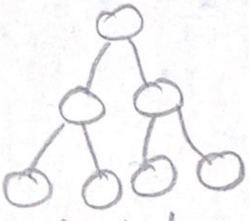
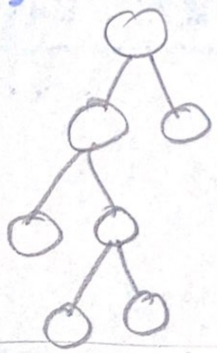
TYPES of TREE (Binary)



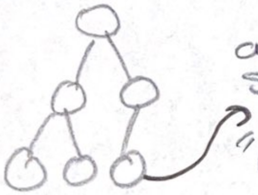
leaf nodes should be at the same level

perfect binary tree



Every node has 2 child

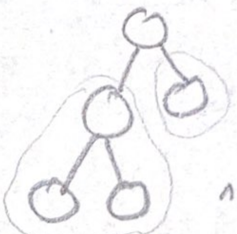
Full binary tree



at the left side

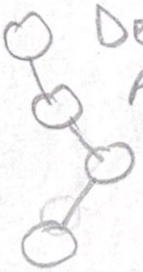
"left aligned"

Complete binary tree

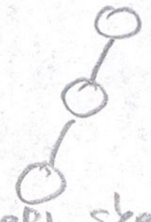


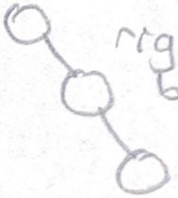
Difference between the left and the right subtree for any node is not more than one

