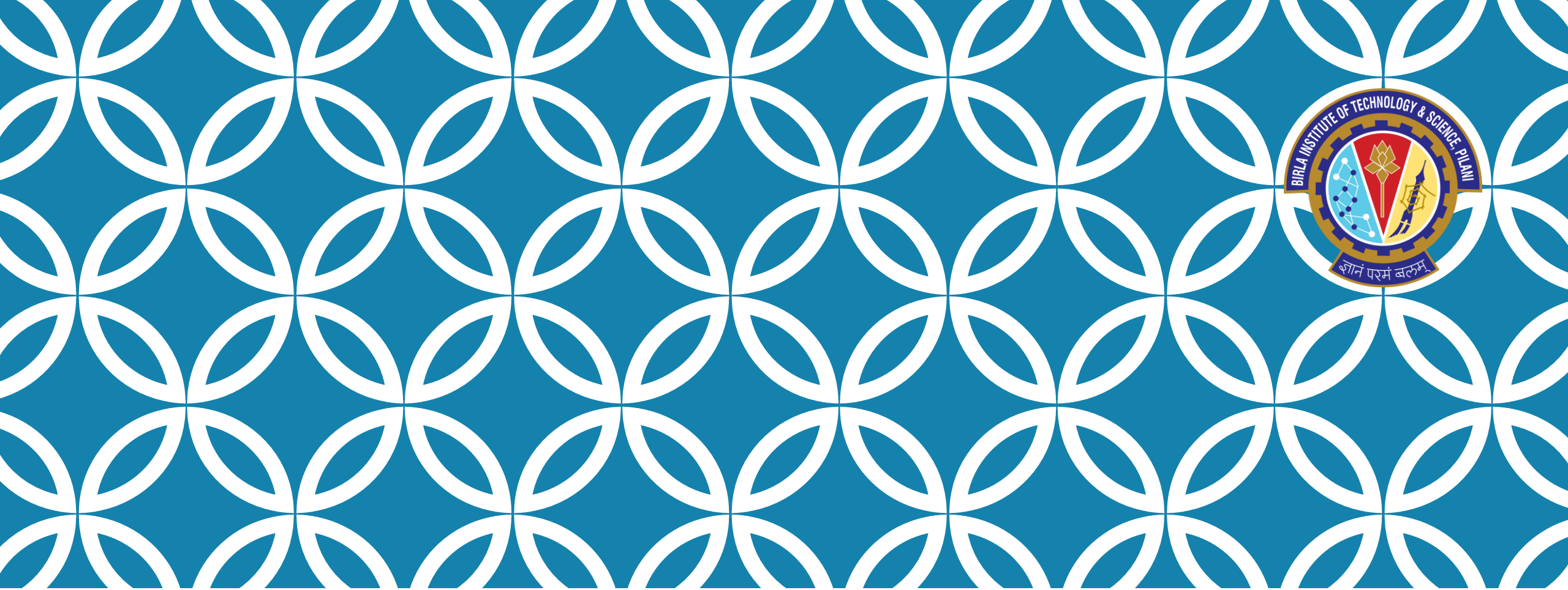


BITS F232: FOUNDATIONS OF DATA STRUCTURES & ALGORITHMS (1ST SEMESTER 2023-24) TREE ADT CONTINUED…

Chittaranjan Hota, PhD
Sr. Professor of Computer Sc.
BITS-Pilani Hyderabad Campus
hota[AT]hyderabad.bits-pilani.ac.in

