaper/2009_019_001_29066.pdf

Applications of Binary Trees: Infix Notation



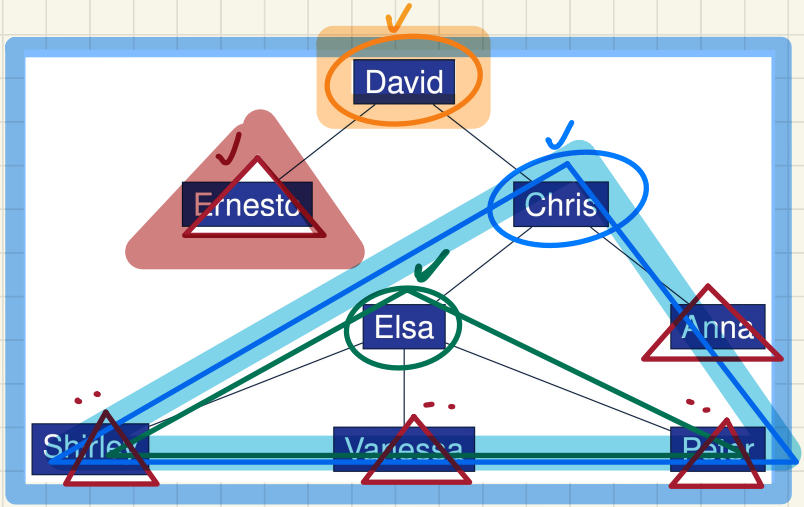
Q. Is the binary tree necessarily proper?

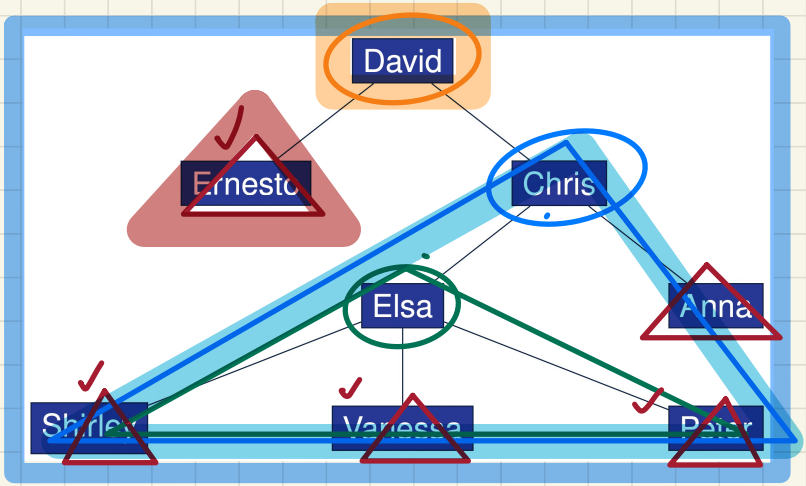
Lecture 5a

Part E

Tree Traversals
Pre-Order, In-Order, Post-Order

General Tree Traversals: Pre-Order vs. Post-Order





Pre-Order Traversal from the Root

Parent, pre-order(child nodes)

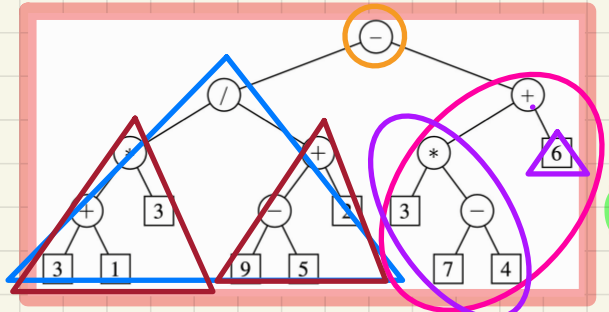
David | Ernesto | Chris | Elsa | Shirley | Vanessa | Peter | Anna

