ADVANCED DATA STRUCTURES AND ALGORITHMS

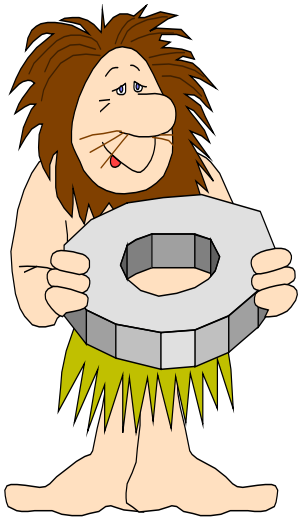
Associate Professor Dr. Raed Ibraheem Hamed

University of Human Development, College of Science and Technology
Computer Science Department

2015 – 2016

What this Lecture is about:

- Traversals
- Traversing Trees
- Types of traversals
- Search Trees (BST)
- How to search a binary tree?
- Some terminology of Binary Trees



Binary Tree Traversal Methods

- It's unclear how we should print a tree.
- Top to bottom? Left to right?
- A tree traversal is a specific order in which to trace the nodes of a tree.

There are 3 common tree traversals.

1. in-order: left, root, right
2. pre-order: root, left, right
3. post-order: left, right, root.

Binary Tree Traversal Methods

- Types of traversals

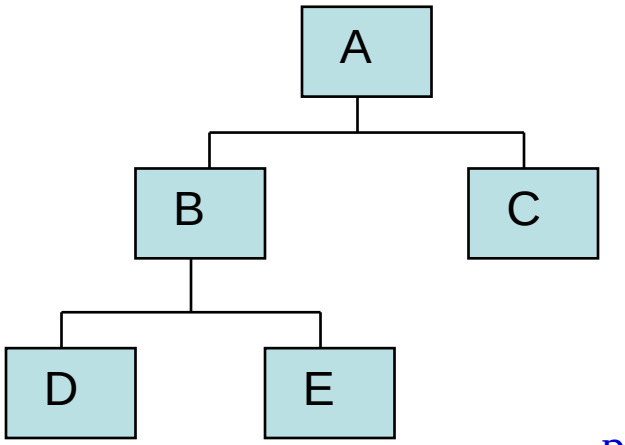
  - Pre-order
    - Visit root, traverse left child, traverse right child
  - In-order
    - Traverse left child, visit root, traverse right child
    - The in-order traversal is probably the easiest to see, because it sorts the values from smallest to largest.
  - Post-Order
    - Traverse left child, traverse right child, visit root
    - It is also called a depth-first search.

Traversing Trees

Pre-order traversal would give:

A, B, D, E, C

Tree

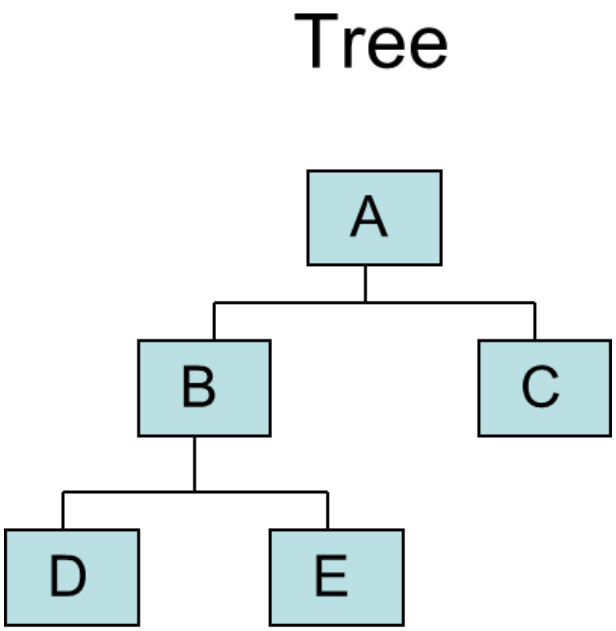


pre-order: root, left, right

Traversing Trees

In-order traversal would give:

D, B, E, A, C

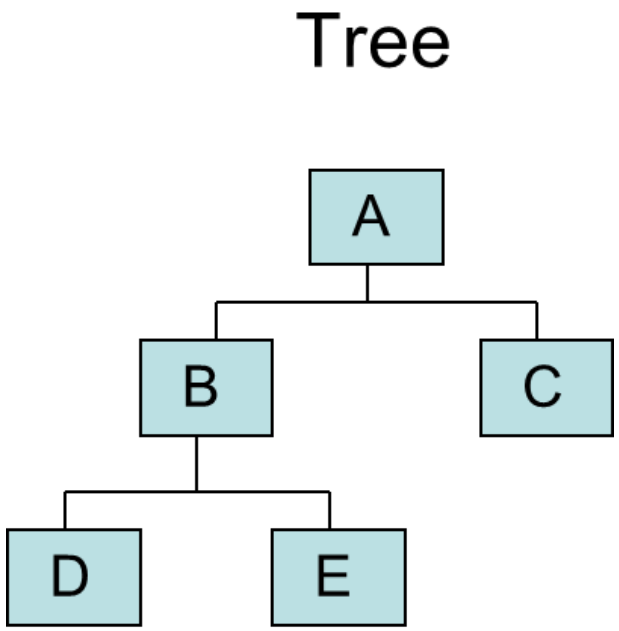


in-order: left, root, right

Traversing Trees

Post-order traversal would give:

