DATA STRUCTURES & ALGORITHMS IN C

TREES

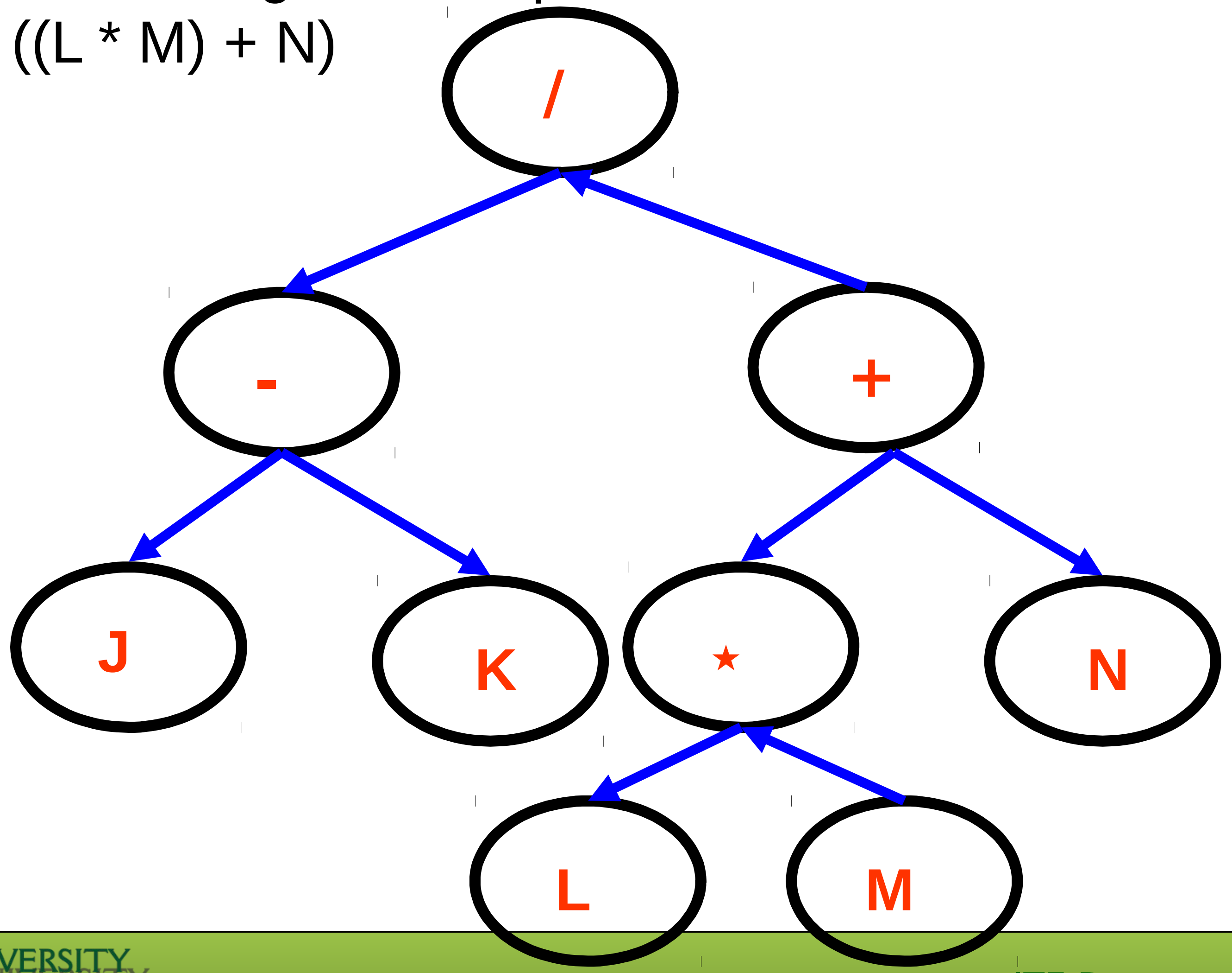
FAR EASTERN UNIVERSITY
East Asia College

Introduction

- Data structure such as Arrays, Stacks, Linked List and Queues are linear data structure. Elements are arranged in linear manner i.e. one after another.
- Tree is a non-linear data structure.
- Tree imposes a ***Hierarchical*** structure, on a collection of items.
- Several Practical applications
  - Organization charts.
  - Family hierarchy
  - Representation of algebraic expression

Introduction

- Representation of algebraic expression
- Z=(J - K) / ((L * M) + N)



:Draw its equivalent algebraic expression tree

$(7 - 2 * 5) + 3$

FAR EASTERN UNIVERSITY
East Asia College

Tree Definition

- A tree is a collection of nodes
  - The collection can be empty
  - If not empty, a tree consists of a distinguished node **R** (the *root*), and zero or more nonempty ***subtrees*** T_1, T_2, ..., T_k, each of whose roots are connected by a directed ***edge*** from **R.**

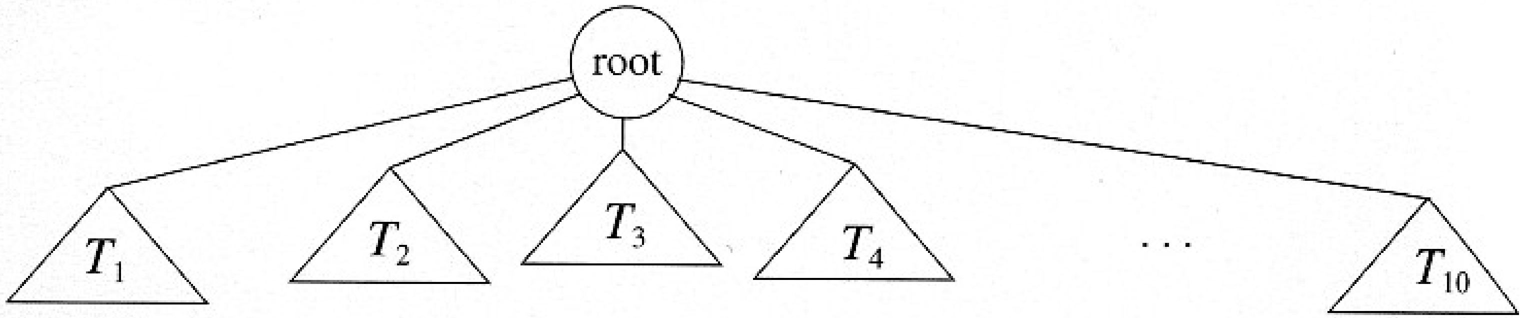


Figure 4.1 Generic tree

Tree

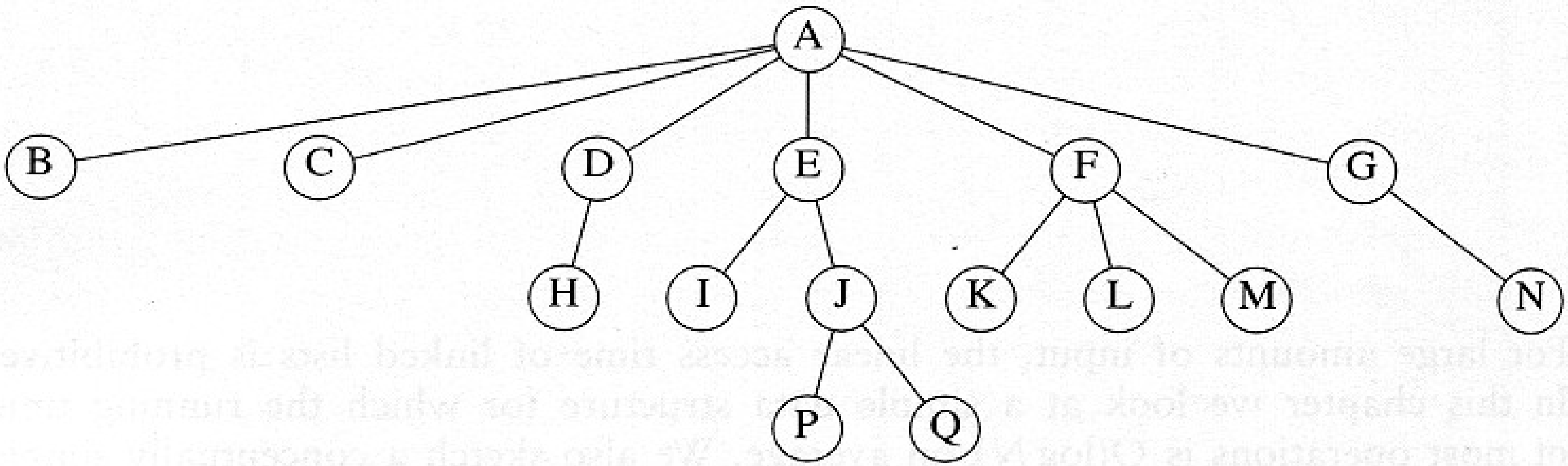
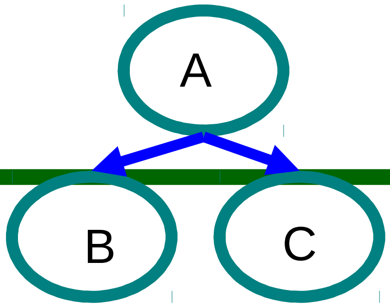


Figure 4.2 A tree

FAR EASTERN UNIVERSITY
East Asia College

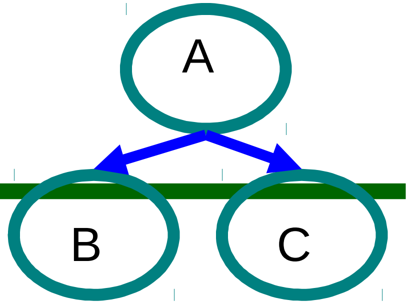
Tree Terminologies



- **Root**
  - It is the mother node of a tree structure. This tree does not have parent. It is the first node in hierarchical arrangement.
- **Node**
  - The node of a tree stores the data and its role is the same as in the linked list. Nodes are connected by the means of links with other nodes.
- **Parent**
  - It is the immediate predecessor of a node. In the figure A is the parent of B and C.

FAR EASTERN UNIVERSITY
East Asia College

Tree Terminologies



Child

- When a predecessor of a node is parent then all successor nodes are called child nodes. In the figure B and C are the child nodes of A

Link / Edge

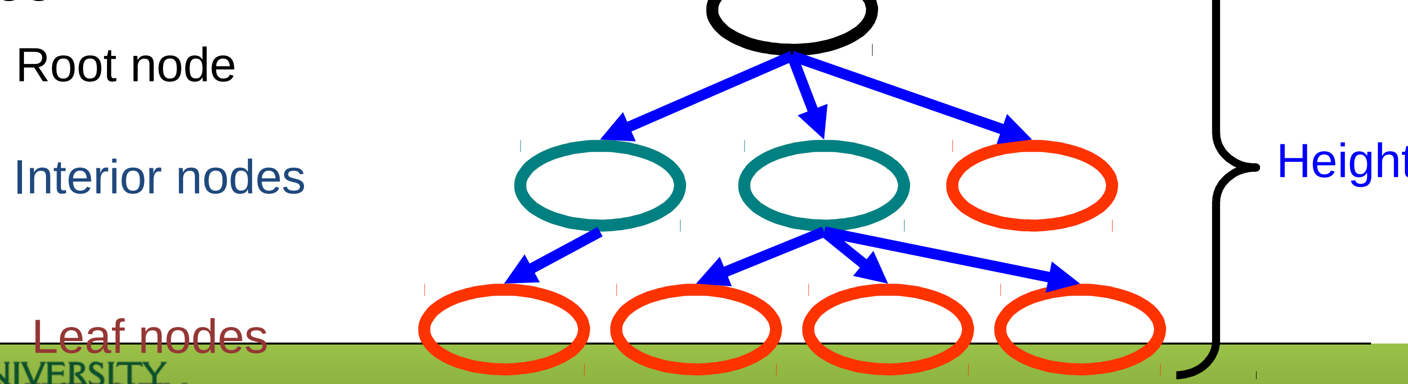
- An edge connects the two nodes. The line drawn from one node to other node is called edge / link. Link is nothing but a pointer to node in a tree structure.

Leaf

- This node is located at the end of the tree. It does not have any child hence it is called leaf node.

Tree Terminologies

- **Level**
  - **Level is the rank of tree hierarchy. The whole tree structured is leveled. The level of the root node is always at 0. the immediate children of root are at level 1 and their children are at level 2 and so on.**
- **Height**
  - **The highest number of nodes that is possible in a way starting from the first node (ROOT) to a leaf node is called the height of tree. The formula for finding the height of a tree** $h = i_{max} + 1$ **, where h is the height and i is the max level of the tree**



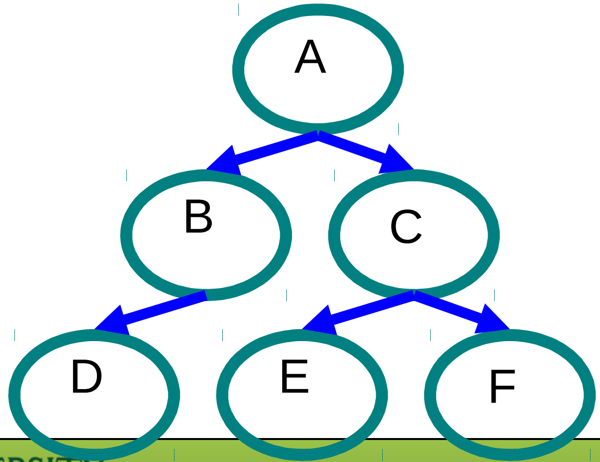
FAR EASTERN UNIVERSITY
East Asia College

Tree Terminologies

- **Sibling**
  - **The child node of same parent are called sibling. They are also called brother nodes.**
- **Degree of a Node**
  - **The maximum number of children that exists for a node is called as degree of a node.**
- **Terminal Node**
  - **The node with degree zero is called terminal node or leaf.**
- **Path length.**
  - **Is the number of successive edges from source node to destination node.**
- **Ancestor and descendant**
  - **Proper ancestor and proper descendant**

Tree Terminologies

- **Depth**
  - **Depth of a binary tree is the maximum level of any leaf of a tree.**
- **Forest**
  - **It is a group of disjoint trees. If we remove a root node from a tree then it becomes the forest. In the following example, if we remove a root A then two disjoint sub-trees will be observed. They are left sub-tree B and right sub-tree C.**



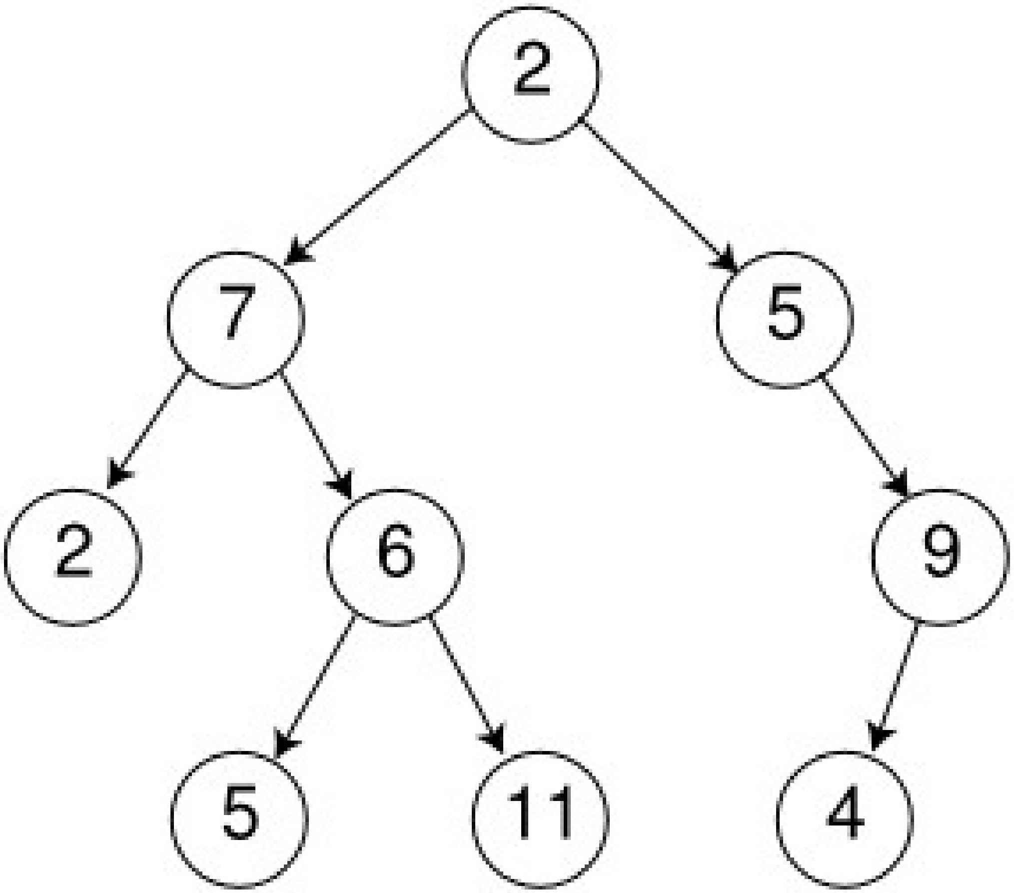
Binary Trees

The simplest form of tree is a **binary tree**. A binary tree consists of

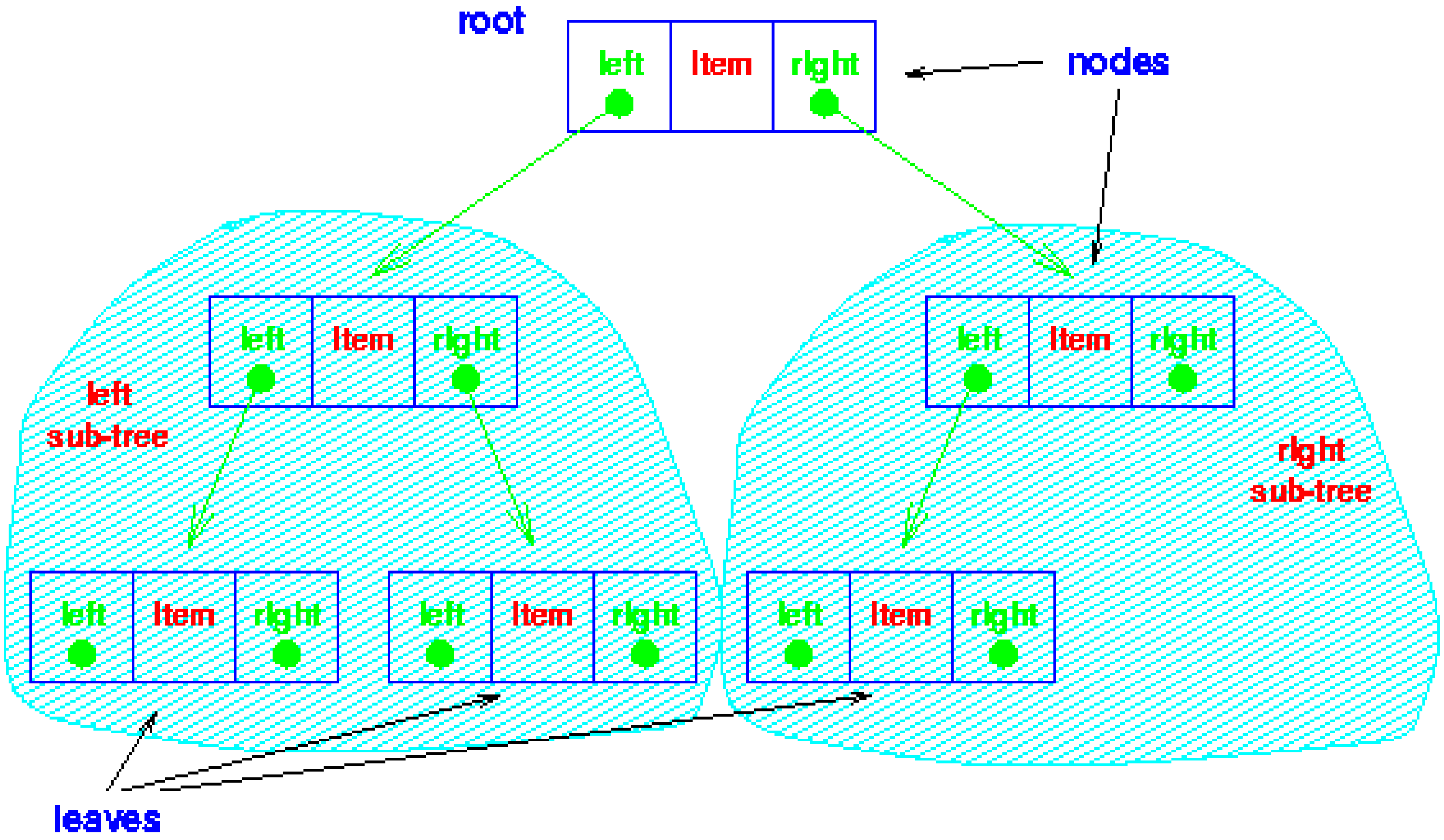
- a *node* (called the **root** node) and
- left and right sub-trees.
- Both the sub-trees are themselves binary trees

We now have a *recursively defined data structure*. Also, a tree is binary if each node of it has a maximum of two branches i.e. a node of a binary tree can have maximum two children.

