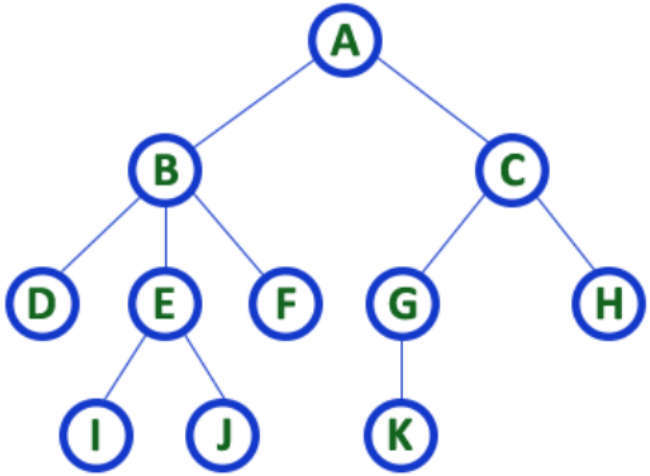
DIT181: Data Structures and Algorithms

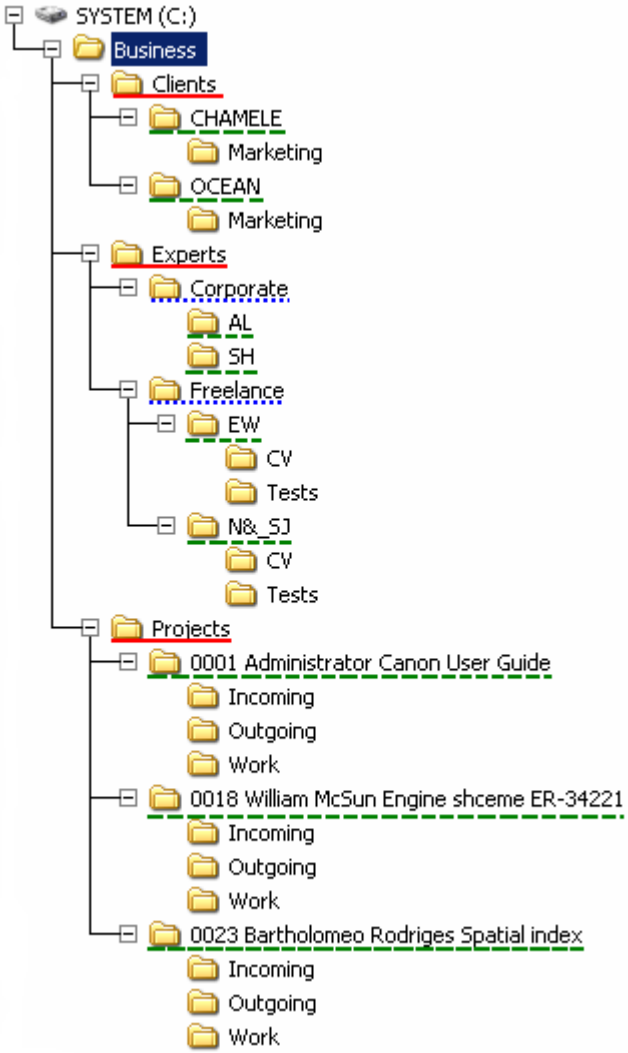
Trees &
Binary Search Trees

Gül Calikli
Email: calikli@chalmers.se

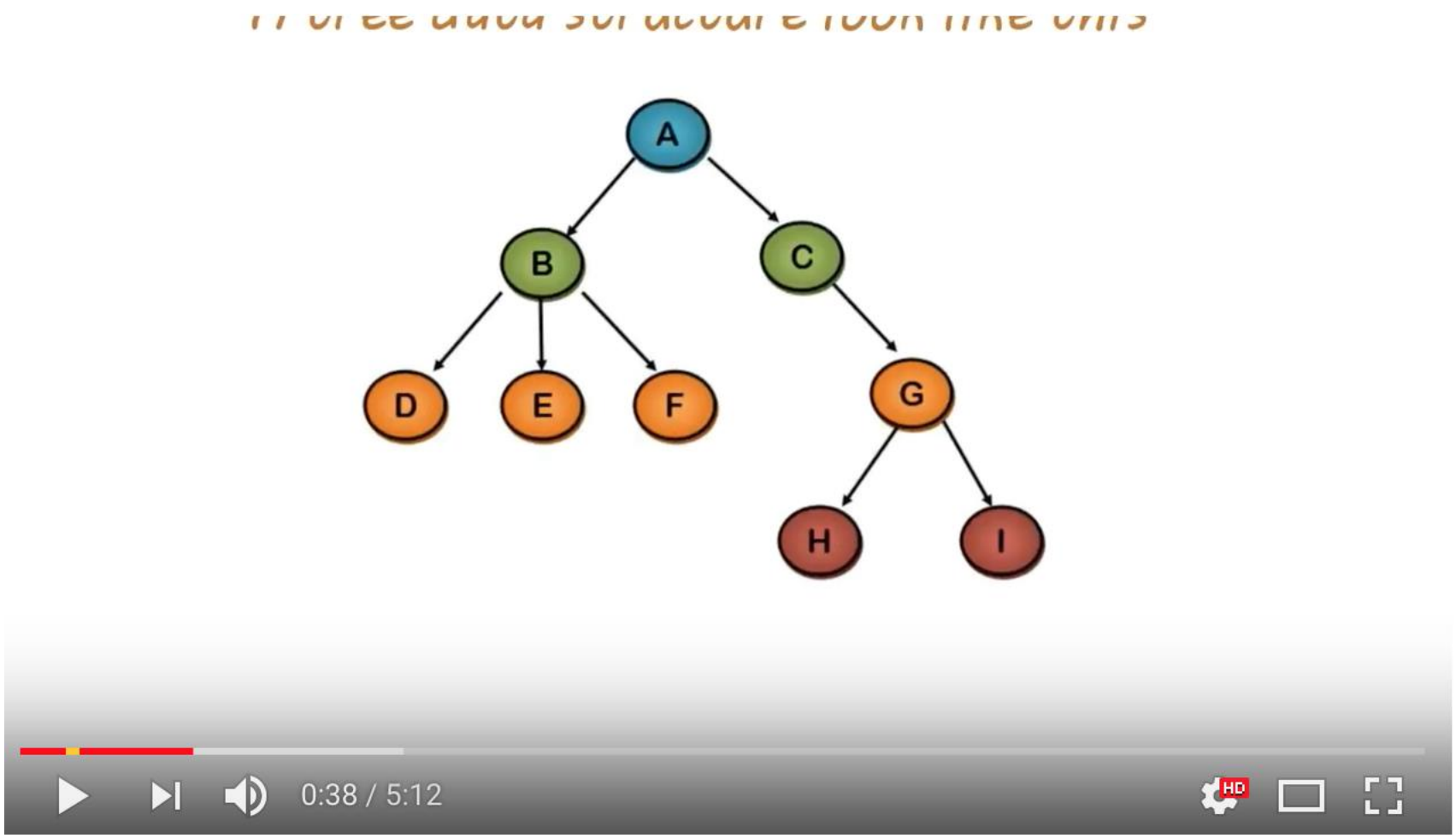
General Trees

- Fundamental data structures in computer science
- Where are trees used?
  - <u>One example</u>: Almost all operating systems store files in trees or tree-like structures