D, E, B, C, A

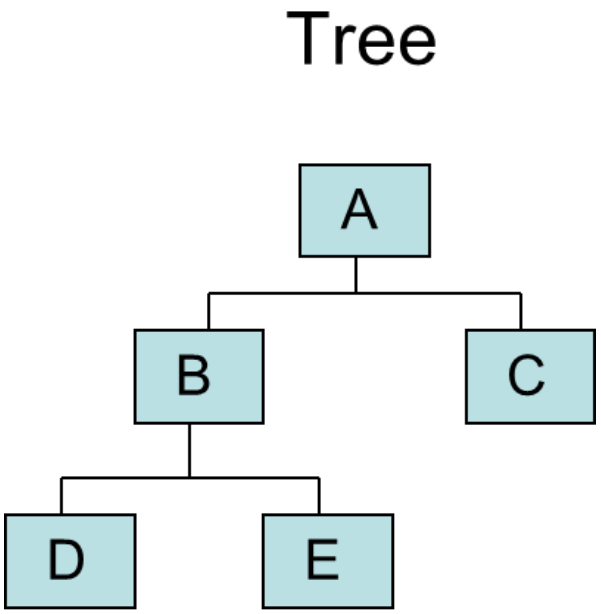


post-order: left, right, root

Traversing Trees

Level-order Traversal would give:

A, B, C, D, E



Traversing Trees

- Preorder: Root, then Children

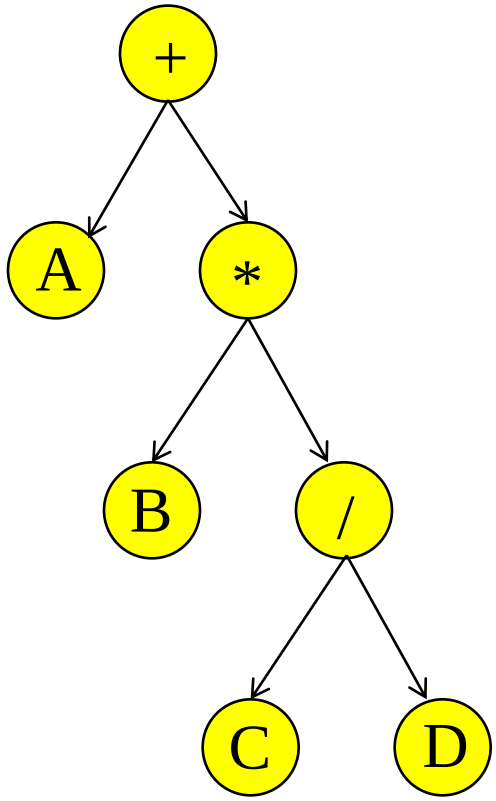
  + A * B / C D

- Postorder: Children, then Root

  A B C D / * +

- Inorder: Left child, Root, Right child

  A + B * C / D



Preorder, Postorder and Inorder Pseudo Code

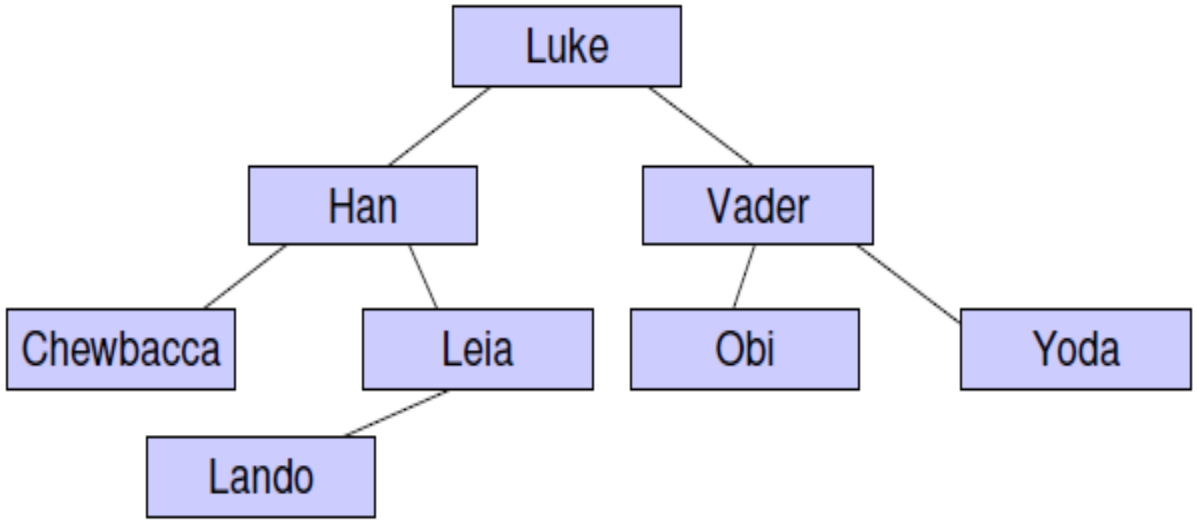
```
Algorithm Preorder(x)
Input: x is the root of a subtree.
1.   if x ≠ NULL
2.      then output key(x);
3.           Preorder(left(x));
4.           Preorder(right(x));
```

```
Algorithm Postorder(x)
Input: x is the root of a subtree.
1.   if x ≠ NULL
2.      then Postorder(left(x));
3.           Postorder(right(x));
4.           output key(x);
```

```
Algorithm Inorder(x)
Input: x is the root of a subtree.
1.   if x ≠ NULL
2.      then Inorder(left(x));
3.           output key(x);
4.           Inorder(right(x));
```

