ITD62-124 Data Structure

Tree

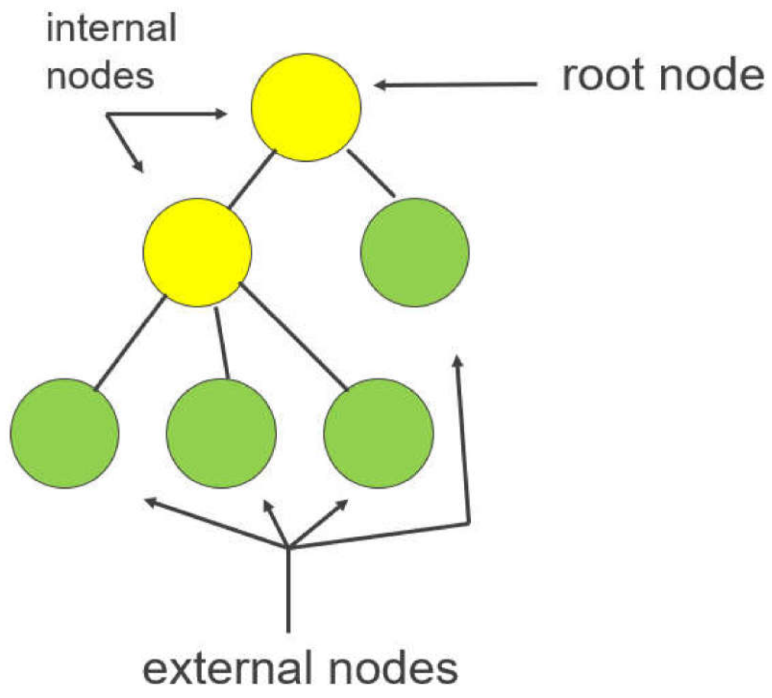
Outline:

- Introduction
- Binary Tree
- Binary Tree Traversal

Introduction

- A *tree* is a data structure accessed beginning at a *root* node
- Each node is either an *external node* (leaf) or an *internal node*
  - ✓ An internal node has 1 or more *children*, nodes that can be reached directly from that internal node
  - ✓ The internal node is said to be the *parent* of its child nodes

Introduction

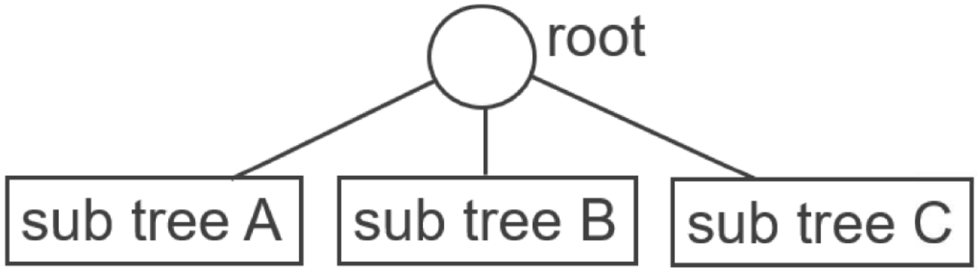


Introduction

- Formal Definition of a Tree

✓ A tree is either empty (no nodes) or

✓ a root connected to 0 or more trees (called sub trees)