Balanced binary tree



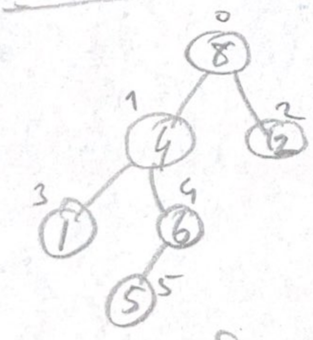
Degenerate on Pathological Tree





right -skewed binary tree

left-skewed binarytree



*All the nodes of the left subtree are less than the root node

* All the nodes of the right subtree are greater than the root node

index of left child → 2k+1

index of right child → 2k+2

k is parent

Binary Search Tree

delete operation on a BST

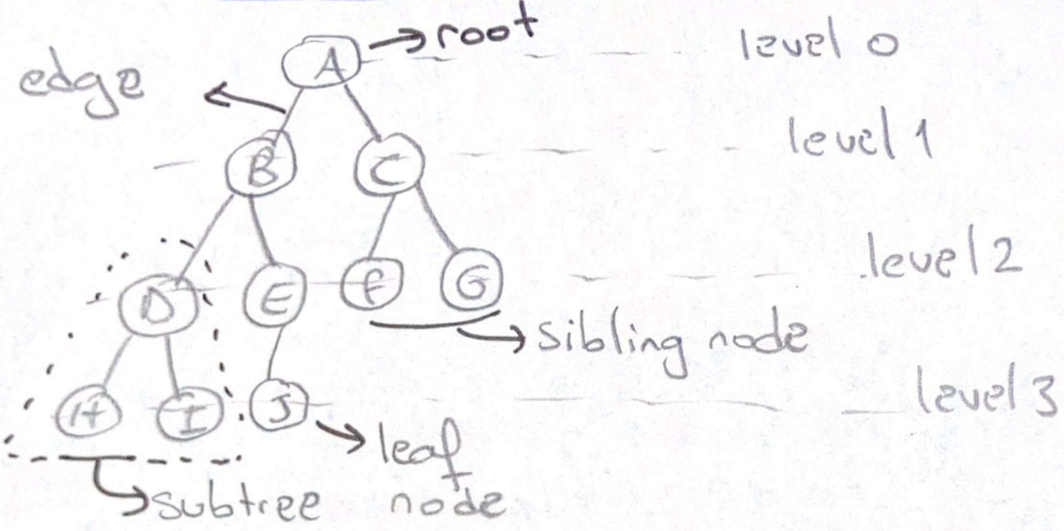
Case I → Not found

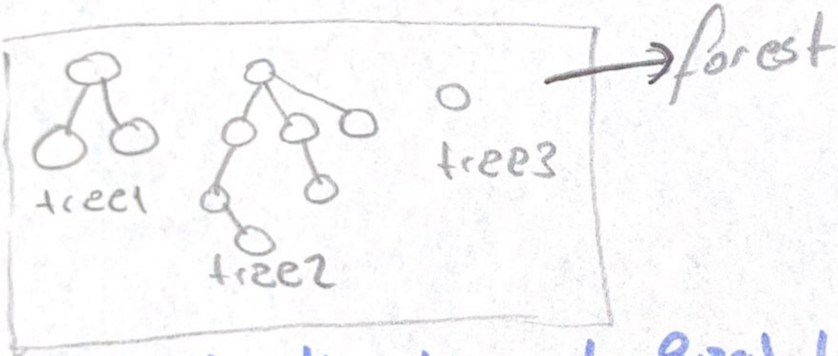
Case II → Leaf Node

Case III → internal node with a single child

Case IV → internal node with 2 child

TREE DATA STRUCTURE



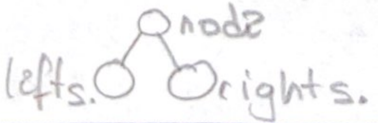


Tree Application In Real Life

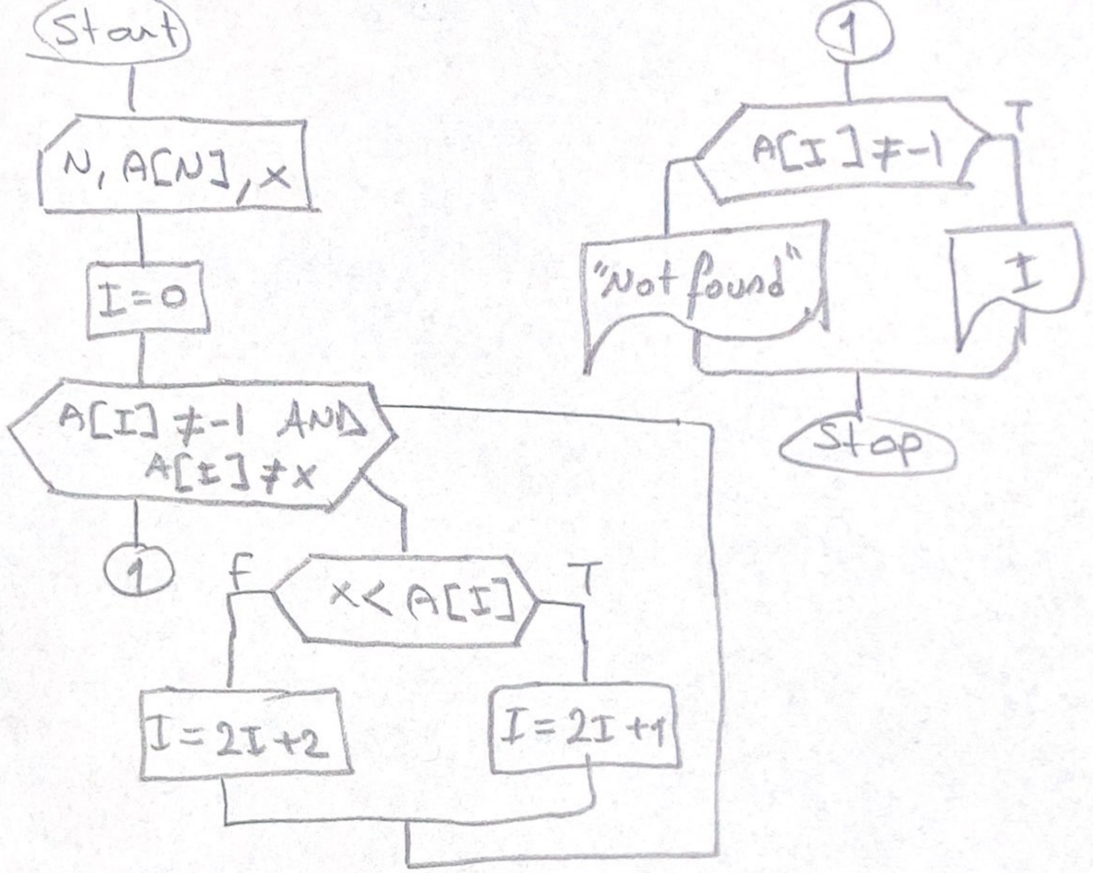
→ File Manogement
→ Compression Algorithms (Huffmann)
→ Databases
→ Compilers (Sytax Tree)
→ Priority Queue
→ AI (Decision Tree, Random Tree)
→ Indexing Multi-dimensional information