Tree Traversal Example

Ex. Write the 3 traversals of the given tree.



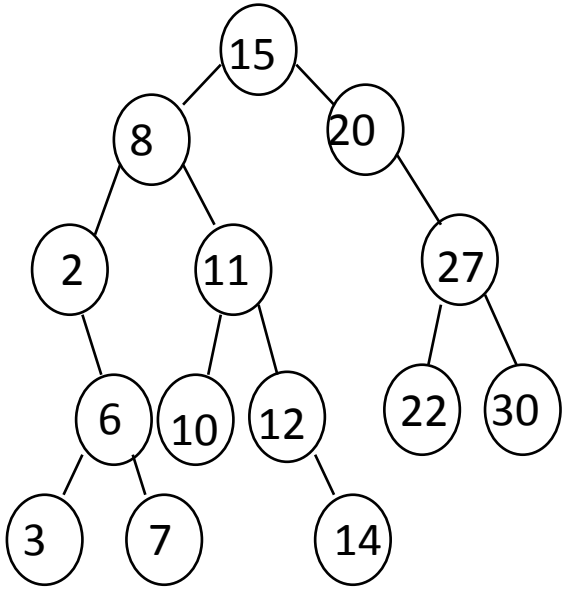
In-order: Chewbacca, Han, Lando, Leia, Luke, Obi, Vader, Yoda
Pre-order: Luke, Han, Chewbacca, Leia, Lando, Vader, Obi, Yoda
Post-order: Chewbacca, Lando, Leia, Han, Obi, Yoda, Vader, Luke

Illustrations for Traversals

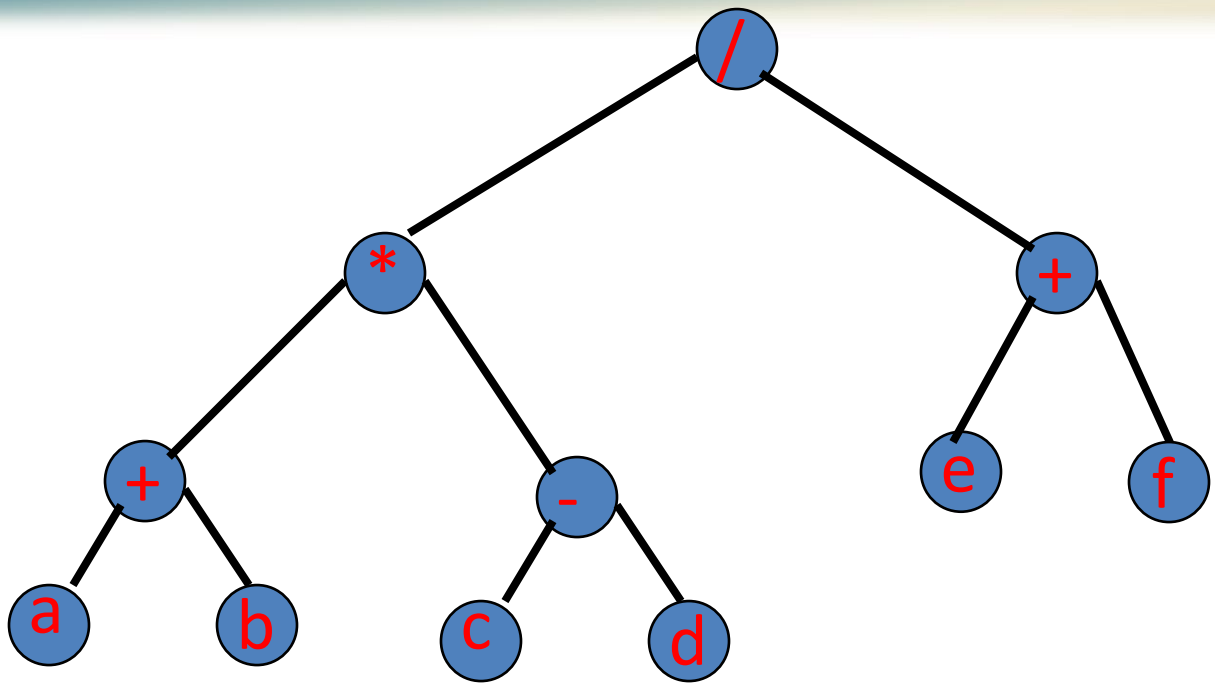
- Assume: visiting a node is printing its data
- Preorder: 15 8 2 6 3 7
  11 10 12 14 20 27 22 30