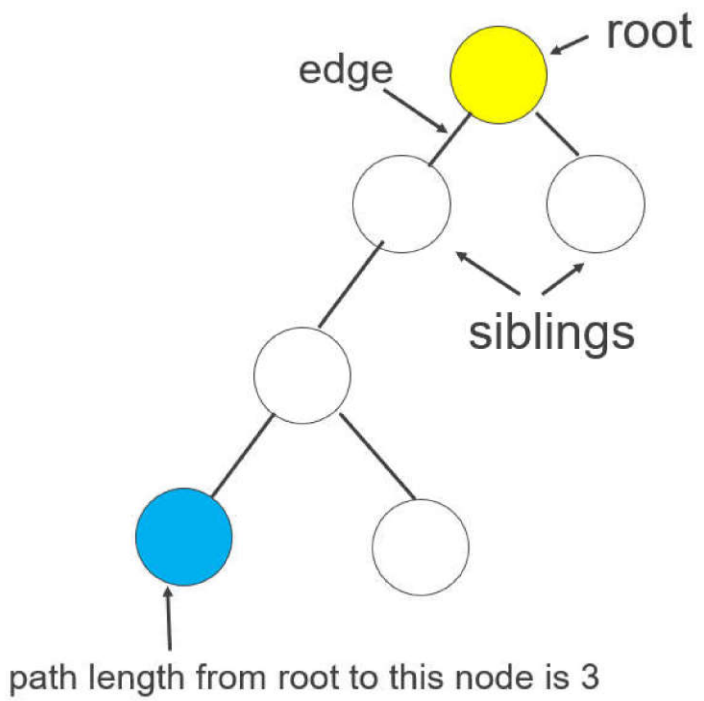


Introduction

- Properties of Trees and Nodes
  - ✓*edge*: the link from one node to another
  - ✓*siblings*: nodes that have the same parent
  - ✓*descendants*: any nodes that can be reached via 1 or more edges from this node
  - ✓*ancestors*: any nodes for which this node is a descendant
  - ✓*path length*: the number of edges that must be traversed to get from one node to another

Introduction

- Properties of Trees and Nodes



Introduction

- Properties of Trees and Nodes
  - ✓*depth:* the path length from the root of the tree to this node
  - ✓*degree:* the number of child node
  - ✓*height of a node:* The maximum distance (path length) of any leaf from this node
    - a leaf has a height of 0
    - the height of a tree is the height of the root of that tree

Introduction

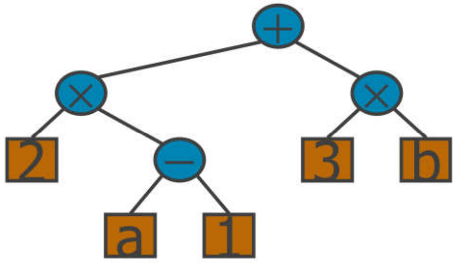
- Tree example: Arithmetic Expression Tree

