Trees

Kuan-Yu Chen (陳冠宇)

2018/10/08 @ TR-212, NTUST

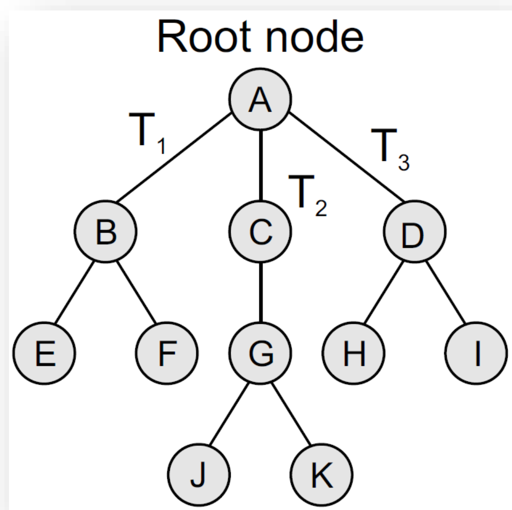
Review

- Queue
  - FIFO
  - Array & Link List
  - Variations
    - Circular Queue
    - Deque
    - Priority Queue
    - Multiple Queue

Trees

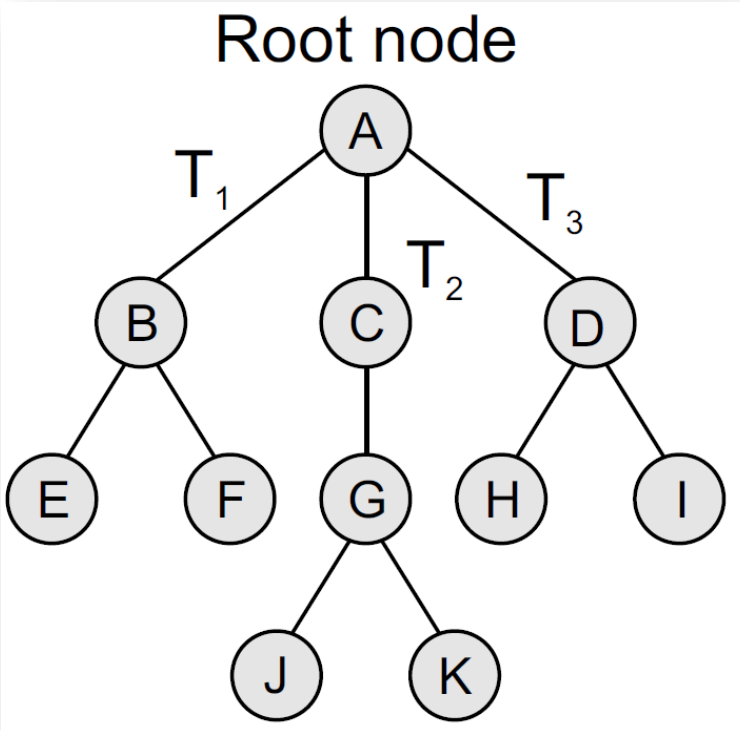
- So far, we have discussed **linear** data structures
  - Arrays
  - Stacks
  - Queues

- A tree is a **non-linear** data structure, which is mainly used to store data that is **hierarchical** in nature
  - The tree is recursively defined as a set of one or more nodes



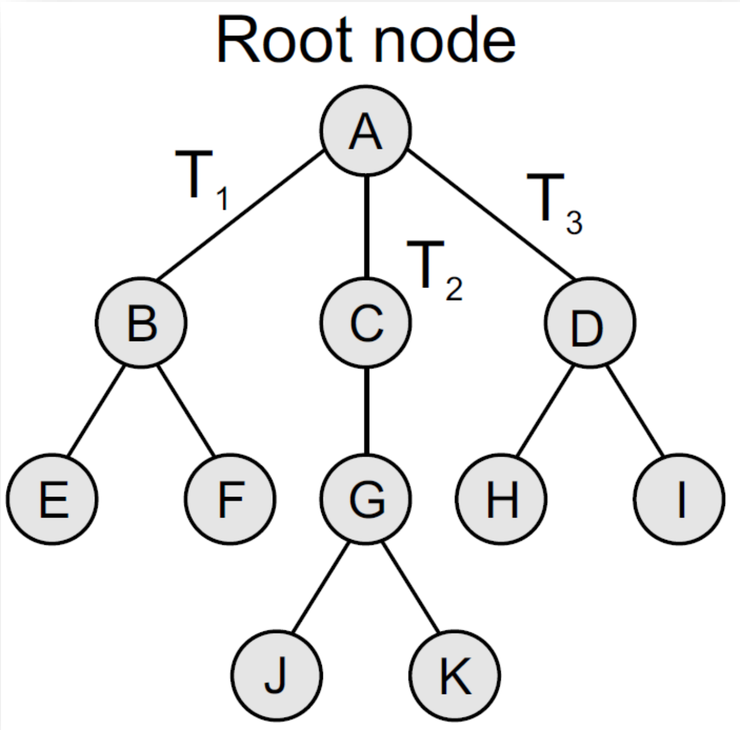
Basic Terminology.