- Inorder: 2 3 6 7 8 10 11
  12 14 15 20 22 27 30

- Postorder: 3 7 6 2 10 14
  12 11 8 22 30 27 20 15



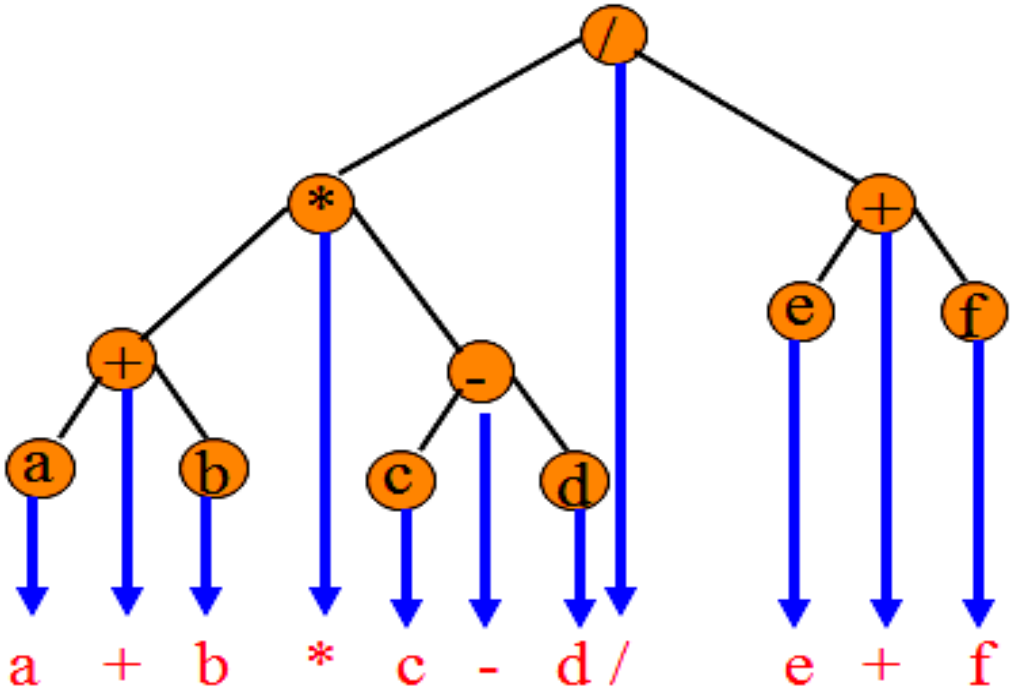
Preorder Of Expression Tree



/ * + a b - c d + e f

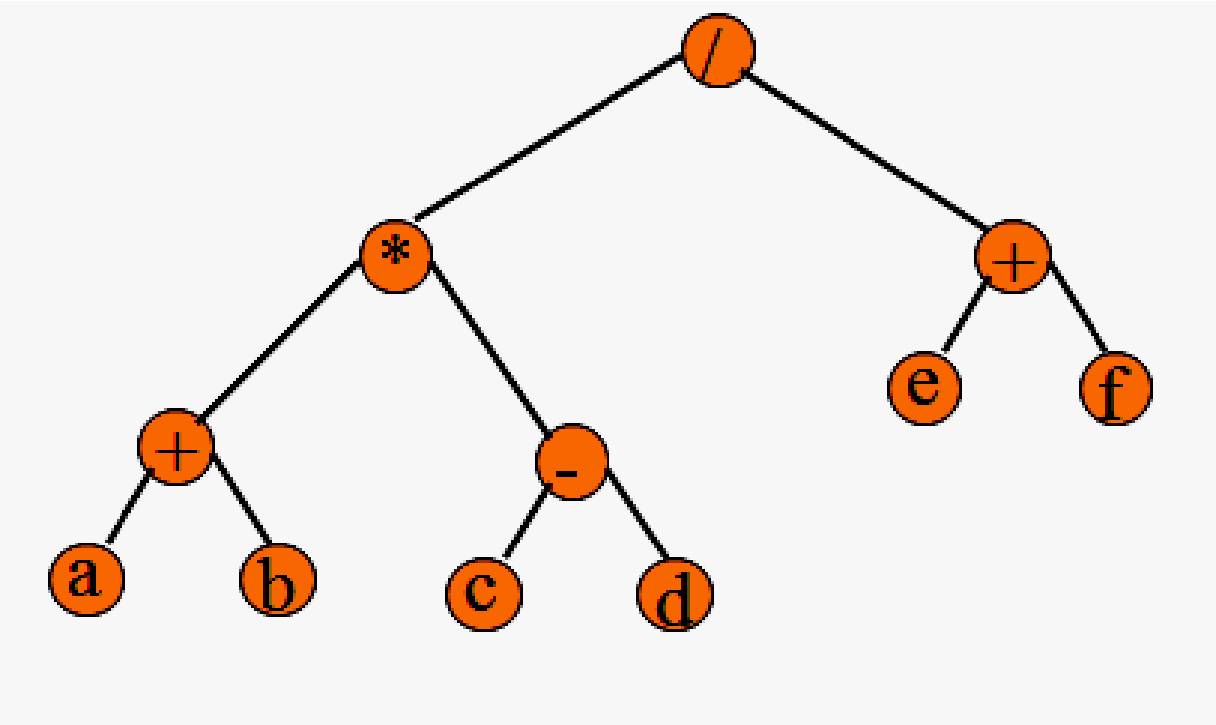
Gives prefix form of expression!