A Binary Tree



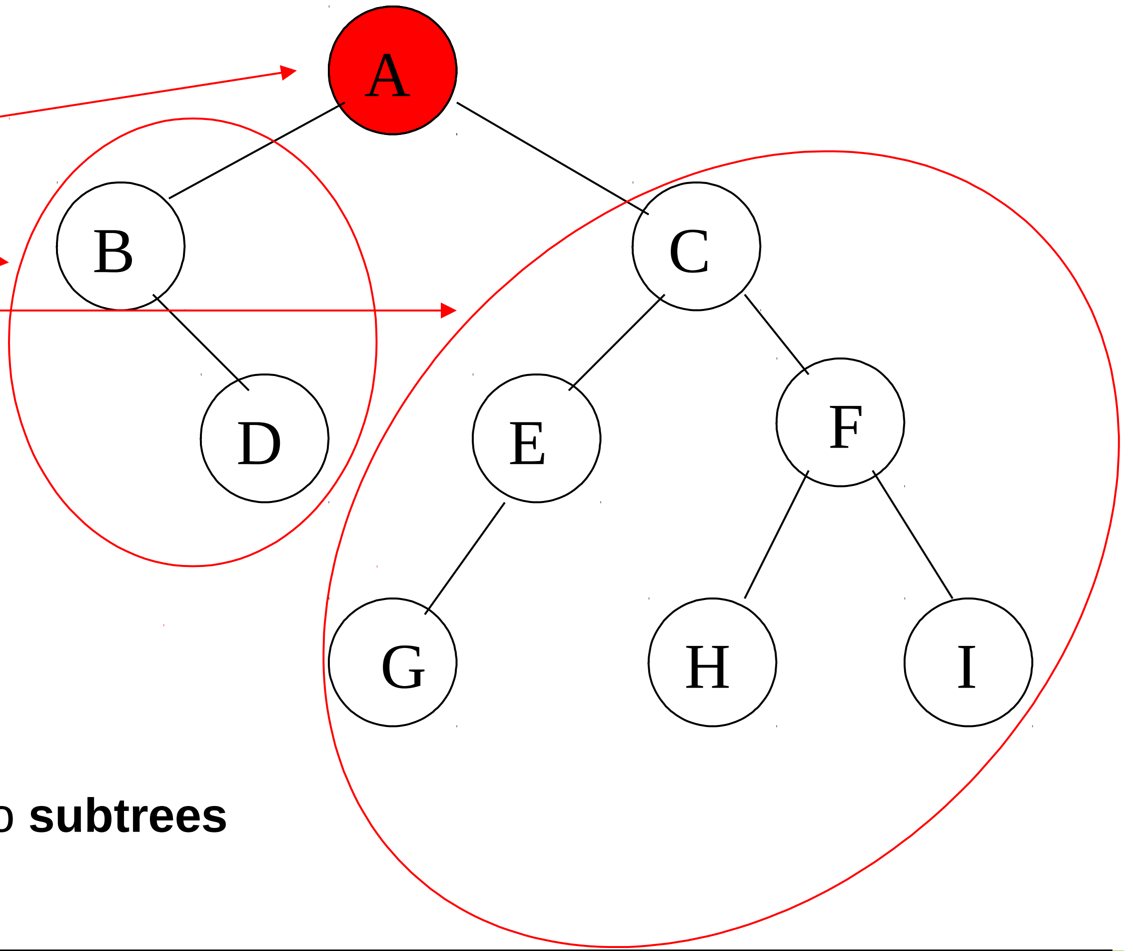
Binary Tree


root
left
Item
right
nodes
left
Item
right
left
sub-tree
left
Item
right
right
sub-tree
left
Item
right
left
Item
right
left
Item
right
leaves