- Root node
  - The root node R is the topmost node in the tree
  - If $R = NULL$, then it means the tree is empty
- Sub-trees
  - The trees T_1, T_2 and T_3 are called the sub-trees of R
- Leaf node
  - A node that has no children is called the leaf node or the terminal node
    - E, F, J, K, H, I
- Path
  - A sequence of consecutive edges is called a path
    - The path from the node A to node I is A, D, I



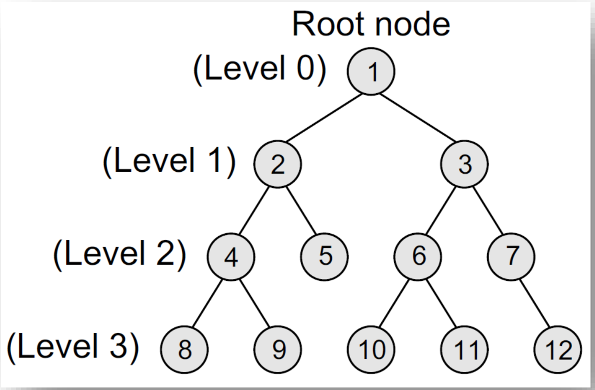
Basic Terminology..

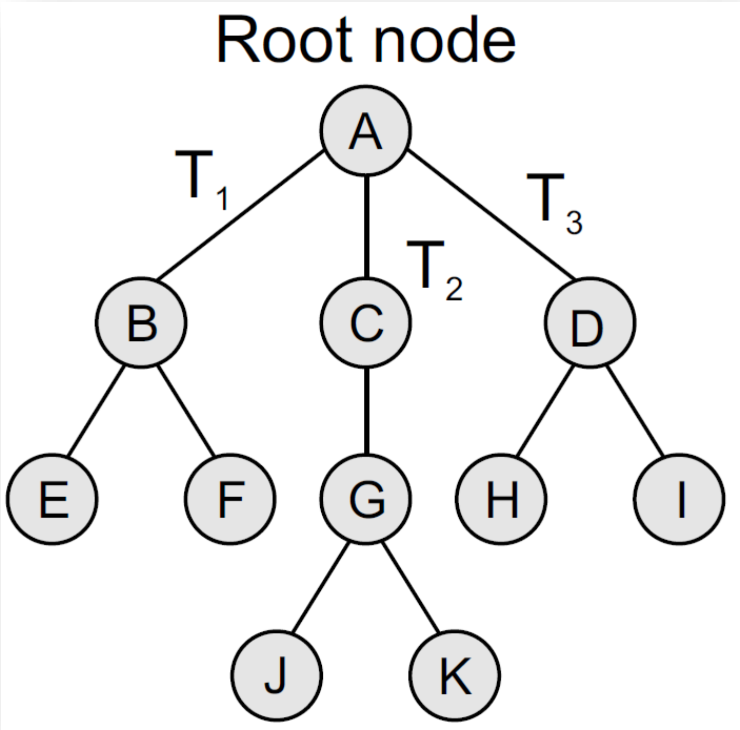
- Ancestor node
  - An ancestor of a node is any predecessor node on the path from root to that node
    - The root node does not have any ancestors
    - Nodes *A*, *C* and *G* are the ancestors of node *K*
- Descendant node
  - A descendant node is any successor node on any path from the node to a leaf node
    - Leaf nodes do not have any descendants
    - Nodes *E* and *F* are the descendants of node *B*



Basic Terminology...

