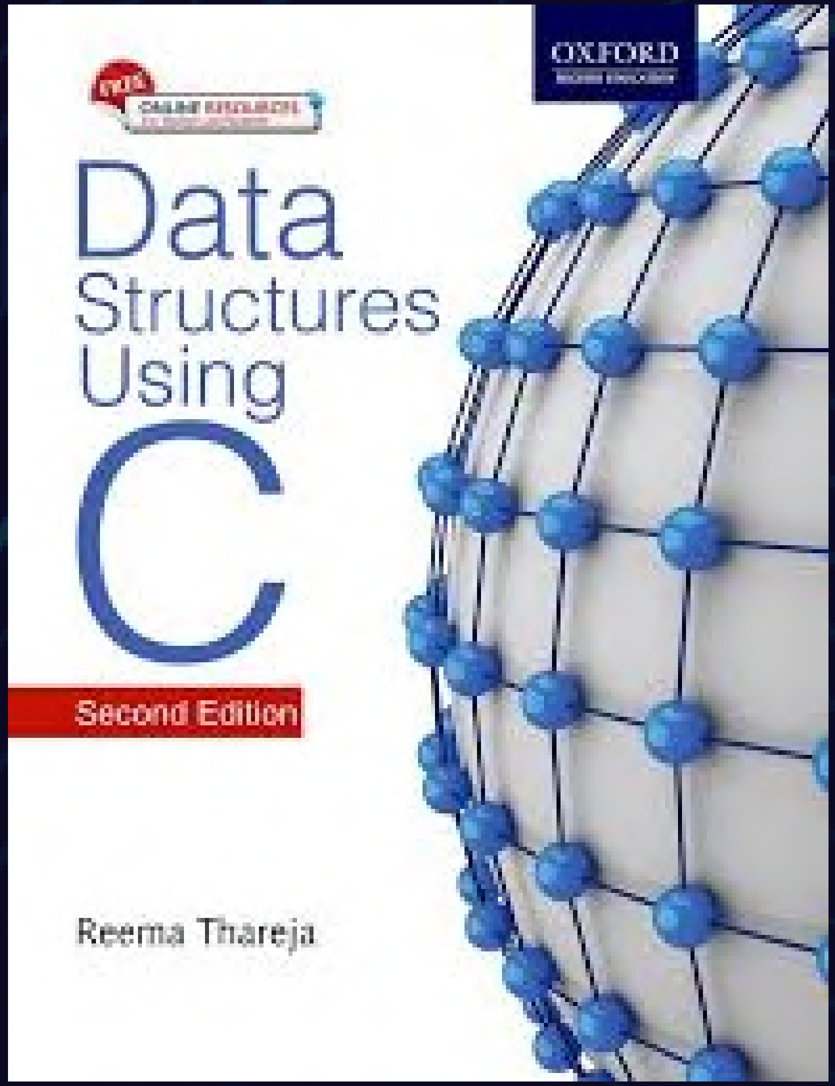
SESSION 7

INTRO TO TREES AND BINARY TREES

Instructor :
Harsh Raj
(BTech. AI&DS 2023-2027)

Contents inspired from.....




चलिये शुरू
करते हैं

What is a **Tree** data structure?

TechnoCrats GSV

A tree is a type of non-linear data structure that organizes data in a hierarchical (top to bottom) way.

```
        A
      /   \
    B       C
   / \     / \
  D   E   F   G
```

TechnoCrats
GSV

How's it different from linear data structures?

Feature	Linear Data Structure	Tree Data Structure (Non-Linear)
Structure	Arranged in a **sequence (line)**	Arranged in a **hierarchical (tree-like)** structure
Connection	Each element is connected to **one or two elements** (before/after)	Each node can be connected to **many children**
Traversal	Traversed in **one direction** (e.g., left to right)	Can be traversed **in multiple directions** (top-down, left-right)
Examples	Array, Linked List, Stack, Queue	Tree, Binary Tree, N-ary Tree
Data Access	Mostly **sequential**	Access is **hierarchical and fast** for search

Types of Trees

1. General Trees
2. Forests
3. Binary Trees
4. Binary Search Trees
5. Expression Trees
6. Tournament Trees

TECHNOCRATS GSV

Some Key Terminologies

- root : The topmost node in the tree.
    - It has no parent.
    - Every tree has exactly one root.

- node : A single element or point in the tree.
    - It can be a root, parent, child, or leaf.

- child : A node that comes from another node.
    - It's connected below a parent.

- parent : A node that has one or more children.

- leaf : A node that has no children.
    - It is the end of a branch.

Some Key Terminologies

- subtree : A portion of a tree that forms its own smaller tree.
  - It includes a node and all its descendants.

- level : The distance from the root, measured in steps.
  - Root is at level 0.
  - Its children are at level 1.
  - Their children are at level 2, and so on.

- path: A sequence of consecutive edges is called a path.