Tree Traversals

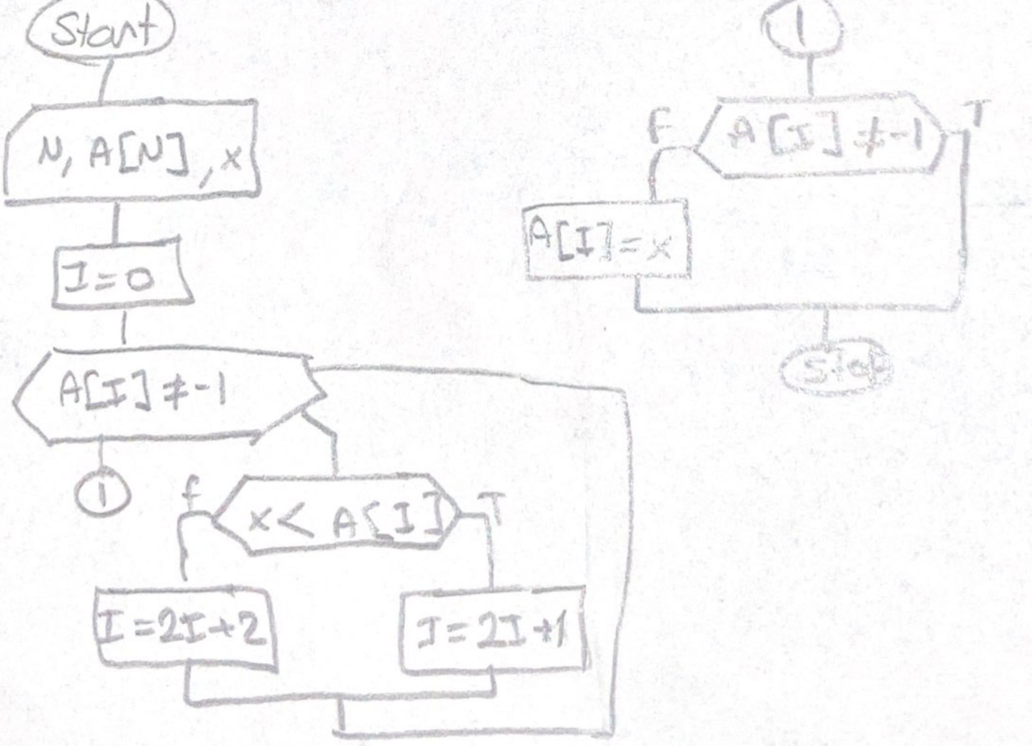
- Inorder ⇒ Left subtree, node, right subtree
- Preorder ⇒ Node, left subtree, right subtree
- Postorder ⇒ Left subtree, right subtree, node



search on a BST



insert on a BST



BST Complexities

	Best	Worst
Search	O(logn)	O(n)
Insert	o(logn)	O(n)
Delete	o ((logn)	O(n)

Because of the unbalanced trees

space complexity → O(n)