Notation

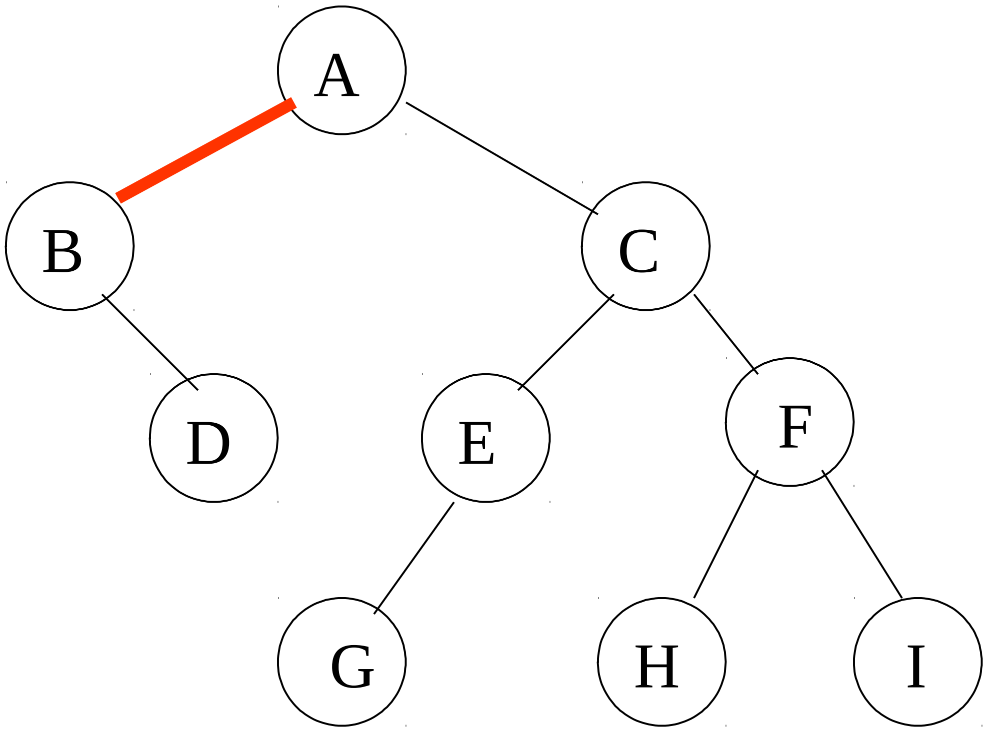
node

- root
  - left subtree
  - right subtree



It consists of a **root** and two **subtrees**

Notation



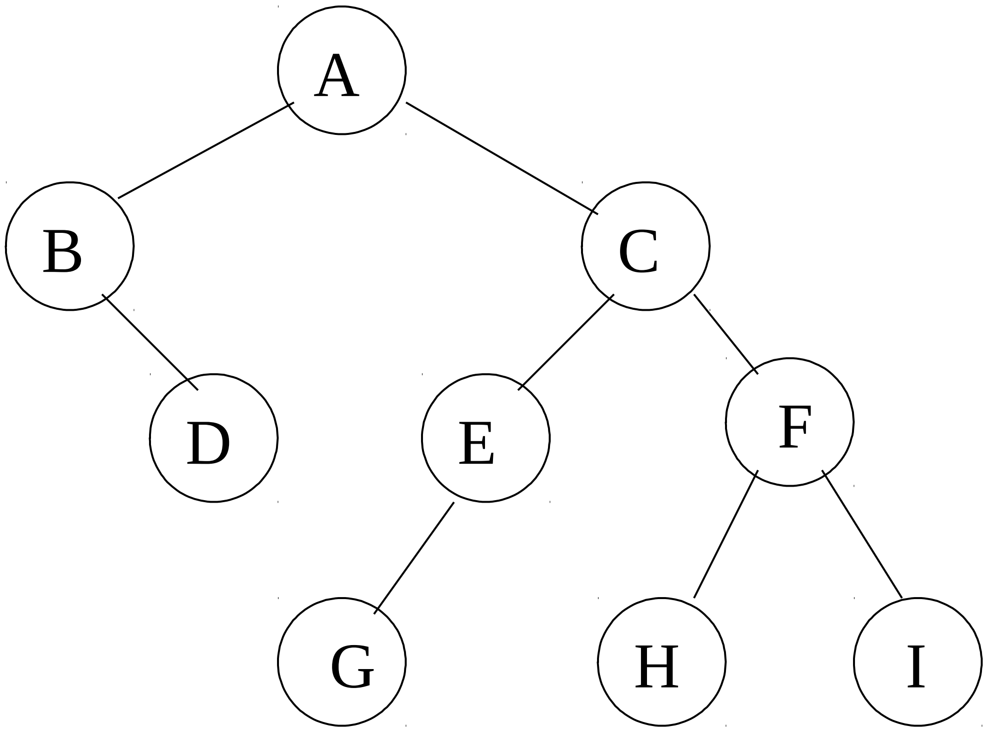
edge –

there is an **edge** from the root to its children

FAR EASTERN UNIVERSITY
East Asia College

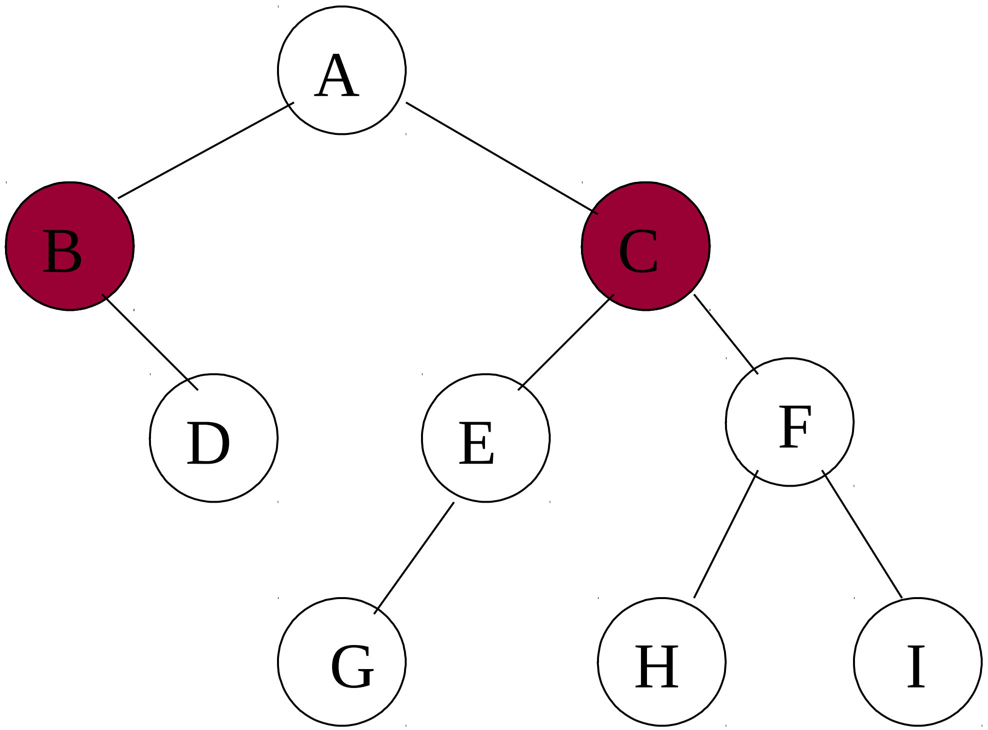
Notation

children



Notation

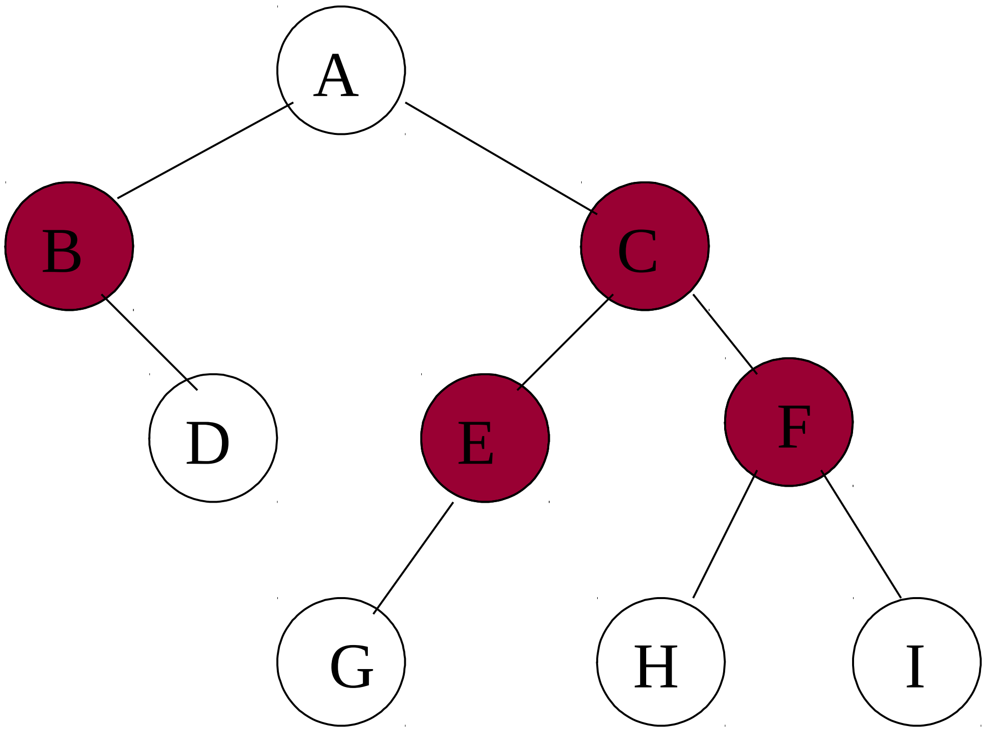
children



?Who are node A’s children

Notation

children



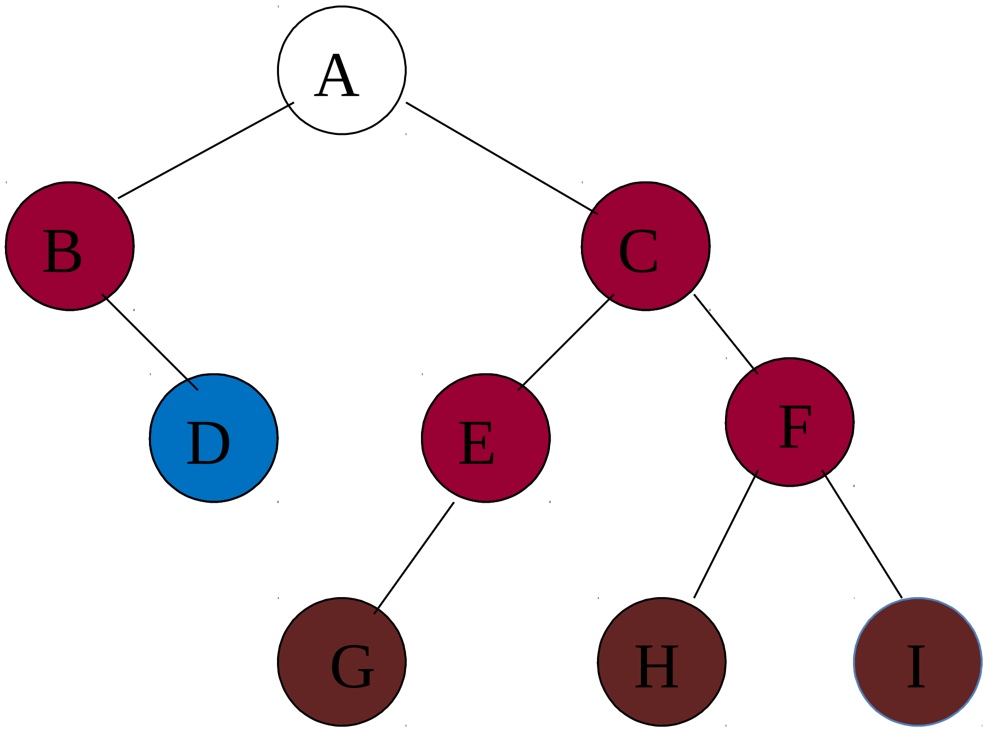
?Who are node C’s children

?Who are node A’s children

FAR EASTERN UNIVERSITY
East Asia College

Notation

descendants



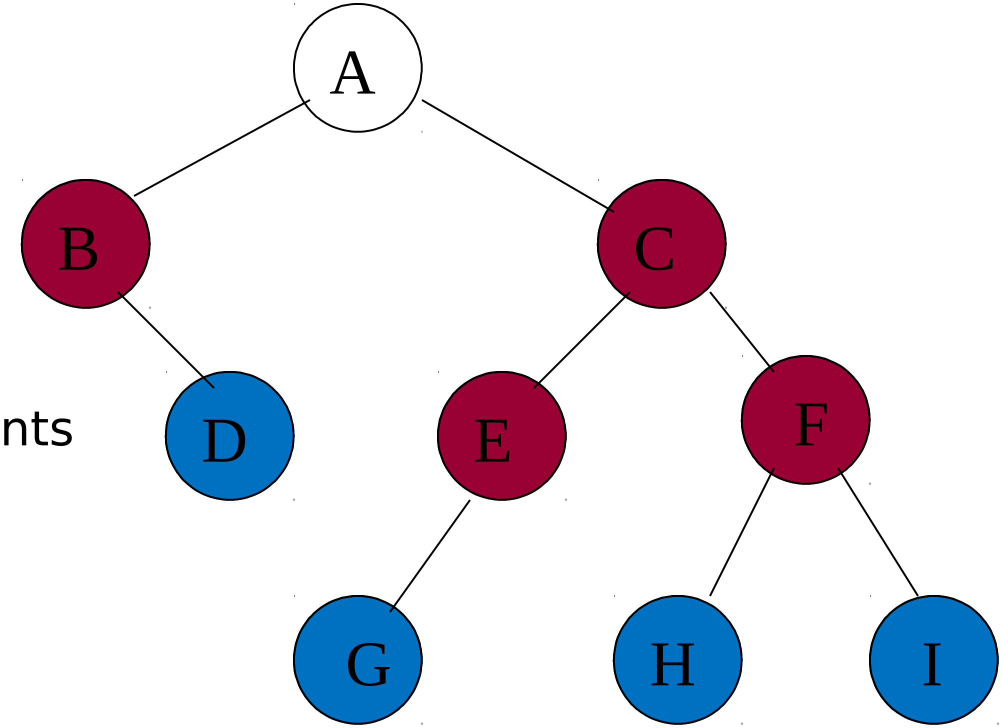
?Who are node A’s descendants

FAR EASTERN UNIVERSITY
East Asia College

Notation

descendants

?Who are node C’s descendants



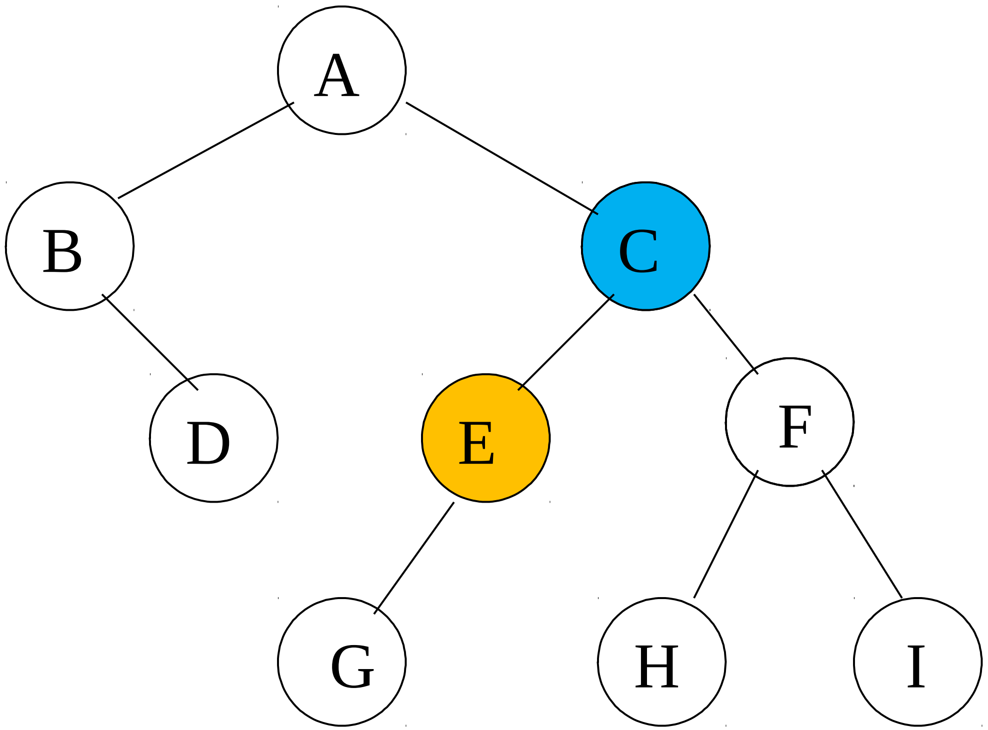
?Who are node A’s descendants

FAR EASTERN UNIVERSITY
East Asia College

Notation

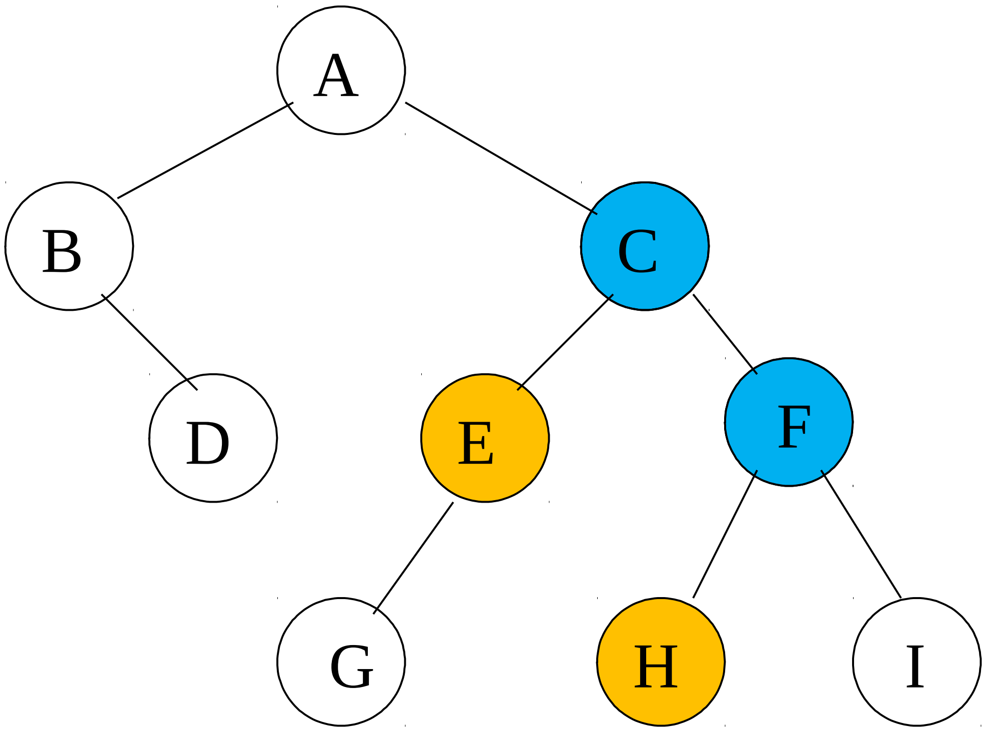
parents

?Who is node E’s parent



Notation

parents

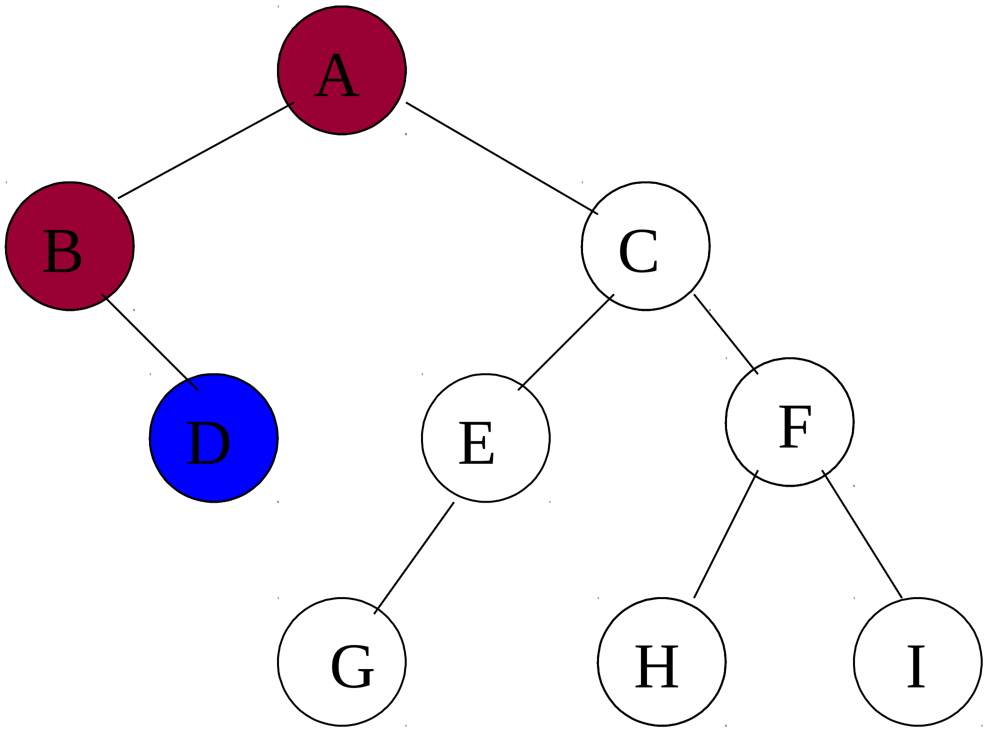


?Who is node E’s parent

?Who are node H’s parent

FAR EASTERN UNIVERSITY
East Asia College

Notation



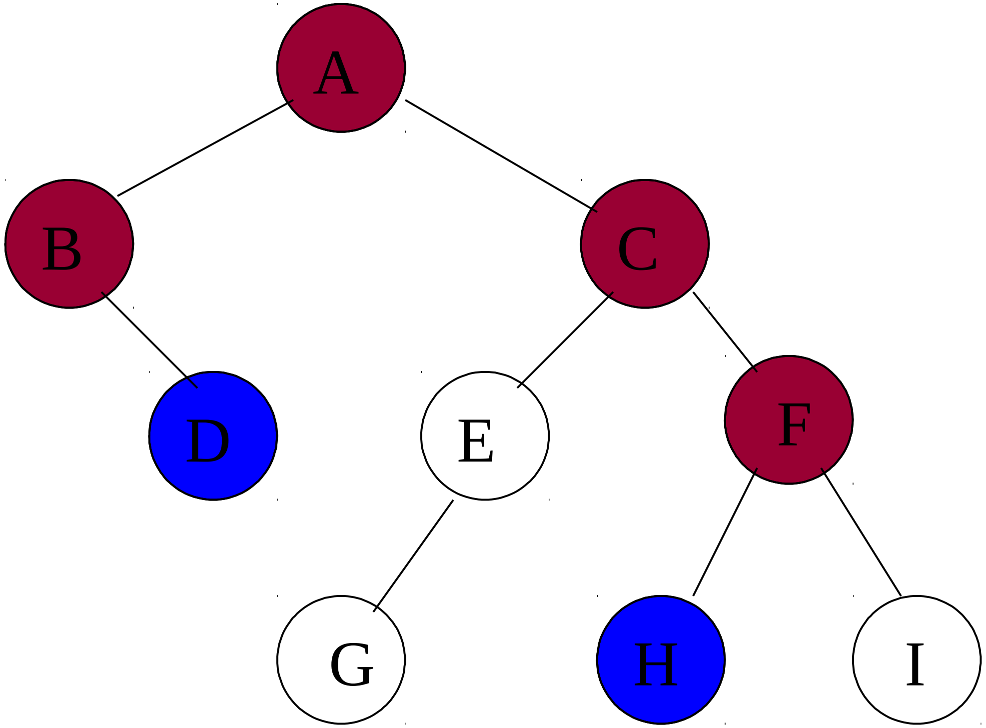
ancestors

?Who are node D’s ancestors

Notation

ancestors

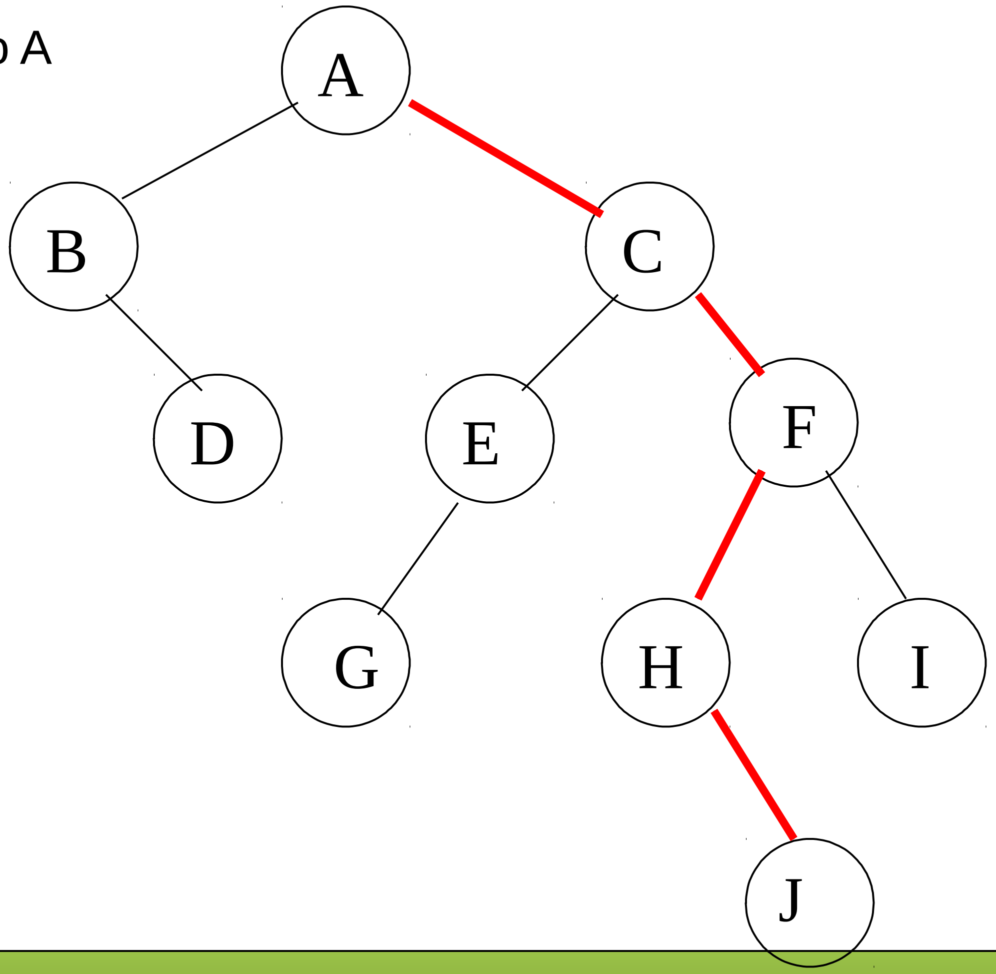
?Who are node D’s ancestors



?Who are node H’s ancestors

Notation

from J to A



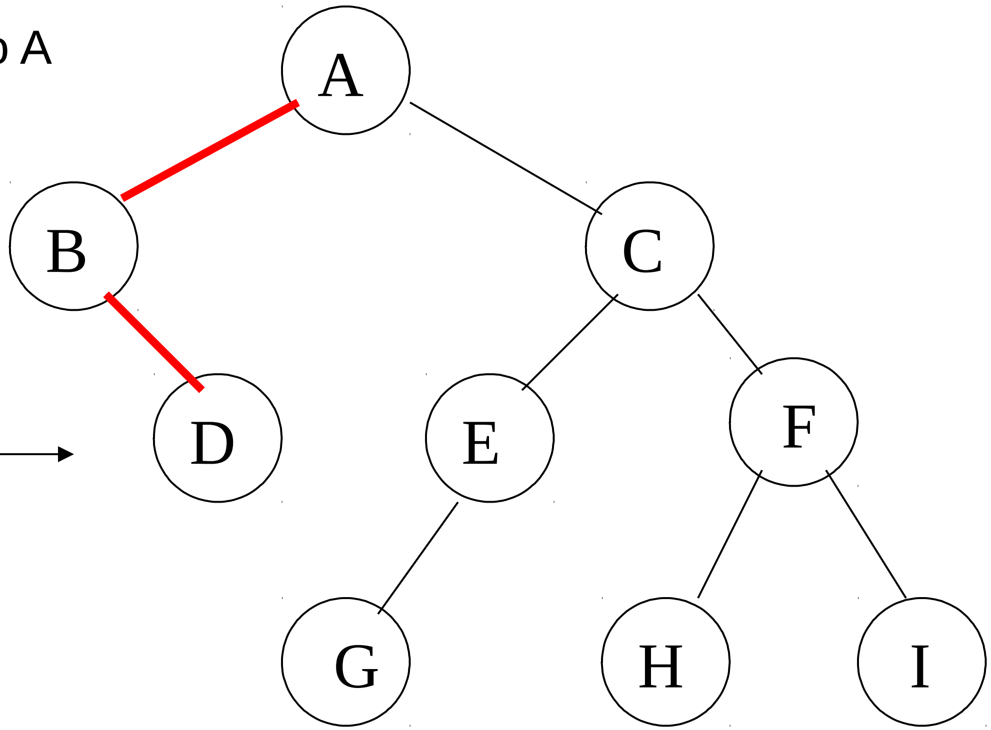
path –

If n_1, n_2,…n_k is a sequence of nodes such that n_i is the parent of n_i+1, then that sequence is a **path.**

.The length of the path is k-1 ▪

Notation

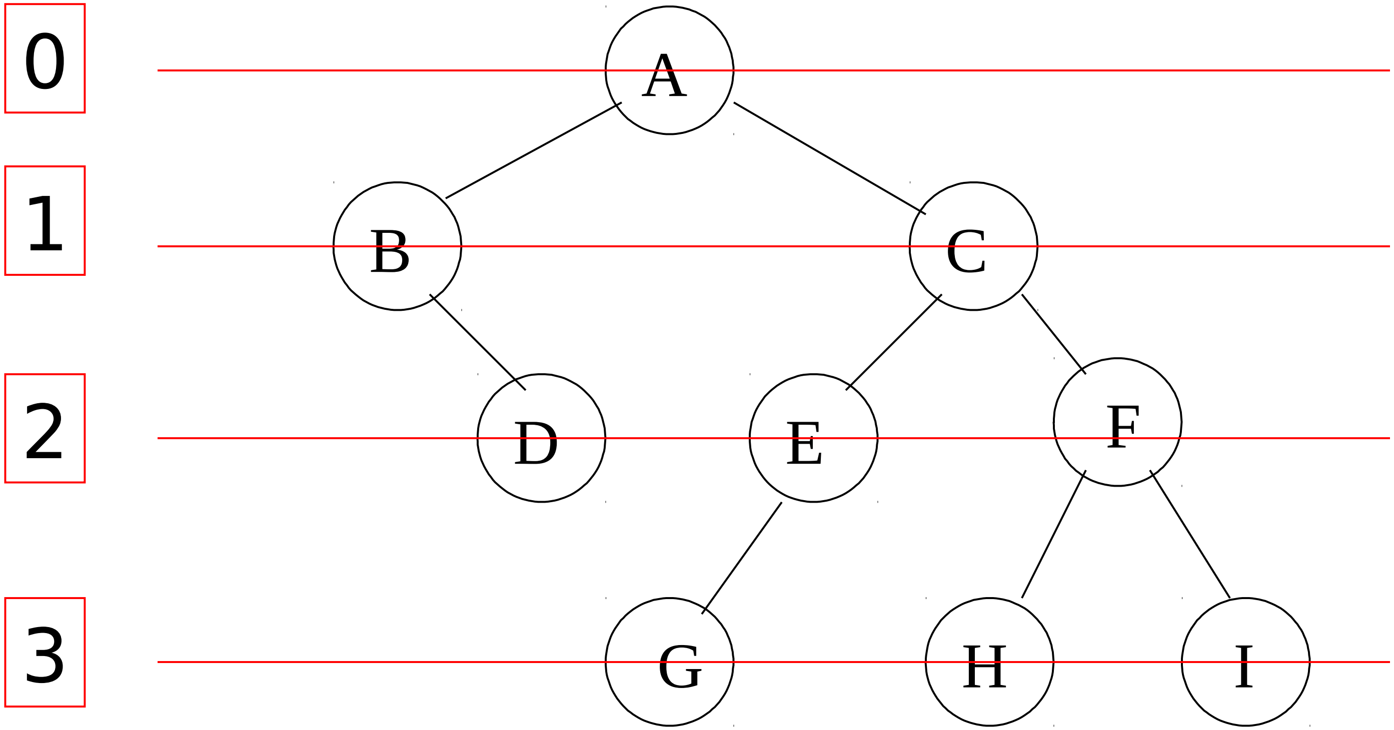
From D to A



2

depth –
the length of the path from the root of the tree to the node

Notation

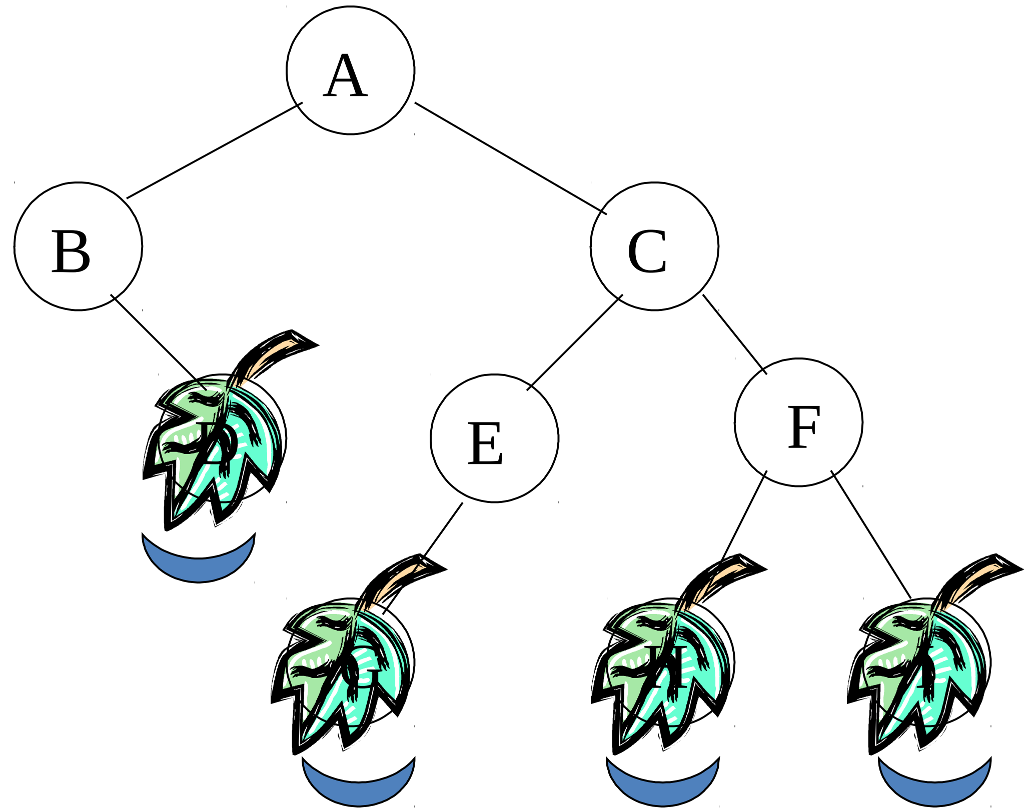


level –

all nodes of depth d are at level d in the tree

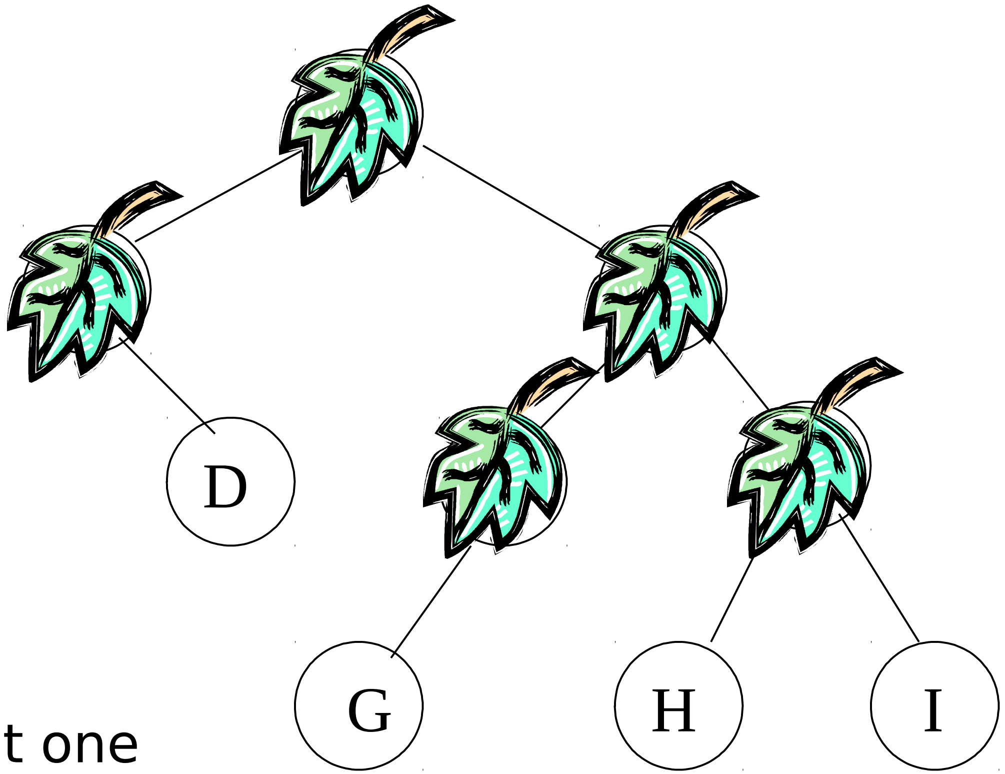
FAR EASTERN UNIVERSITY
East Asia College

Notation



leaf node –
any node that has two empty children

Notation



internal node –

any node that has at least one non-emptyChild

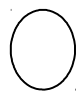
Or

An *internal node* of a *tree* is any *node* which has degree greater than one.