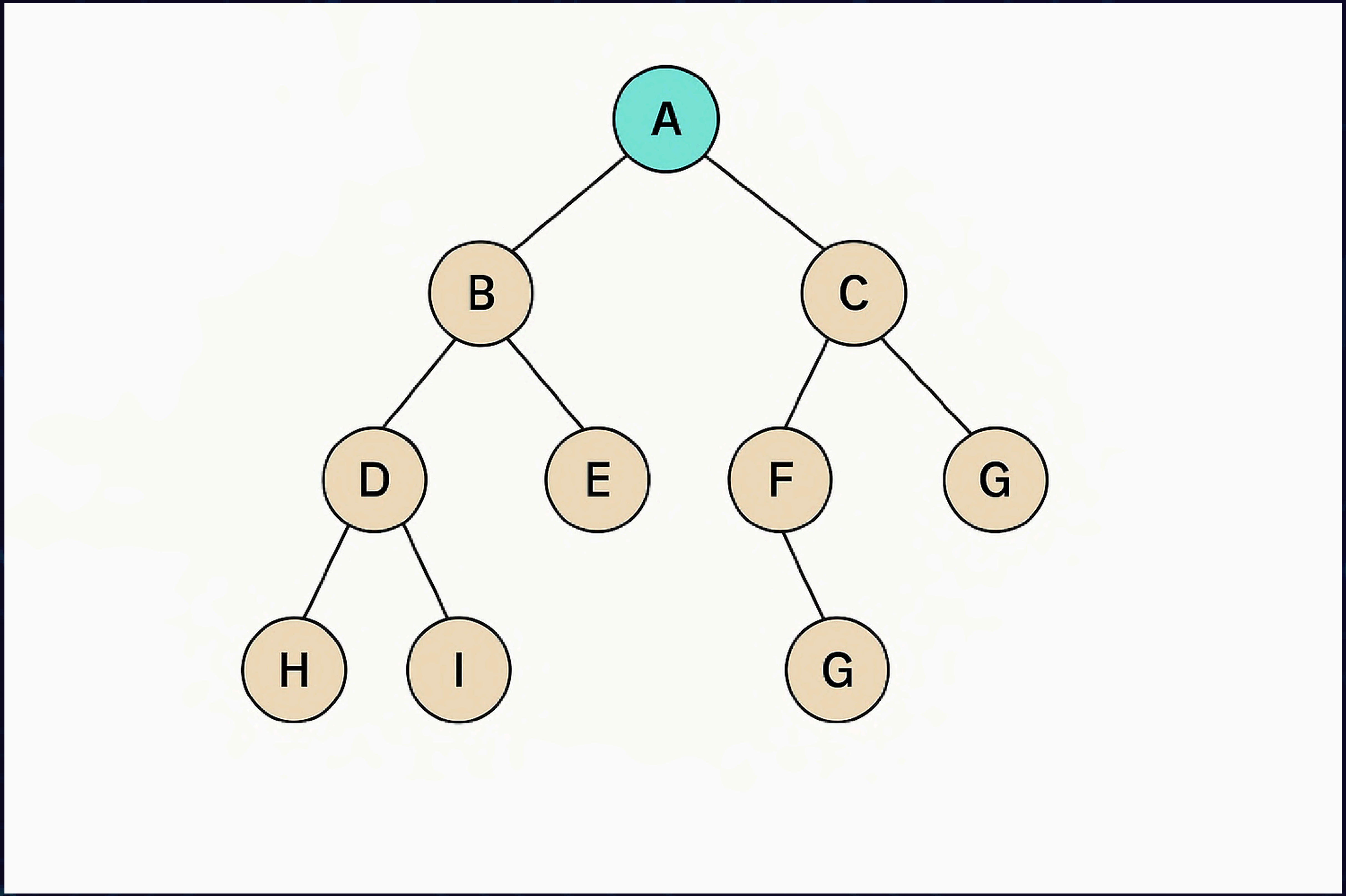
- Ancestor node: An ancestor of a node is any predecessor node on the path from root to that node

Some Key Terminologies

- Descendant node: A descendant node is any successor node on any path from the node to a leaf node.
- Degree: Degree of a node is equal to the number of children that a node has.
- In-degree: In-degree of a node is the number of edges arriving at that node.
- Out-degree: Out-degree of a node is the number of edges leaving that node.


Maza nahi aa raha hai

Let’s visualize this



Applications

1. File systems: Your computer’s file explorer is structured like a tree.

- Folders can contain subfolders/files (hierarchical)
- Easy to navigate, search, and manage

```
Root (C:/)
├── Users
│   └── Alice
│       ├── Documents
│       │   └── Resume.docx
│       └── Pictures
│           └── Photo.jpg
└── Program Files
    └── App
```

