Module Code: CSE401

Session 12a: Tress

Session Speaker:

Prema Monish
premamonish.tlll@msruas.ac.in

Objectives

- Explain tree terminology and concepts
- Binary tree traversals

Contents

- Trees
- Binary trees
- Tress traversals

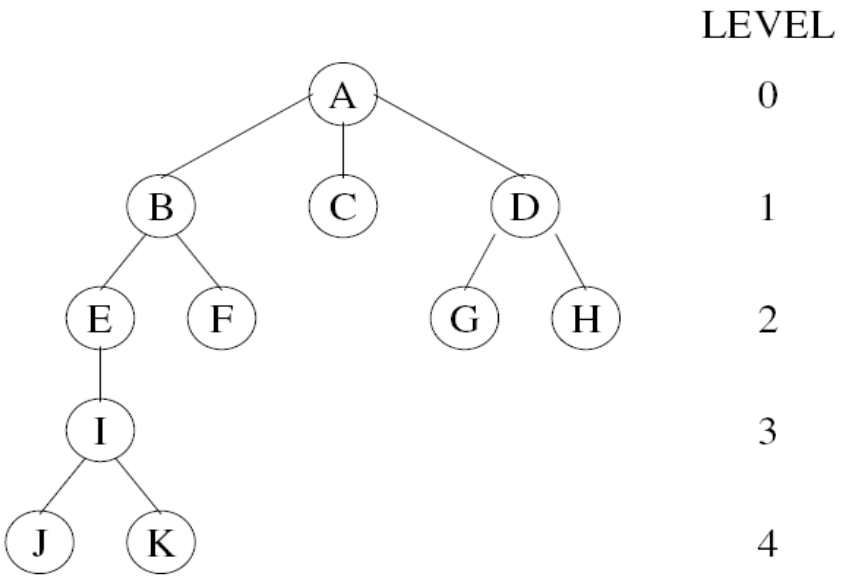
Trees

- Trees in Computer Science terminology refer to hierarchical structured data representations

- Trees consist of *nodes* with a parent-child relation

Tree Data Structures

- Tree has *root* at the top and branches grown *down* ending in *leaves*
- Each link in the root node refers to a child
- The left child is the first node in the left subtree
- Right child is the first node in the right subtree

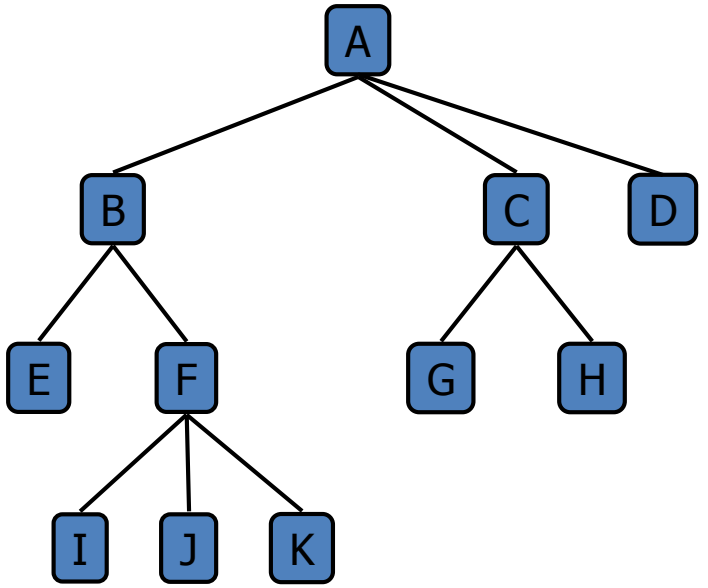


Terminology

- **Root**: A node without a parent
- **Internal Node**: A node with at least one child
- **Leaf (or External Node):** A node without a child
- **Ancestor of a node:** Parent, grand-parent, grand-grand-parent, etc.
- **Descendent of a node**: Child, grand-child, great-grand-child, etc.
- **Depth of a node:** Number of ancestors of the node
- **Height of the Tree:** Maximum depth of any node in the Tree

Example of a Tree

- *root* is node A
- *Internal* (branch) nodes are nodes A, B, C, F
- *External* nodes (*leaves*) are nodes E, I, J, K, G, H, D
- *Depth* of node F is 2
- *Height* of T is 3
- *Ancestors* of node H are C and A
- *Children* of node A are B, C and D
- Nodes B, C and D are *siblings*
- *Descendants* of node B are E, F, I, J and K