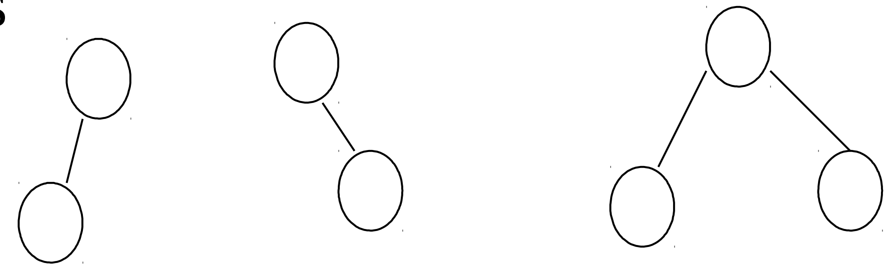
Binary Trees

Some Binary Trees

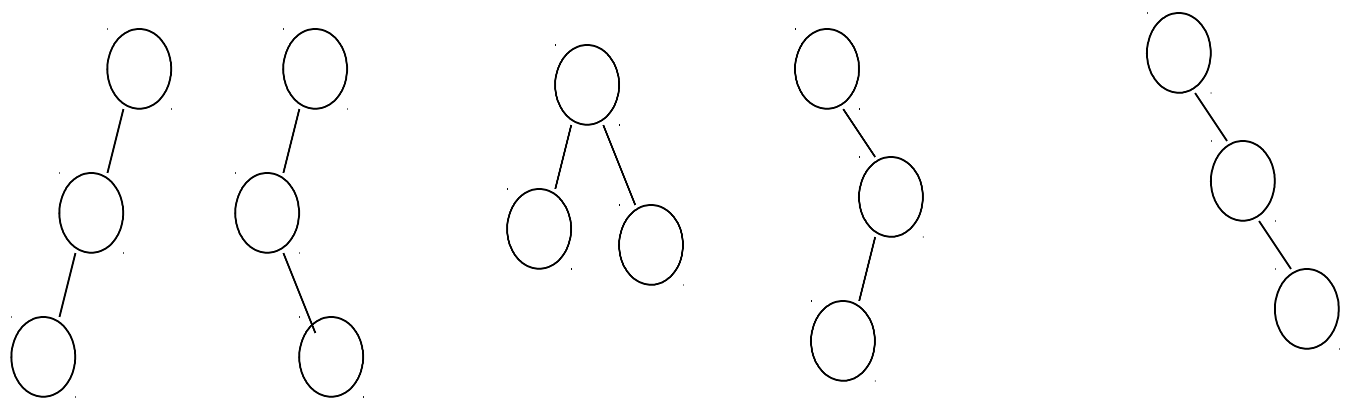
One node



Two nodes



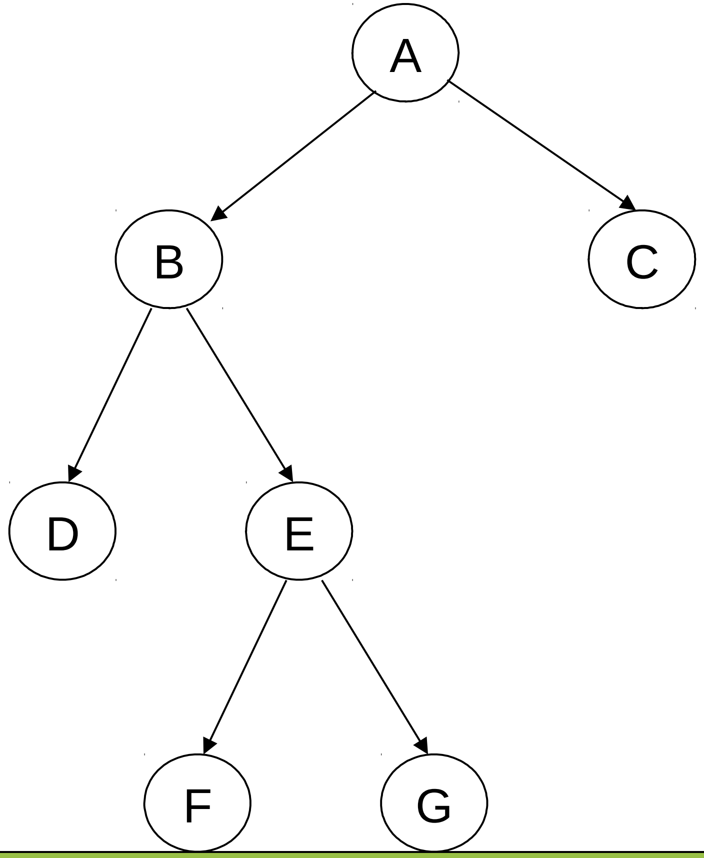
Three nodes



FAR EASTERN UNIVERSITY
East Asia College

Strictly Binary Tree

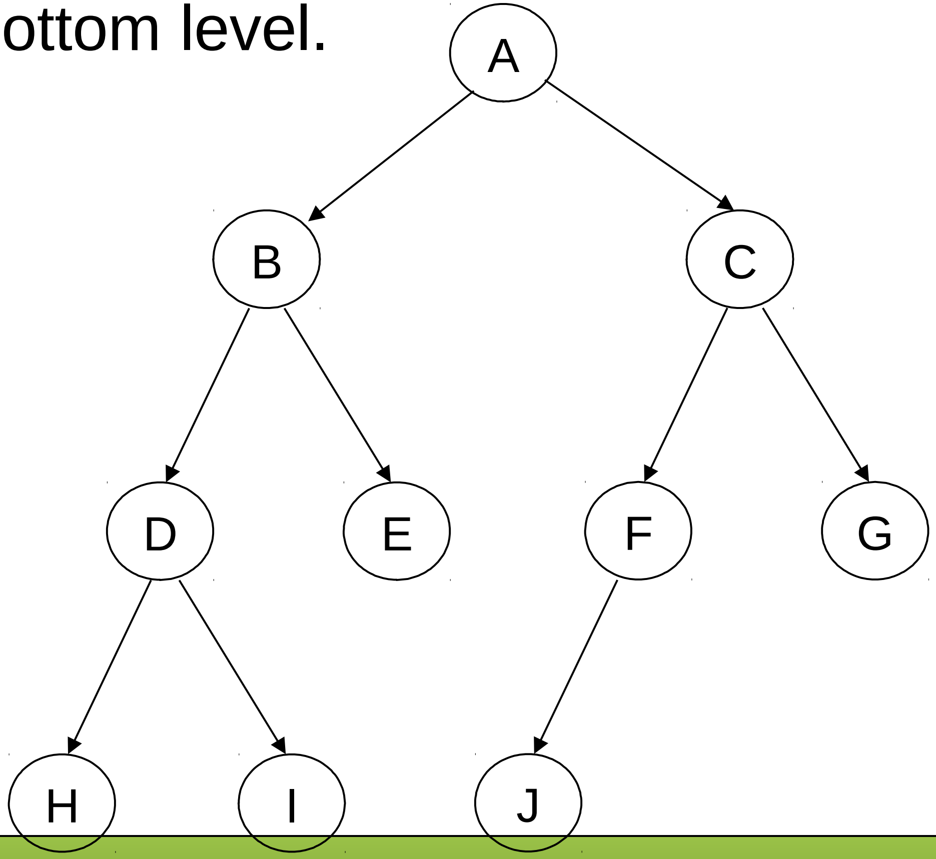
- When every non-leaf node in binary tree is filled with left and right sub-trees, the tree is called strictly binary tree.



Complete Binary Tree

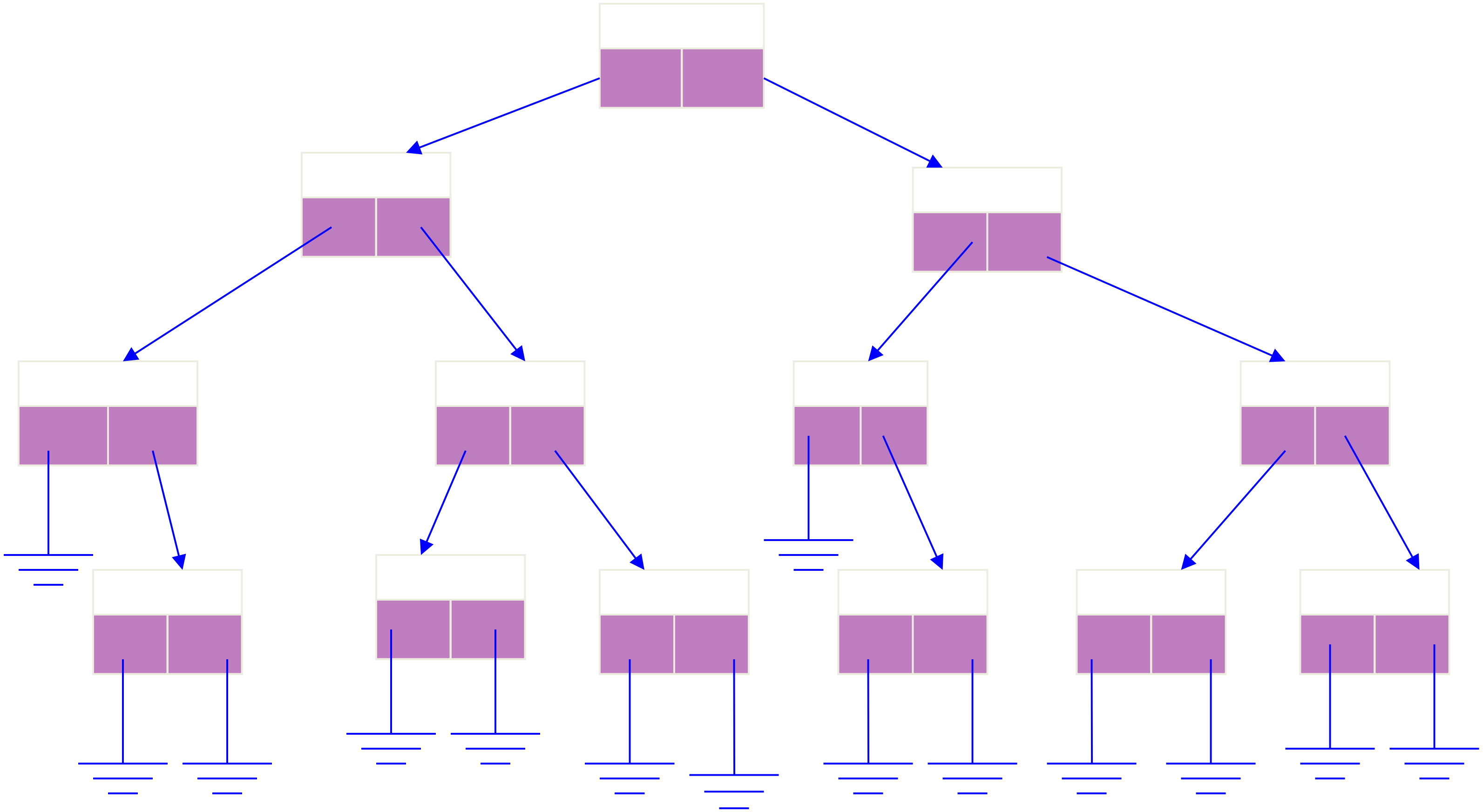
A *Complete binary* tree is:

- A tree in which each level of the tree is completely filled.
- Except, possibly the bottom level.



Dynamic Implementation of Binary Tree

Linked Implementation



Structure Definition of Binary Tree Using Dynamic Implementation

The fundamental component of binary tree is node. In binary tree node should consist of three things.

- **Data**
  Stores given values
- **Left child**
  is a link field and hold the address of its left node
- **Right child.**
  Is a link field and holds the address of its right node.

```
struct node
{
     int data
     node *left_child;
     node *right_child;
};
```

Operations on Binary Tree

Create

- **Create an empty binary tree**

Empty

- **Return true when binary tree is empty else return false.**

Lchild

- **A pointer is returned to left child of a node, when a node is without left child, NULL pointer is returned.**

Rchild

- **A pointer is returned to right child of a node, when a node is without left child, NULL pointer is returned.**

Father/Parent

- **A pointer to father of a node is returned.**

Operations on Binary Tree

Sibling

- A pointer to brother of the node is returned or else NULL pointer is returned.

Tree Traversal

- Inorder Traversal
- Preorder Traversal
- Postorder Traversal

Insert

- To insert a node

Deletion

- To delete a node

Search

- To search a given node

Copy

- Copy one tree into another.

FAR EASTERN UNIVERSITY
East Asia College

Example: Expression Trees

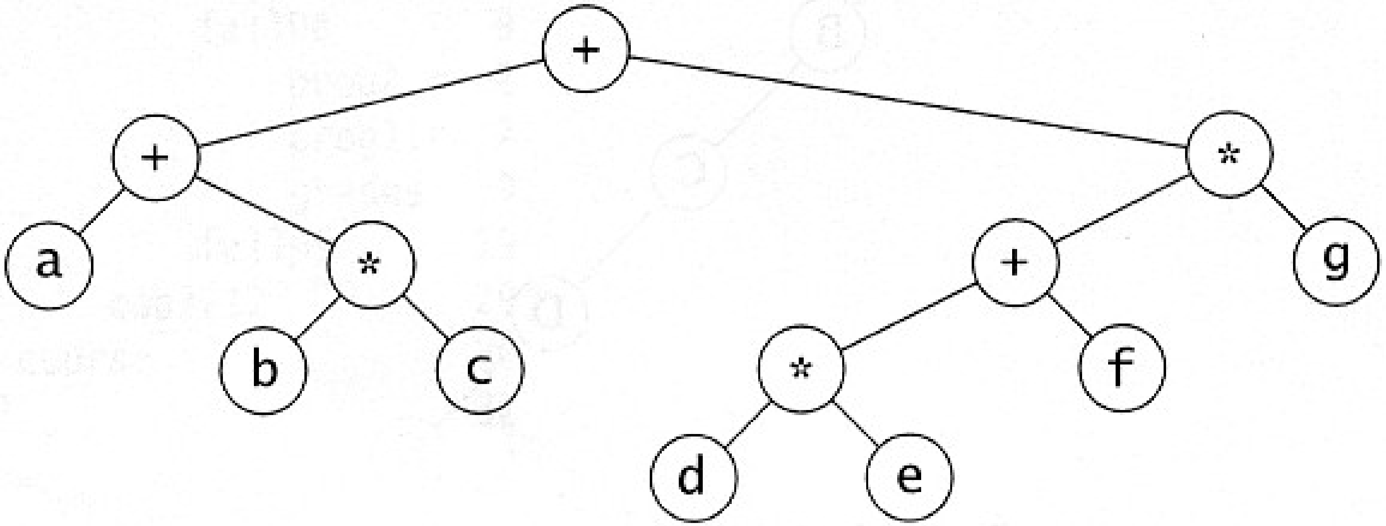


Figure 4.14 Expression tree for (a + b * c) + ((d * e + f) * g)

- Leaves are operands (constants or variables)
- The other nodes (internal nodes) contain operators

Traversal of a Binary Tree

- Used to display/access the data in a tree in a certain order.
- In traversing always right sub-tree is traversed after left sub-tree.
- Three methods of traversing
  - **Preorder Traversing**
    - Root – Left –Right
  - **Inorder Traversing**
    - Left – Root – Right
  - **Postorder Traversing**
    - Left – Right - Root

Preorder Traversal

- **Preorder traversal**
  - **Node – Left – Right**
  - **Prefix expression**
    - **++a*bc*+*defg**

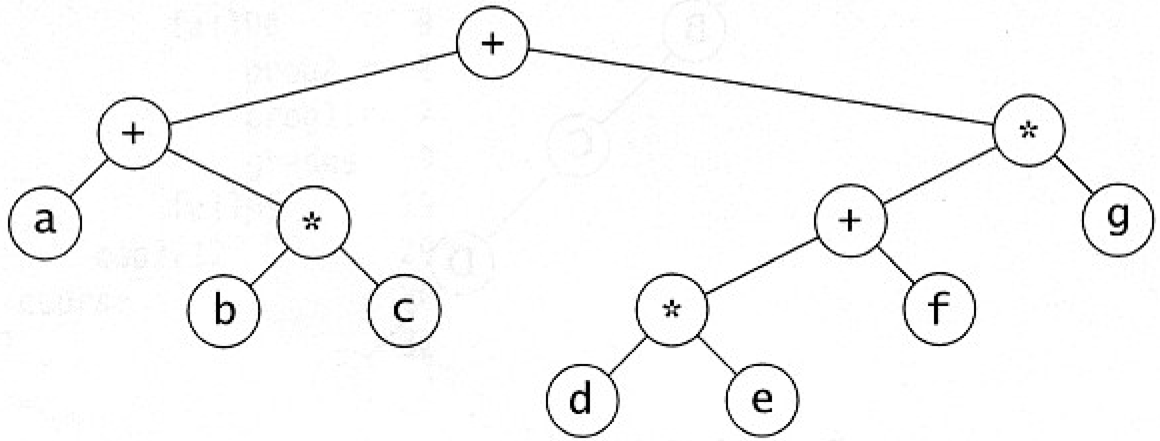
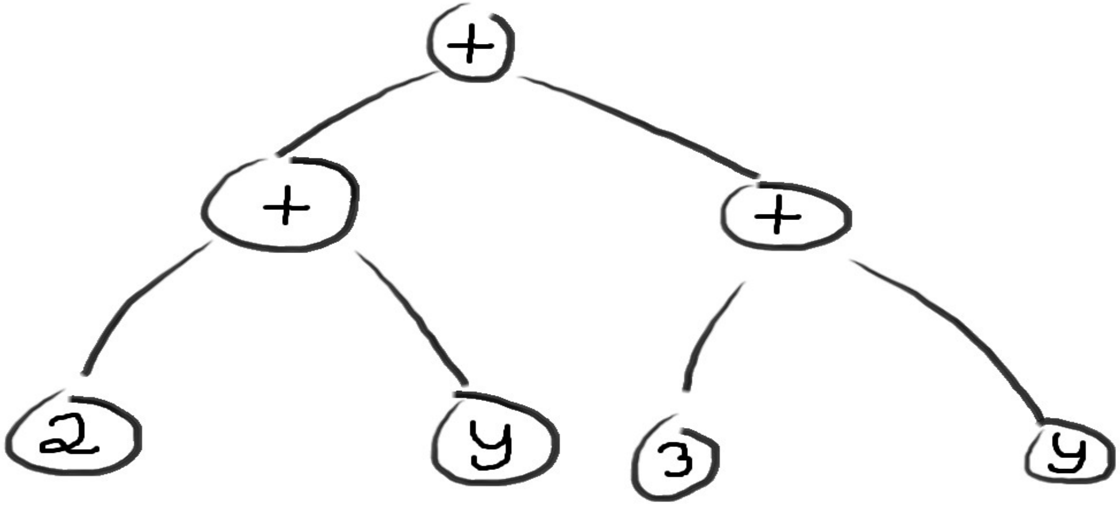