Inorder Of Expression Tree



Gives infix form of expression

Postorder Of Expression Tree



a b + c d - * e f + /

Gives postfix form of expression!

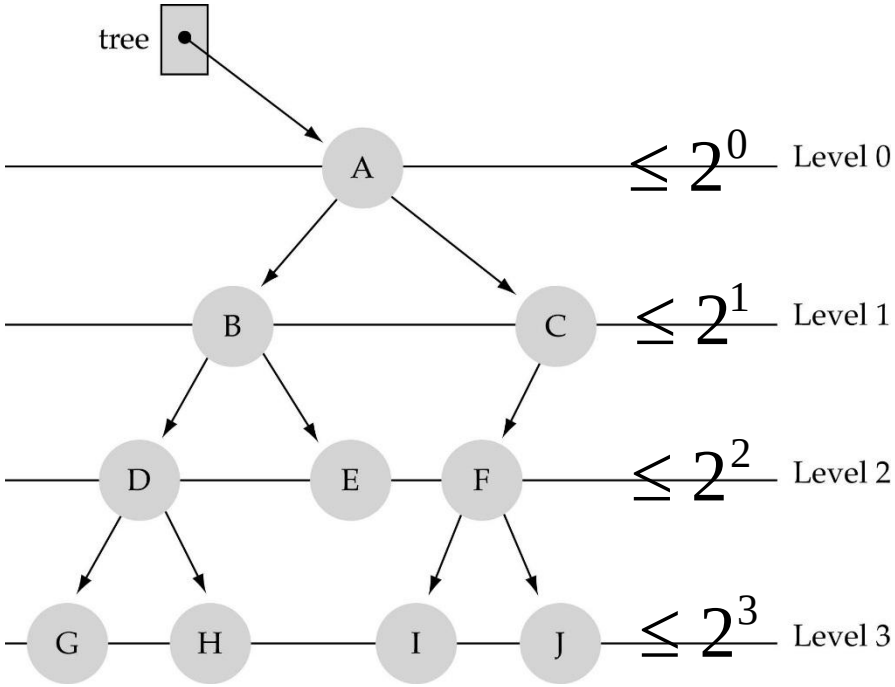
Some terminology of Binary Trees

- The successor nodes of a node are called its **children**
- The predecessor node of a node is called its **parent**
- The "beginning" node is called the **root** (has no parent)
- A node without children is called a **leaf**

Some terminology of Binary Trees

What is the max #nodes at some level i ?

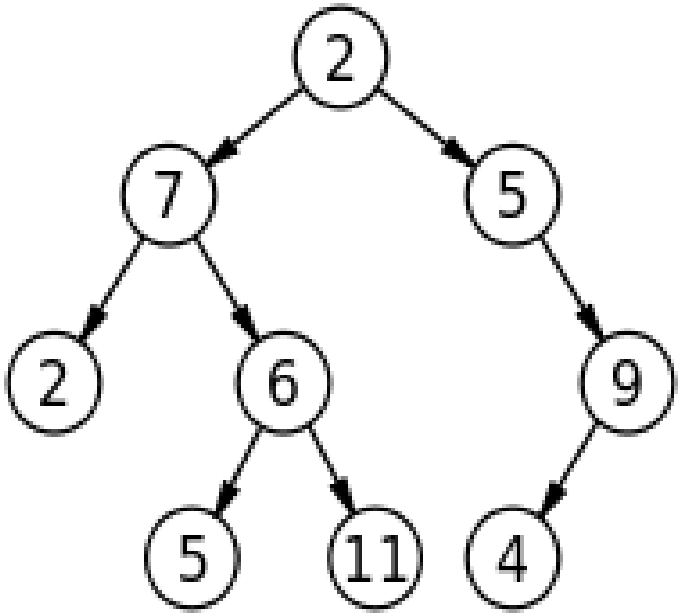
The max # nodes at **level i** is 2^i where i = 0,1,2, ...,L-1



How to search a binary tree?