Tree Traversal

- The fundamental *algorithm* on Trees is Traversal
  - Visit each node of the Tree
  - Optionally perform some operation during each visit
- Types of Tree traversals
  1. Preorder traversal
  2. Postorder traversal
  3. Inorder traversal

Preorder Traversal

- Each node is visited before all of its descendents
- The steps for a preOrder traversal
  1. Process the value in the node
  2. Traverse the left subtree preorder
  3. Traverse the right subtree preorder

- The value in each node is processed as the node is visited
- Useful for prefix expression evaluation and processing structured documents (e.g., XML)

Postorder Traversal

- Each node is visited *after* its descendents
- The steps for a postOrder traversal
  1. Traverse the left subtree postOrder
  2. Traverse the right subtree postOrder
  3. Process the value in the node

- The value in each node is not printed until the values of its children are printed
- Useful for postfix expression evaluation and accumulation of values at ancestor nodes (*e.g.,* hierarchical disk usage information)

Inorder Traversal

- The steps for an inOrder traversal
  1. Traverse the left subtree inOrder
  2. Process the value in the node
  3. Traverse the right subtree inOrder

- The value in a node is not processed until the values in its left subtree are processed
- The inOrder traversal of a binary search tree prints the node values in ascending order

Traversal - Example

- Preorder

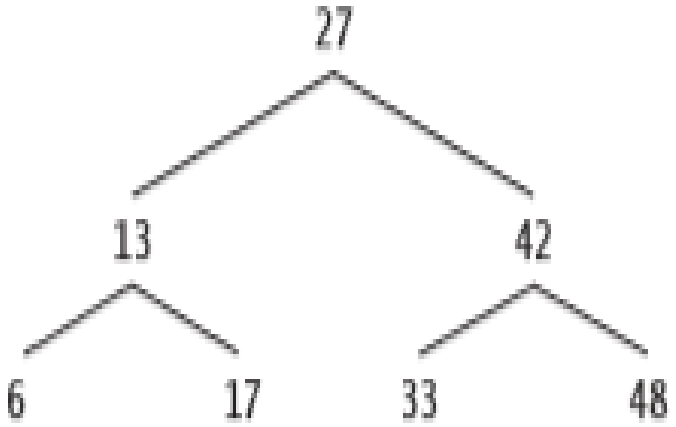
  27 13 6 17 42 33 48

- Postorder

  6 17 13 33 48 42 27

- Inorder

  6 13 17 27 33 42 48



Binary Trees

- Each node has at most two children
  - Left child and right child

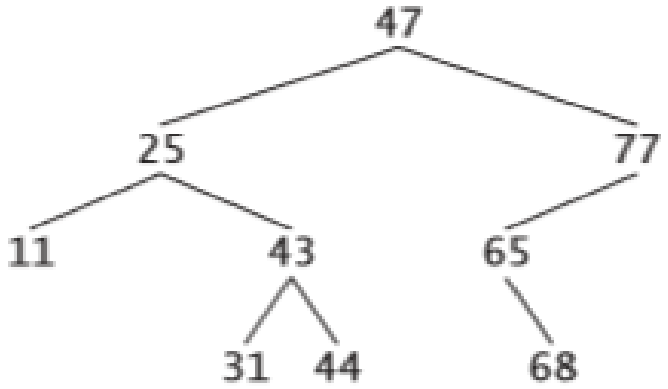
- Alternative *recursive definition*
  - A Binary Tree is either a Tree with a single node

    Or

  - A Tree whose root has an ordered pair of children, each of which is a Binary Tree
- Applications
  - Binary (*e.g.,* arithmetic) expression evaluation
  - Binary search algorithms
  - Compilers use binary tree to store data and retrieve data

Binary Search Tree

- A binary search tree (with no duplicate node values)
  - Values in any left subtree are less than the value in its parent node
  - Values in any right subtree are greater than the value in its parent
- The shape of the binary search tree that corresponds to a set of data can vary, depending on the order in which the values are inserted into the tree



Code for creating a node

```
struct node {
int data;
struct node *lchild, *rchild;
} ;
```

Crating new node:

```
node *get_node() {
 node *temp;
 temp = (node *) malloc(sizeof(node));
temp->lchild = NULL;
temp->rchild = NULL;
return temp;
}
```

To insert elements into node

```
void insert(node *root, node *new_node) {
 if (new_node->data < root->data) {
   if (root->lchild == NULL)
     root->lchild = new_node;
   else
     insert(root->lchild, new_node);
  }

  if (new_node->data > root->data) {
    if (root->rchild == NULL)
      root->rchild = new_node;
    else
      insert(root->rchild, new_node);
    }
}
```