Figure 4.14 Expression tree for (a + b * c) + ((d * e + f) * g)

FAR EASTERN UNIVERSITY
East Asia College

Preorder Traversal

- **Preorder traversal**
  - **Node – Left – Right**
  - **Prefix expression**



Inorder Traversal

- **Inorder traversal**
  - left, node, right.
  - infix expression
    - a+b*c+d*e+f*g

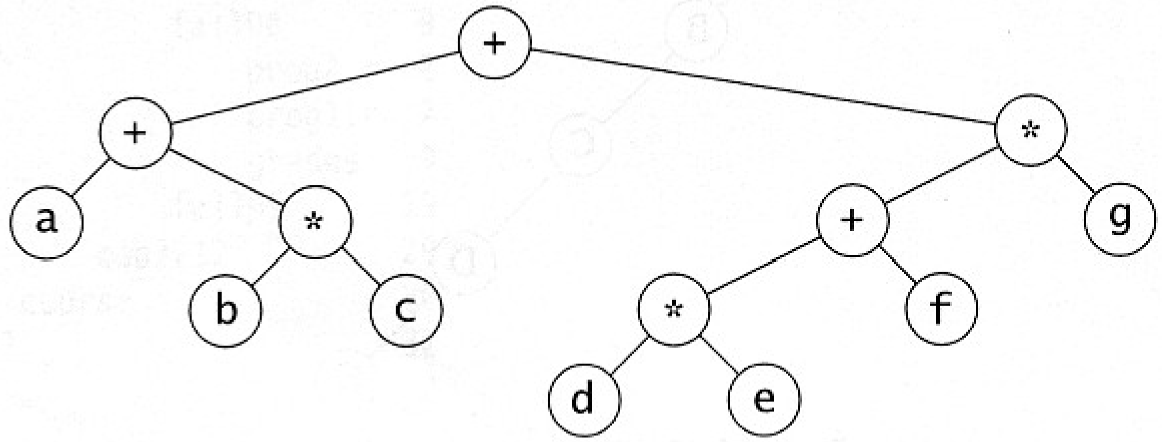
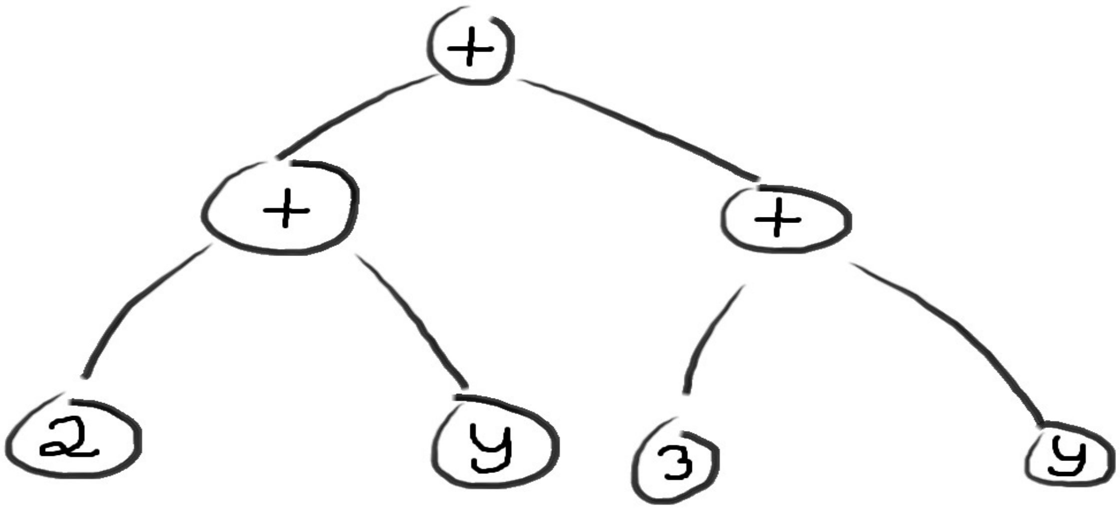


Figure 4.14 Expression tree for (a + b * c) + ((d * e + f) * g)

Inorder Traversal

- **Inorder traversal**
  - **left, node, right.**
  - **infix expression**



Postorder Traversal

- **Postorder Traversal**
  - **left, right, node**
  - **postfix expression**
    - **abc*+de*f+g*+**

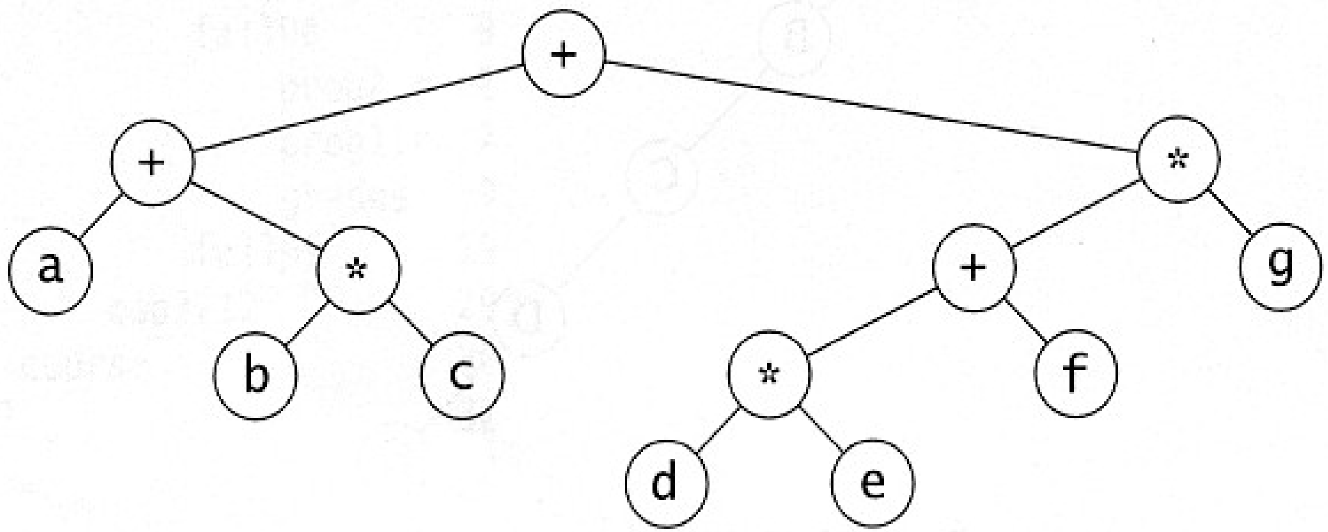
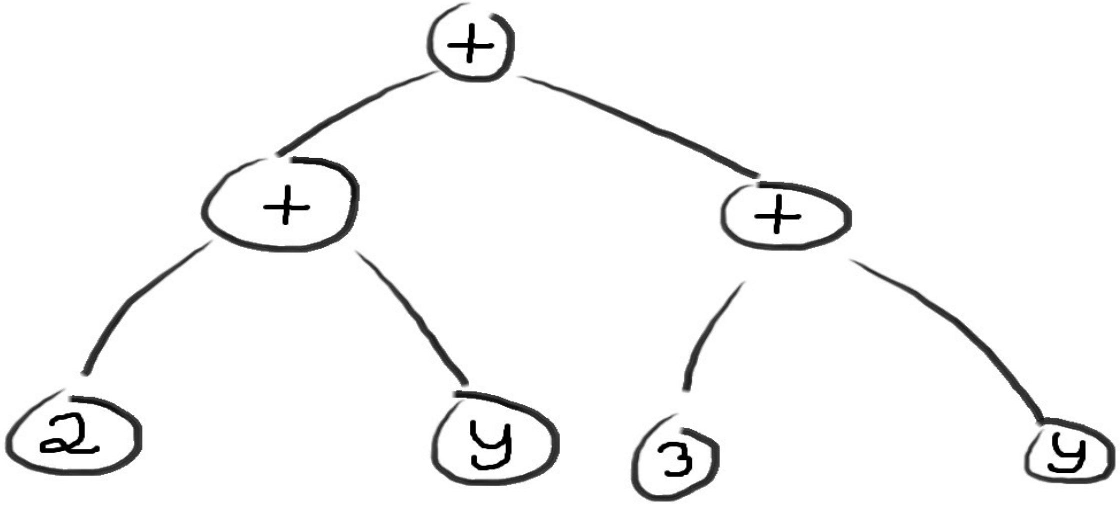


Figure 4.14 Expression tree for (a + b * c) + ((d * e + f) * g)

FAR EASTERN UNIVERSITY
East Asia College

Postorder Traversal

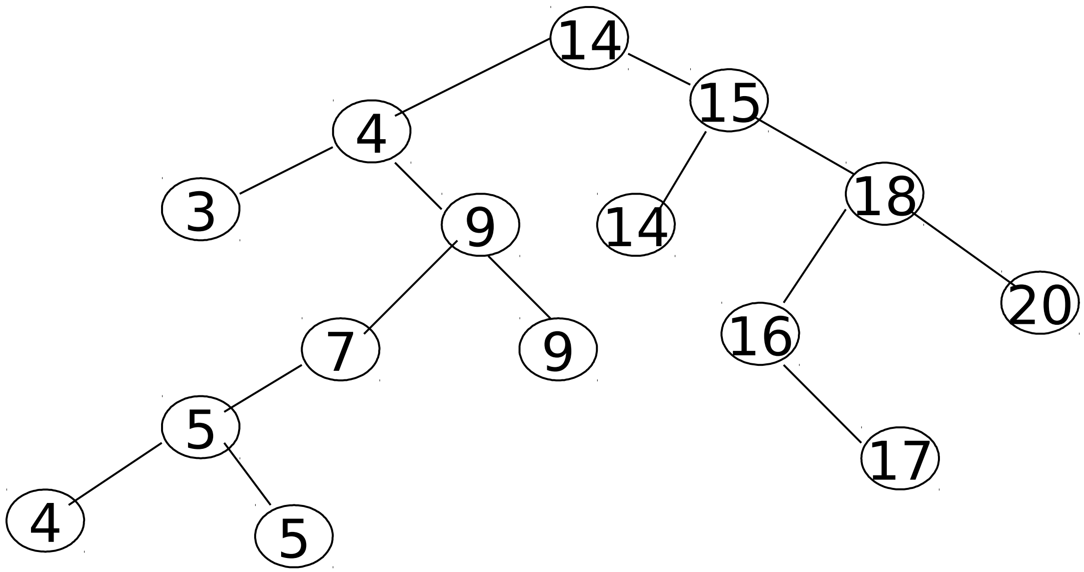
- **Postorder Traversal**
  - **left, right, node**
  - **postfix expression**



FAR EASTERN UNIVERSITY
East Asia College

Traversal Exercise

.Traverse the following tree



? Pre-order Traversal
? Post-order Traversal
? In-order Traversal

FAR EASTERN UNIVERSITY
East Asia College

Binary Search Tree

Data Structures & Algorithms in C++

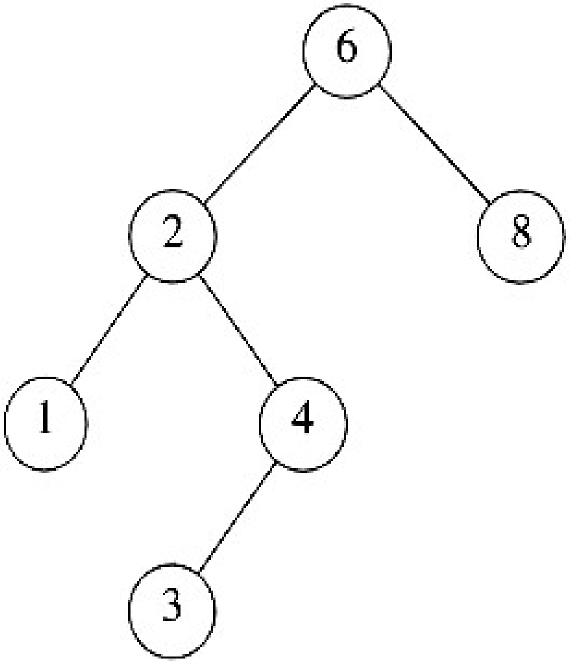
Binary Search Tree

- Stores keys in the nodes in a way so that searching, insertion and deletion can be done efficiently.

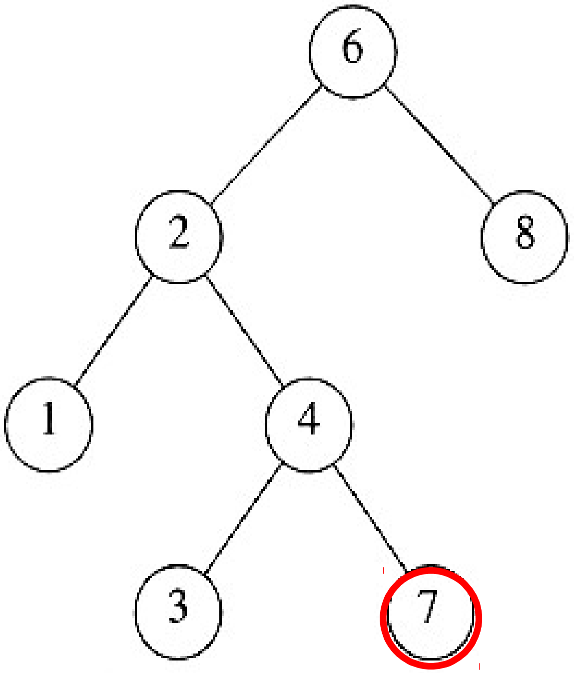
- Binary search tree is either empty or each node N of tree satisfies the following property
  - The Key value in the left child is not more than the value of root
  - The key value in the right child is more than or identical to the value of root
  - All the sub-trees, i.e. left and right sub-trees follow the two rules mention above.

FAR EASTERN UNIVERSITY
East Asia College

Examples



A binary search tree

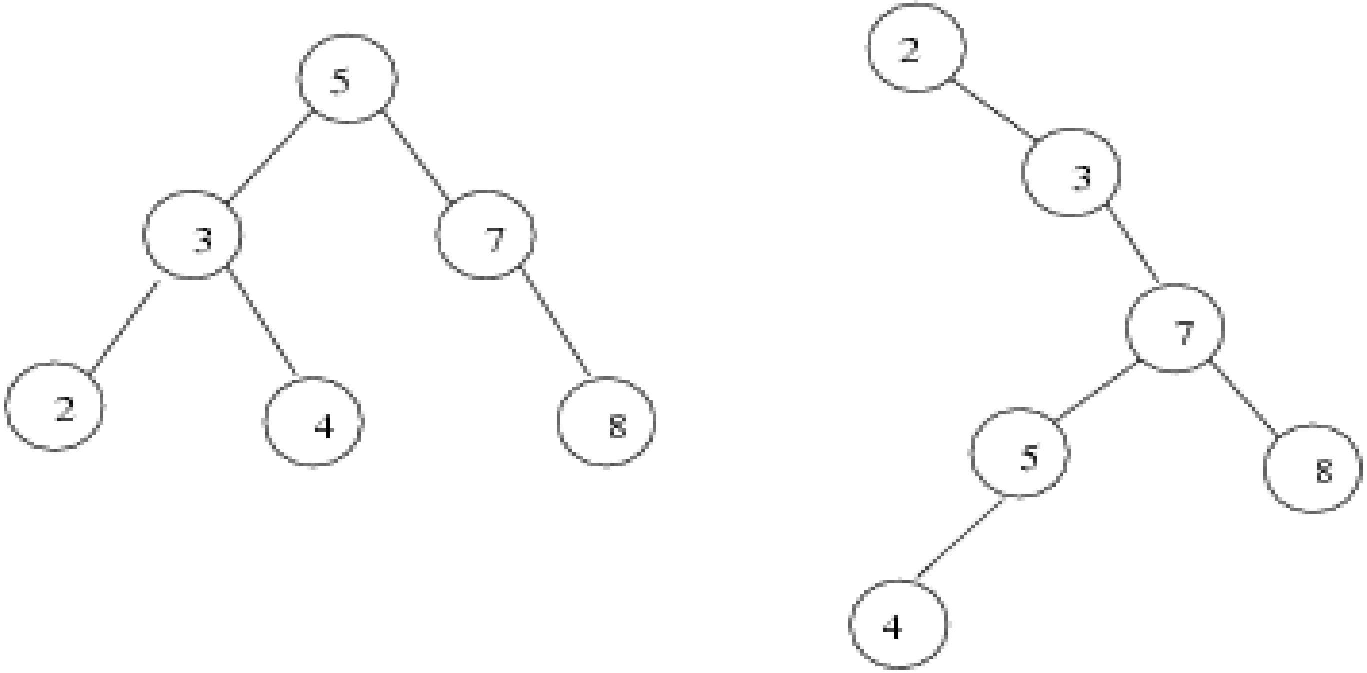


Not a binary search tree

FAR EASTERN UNIVERSITY
East Asia College

Example 2

Two binary search trees representing the same Data set:

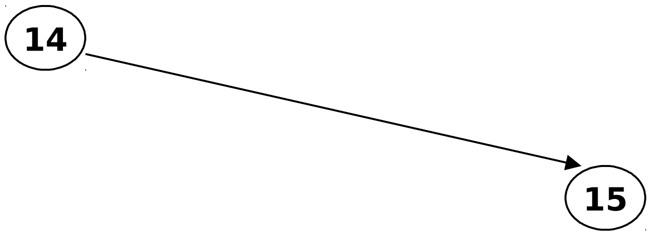


FAR EASTERN UNIVERSITY
East Asia College

Example

- Input list of numbers:

14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

14 15 4 9 7 18 3 5 16 4 20 17 9 14 5

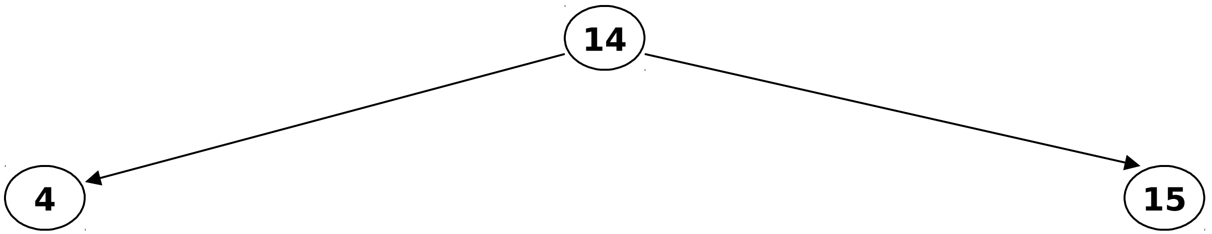
14

FAR EASTERN UNIVERSITY
East Asia College

Example

- Input list of numbers:

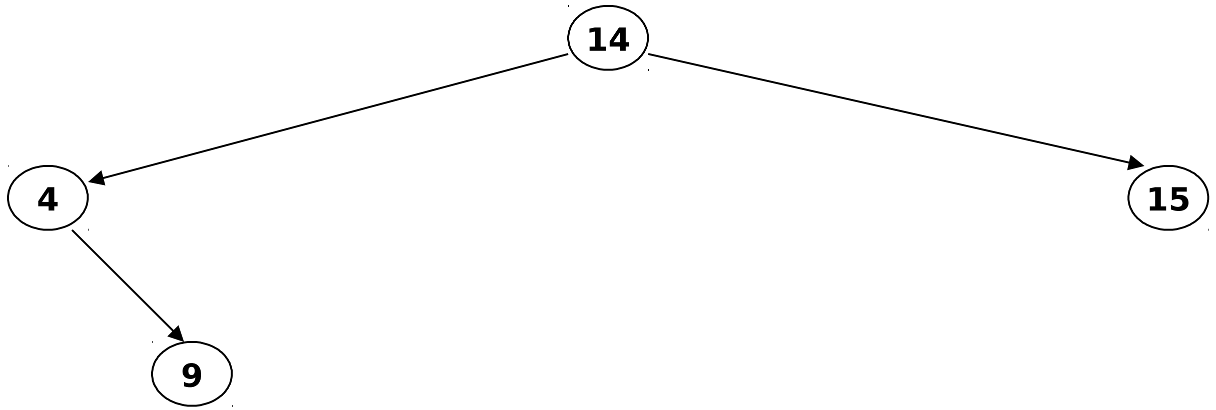
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

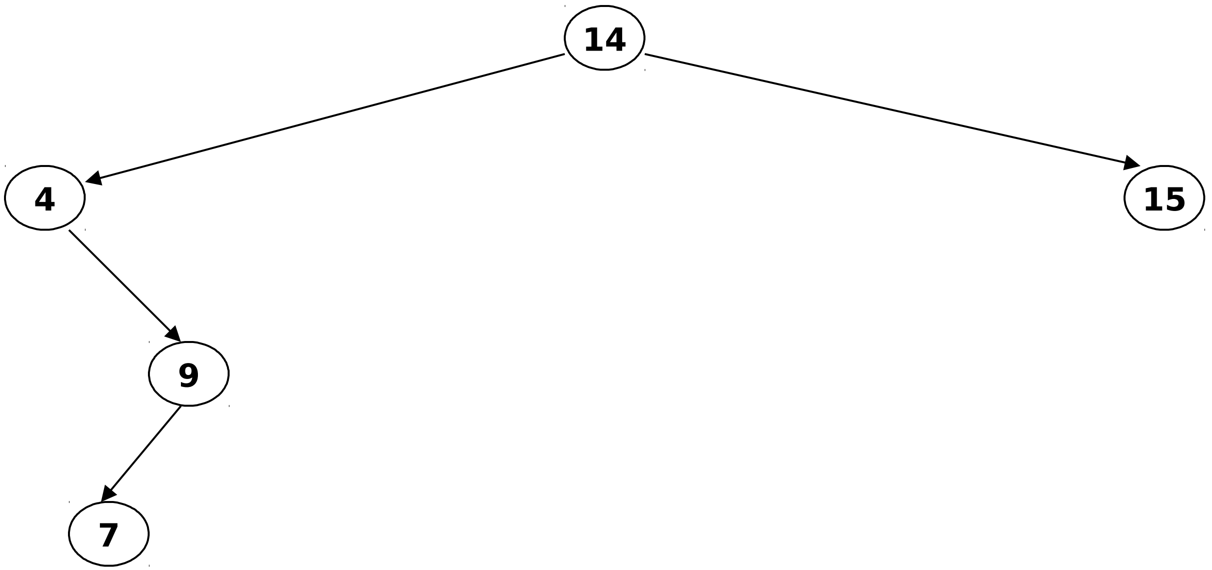
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

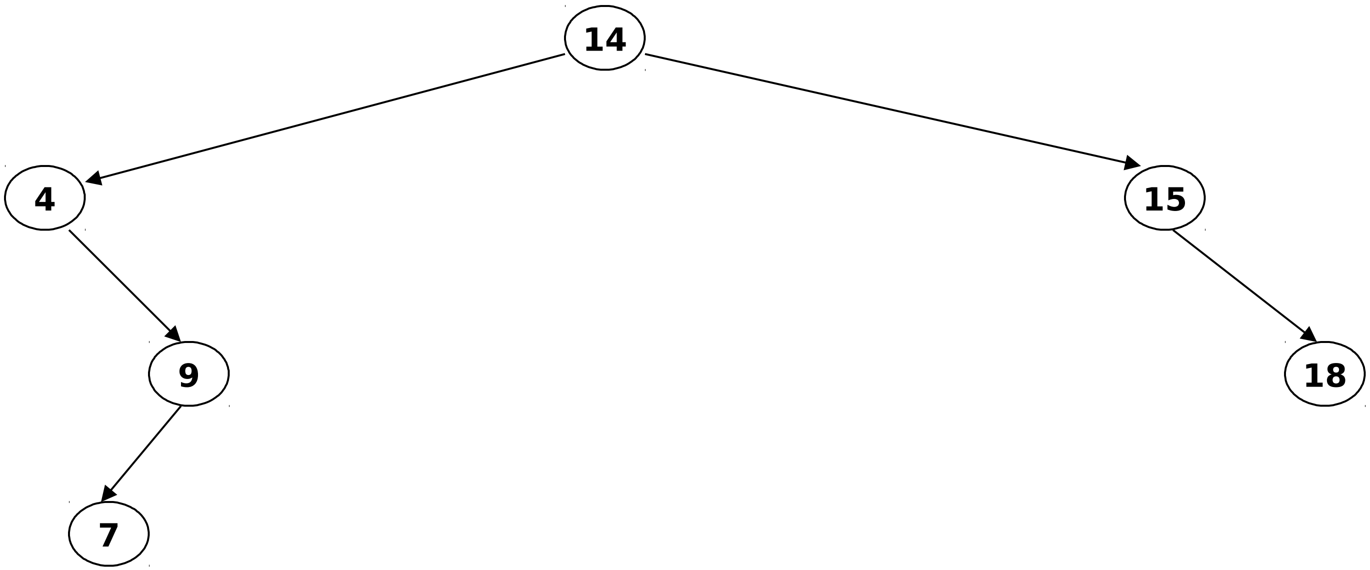
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

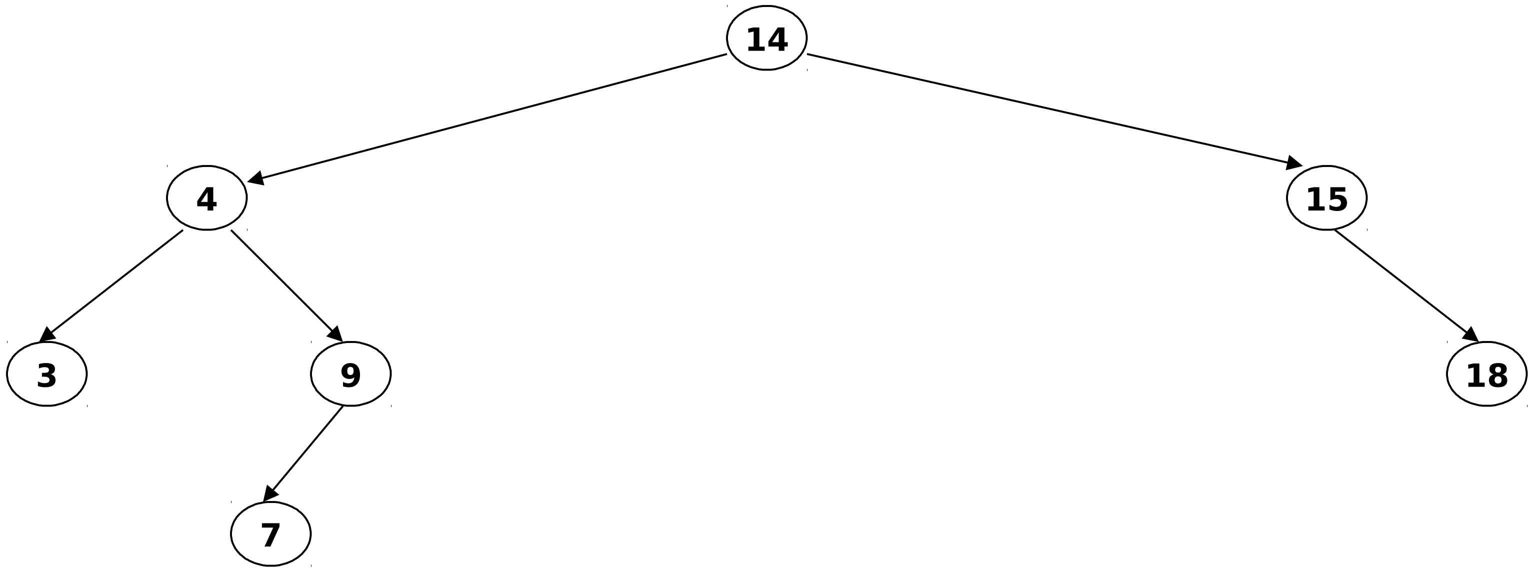
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

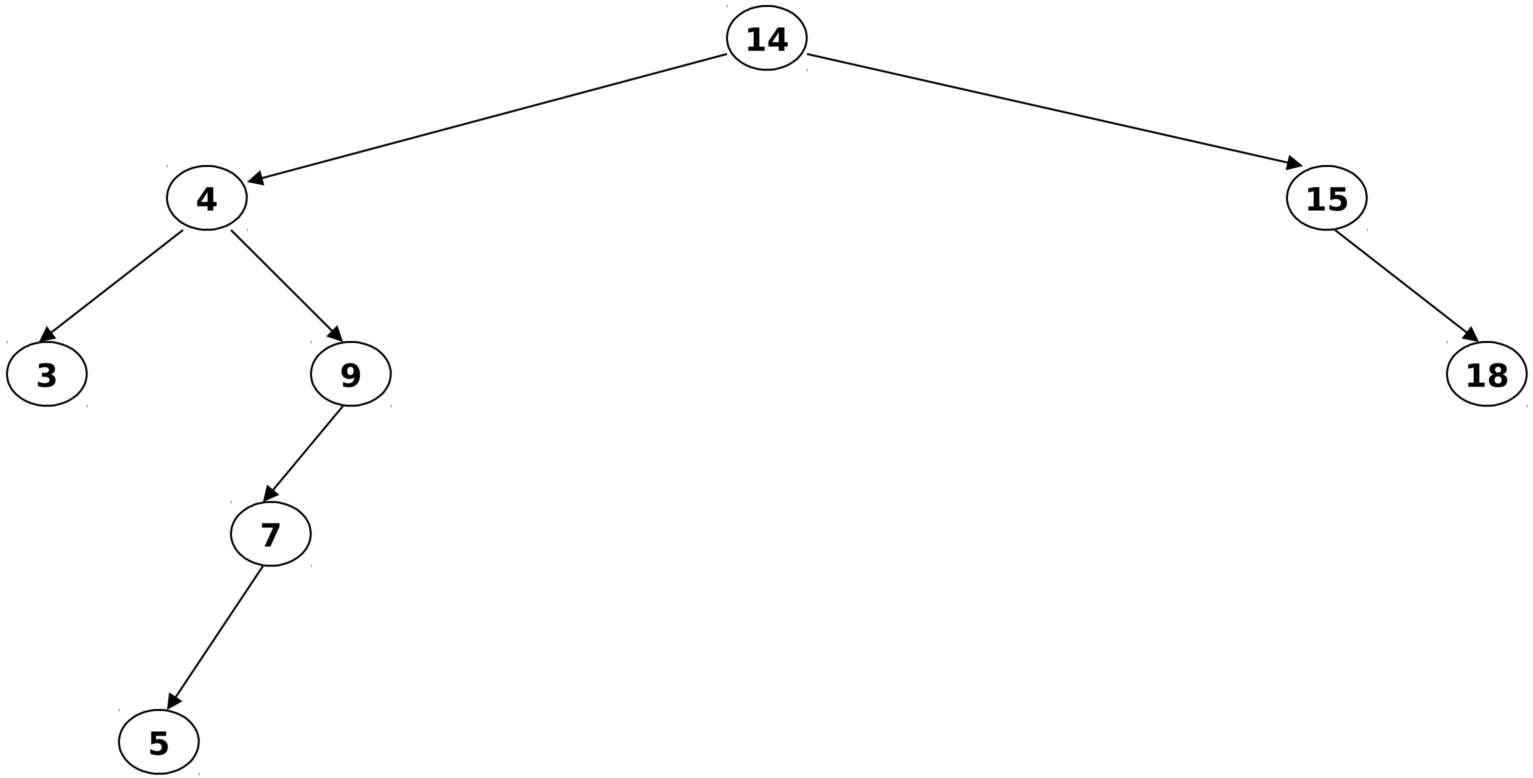
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

14 15 4 9 7 18 3 5 16 4 20 17 9 14 5

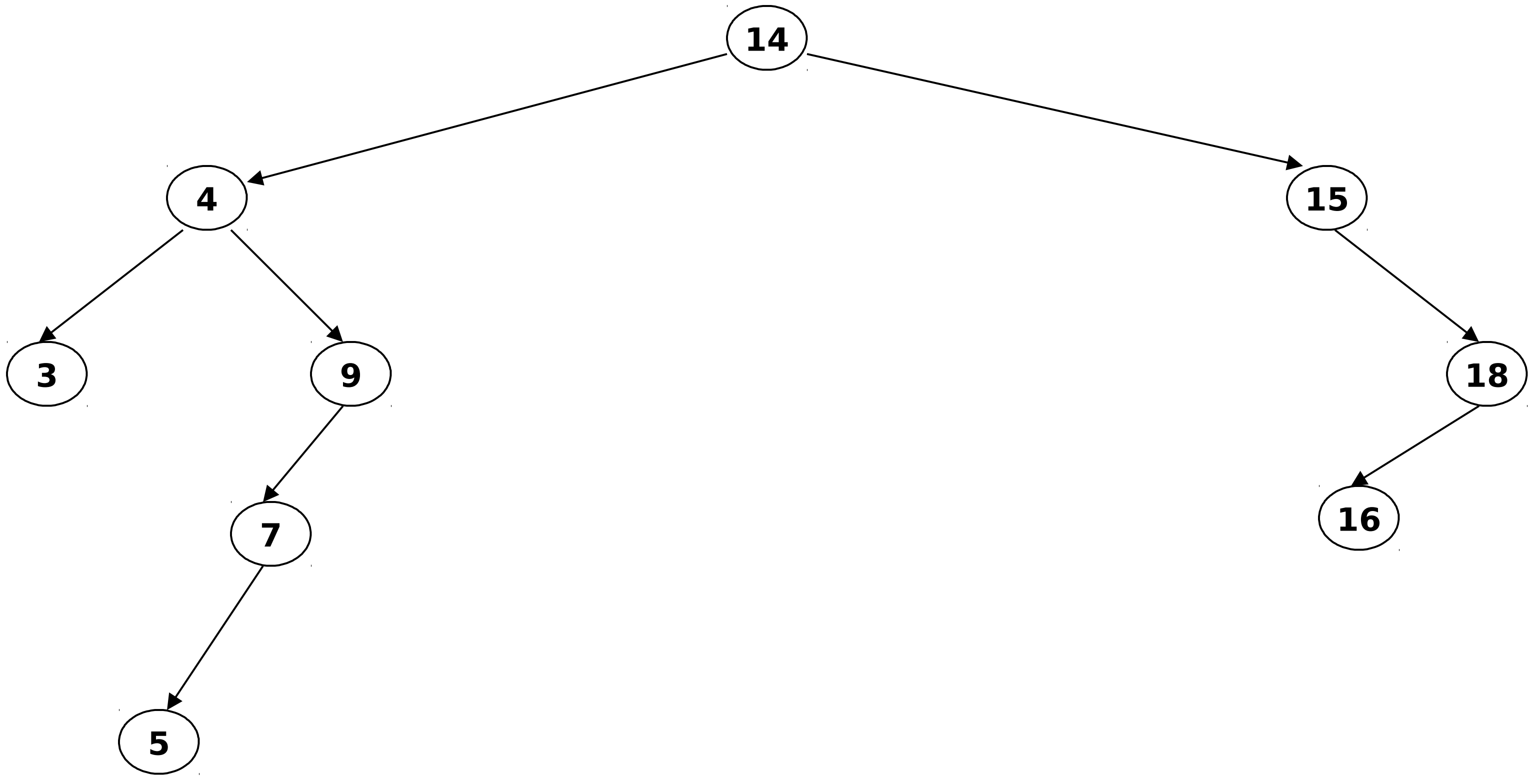


FAR EASTERN UNIVERSITY
East Asia College

Example

- Input list of numbers:

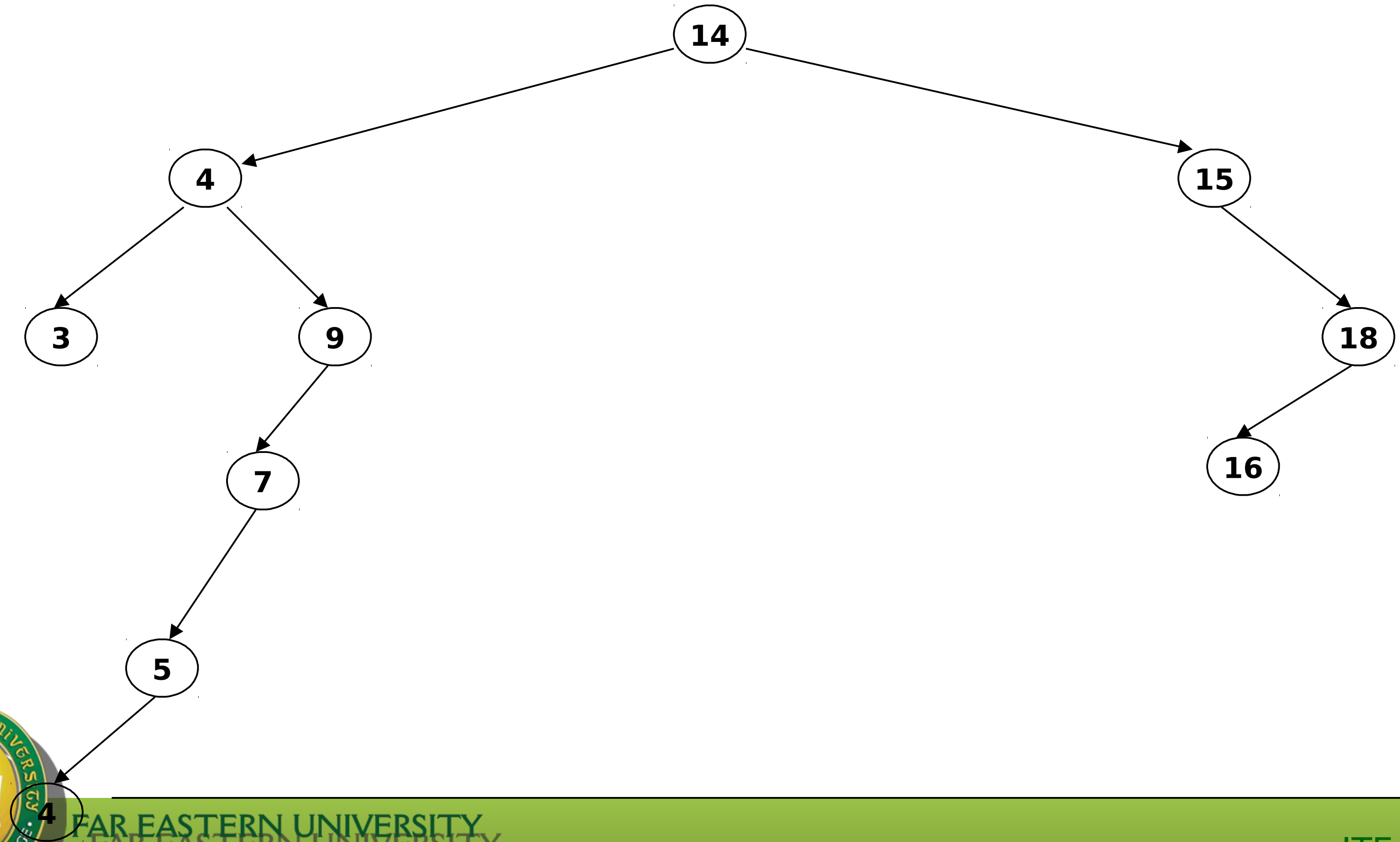
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