(1) Start at the root

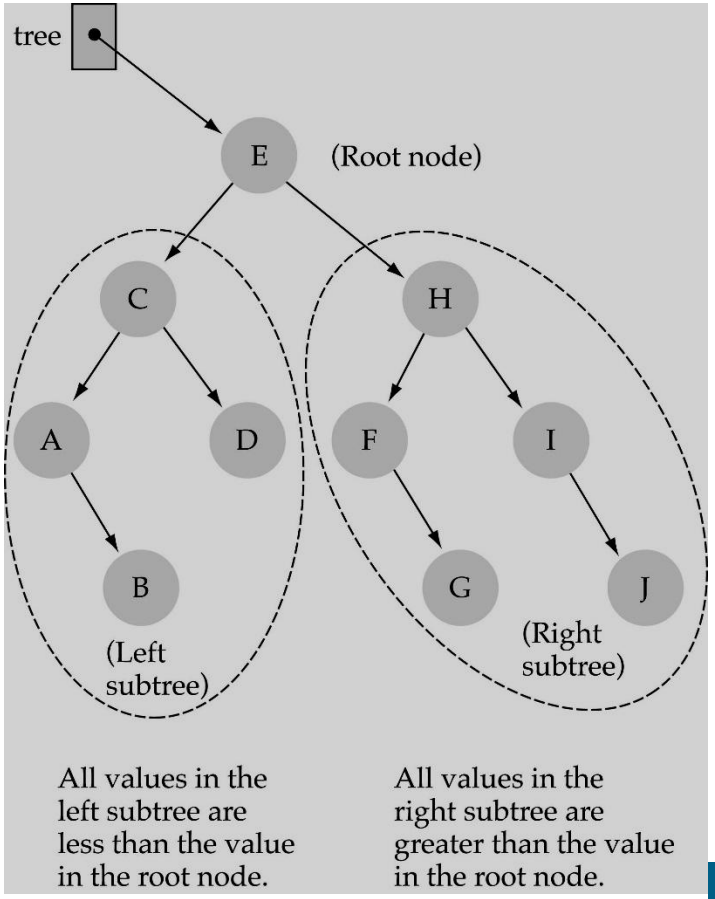
(2) Search the tree level by level, until you find the element you are searching for or you reach a leaf.



Binary Search Trees (BSTs)

- **Binary Search Tree Property:**

  The value stored at a node is *greater* than the value stored at its left child and *less* than the value stored at its right child



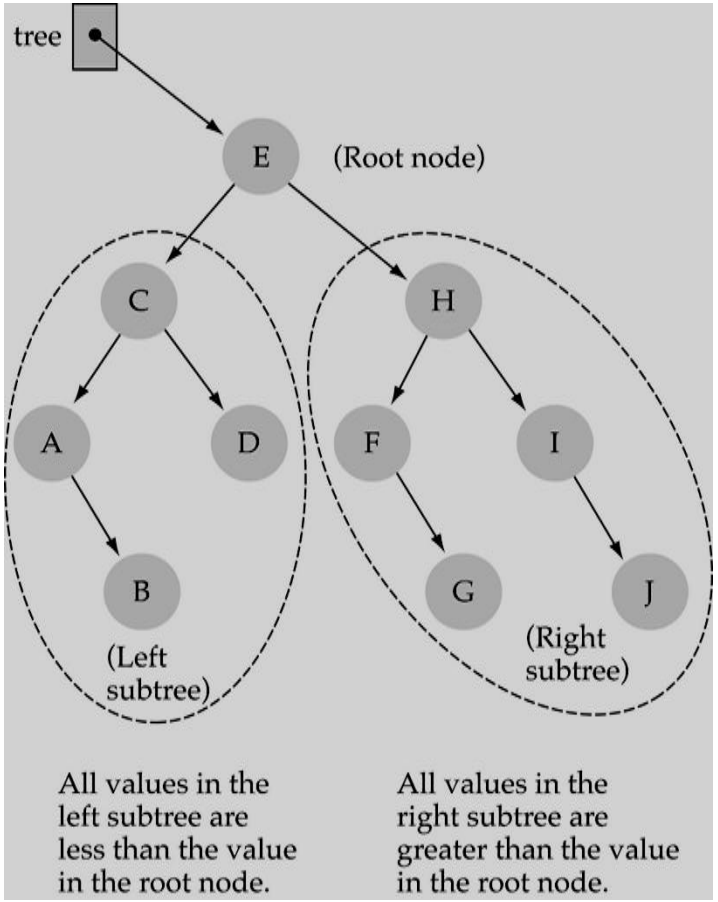
Binary **Search** Trees (BSTs)

Where is the smallest element?

