

**CSE1303 Part A**  
**Data Structures and Algorithms**  
**Summer Semester 2003**

**Lecture A12 – Binary Trees**

Kymerly Fergusson

**Overview**

- Trees.
- Terminology.
- Traversal of Binary Trees.
- Expression Trees.
- Binary Search Trees.

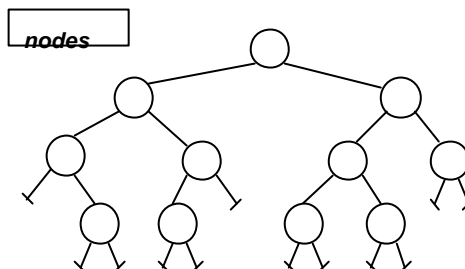
2

**Trees**

- Family Trees.
- Organisation Structure Charts.
- Program Design.
- Structure of a chapter in a book.

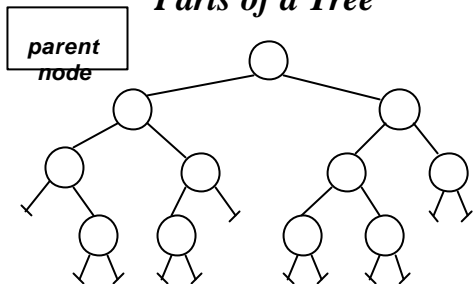
3

**Parts of a Tree**



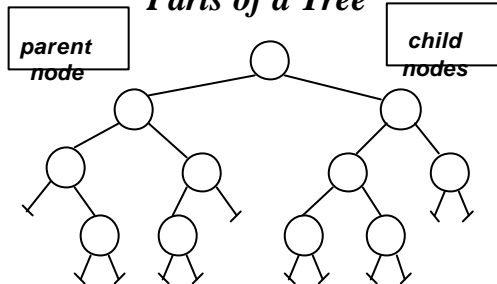
4

**Parts of a Tree**

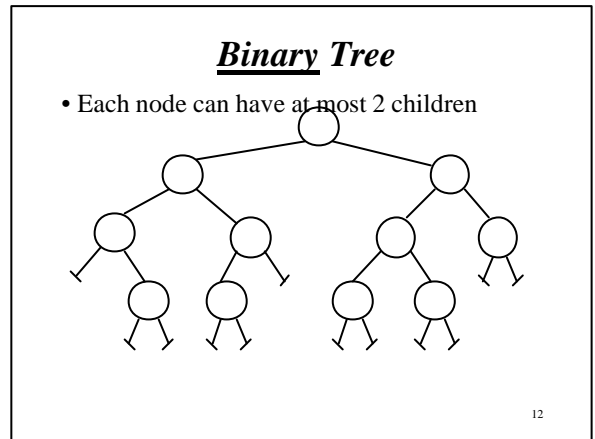
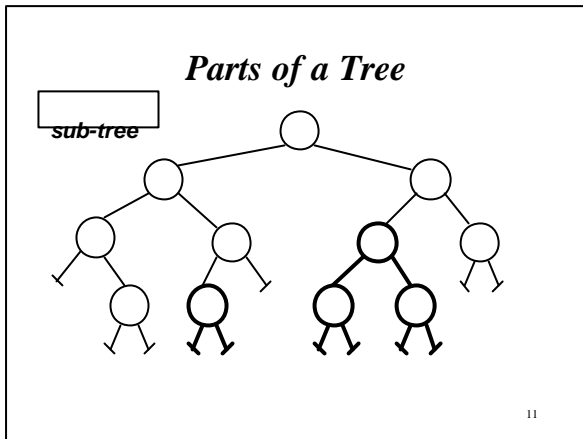