- Level number
  - Every node in the tree is assigned a level number
    - The root node is at level 0
    - Children of the root node (i.e., B, C and D) are at level number 1





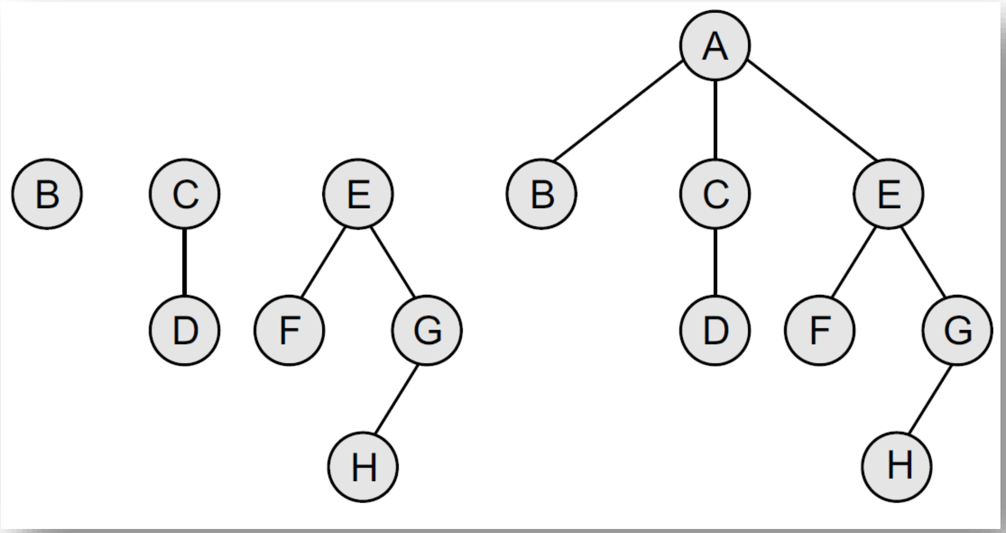
- Degree
  - Degree of a node is equal to the number of children that a node has
    - The degree of J is 0
    - The degree of B is 2

Types of Trees

- Trees can be classified into six classes
  - General Trees
  - Forests
  - Binary Trees
  - Binary Search Trees
  - Expression Trees
  - Tournament Trees

Forest

- A forest is a disjoint union of trees
  - A set of disjoint trees (or forests) is obtained by deleting the root and the edges connecting the root node to nodes at level 1
    - We can convert a forest into a tree by adding a single node as the root node of the tree

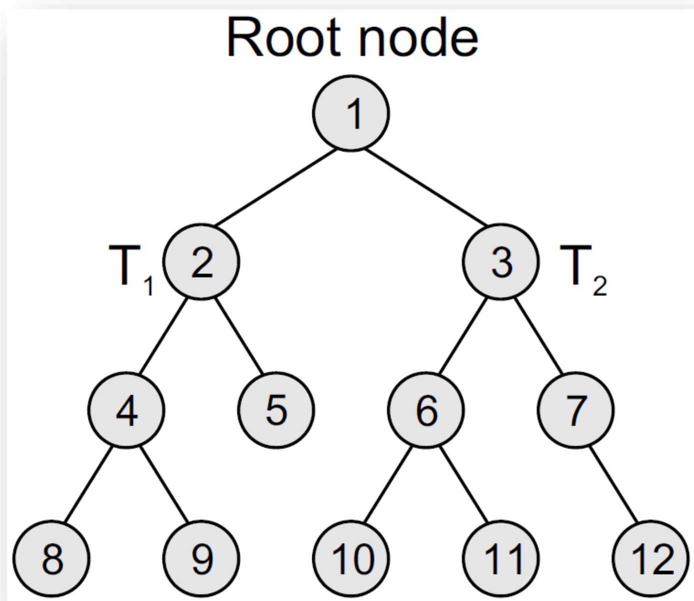


  - While a general tree must have a root, **a forest may be empty** because by definition it is a set, and sets can be empty

Binary Trees.

- A binary tree is a data structure that is defined as a collection of elements called nodes

- In a binary tree, the topmost element is called the root node, and each node has 0, 1, or at the most 2 children