© C R Hota, BITS Hyderabad

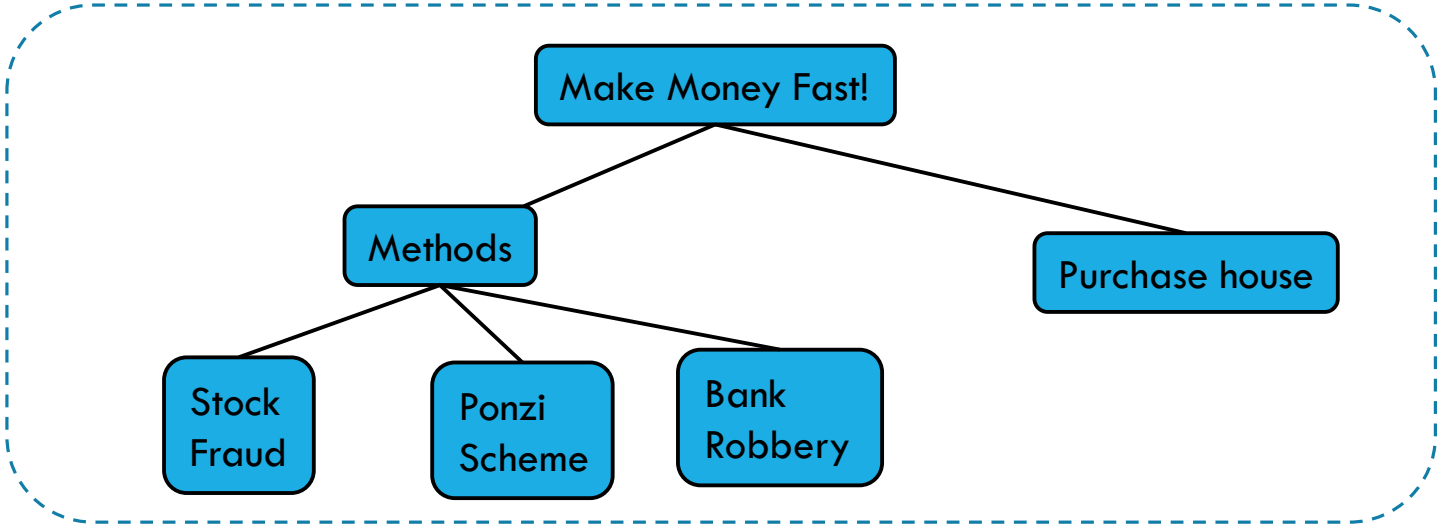
TREE TRAVERSAL ALGORITHMS

```
void iterativeLevelOrder(TreeNode* root)
{
    if (root == NULL)
        return;

    queue<TreeNode*> treeQueue;
    treeQueue.push(root);

    while (treeQueue.empty() == false)
    {
        TreeNode* currNode = treeQueue.front();
        treeQueue.pop();
        cout << currNode->data << " ";

        if (currNode->left != NULL)
            treeQueue.push(currNode->left);

        if (currNode->right != NULL)
            treeQueue.push(currNode->right);
    }
}
```

```
void printLevelOrder(node* root)
{
    int h = height(root);
    int i;
    for (i = 0; i < h; i++)
        print_current_level(root, i);
}
void print_current_level(Node* root, int level_no)  {
    if (!root) return;
    if (level_no == 0) {
       printf("%d -> ", root->value);
    }
    else {
        print_current_level(root->left, level_no - 1);
        print_current_level(root->right, level_no - 1);
    }
}
```

TREE TRAVERSAL: DEPTH-FIRST (PREORDER)

- In a preorder traversal, a node is visited before its descendants.
- Applications: print a structured document, get the prefix expression on an expression tree.




```
20  void printPreorder(struct Node* node)
21  {
22      if (node == NULL)
23          return;
24      cout << node->data << " ";
25
26      printPreorder(node->left);
27
28      printPreorder(node->right);
29  }
```

```
Preorder traversal of binary tree is
1 2 4 5 3
```

Algorithm *preOrder* (T, p) {

}

Lab 8 after mid sem

```
procedure iterativePreorder(node)
    if node = null
        return
    stack ← empty stack
    stack.push(node)
    while not stack.isEmpty()
        node ← stack.pop()
        visit(node)
        // right child is pushed first
        if node.right ≠ null
            stack.push(node.right)
        if node.left ≠ null
            stack.push(node.left)
```

iterative

POSTORDER TRAVERSAL

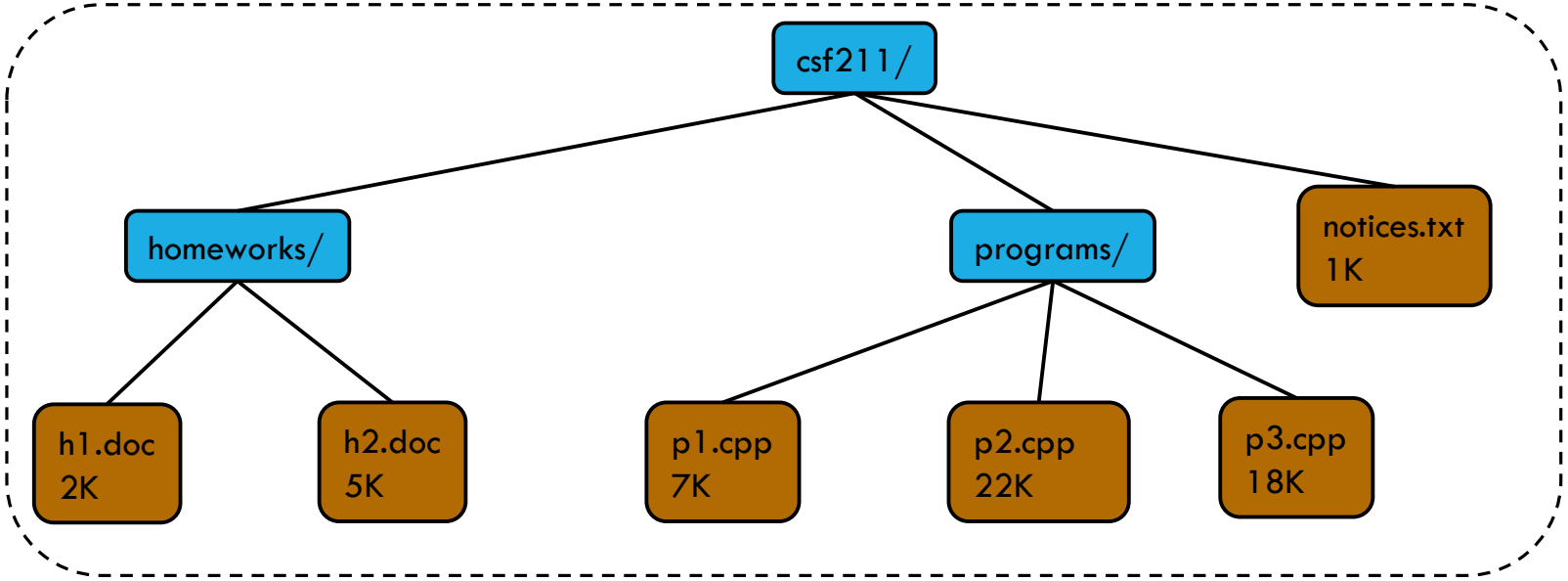
```
23  void printPostorder(struct Node* node)
24  {
25      if (node == NULL)
26          return;
27
28      printPostorder(node->left);
29
30      printPostorder(node->right);
31
32      cout << node->data << " ";
33  }
```