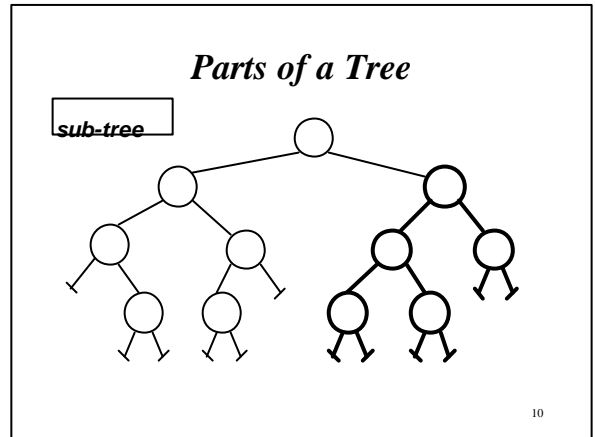
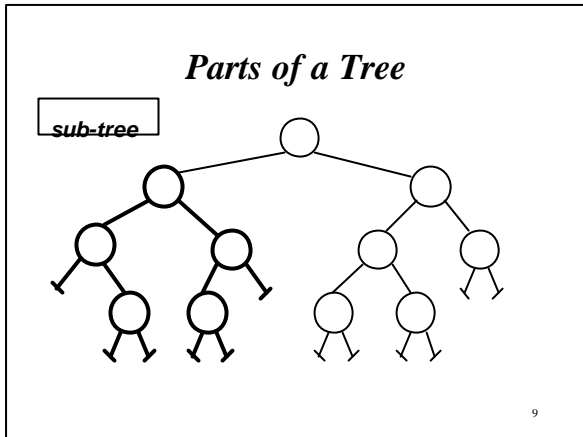
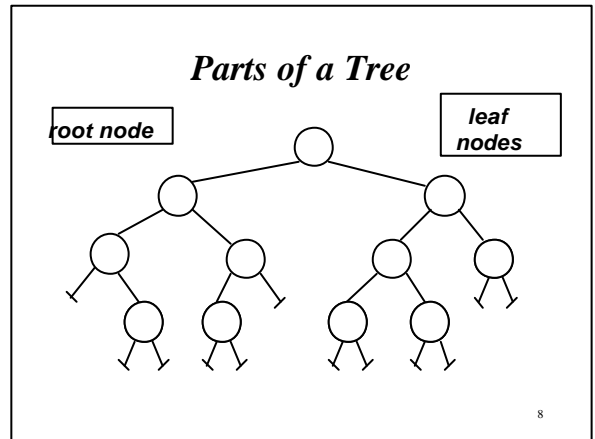
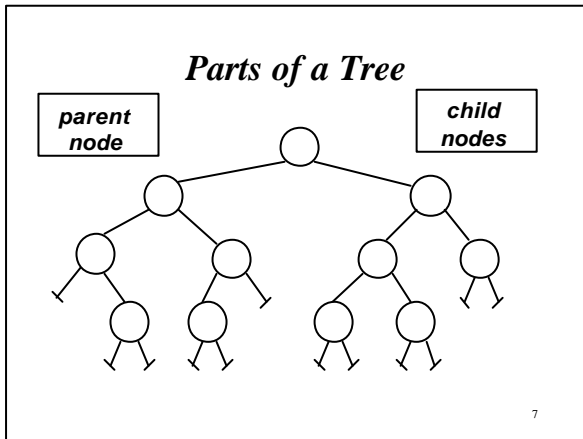


5

**Parts of a Tree**



6



### *Traversal*

- Systematic way of visiting all the nodes.
- Methods:
  - Preorder, Inorder, and Postorder
- They all traverse the left subtree before the right subtree.
- The name of the traversal method depends on when the node is visited.

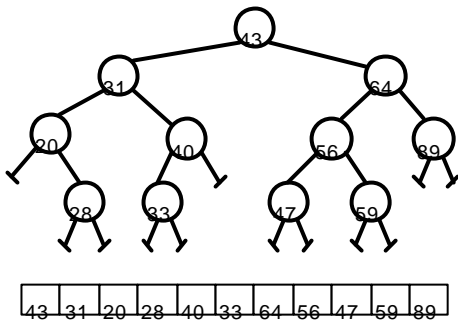
13

### *Preorder Traversal*

- Visit the node.
- Traverse the left subtree.
- Traverse the right subtree.