Applications

2. Decision Tress: Used to make decisions, like whether to approve a loan.

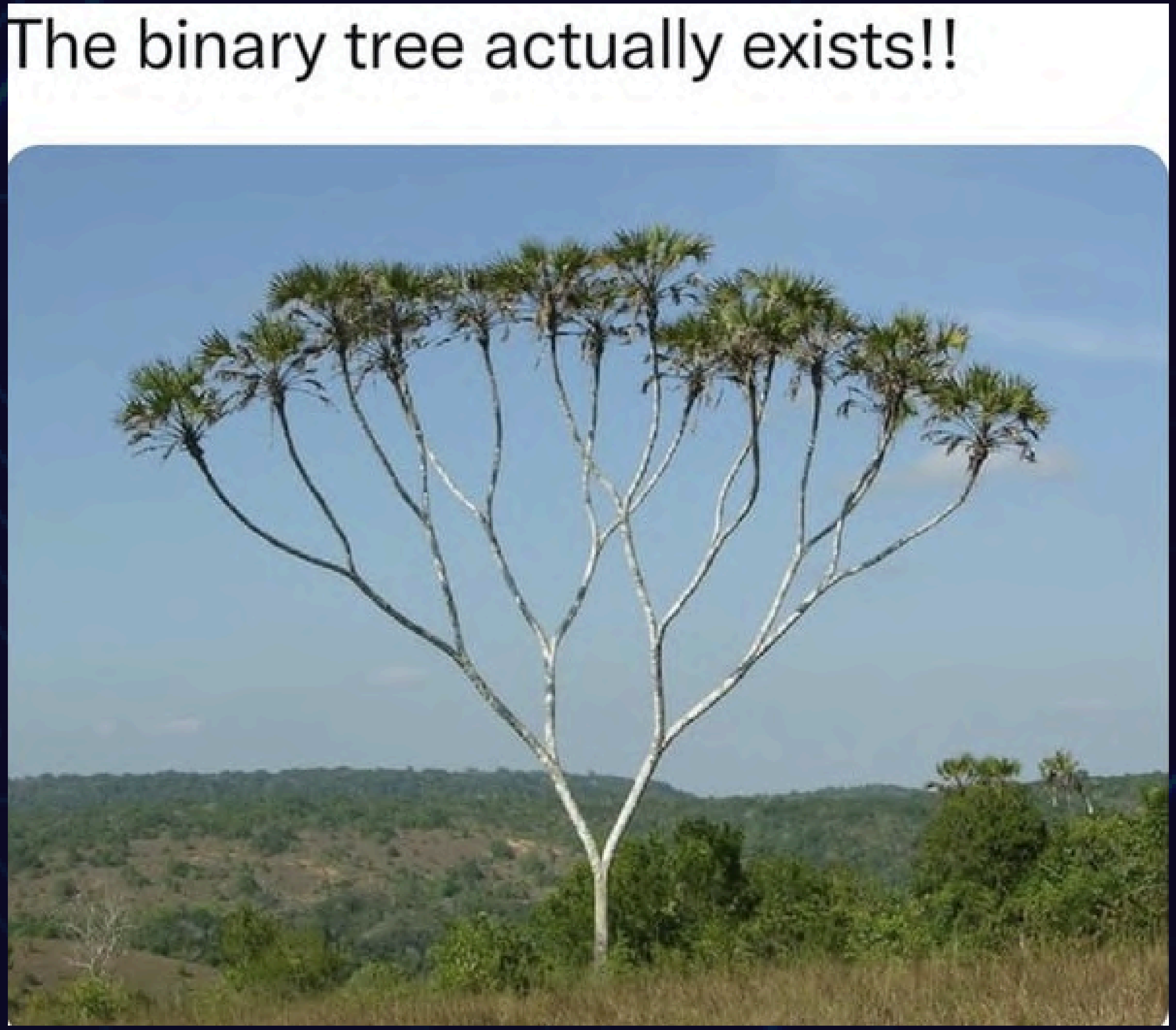
- Easy to understand and visualize decision logic
- Widely used in AI, data mining, and game strategy

```
Is income > $50K?
├── Yes → Has good credit?
│   ├── Yes → Approve Loan
│   └── No  → Reject Loan
└── No → Reject Loan
```

Binary Trees

A binary tree is a tree data structure having 0,1 and atmost 2 children.

Every node contains a data element, a left pointer and a right pointer.



Types of Binary Trees

1. Complete Binary Trees

A complete binary tree is a binary tree that satisfies two properties:

1. Every level, except last, is completely filled. Why? It is filled from left to right
2. All nodes appear as far left as possible.