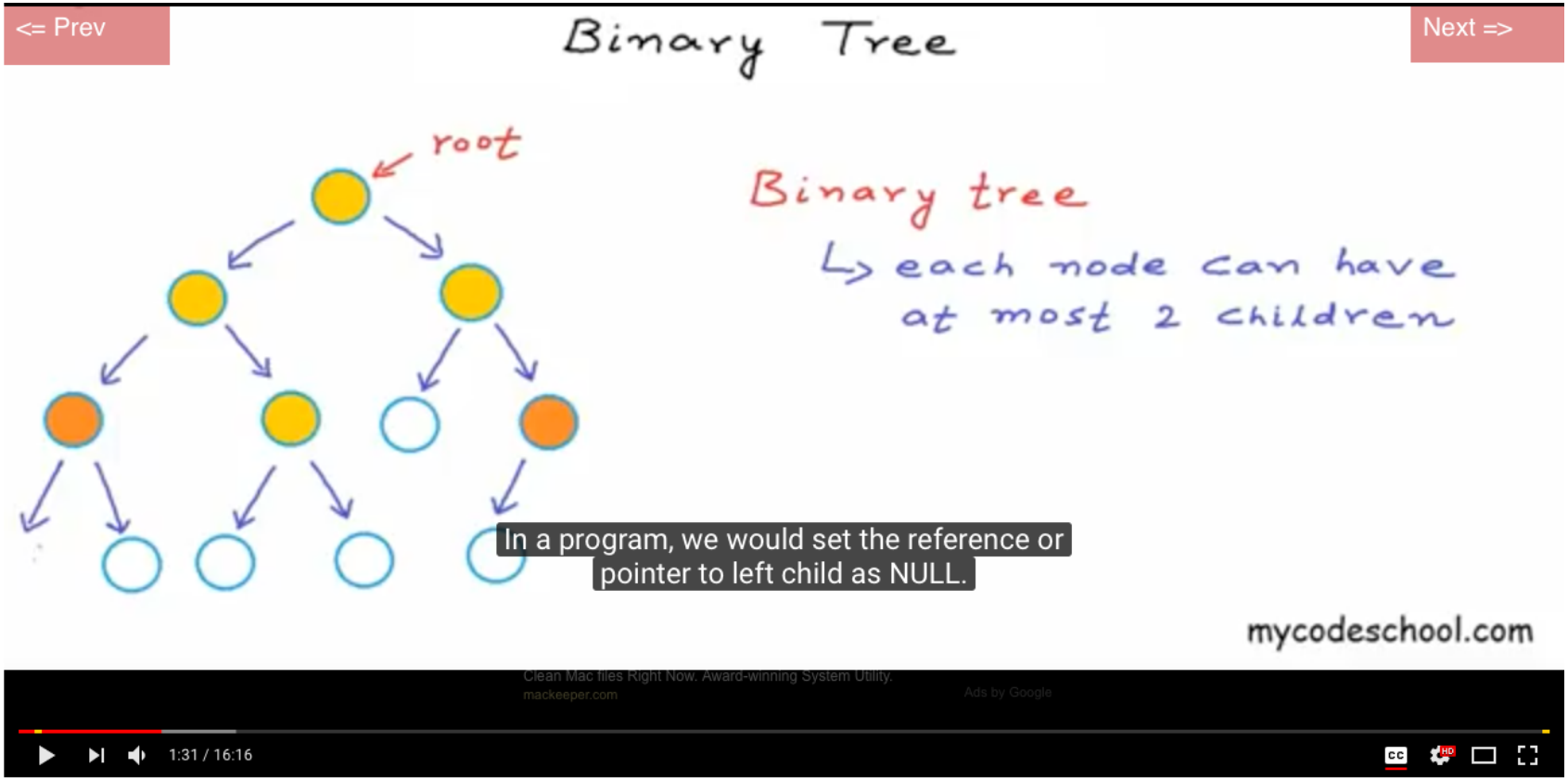


Introduction to Trees



Video link: https://www.youtube.com/watch?v=ikPPdBDZnz4

Binary Trees



Video link: https://www.youtube.com/watch?v=H5JubkIy_p8&t=8s

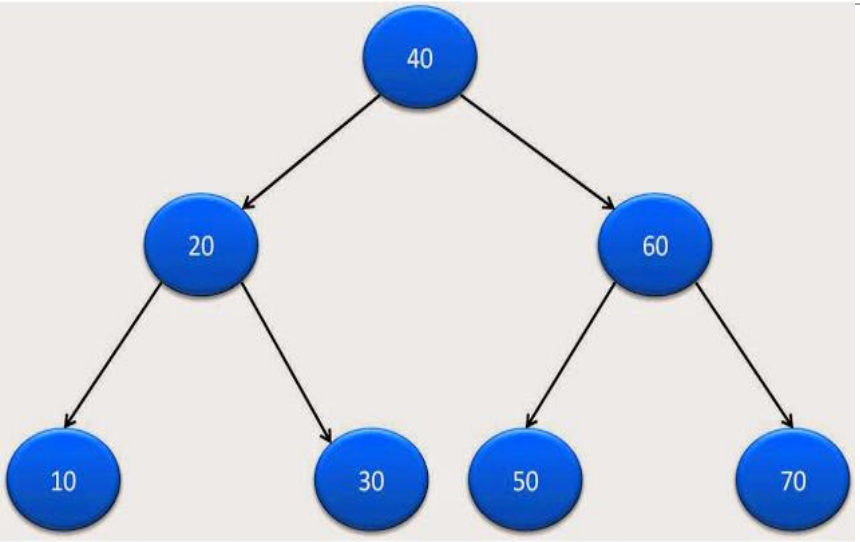
Binary Trees: Tree Traversal

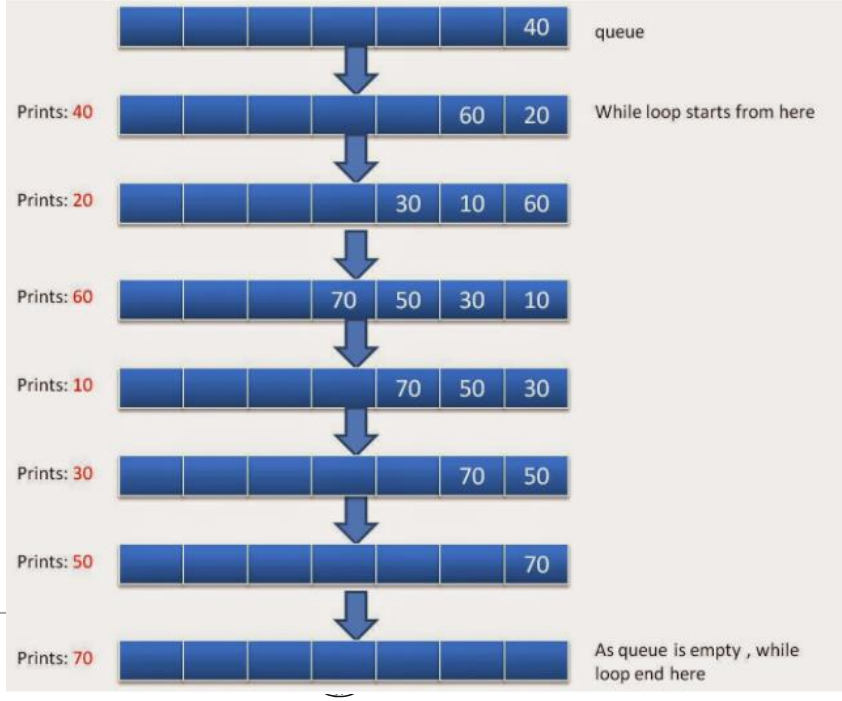
- Breadth First Traversal (BST) – Level order Traversal
- Depth First Traversal (DST)
  - Pre-order Traversal
  - Post-order Traversal
  - In-order Traversal

Binary Trees: Tree Traversal

- Breadth First Traversal (BST) – Level order Traversal
- Depth First Traversal (DST)
  - Pre-order Traversal
  - Post-order Traversal
  - In-order Traversal