14

### *Example: Preorder*



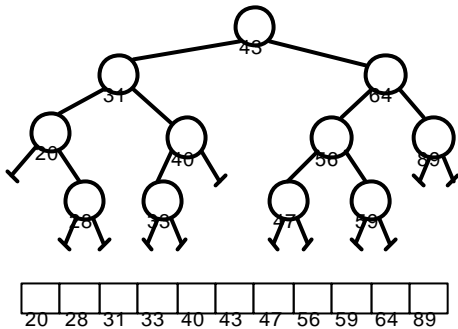
15

### *Inorder Traversal*

- Traverse the left subtree.
- Visit the node.
- Traverse the right subtree.

16

### *Example: Inorder*



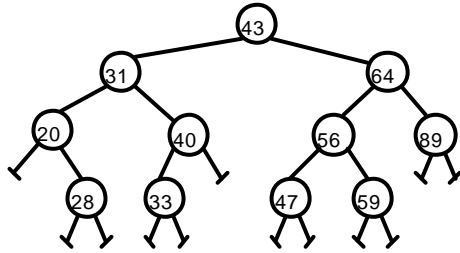
17

### *Postorder Traversal*

- Traverse the left subtree.
- Traverse the right subtree.
- Visit the node.

18

**Example: Postorder**



28	20	33	40	31	47	59	56	89	64	43
----	----	----	----	----	----	----	----	----	----	----

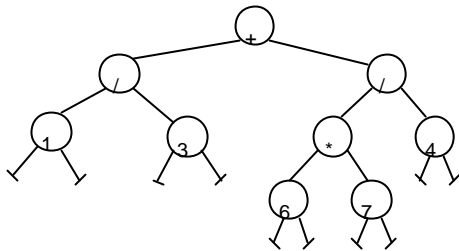
19

**Expression Tree**

- A Binary Tree built with operands and operators.
- Also known as a parse tree.
- Used in compilers.

20

**Example: Expression Tree**



1	/	3	+	6	*	7	/	4
---	---	---	---	---	---	---	---	---

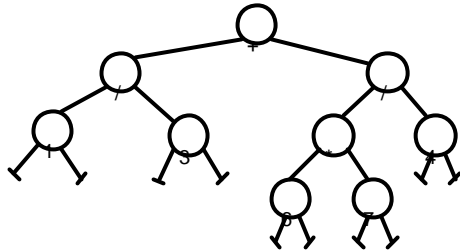
21

**Notation**

- Preorder
  - Prefix Notation
- Inorder
  - Infix Notation
- Postorder
  - Postfix Notation