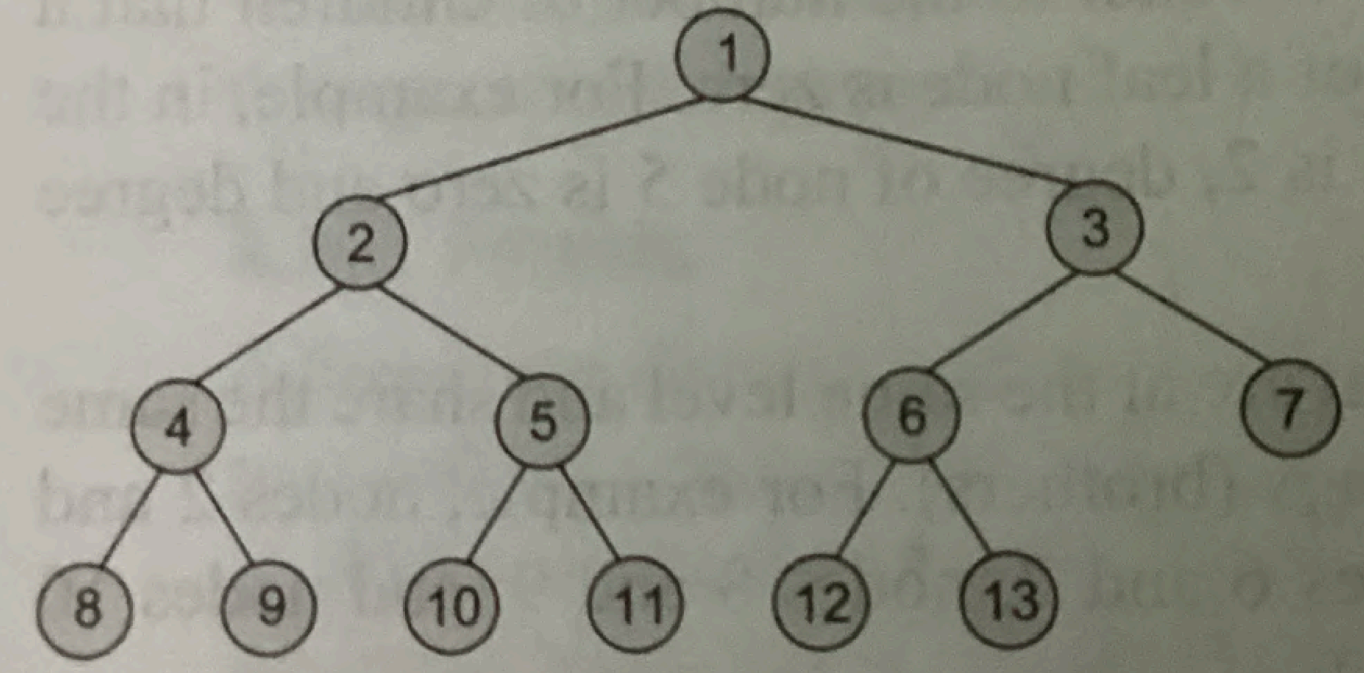


Figure 9.7 Complete binary tree

Some numericals based on CBT

1.Number of nodes at a level n(at most) = 2^n

2. Left child node = 2*k

3. Right child node = 2*k + 1

4. Parent of node = child node’s k//2 (floor division)

5. Height of tree = |log base 2 (number of nodes + 1)|

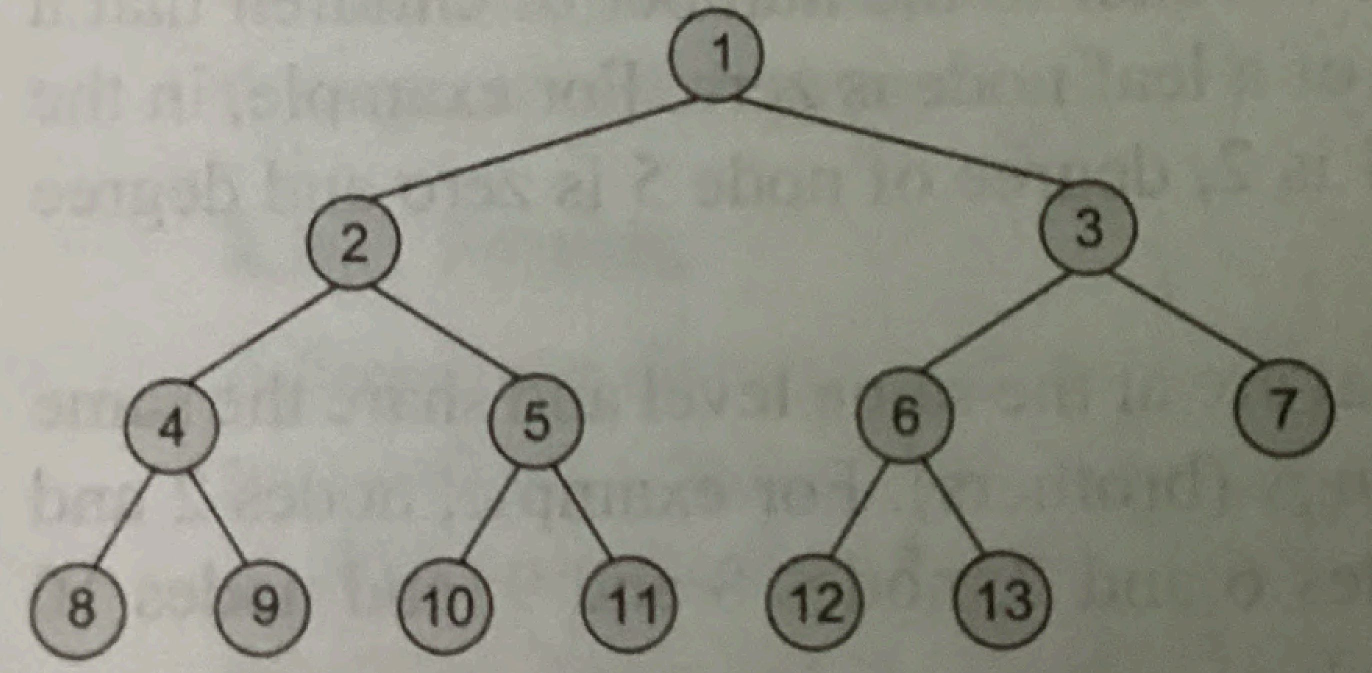
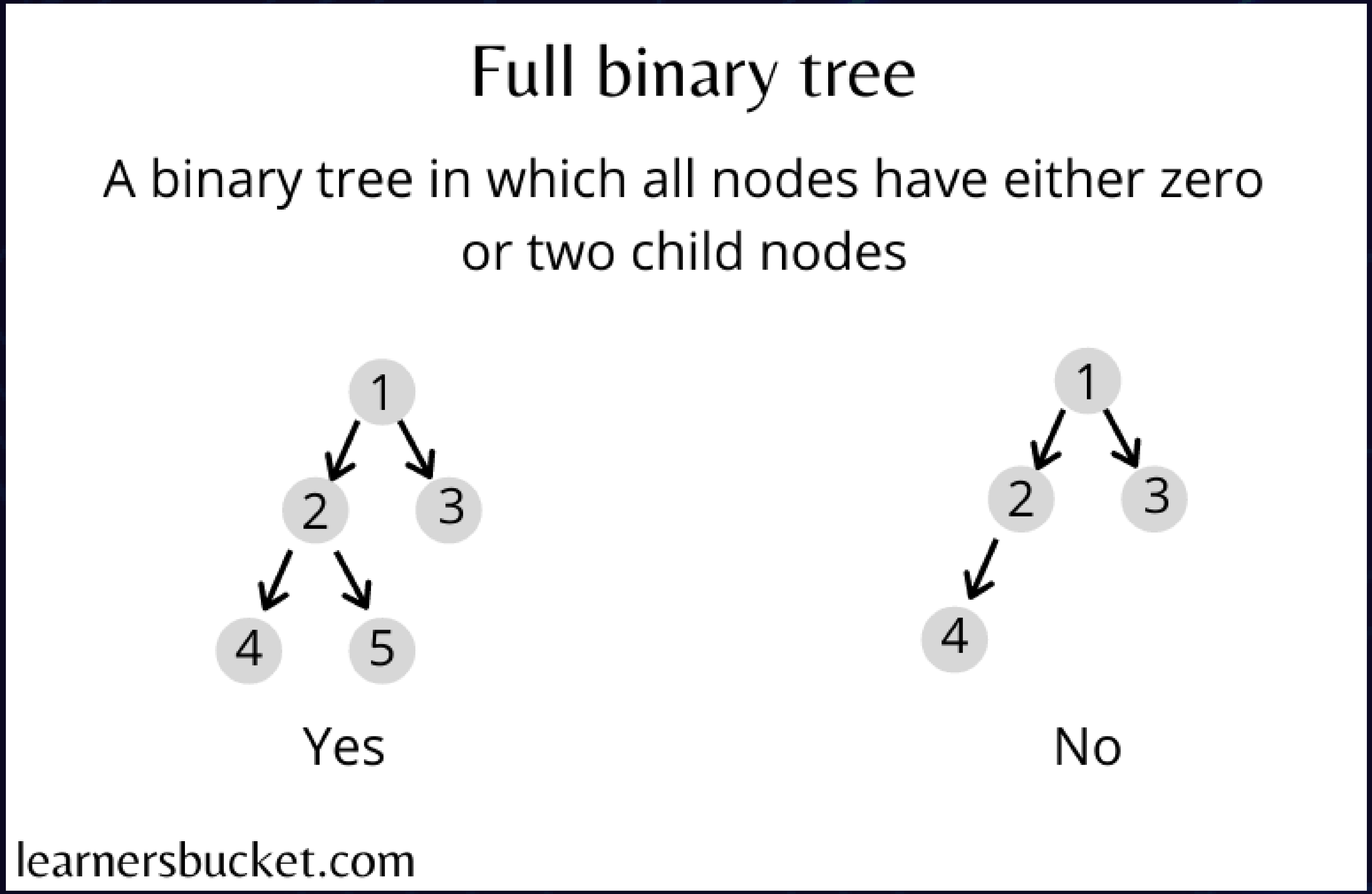