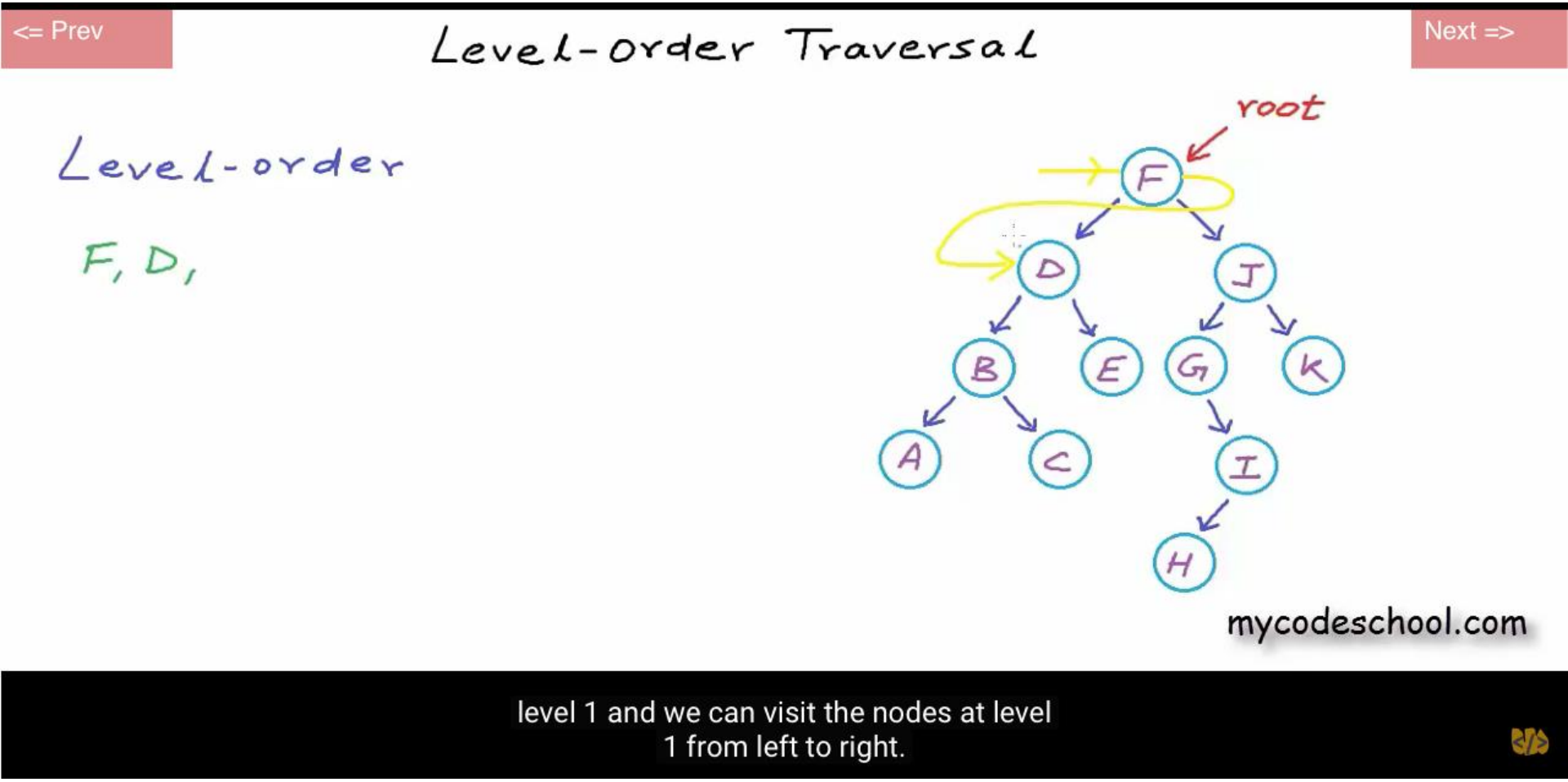
Level Order Search





- **Pseudocode**:
- Create an empty queue and push root node to it.
- While queue is not empty, do the following:
  - Dequeue a node from queue and print it
  - If left child of the dequeued node is not null, enqueue it to the queue.
  - If right child of the queue is not null, enqueue it to the queue.

Level Order Traversal