22

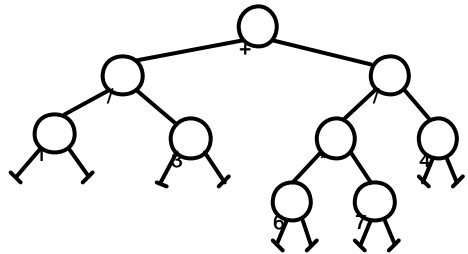
**Example: Infix**



1	/	3	+	6	*	7	/	4
---	---	---	---	---	---	---	---	---

23

**Example: Postfix**

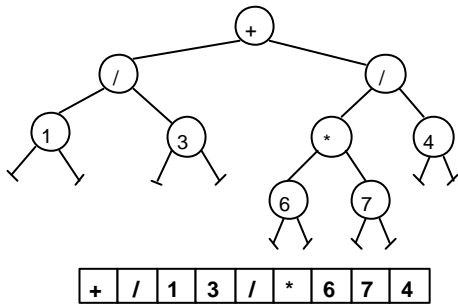


Recall: Reverse Polish Notation

1	3	/	6	7	*	4	/	+
---	---	---	---	---	---	---	---	---

24

**Example: Prefix**



25

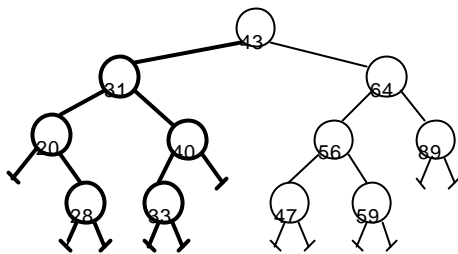
**Binary Search Tree**

A Binary Tree such that:

- Every node entry has a **unique** key.
- **All** the keys in the **left subtree** of a node are **less** than the key of the node.
- **All** the keys in the **right subtree** of a node are **greater** than the key of the node.

26

**Example 1:** key is an integer



27

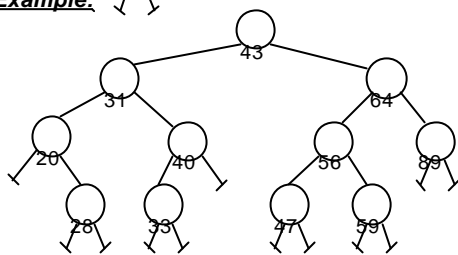
**Insert**

- Create new node for the item.
- Find a parent node.
- Attach new node as a leaf.

28

**Insert**

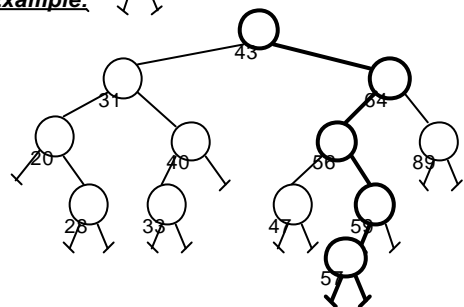
**Example:**



29

**Insert**

**Example:**



30

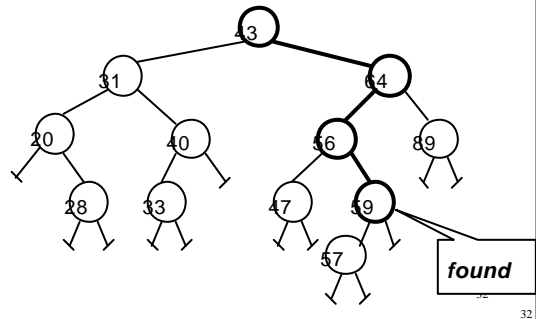
### Search: Checklist

- if target key is less than current node's key, search the left sub-tree.
- else, if target key is greater than current node's key, search the right sub-tree.
- returns:
  - if found, pointer to node containing target key.
  - otherwise, NULL pointer.

31

### Search

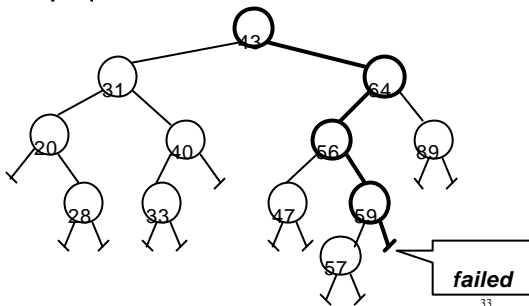
Example: 59



32

### Search

Example: 61



33

### Revision

- Binary Tree
- Preorder, Inorder, Postorder Traversal
- Expression Trees
- Prefix, Infix, and Postfix notation
- Insert and Search

### Revision: Reading

- Kruse 9
- Standish 9
- Langsam 5
- Deitel & Deitel 12.7

### Preparation

Next lecture: Binary Search Trees (Information Retrieval)

- Read Chapter 9.2 in Kruse et al.

34