Figure 9.7 Complete binary tree

Types of Binary Trees

2.Full Binary Trees

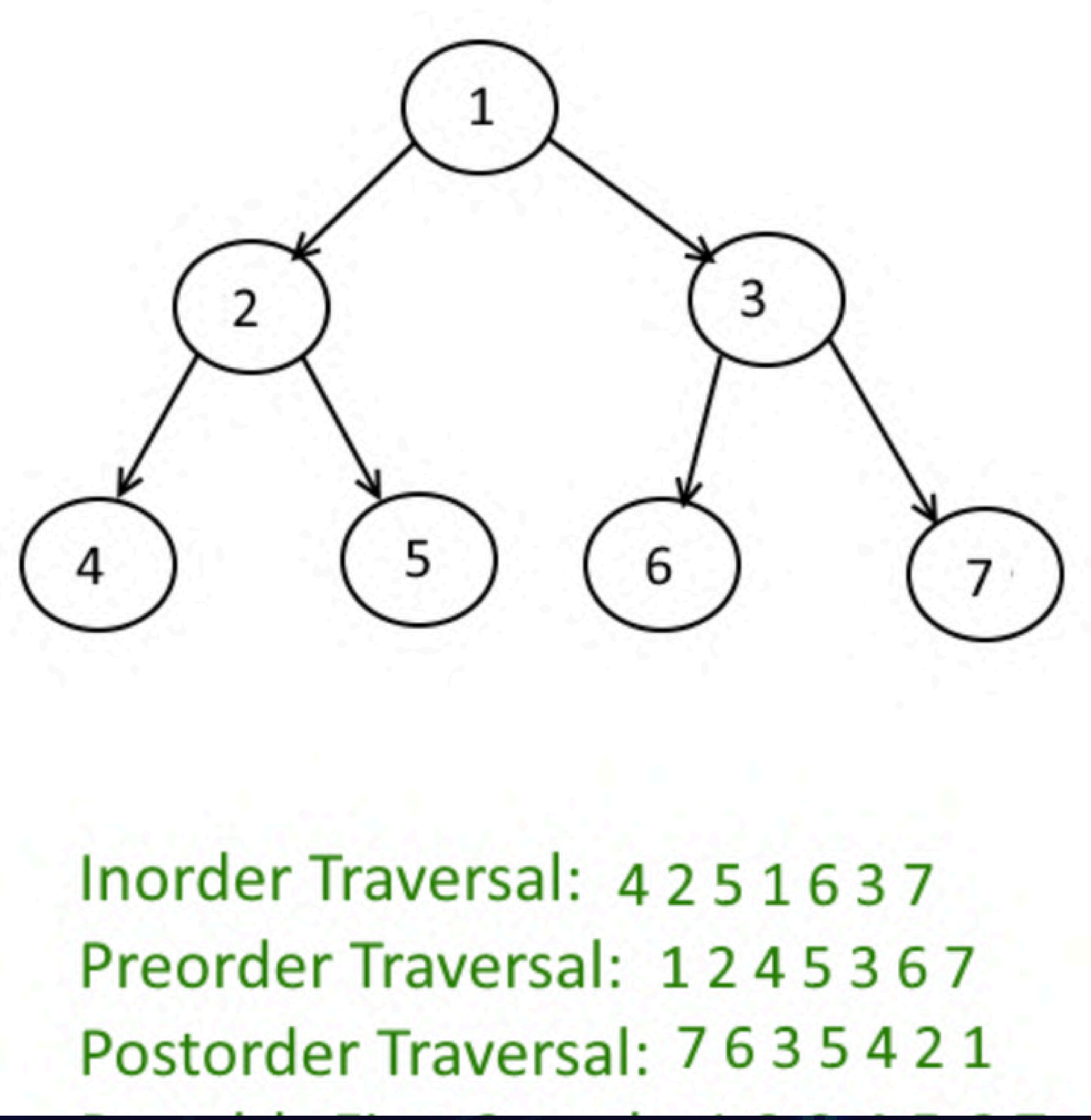
A full binary tree is a binary tree where every node has either zero or two children. This structure ensures a balanced tree