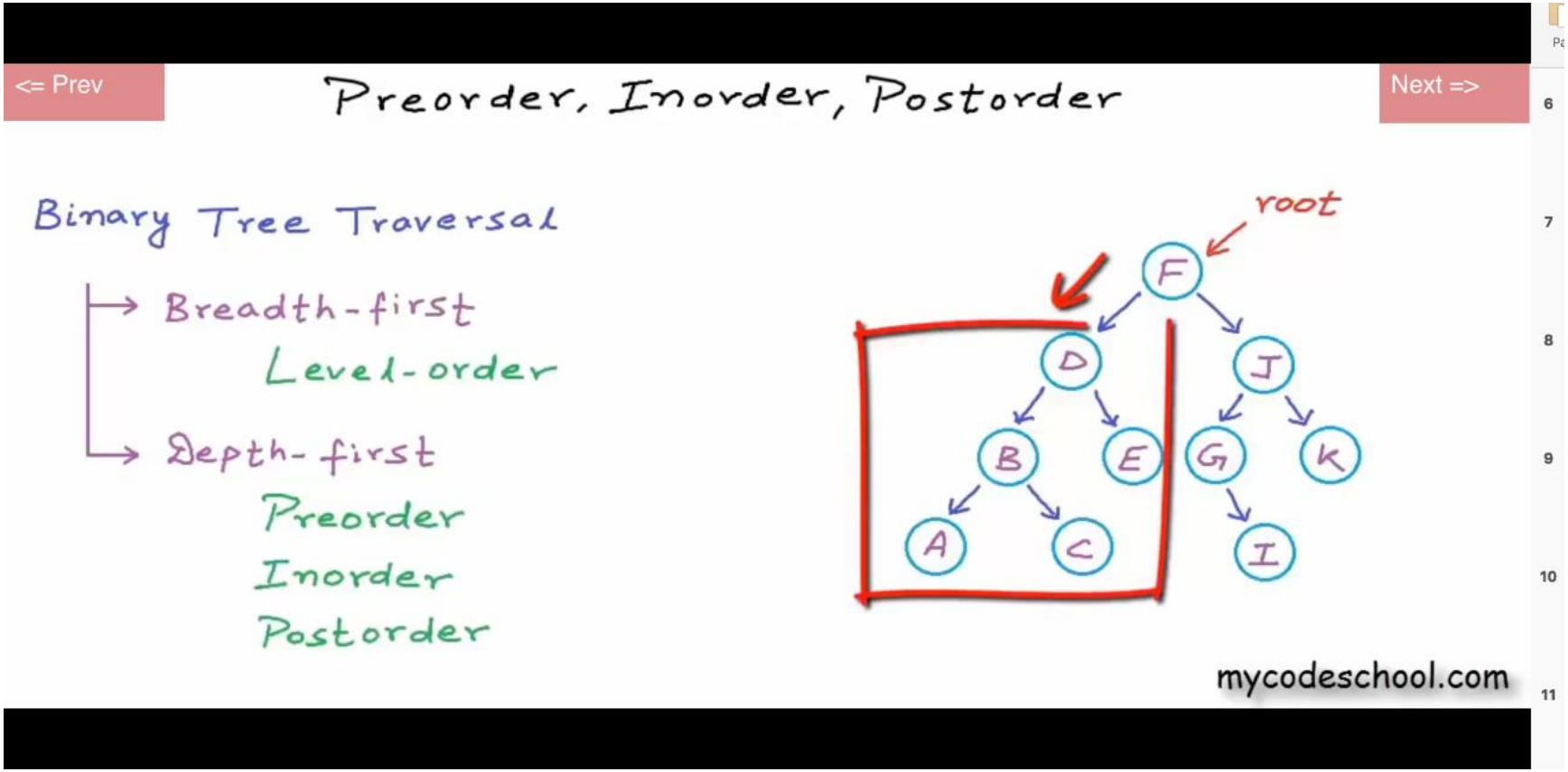


Video link: https://www.youtube.com/watch?v=86g8jAQug04

Binary Trees: Tree Traversal

- Breadth First Traversal (BST) – Level order Traversal
- Depth First Traversal (DST)
  - Pre-order Traversal (Root, Left child, Right child)
  - In-order Traversal (Left child, Root, right child)
  - Post-order Traversal (Left Child, Right child, rRoot)