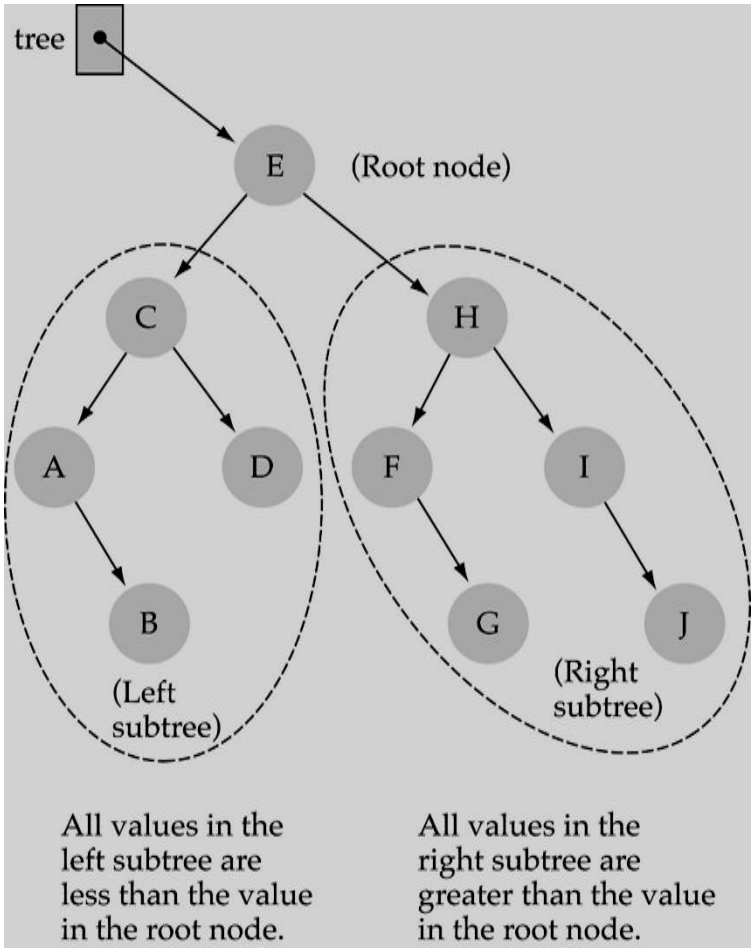
Ans: leftmost element

Where is the largest element?

Ans: rightmost element



How to search a binary search tree?

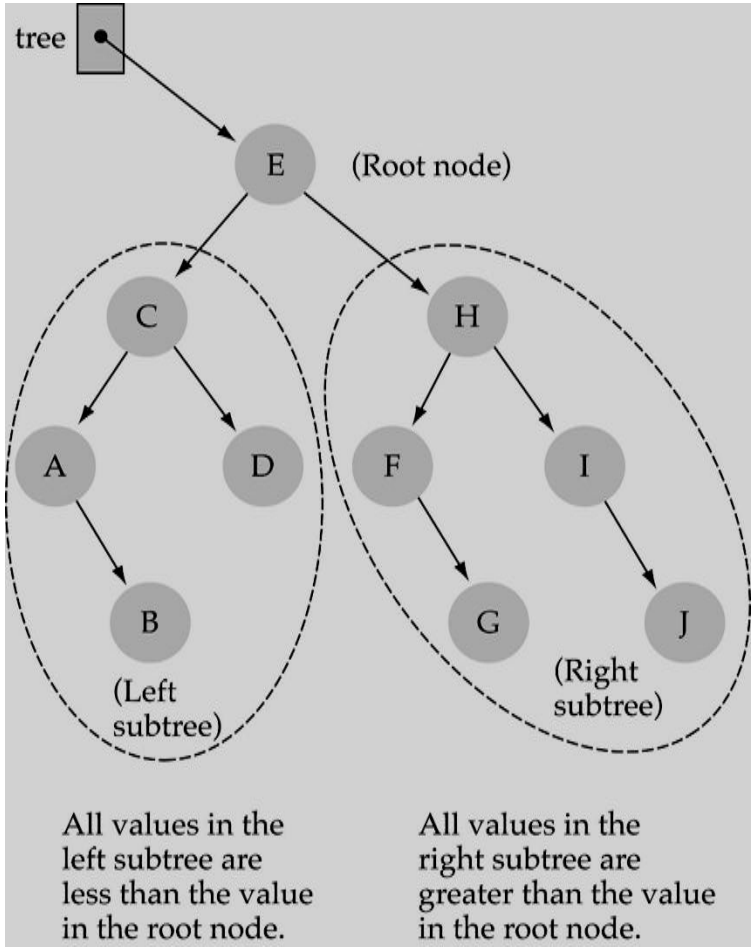


(1) Start at the root

(2) Compare the value of the item you are searching for with the value stored at the root

(3) If the values are equal, then *item found*; otherwise, if it is a leaf node, then *not found*

How to search a binary search tree?



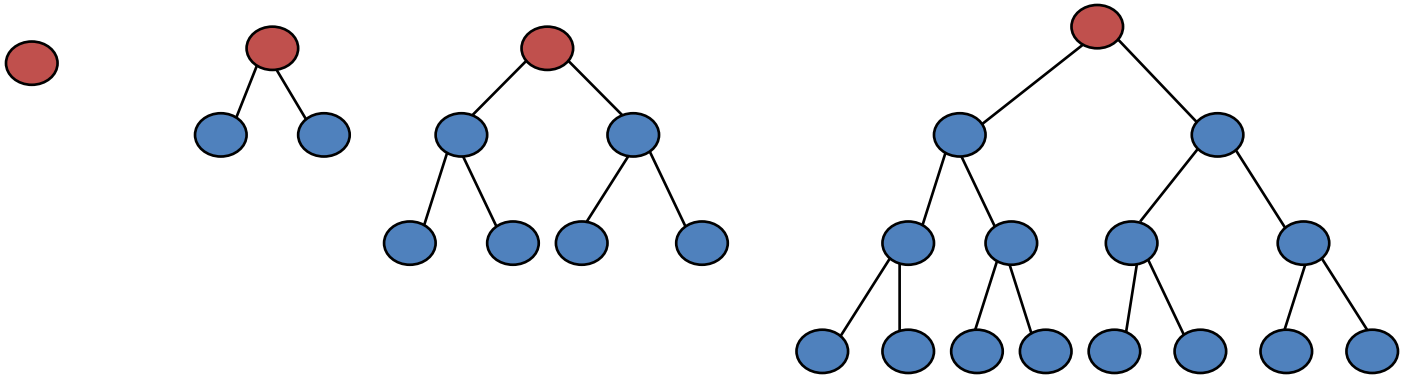
(4) If it is **less** than the value stored at the root, then search the **left subtree**