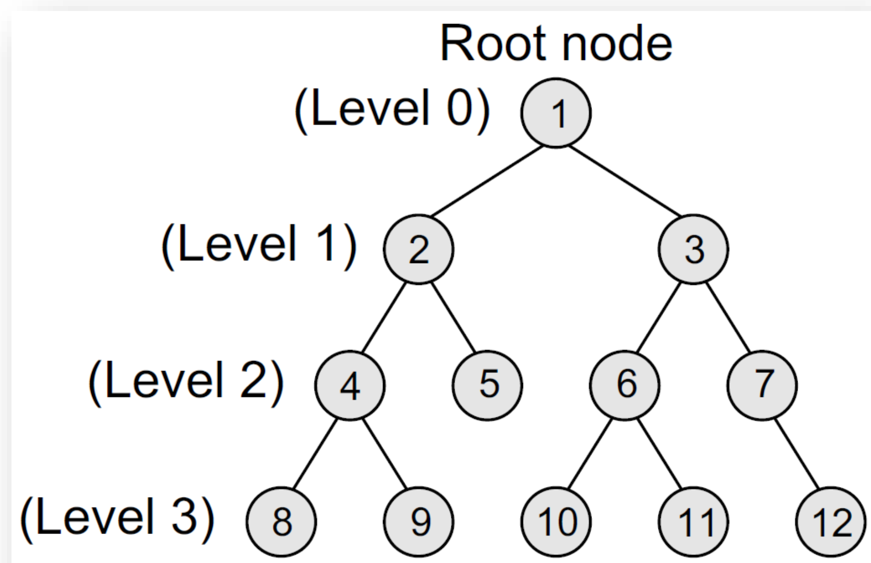


Binary Trees..

- Parent
  - Every node other than the root node has a parent
    - Node 2 is the parent of nodes 4 and 5

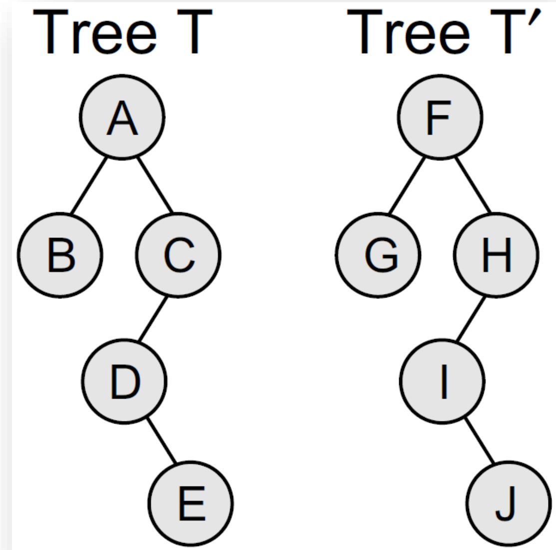
- Sibling
  - All nodes that are at the same level and share the same parent are called ⊠⊠⊠⊠⊠ (brothers)
    - Nodes 4 and 5 are siblings

- Height of a Tree
  - It is the total number of nodes on the path from the root node to the deepest node in the tree
    - A binary tree of height h has at least h nodes and at most $2^h - 1$ nodes

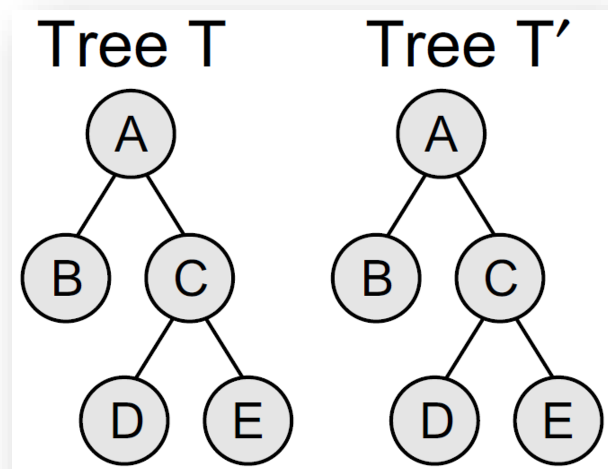


Binary Trees…

- Similar Binary Trees
  - Two binary trees T and T' are said to be similar if both these trees have the **same structure**