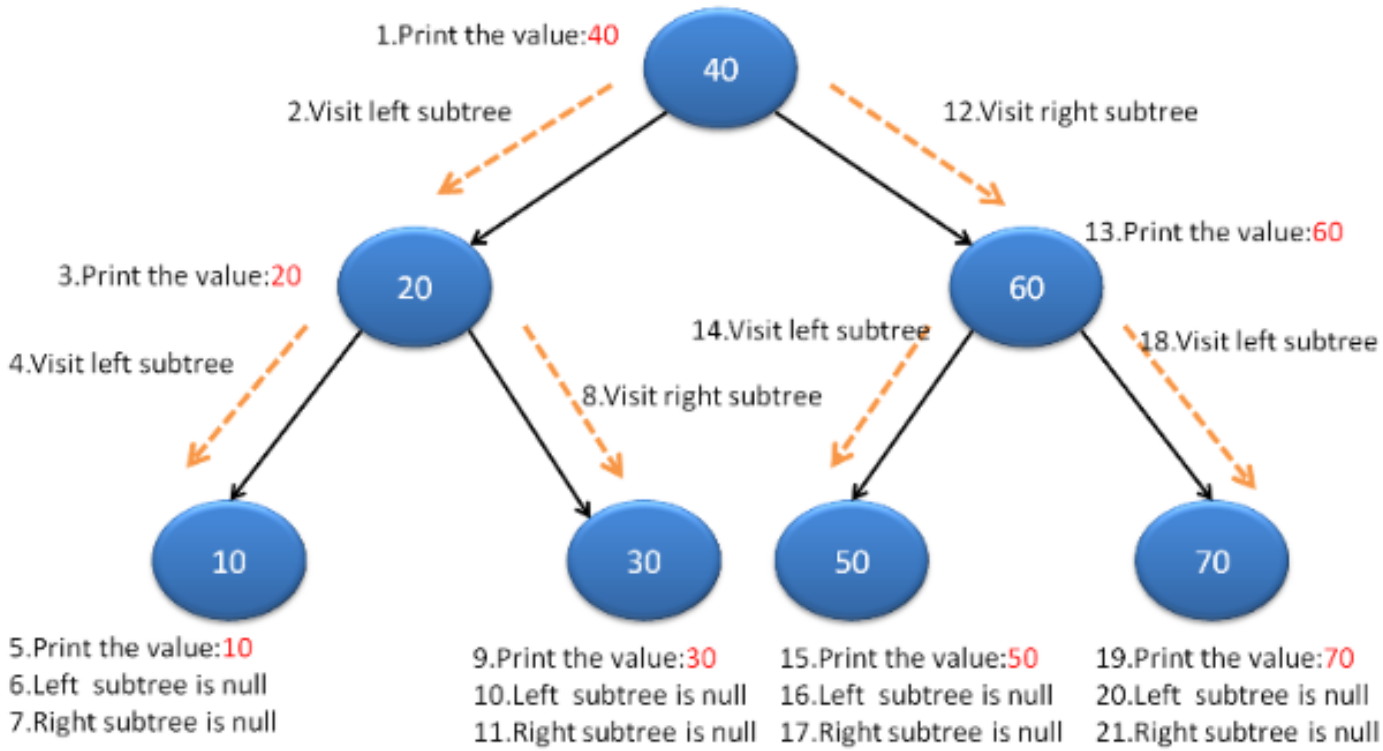
Depth First Search (Pre-order, In-order and Post-order Traversal)



Video link: https://www.youtube.com/watch?v=gm8DUJJhmY4

Pre-Order Binary Tree Traversal: Recursive Solution

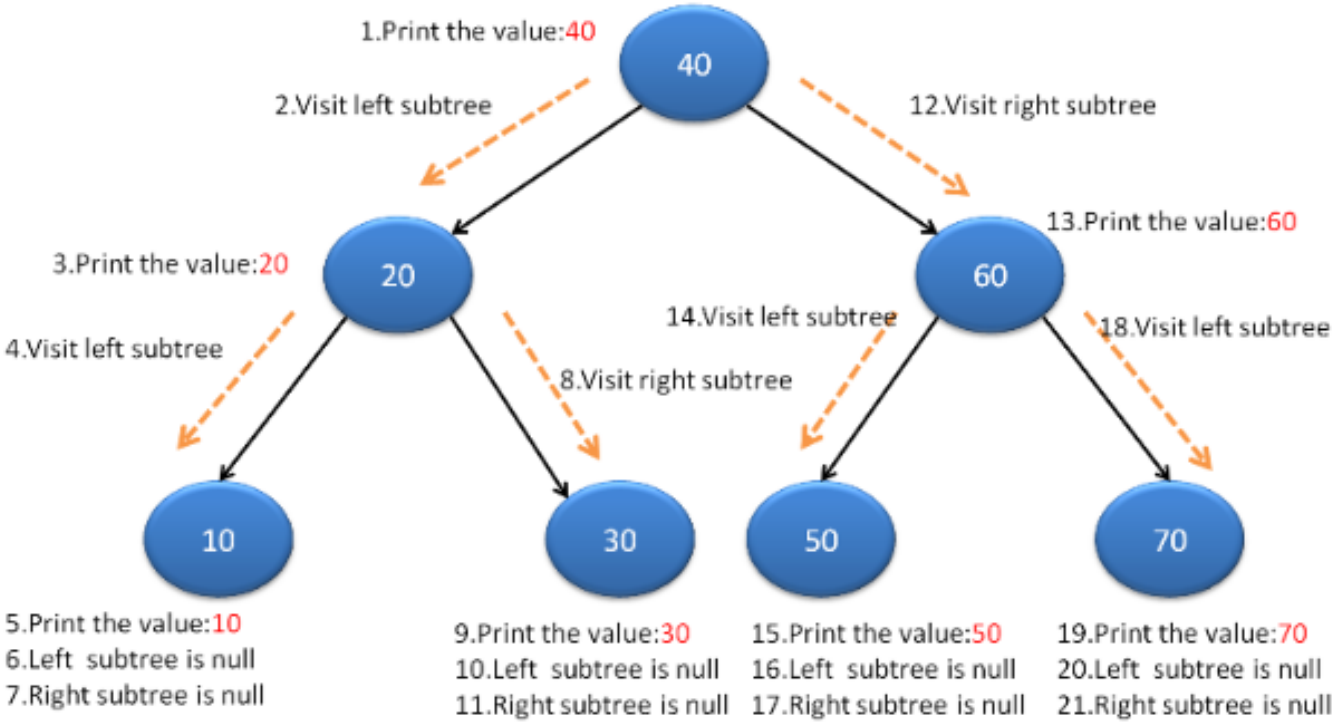
- Pseudocode:
  - Visit the node.
  - Traverse the left subtree in Pre-Order.
  - Traverse the right subtree in Pre-Order.