Trees Traversal

There are three types of Traversals in BT:

1. In-order
2. Pre-order
3. Post-Order



Trees Traversal

In-order:

1. Traversing the left sub tree
2. Visiting the root node
3. Traversing the right sub tree

Pseudo Code:

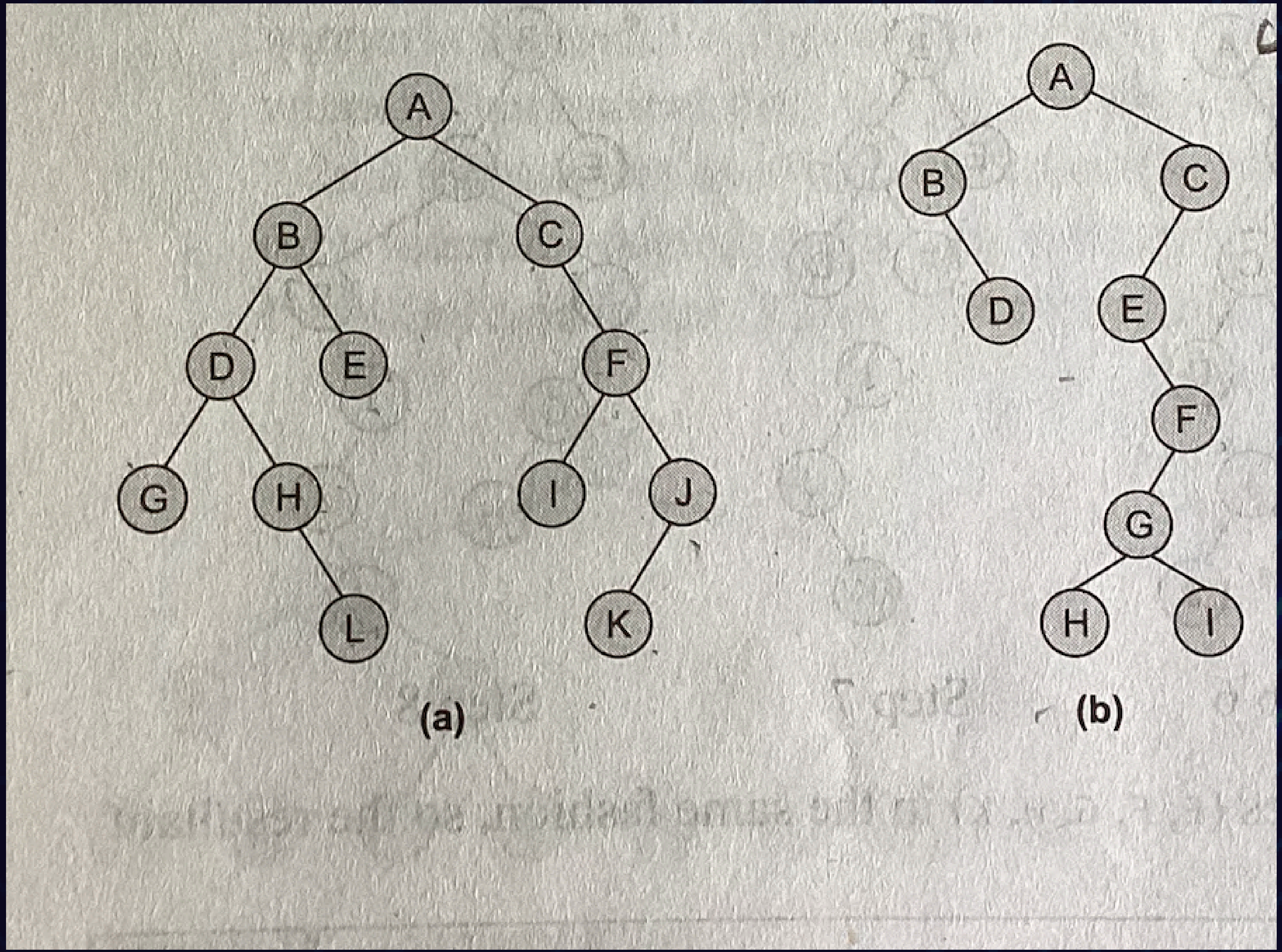
```
Function InorderTraversal(node):
    if node is NULL:
        return
    InorderTraversal(node.left)
    Visit(node)  // e.g., print node.value
    InorderTraversal(node.right)
```