Binary Tree traversal

Inorder traversal

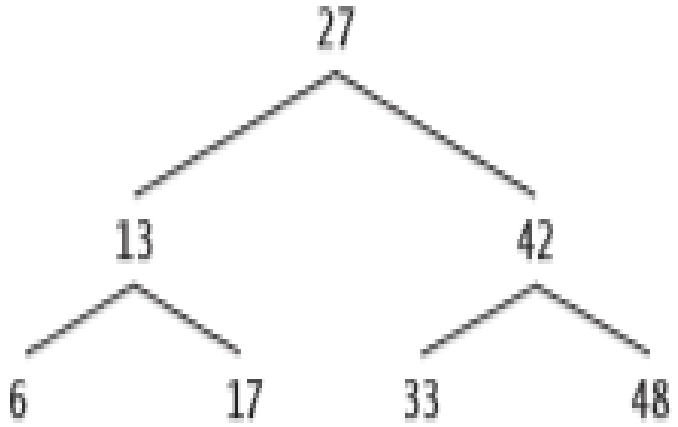
```
void inorder(node *temp) {
  if (temp != NULL) {
    inorder(temp->lchild);
    printf("%d", temp->data);
    inorder(temp->rchild);
  }
}
```

Post_order traversal:

```
void postorder(node *temp) {
  if (temp != NULL) {
    postorder(temp->lchild);
    postorder(temp->rchild);
    printf("%d", temp->data);
  }
}
```

Pre_order traversal:

```
void preorder(node *temp) {
  if (temp != NULL) {
    printf("%d", temp->data);
    preorder(temp->lchild);
    preorder(temp->rchild);
  }
}
```

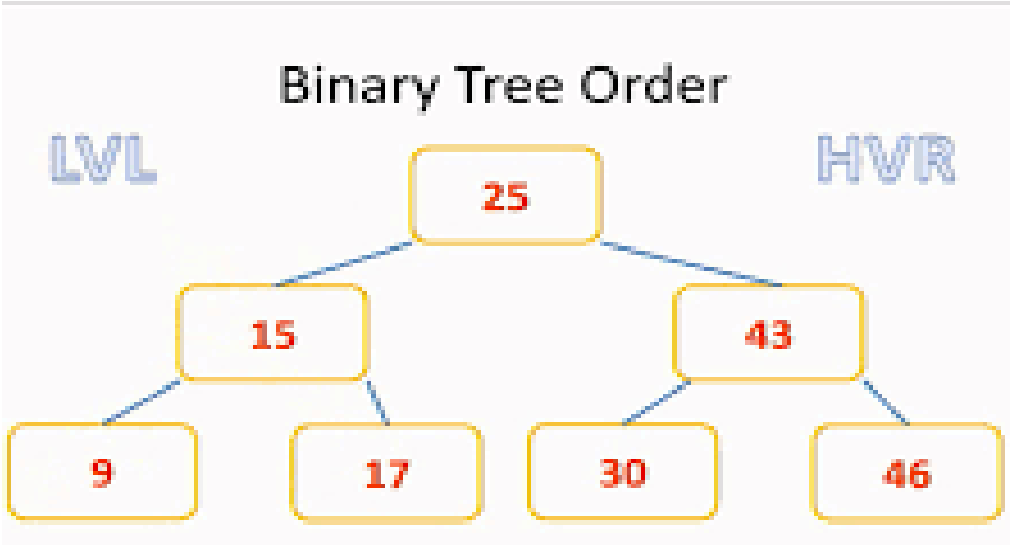


Binary Tree Sort

- Let's imagine we are given array of numbers

  25, 15, 43, 30, 46, 9, 17

- We are simply ordering numbers according to less value to the left and higher value to the right.



- Finally we get sorted array of numbers

  9, 15, 17, 25, 30, 43, 46 (Inorder traversal)

Binary Tree Sort

- Sort the given array of names using binary tree sort.

  Joe, harry, kevin, sally, Lyn, Dave, Freddie

Complete Binary Tree: A complete binary tree is a binary tree in which every level except possibly the last, is completely filled, and all nodes are as far left as possible.

Summary

- Basics of Concepts and terminology of trees such as parent, child, root, branch, degree, out-degree, in-degree are discussed
- Three binary tree traversals are:
  - Inorder
  - Preorder
  - postorder
- A binary tree is a tree in which no node can have more than two subtrees; the maximum outdegree for a node is two.