PreOrder traversal of above graph is :40,20,10,30,60,50,70

Pre-Order Binary Tree Traversal: Iterative Solution

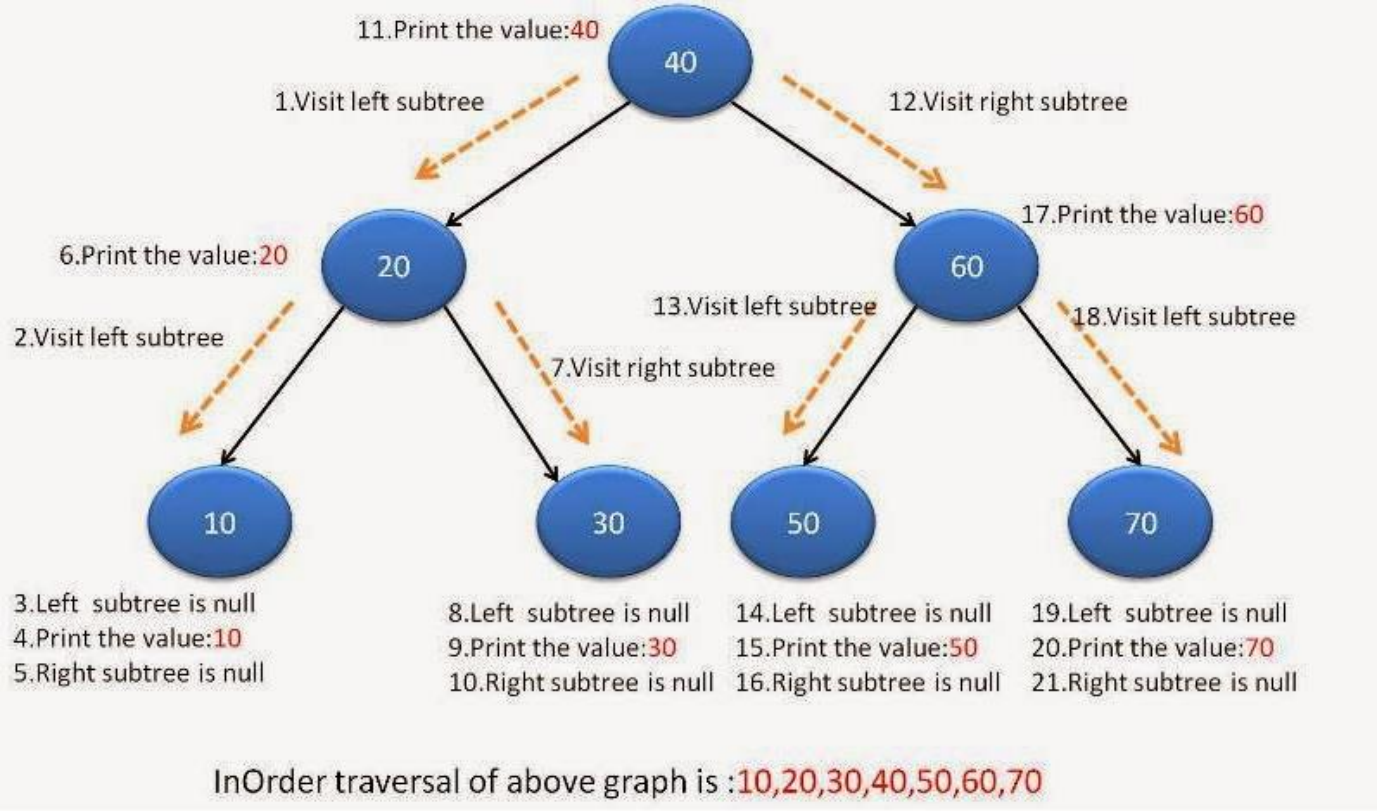
- **Pseudocode:**
- Create empty stack and push root node to it.
- Do the following when stack is not empty
  - Pop a node from stack and print it
  - Push right child of popped node to stack
  - Push left child of popped node to stack



PreOrder traversal of above graph is :40,20,10,30,60,50,70

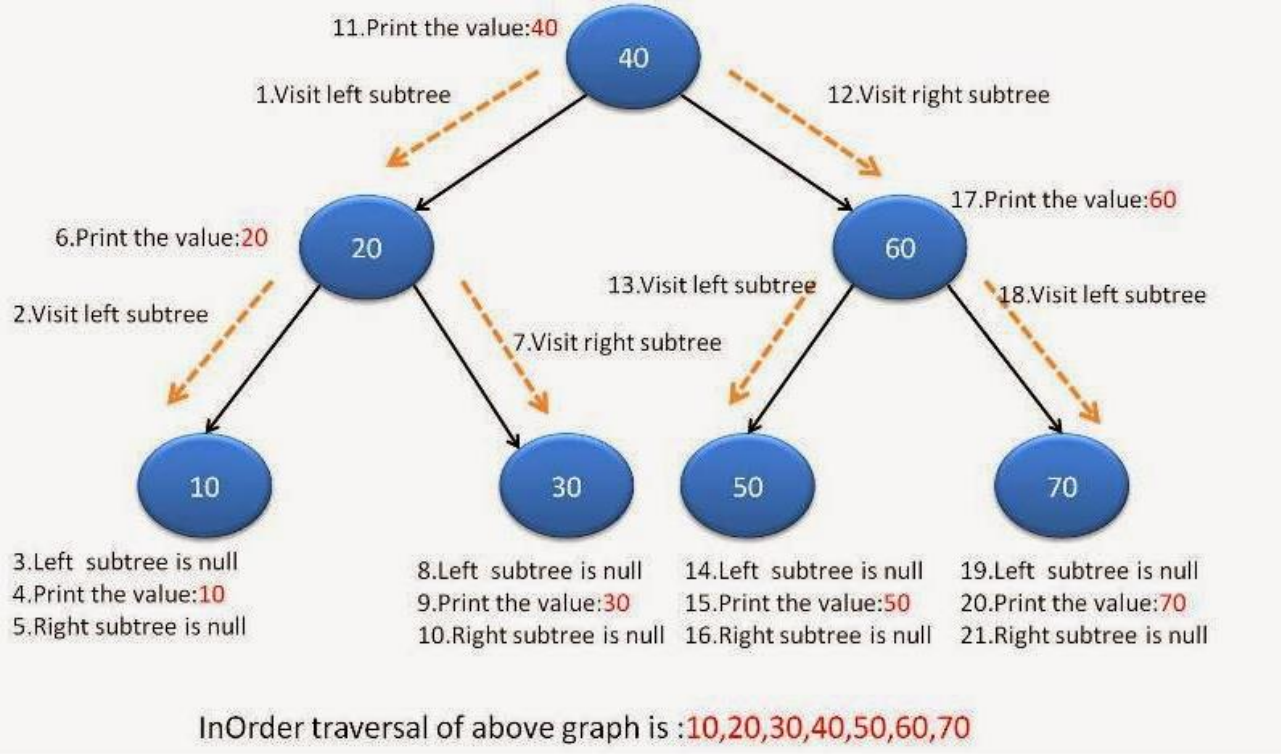
In-Order Binary Tree Traversal: Recursive Solution