Algorithm

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:             INORDER(TREE -> LEFT)
Step 3:             Write TREE -> DATA
Step 4:             INORDER(TREE -> RIGHT)
        [END OF LOOP]
Step 5: END
```

Figure 9.17 Algorithm for in-order traversal

Examples:



Pre-order:

1. Visiting the root node
2. Traversing the left sub tree
3. Traversing the right sub tree

Pseudo Code:

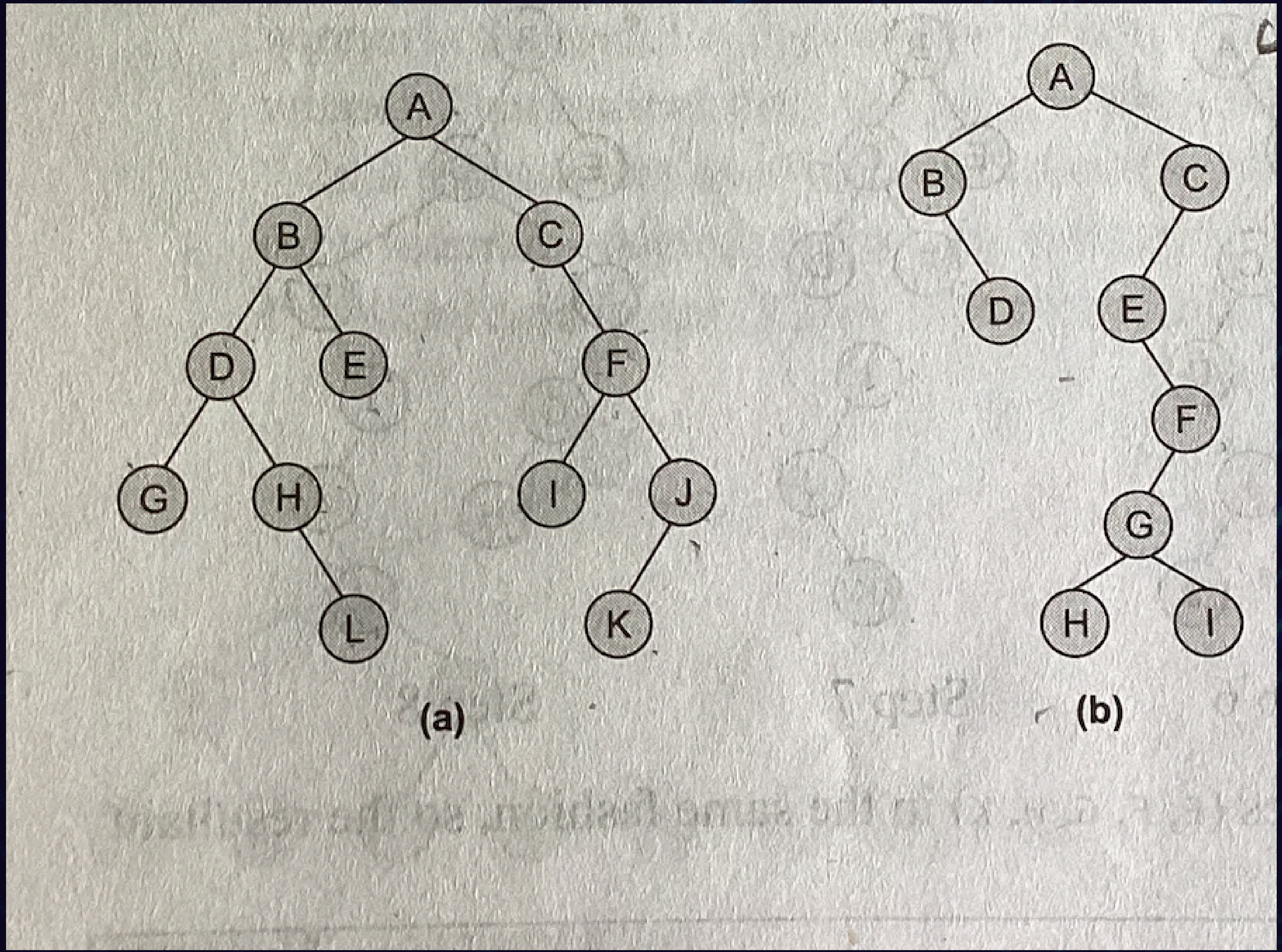
```
Function PreorderTraversal(node):
    if node is NULL:
        return
    Visit(node)          // e.g., print node.value
    PreorderTraversal(node.left)
    PreorderTraversal(node.right)
```

Algorithm

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:         Write TREE -> DATA
Step 3:         PREORDER(TREE -> LEFT)
Step 4:         PREORDER(TREE -> RIGHT)
            [END OF LOOP]
Step 5: END
```

Figure 9.16 Algorithm for pre-order traversal

Examples



Post-order:

1. Traversing the left sub tree
2. Traversing the right sub tree
3. Visiting the root node

Pseudo Code:

```
Function PostorderTraversal(node):
    if node is NULL:
        return
    PostorderTraversal(node.left)
    PostorderTraversal(node.right)
    Visit(node)     // e.g., print node.value
```