(5) If it is **greater** than the value stored at the root, then search the **right subtree**

(6) Repeat steps 2-6 for the root of the subtree chosen in the previous step 4 or 5

Other Kinds of Binary Trees

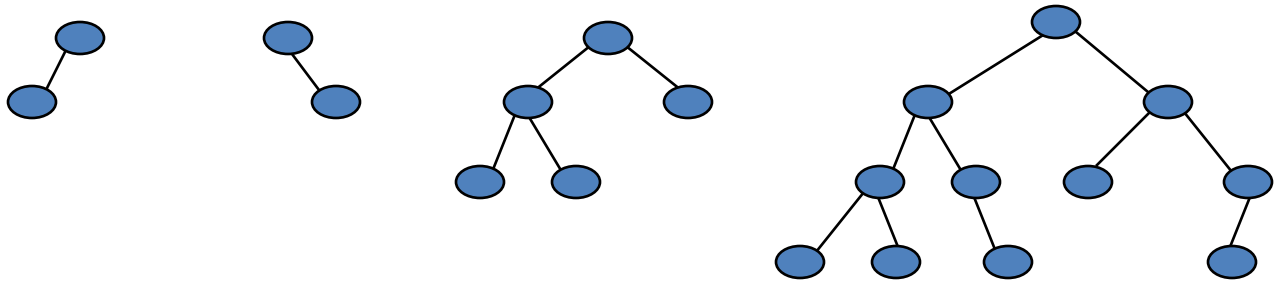
- **Full Binary Tree**: A full binary tree is a binary tree where all the leaves are on the same level and every non-leaf has two children
- The first four full binary trees are:



Examples of Non-Full Binary Trees

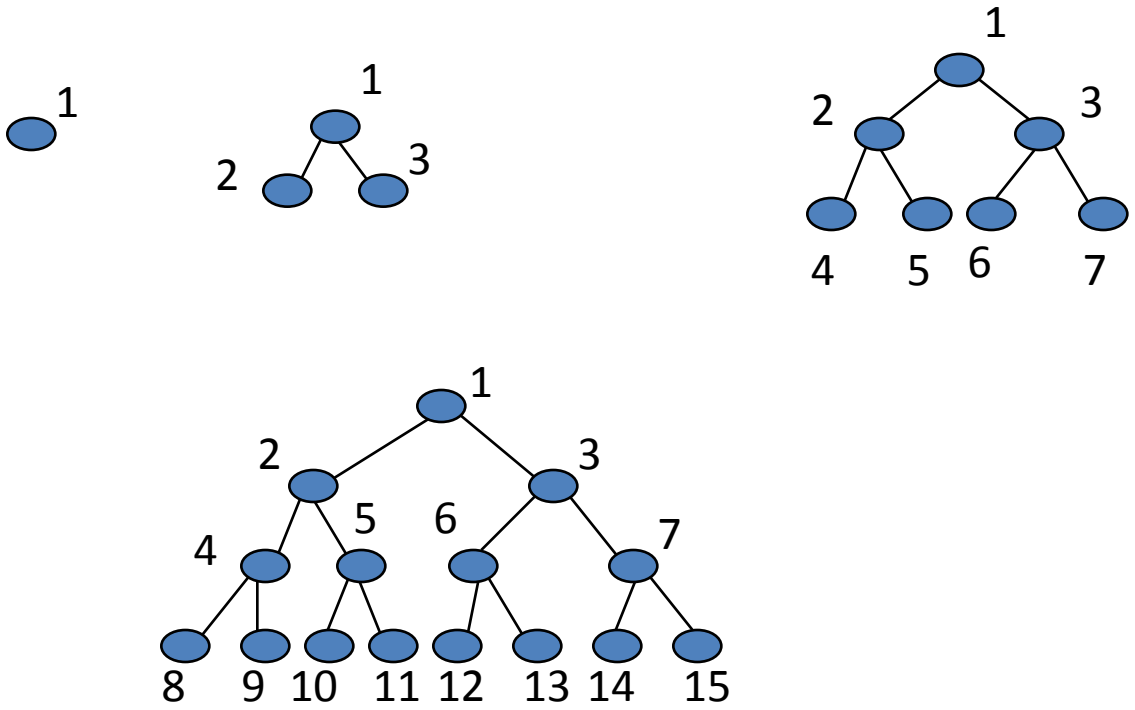
- These trees are NOT full binary trees:

  (do you know why?)



Labeling of Full Binary Trees

- Label the nodes from 1 to n from the top to the bottom, left to right



Thank you ???