- **Pseudocode:**
- Traverse the left subtree in In-Order.
- Visit the node.
- Traverse the right subtree in In-Order.



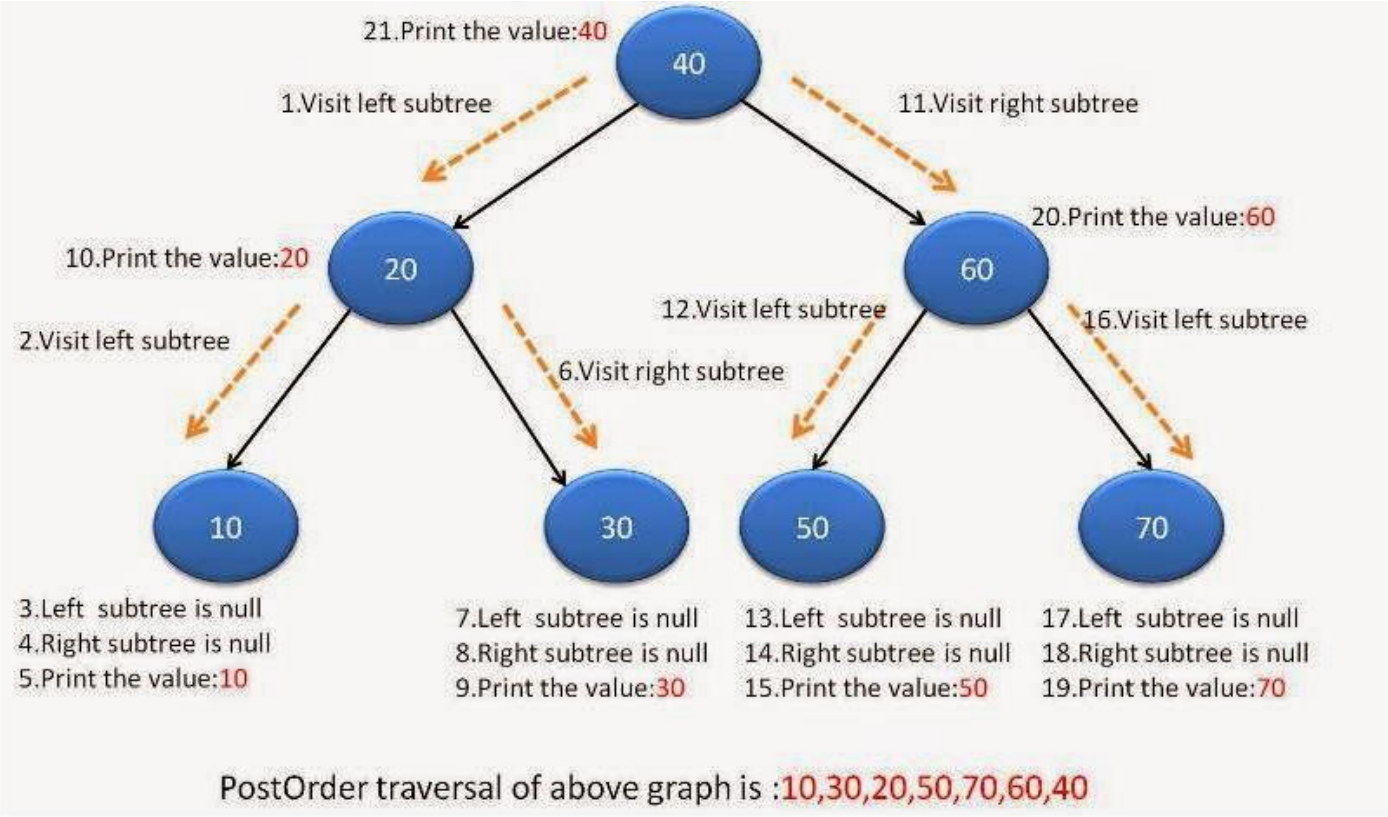
In-Order Binary Tree Traversal: Iterative Solution

- **Pseudocode:**
- Create an empty stack s and Initialize current node as root
- Push the current node to s and set currentNode = currentNode.left until currentNode is NULL
- If currentNode is NULL and s is not empty then
  - Pop the top node from stack and print it
  - set currentNode = currentNode.right
  - go to step 2