Algorithm

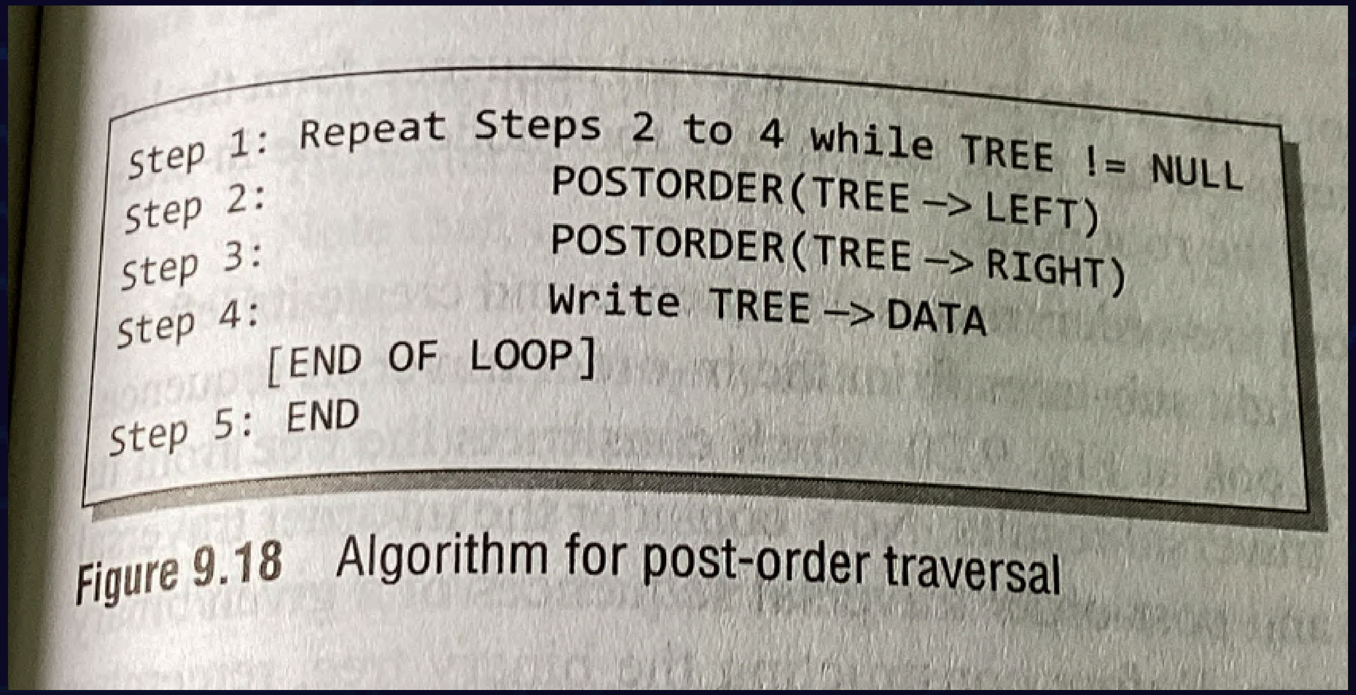
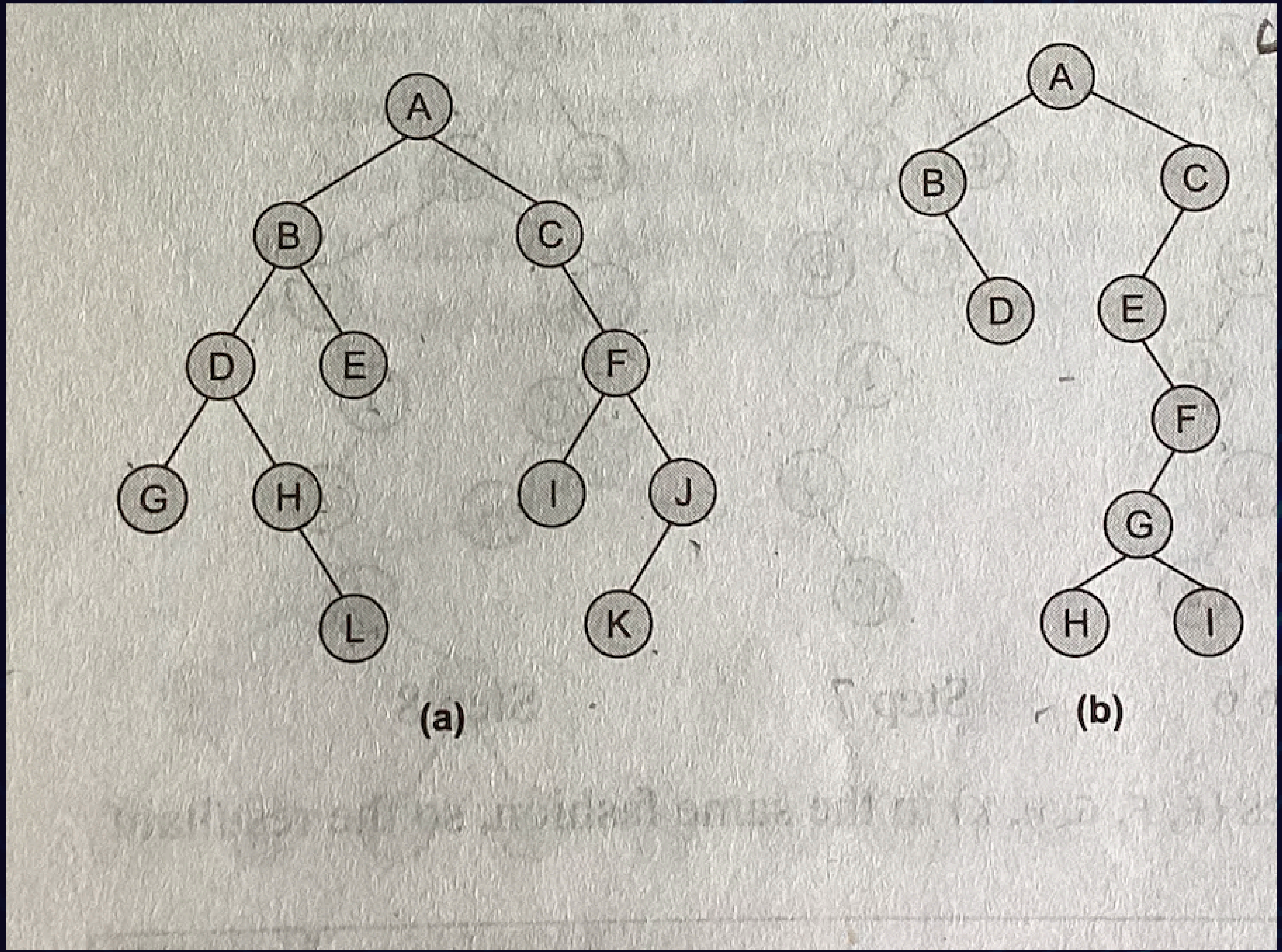

Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2: POSTORDER(TREE -> LEFT)
Step 3: POSTORDER(TREE -> RIGHT)
Step 4: Write TREE -> DATA
[END OF LOOP]
Step 5: END

Figure 9.18 Algorithm for post-order traversal

Examples




THANK YOU
FOR YOUR ATTENTION