Lab 8 after mid sem

```
Postorder traversal of binary tree is
4 5 2 3 1
```

- In a postorder traversal, a node is visited after its descendants.
- Application: compute space used by files in a directory and its subdirectories, delete the tree, compute postfix expression…

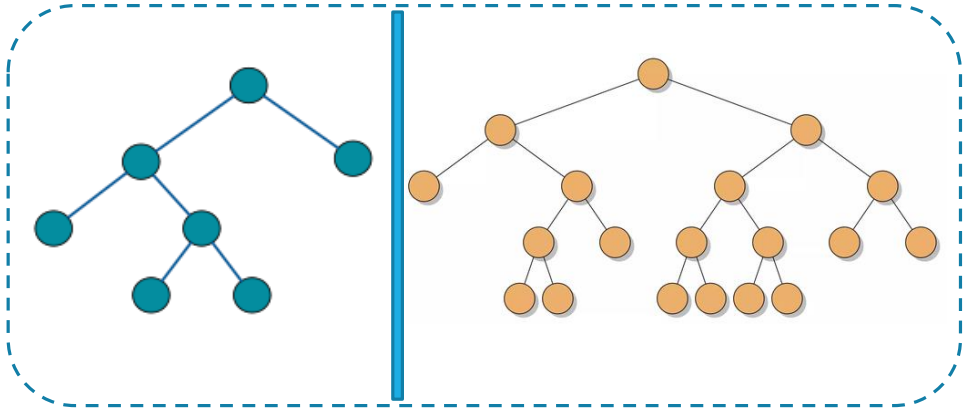
```
Algorithm postOrder(v){
    for each child w of v
        postOrder (w);
    visit(v);
}
```



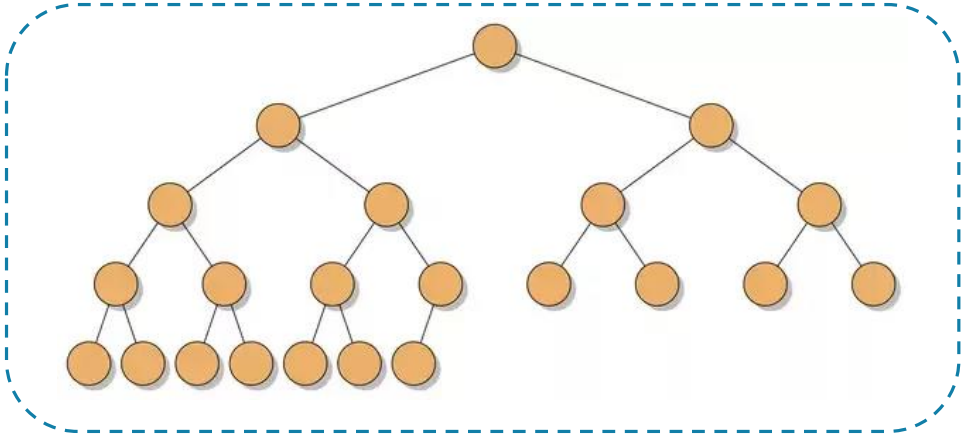
DEFINITIONS: BINARY TREE

- What is a binary tree?
- What is a full binary tree or proper binary tree?
  - All internal nodes have exactly ??? children
- What is a complete binary tree?
  - Every level except the last is filled. Last from left.
- Are the children of a binary tree ordered?

- Applications:
  - Problem representation (arithmetic expressions, decision trees)
  - Efficient algorithmic solutions (searching becomes faster, priority queues via heaps)
  - …



(what are these?)



(what is this?)