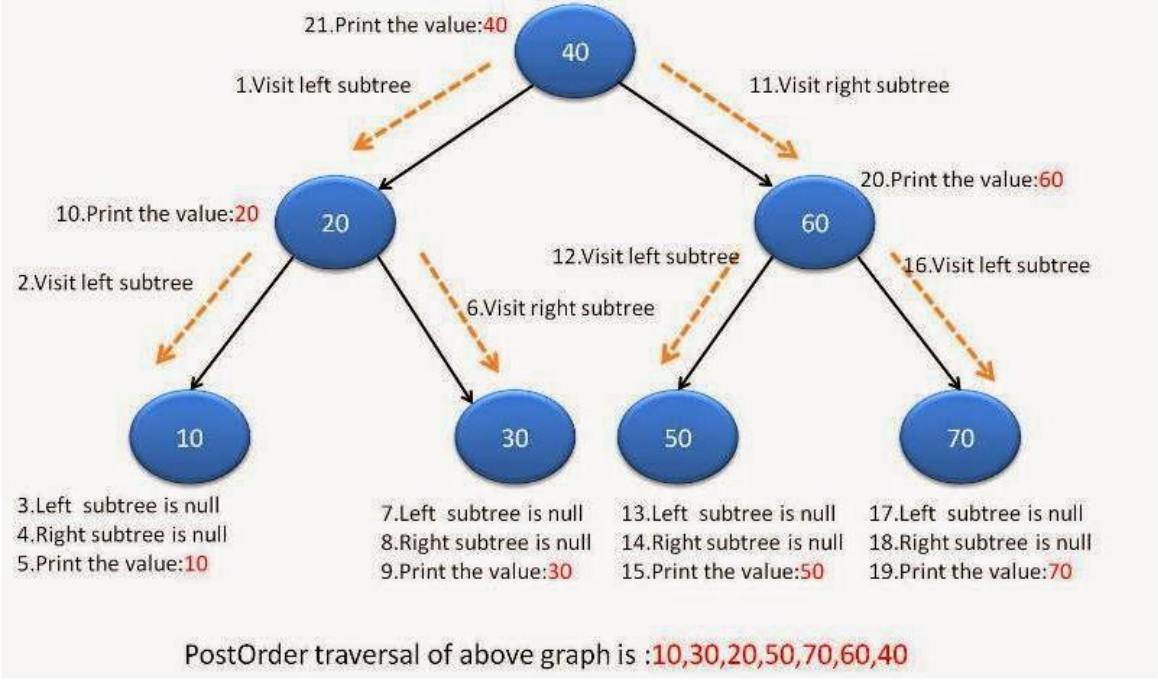
Post-Order Binary Tree Traversal: Recursive Solution

- **Pseudocode:**
- Traverse the left subtree in Post-Order.
- Traverse the right subtree in Post-Order.
- Visit the node.

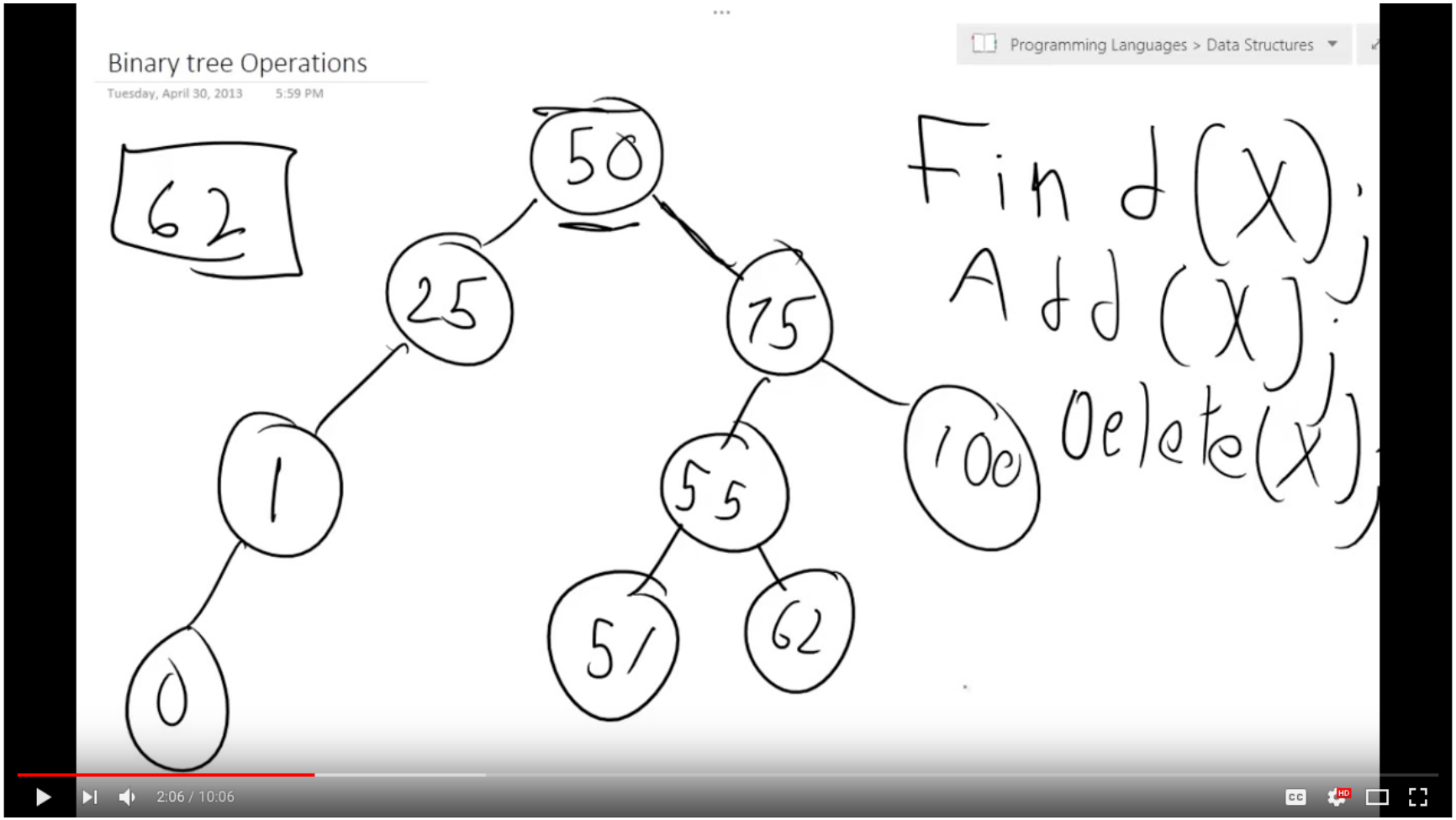


Post-Order Binary Tree Traversal: Iterative Solution

- **Pseudocode:**
- Create an empty stack s and set currentNode =root.
- while currentNode is not NULL Do following
  - Push currentNode ‘s right child and then currentNode to stack.
  - Set currentNode =currentNode .left
- Pop a node from stack and set it as root and set it to currentNode
  - If the popped node has a right child and the right child is at top of stack, then remove the right child from stack, push the current node back and set currentNode as currentNode ‘s right child.
  - Else print currentNode ‘s data and set currentNode as NULL.
- Repeat steps 2 and 3 while stack is not empty.



Binary Search Tree Operations



Binary Search Tree Operations