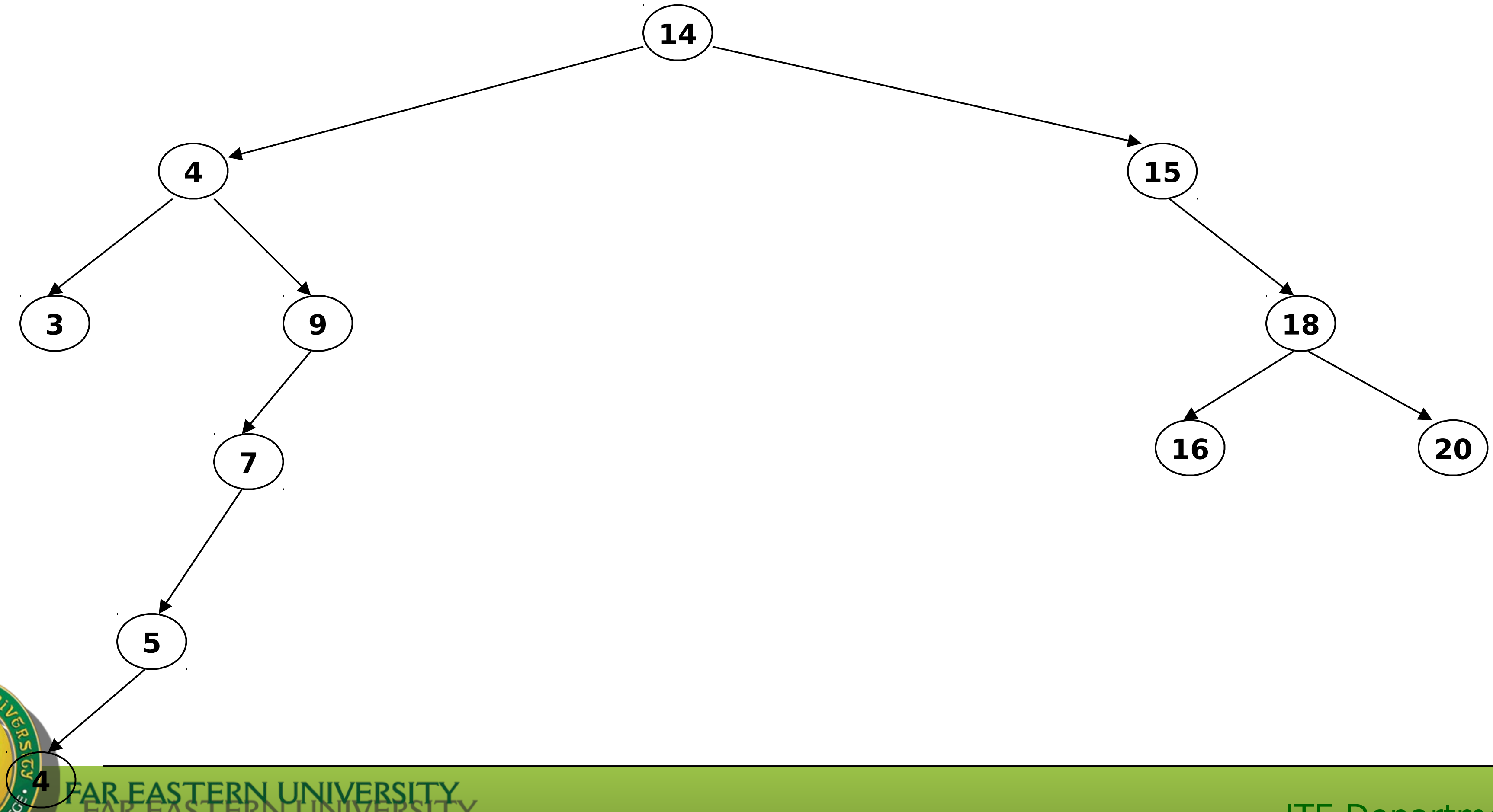
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

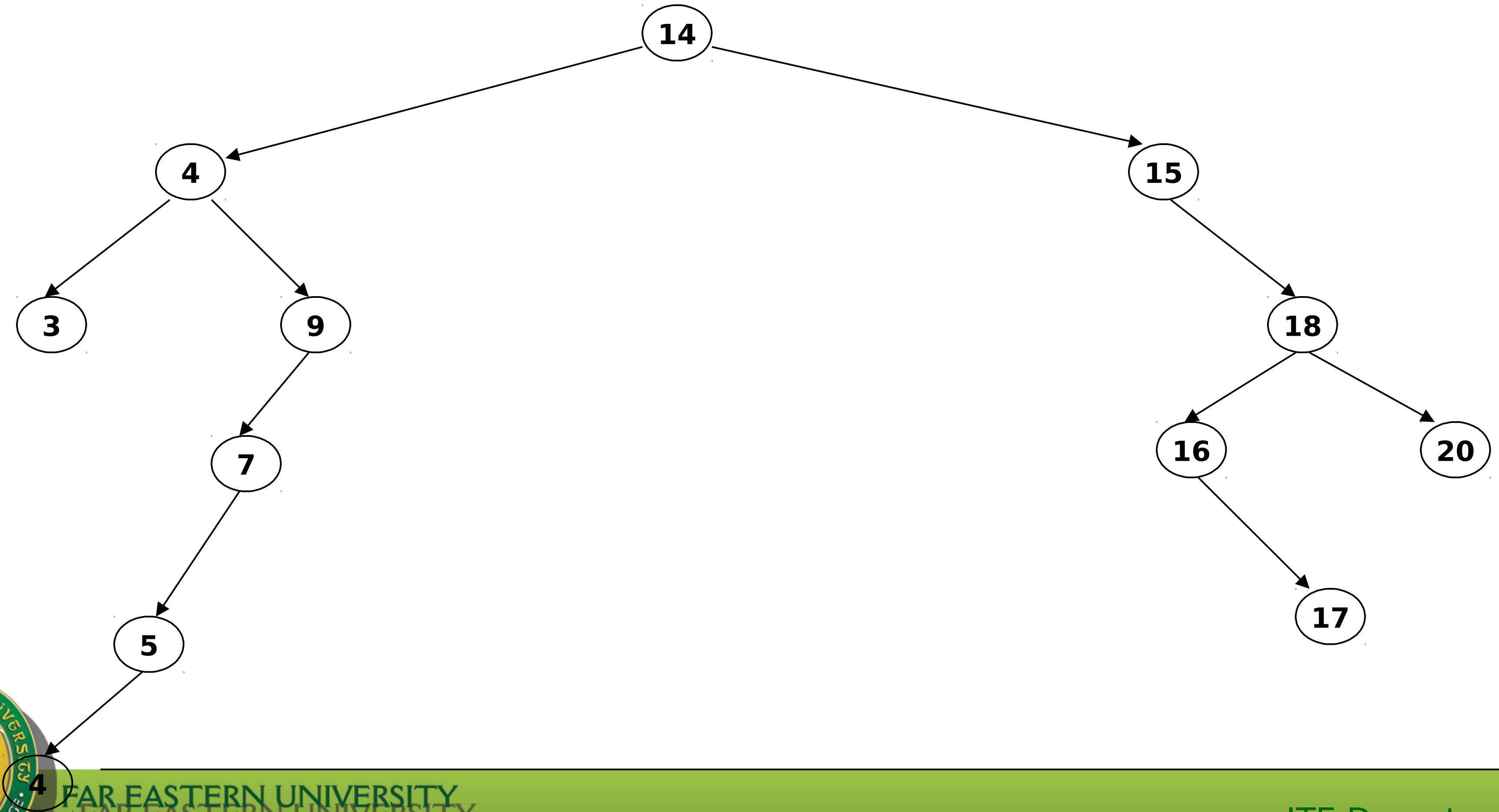
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

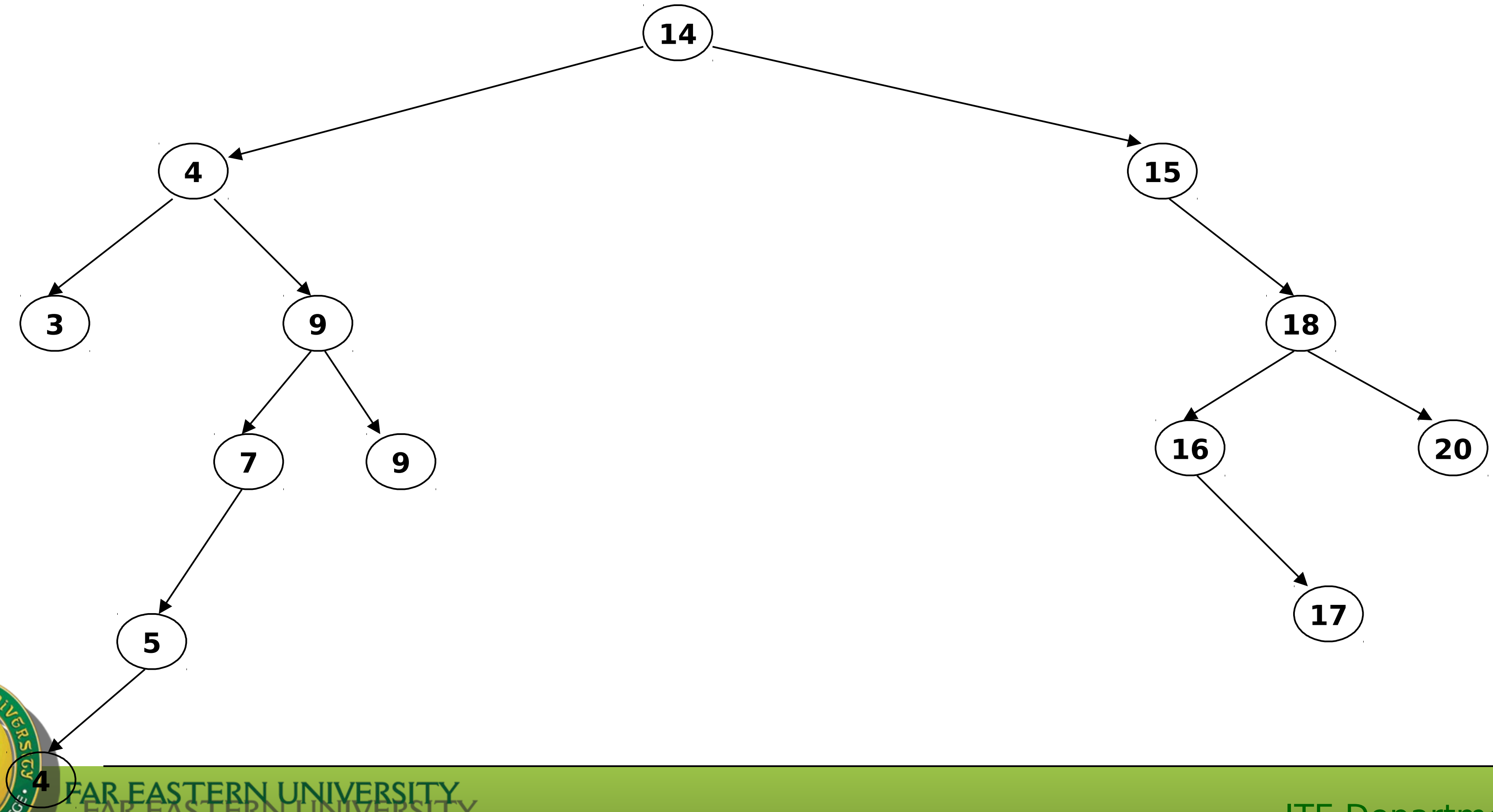
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

14 15 4 9 7 18 3 5 16 4 20 17 9 14 5

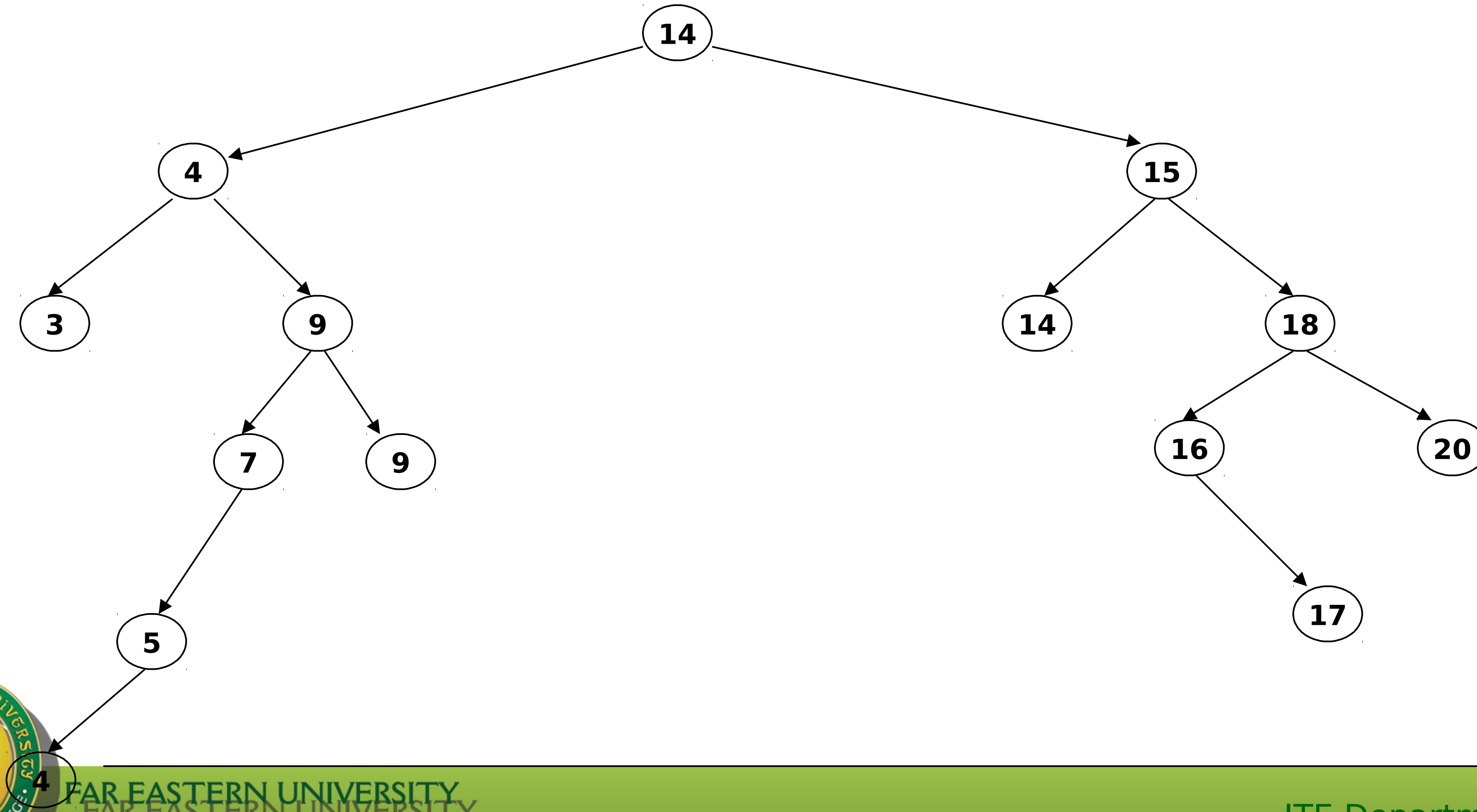


FAR EASTERN UNIVERSITY
East Asia College

Example

- Input list of numbers:

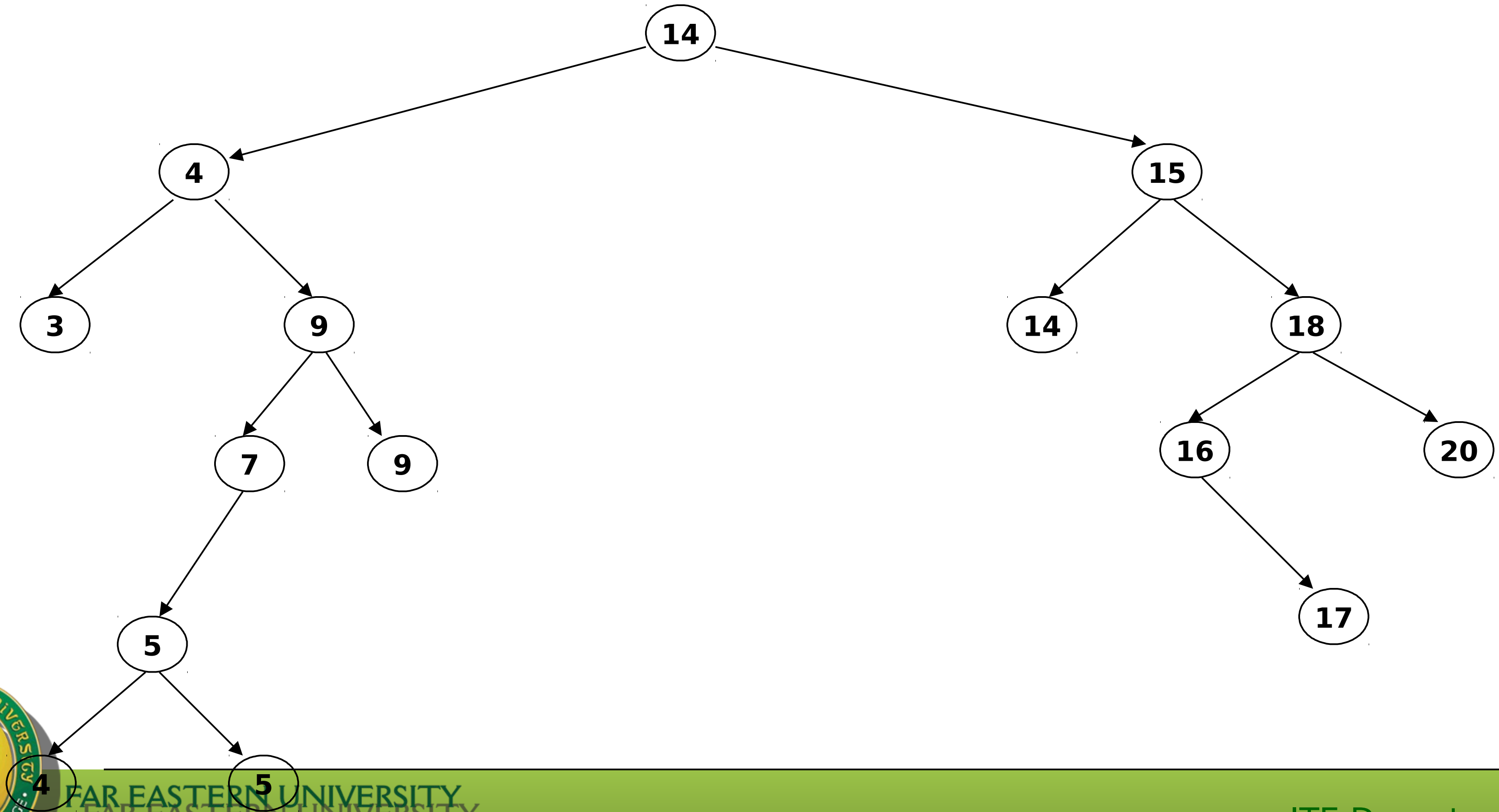
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

- Input list of numbers:

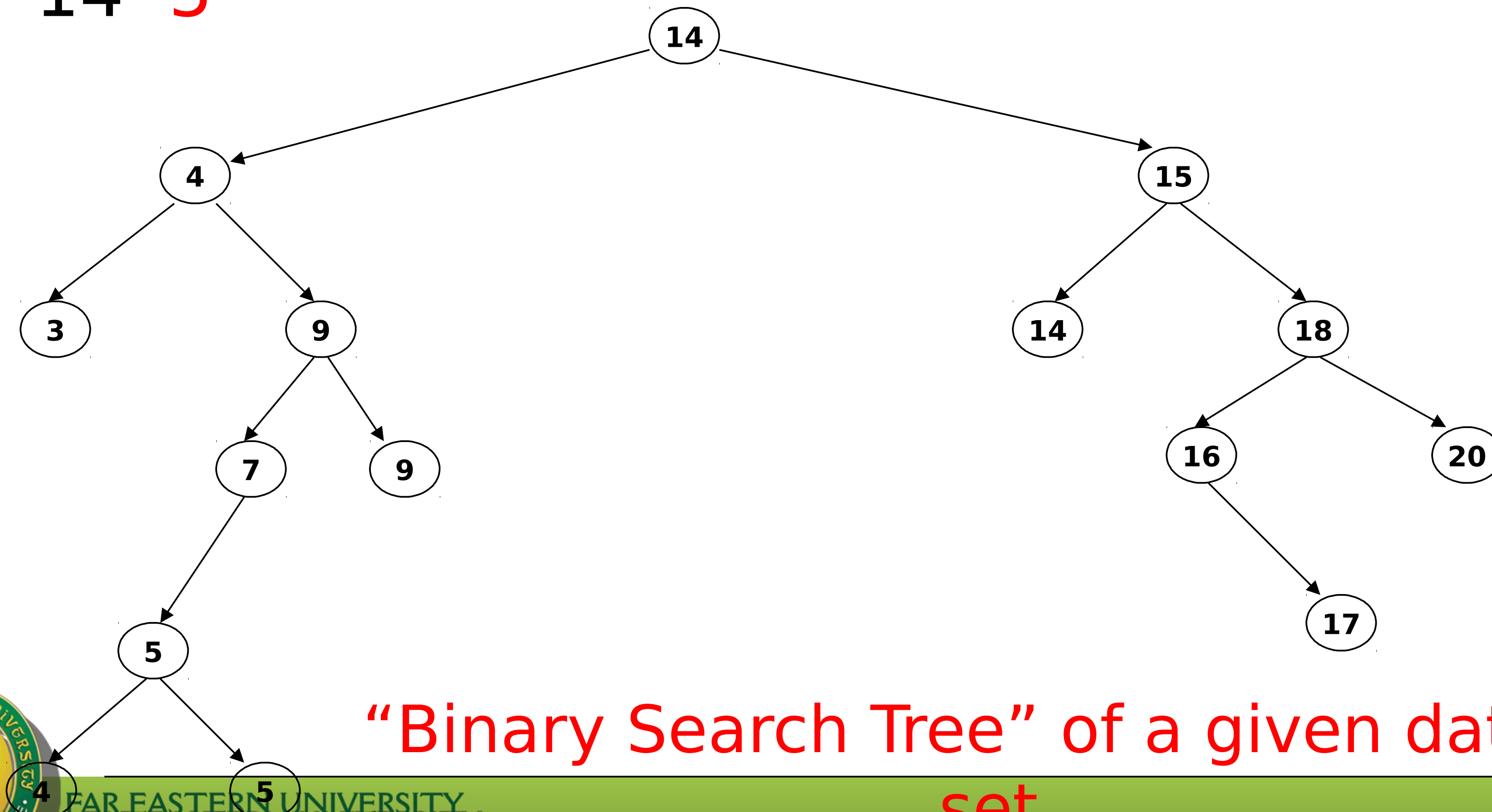
14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



Example

Input list of numbers:

14 15 4 9 7 18 3 5 16 4 20 17 9 14 5



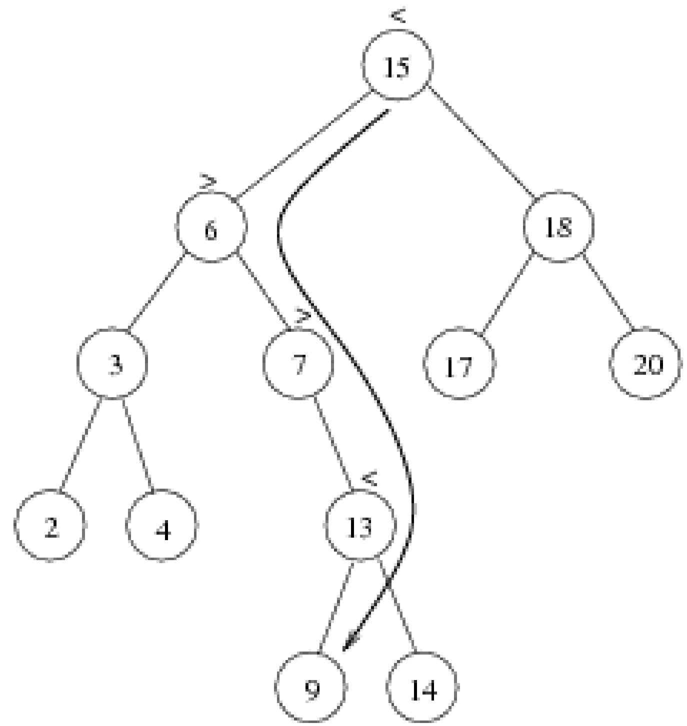
“Binary Search Tree” of a given data set

Searching in Binary Search Tree

- **Three steps of searching**
  - **The item which is to be searched is compared with the root node. If the item is equal to the root, then we are done.**
  - **If its less than the root node then we search in the left sub-tree.**
  - **If its more than the root node then we search in the right sub-tree.**

- **The above process will continue till the item is found or you reached end of the tree.**

Example: Search for 9 ...

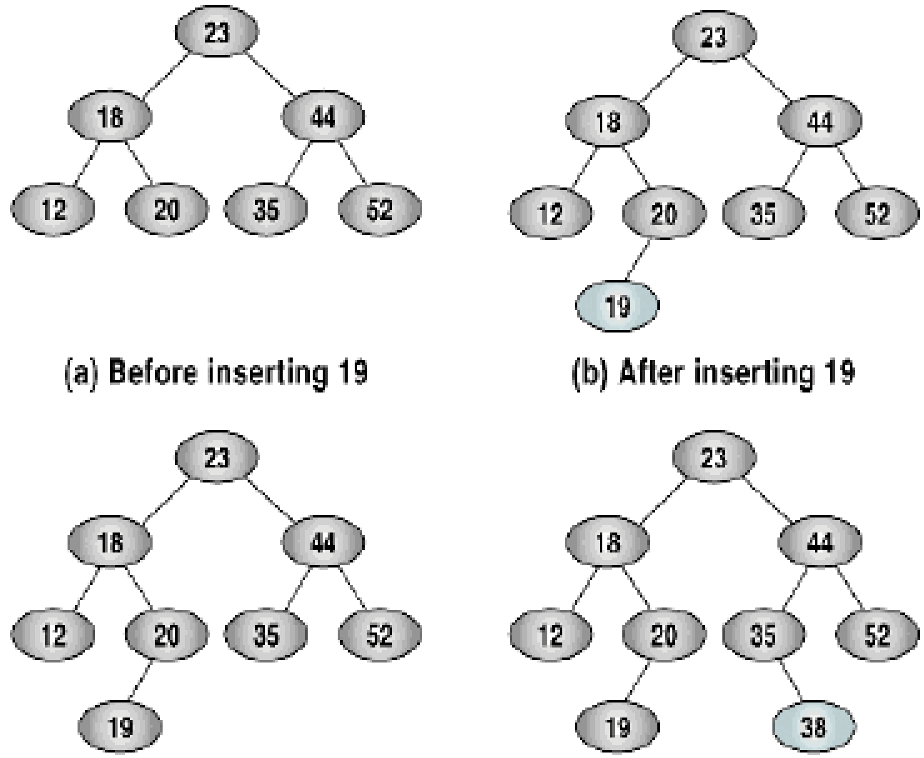


Search for 9:

1. compare 9:15(the root), go to left subtree;
2. compare 9:6, go to right subtree;
3. compare 9:7, go to right subtree;
4. compare 9:13, go to left subtree;
5. compare 9:9, found it!

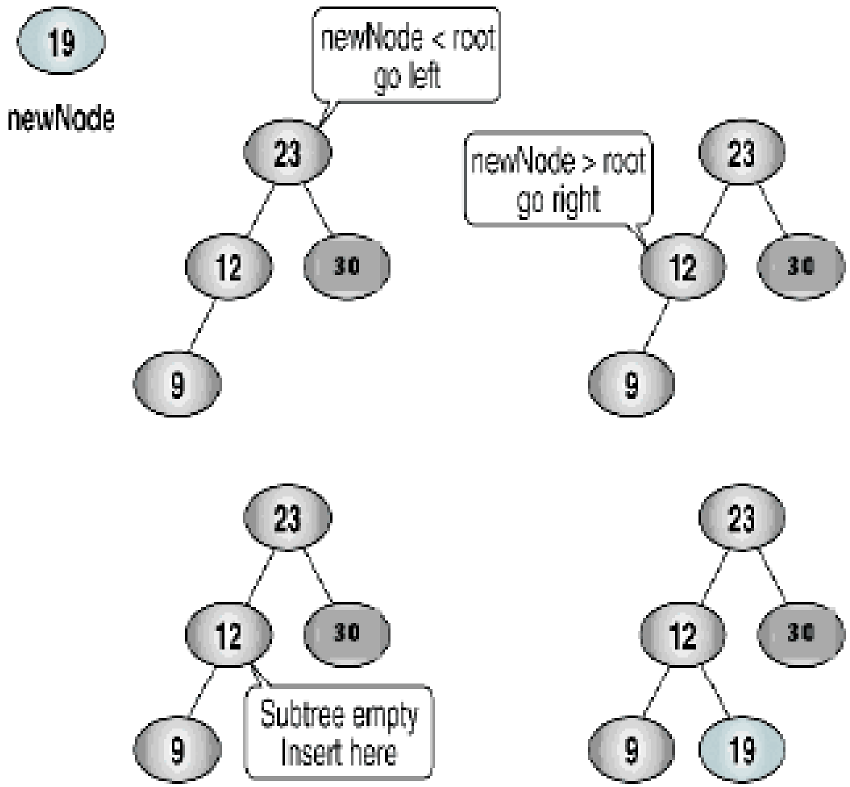
Insertion in BST

- **Three steps of insertion**
  - **If the root of the tree is NULL then insert the first node and root points to that node.**
  - **If the inserted number is lesser than the root node then insert the node in the left sub-tree.**
  - **If the inserted number is greater than the root node then insert the node in the right sub-tree.**



(a) Before inserting 19
(b) After inserting 19
(c) Before inserting 38
(d) After inserting 38

FAR EASTERN UNIVERSITY
East Asia College



FAR EASTERN UNIVERSITY
East Asia College

Deletion in BST

- When we delete a node, we need to consider how we take care of the children of the deleted node.
  - This has to be done such that the property of the search tree is maintained.

Deletion in BST

Three cases:

(1) The node is a leaf

- **Delete it immediately**

(2) The node has one child

- **Adjust a pointer from the parent to bypass that node**

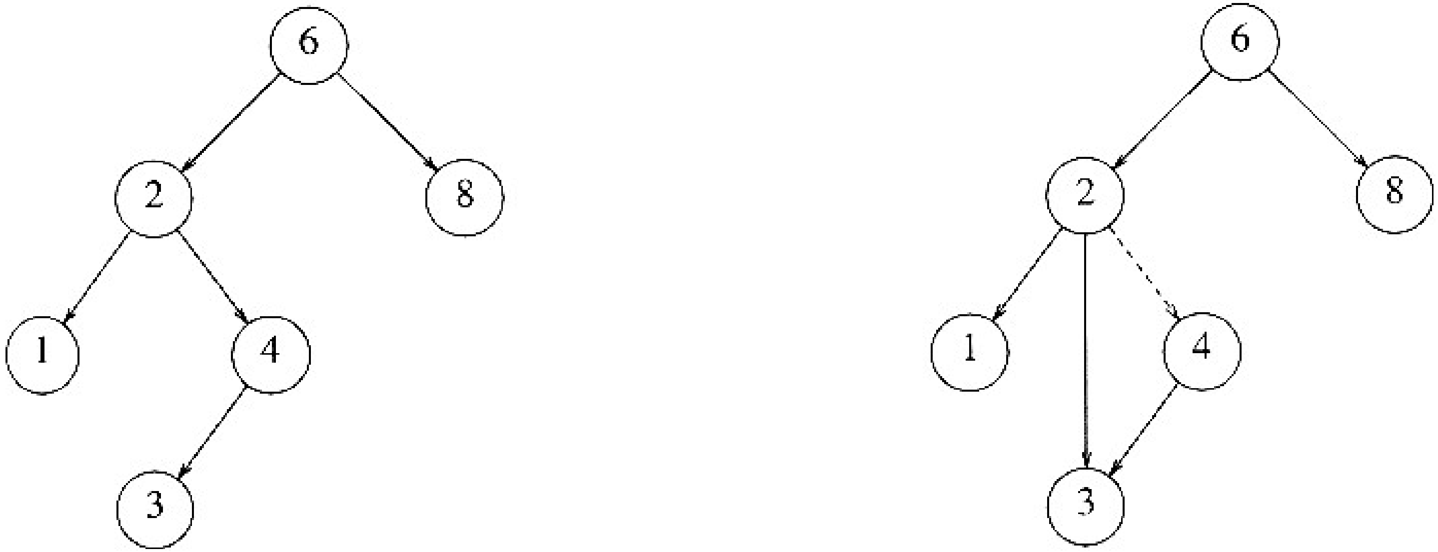


Figure 4.24 Deletion of a node (4) with one child, before and after

FAR EASTERN UNIVERSITY
East Asia College

Deletion in BST

(3) The node has 2 children

- **Replace the key of that node with the minimum element at the right subtree**
- **Delete the minimum element**
  - **Has either no child or only right child because if it has a left child, that left child would be smaller and would have been chosen. So invoke case 1 or 2.**

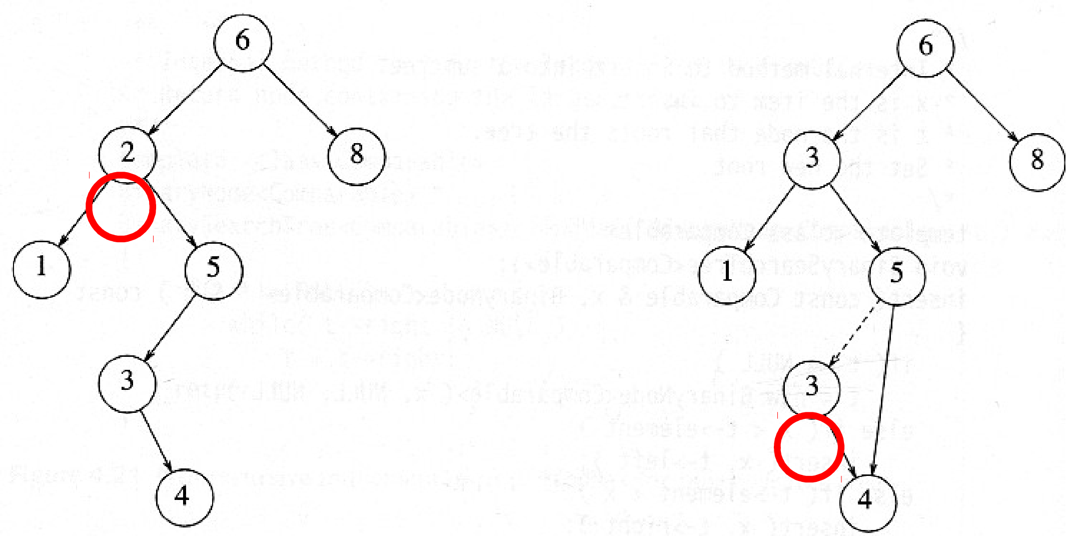


Figure 4.25 Deletion of a node (2) with two children, before and after

Deletion

- There are the following possible cases when we delete a node:
- The node to be deleted has no children. In this case, all we need to do is delete the node.
- The node to be deleted has only a right subtree. We delete the node and attach the right subtree to the deleted node's parent.
- The node to be deleted has only a left subtree. We delete the node and attach the left subtree to the deleted node's parent.
- The node to be deleted has two subtrees. It is possible to delete a node from the middle of a tree, but the result tends to create very unbalanced trees.

FAR EASTERN UNIVERSITY
East Asia College

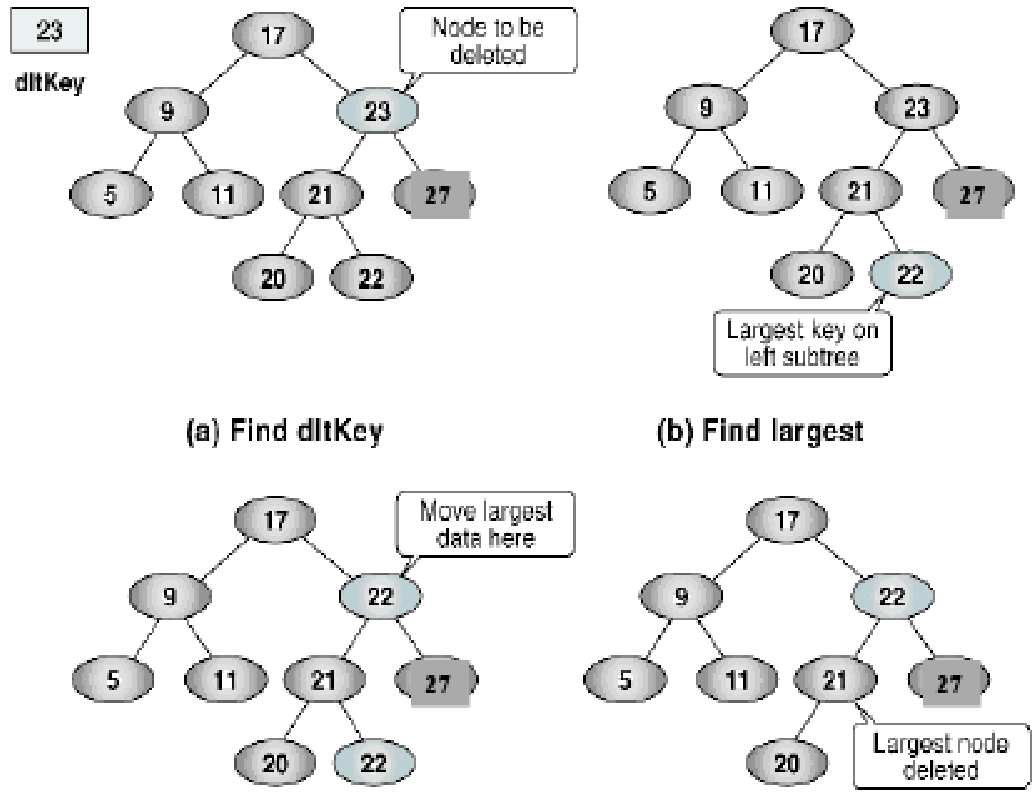
Deletion from the middle of a tree

- Rather than simply delete the node, we try to maintain the existing structure as much as possible by finding data to take the place of the deleted data. This can be done in one of two ways.

Deletion from the middle of a tree

- We can find the largest node in the deleted node’s left subtree and move its data to replace the deleted node’s data.
- We can find the smallest node on the deleted node’s right subtree and move its data to replace the deleted node’s data.
- Either of these moves preserves the integrity of the binary search tree.

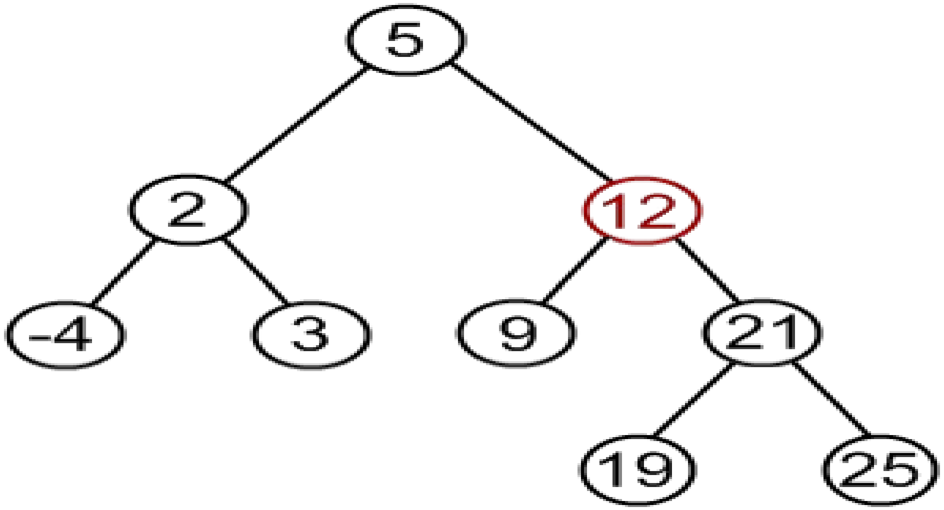
FAR EASTERN UNIVERSITY
East Asia College



(a) Find dltKey (b) Find largest

(c) Move largest data (d) Delete largest node

FAR EASTERN UNIVERSITY
East Asia College



FAR EASTERN UNIVERSITY
East Asia College

-End-