Post-Order Traversal from the Root

post-order(child nodes), Parent

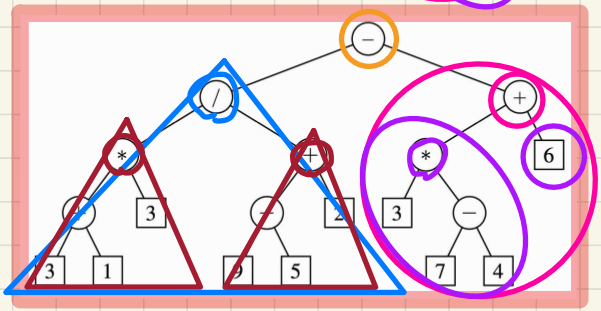
Ernesto | Shirley | Vanessa | Peter | Elsa | Anna | Chris | David

Binary Tree Traversals



Pre-Order Traversal

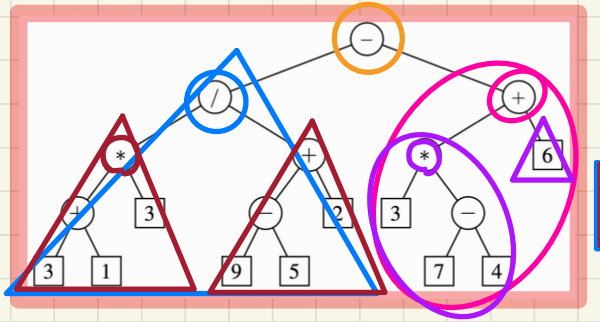
− / * + 3 1 3 + − 9 5 2 + * 3 − 7 4 6



In-Order Traversal

In-Order(LST) Parent In-Order(RST)

3 + 1 * 3 / 9 − 5 + 2 − 3 * 7 − 4 + 6



Post-Order Traversal

3 1 + 3 * 9 5 − 2 + / 3 7 4 − * 6 + −

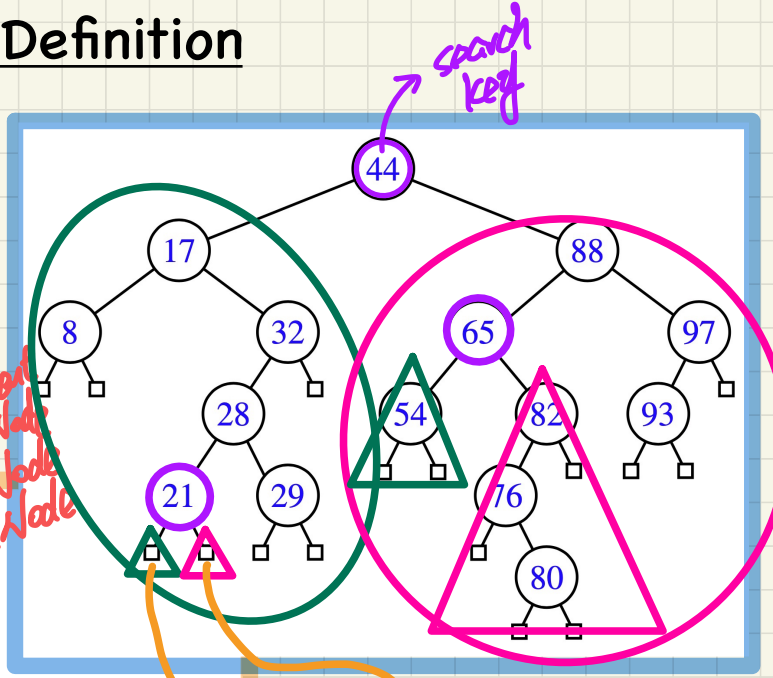
Lecture 5b

Part A

Binary Search Tree - Definition and Property

Binary Search Trees: Recursive Definition

- external node
- internal node
  + LST
  + RST



$\forall x \cdot x \in \emptyset \Rightarrow P(x)$

internal node

≈ element of a Trivially SLLNode/DLLNode

Node p stores (key(p), value(p))

Each node n of LST is such that key(n) < key(p)

Each node n of RST is such that key(n) > key(p)

can't find a violation of search prop.

External node satisfies search property ∵ its LST or RST contains no nodes

Binary Search Trees: Sorting Property

- BST: Non-Linear Structure
- In-Order Traversal

8 17 21 28 29 32 44 54 65 76 80 82 88 93 97

in-order of RST

in-order of LST