✓ Binary tree associated with an arithmetic expression

  - ➢ internal nodes: operators
  - ➢ external nodes: operands

- Example: arithmetic expression tree for the expression

$$(2 \times (a - 1) + (3 \times b))$$



Introduction

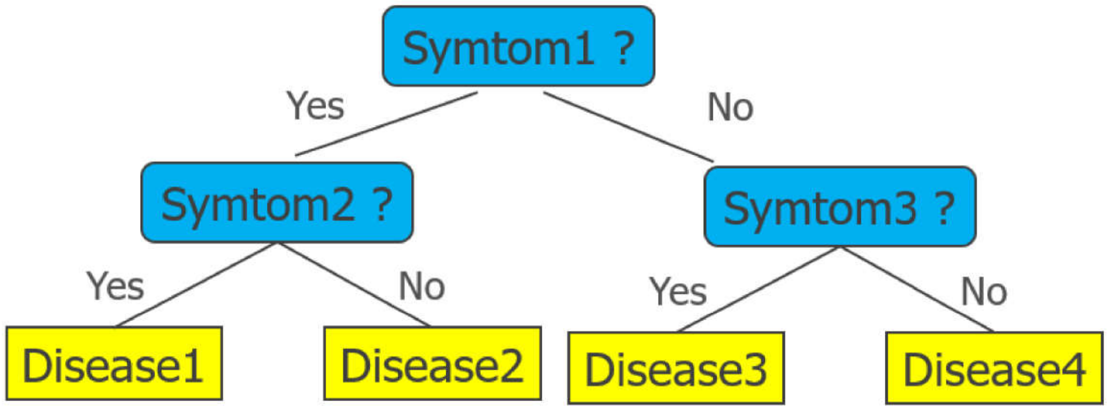
- Tree example: Decision Tree

✓ Binary tree associated with a decision process

  - ➢ internal nodes: questions with yes/no answer
  - ➢ external nodes: decisions

Introduction

- Tree example: Decision Tree

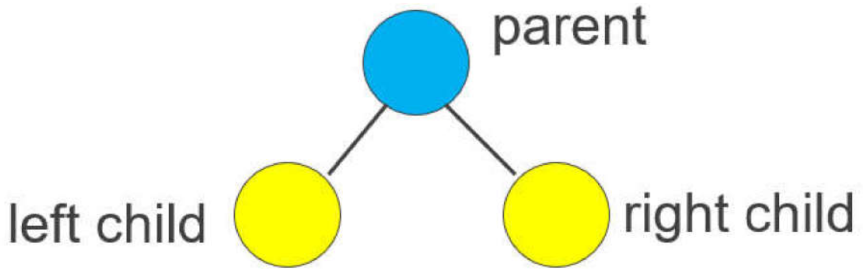


Binary Tree

- Binary tree:

✓ a tree with at most two children for each node

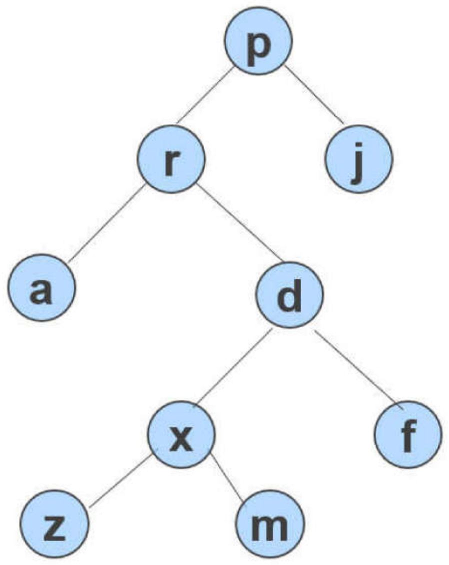
✓ the possible children are normally referred to as the left and right child



Binary Tree

- Full binary tree:

✓ a binary tree which each node was exactly 0 or 2 children



Binary Tree

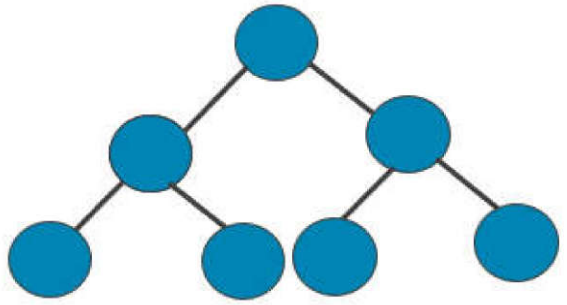
- Complete binary tree:

✓ a binary tree with all leaf nodes at the same depth

✓ all internal nodes have exactly two children

✓ has $2^{(n+1)} - 1$ nodes where *n* is the height of a tree

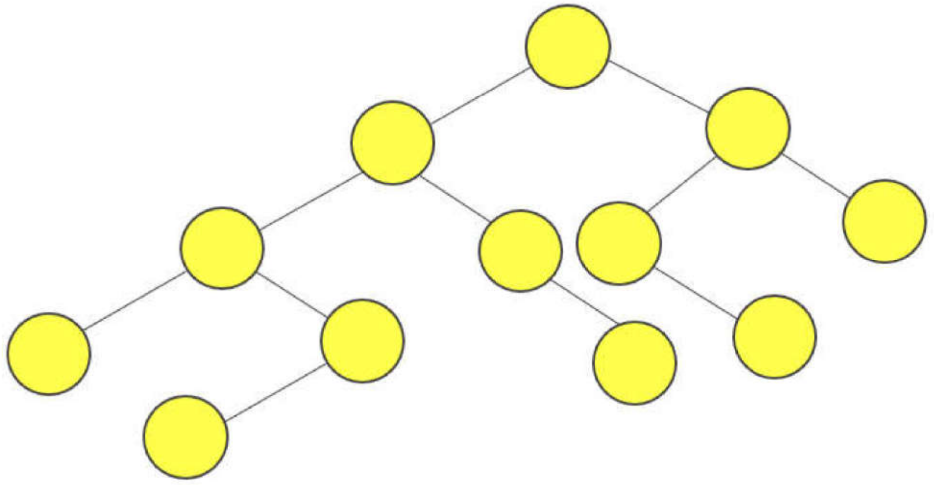
- ➢ height = 0 -> 1 node
- ➢ height = 1 -> 3 nodes
- ➢ height = 2 -> 7 nodes



Binary Tree

- Balanced binary tree:

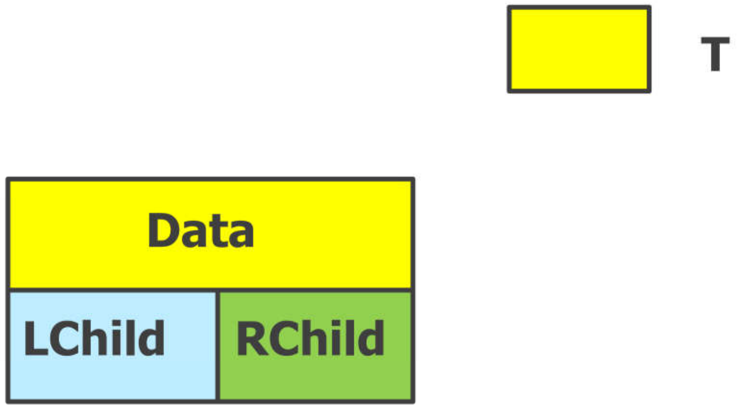
✓ a binary tree which the height of right subtree of any node differ from the height of left subtree ≤ 1



Binary Tree

- Binary tree implementation : Pointer

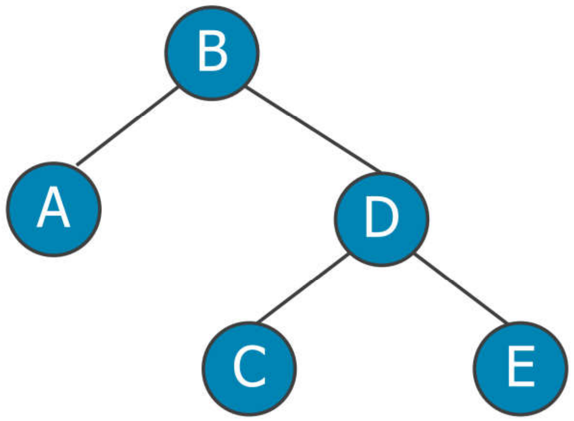
✓ A node is represented by an object storing
  - ➢ Data
  - ➢ Left child node
  - ➢ Right child node



Binary Tree

- Binary tree implementation : Pointer

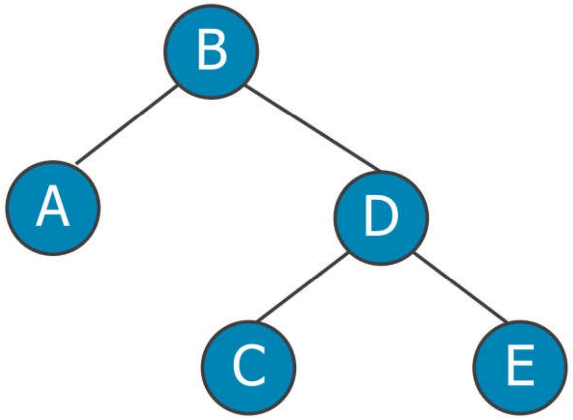
✓ Example:

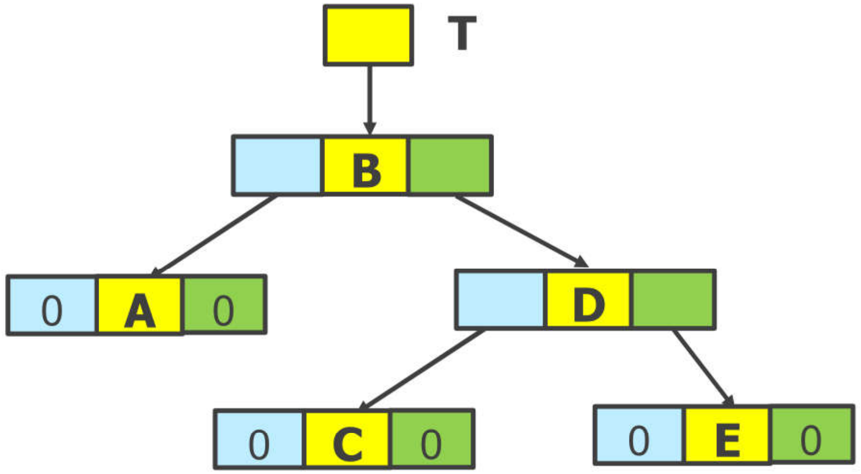


Binary Tree

- Binary tree implementation : Pointer

✓ Example:





Binary Tree Implementation: Python

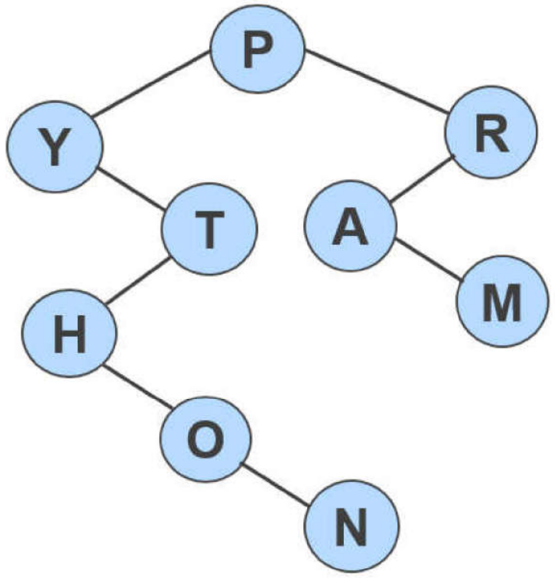
- Example:

```
class Node:
    def __init__(self):
        self.data = None
        self.leftChild = None
        self.rightChild = None


tree = Node( )
```

Binary Tree Traversal

- Traversal is the process of visiting every node once



Binary Tree Traversal

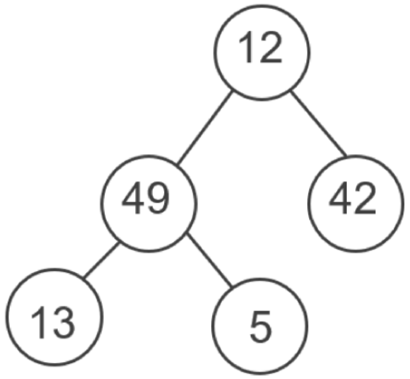
- There are 3 traditional types of traversals
  - ✓ Pre-order tree traversal: process the root, then process all subtrees (left to right)
  - ✓ In-order tree traversal: process the left subtree, process the root, process the right subtree
  - ✓ Post-order tree traversal: process the left subtree, process the right subtree, then process the root

Binary Tree Traversal

Pre-order Traversal:
1. Visit the root
2. Traverse left subtree
3. Traverse right subtree

Binary Tree Traversal

- Example:



✓Pre-order tree traversal: 12 49 13 5 42

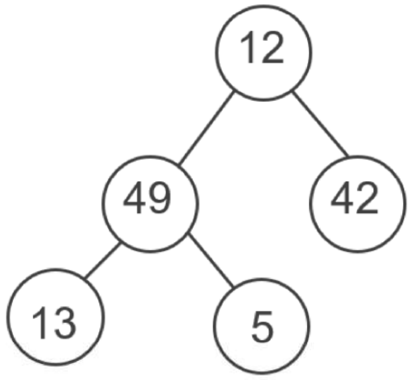
Binary Tree Traversal

<u>In-order Traversal</u>:

1. Traverse left subtree
2. Visit the root
3. Traverse right subtree

Binary Tree Traversal

- Example:



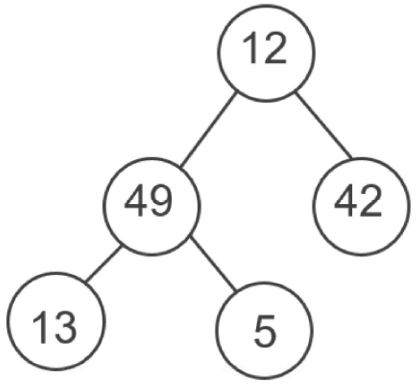
✓In-order tree traversal: 13 49 5 12 42

Binary Tree Traversal

Post-order Traversal:
1. Traverse left subtree
2. Traverse right subtree
3. Visit the root

Binary Tree Traversal

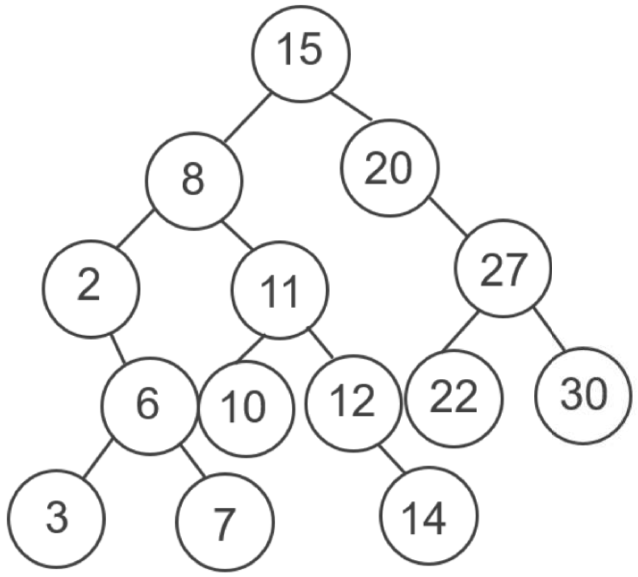
- Example:



✓Post-order tree traversal: 13 5 49 42 12

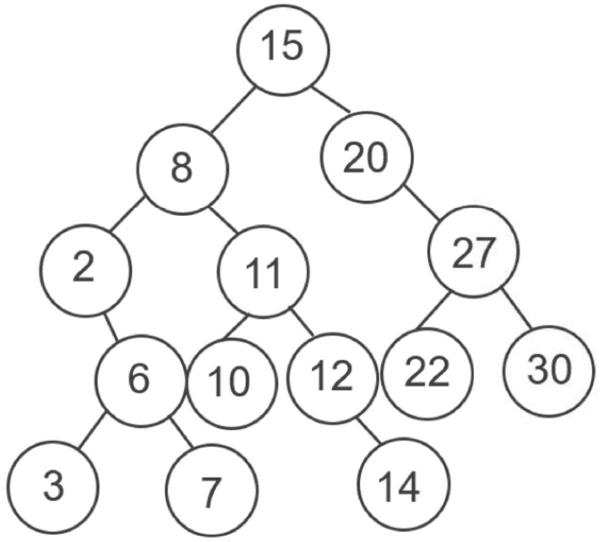
Binary Tree Traversal

- Example:



Binary Tree Traversal

- Example:



✓Pre-order tree traversal: 15 8 2 6 3 7 11 10 12 14 20 27 22 30

✓In-order tree traversal: 2 3 6 7 8 10 11 12 14 15 20 22 27 30

✓Post-order tree traversal: 3 7 6 2 10 14 12 11 8 22 30 27 20 15

Class Activity

Q & A

Formative Assessment