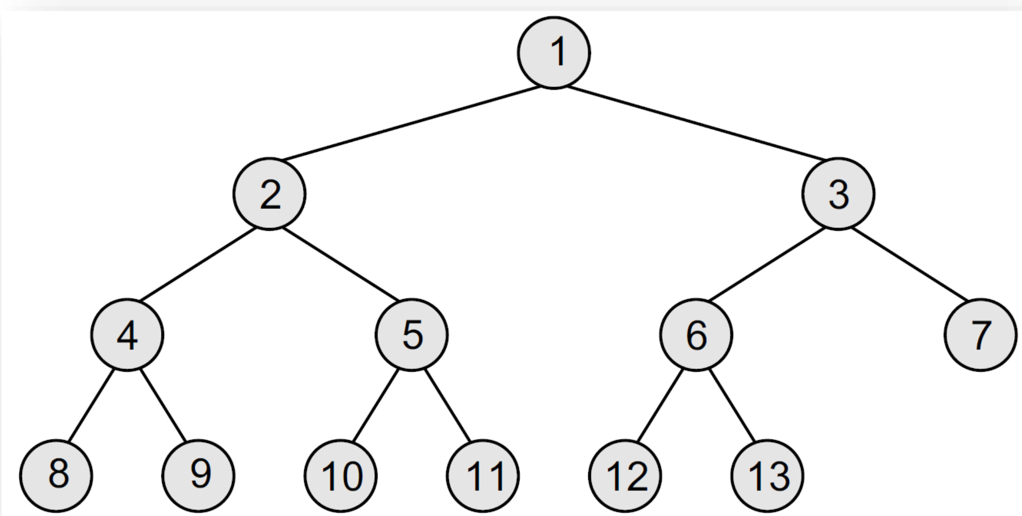


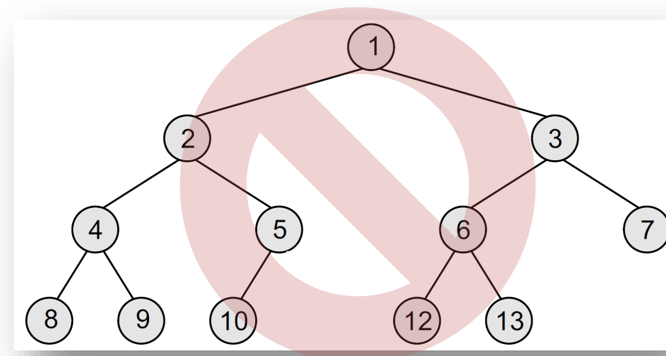
- Copies
  - Two binary trees T and T' are said to be copies if they have **similar structure** and if they have **same content** at the corresponding nodes

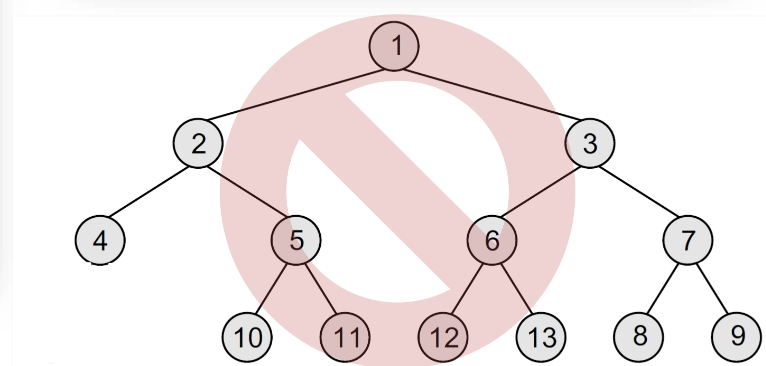


Binary Trees.... – Complete Binary Tree

- Complete Binary Trees
  - A complete binary tree is a binary tree that satisfies two properties
    1. Every level, except possibly the last, is completely filled
    2. All nodes appear as far left as possible

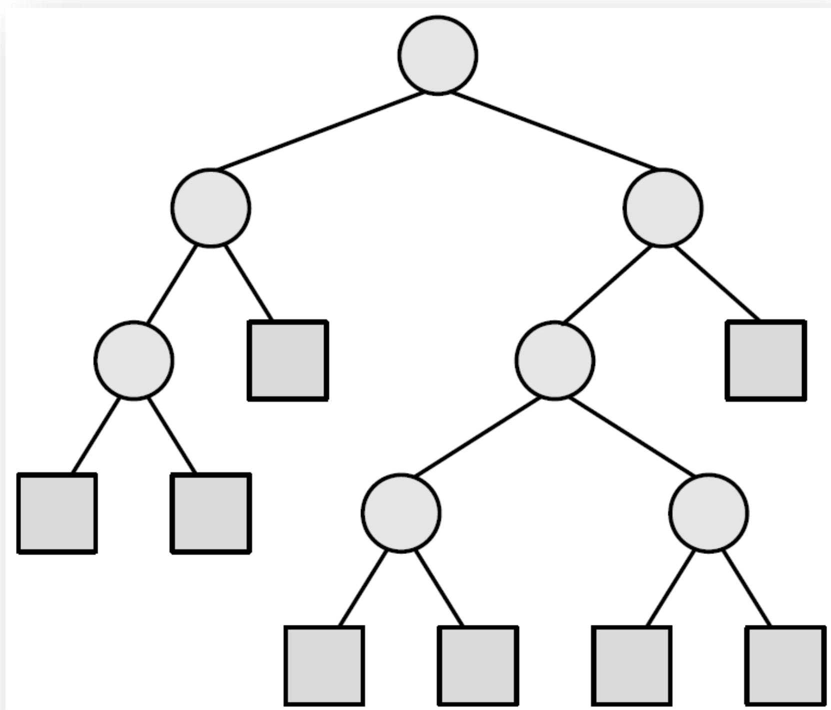






Binary Trees…. – Extended Binary Tree

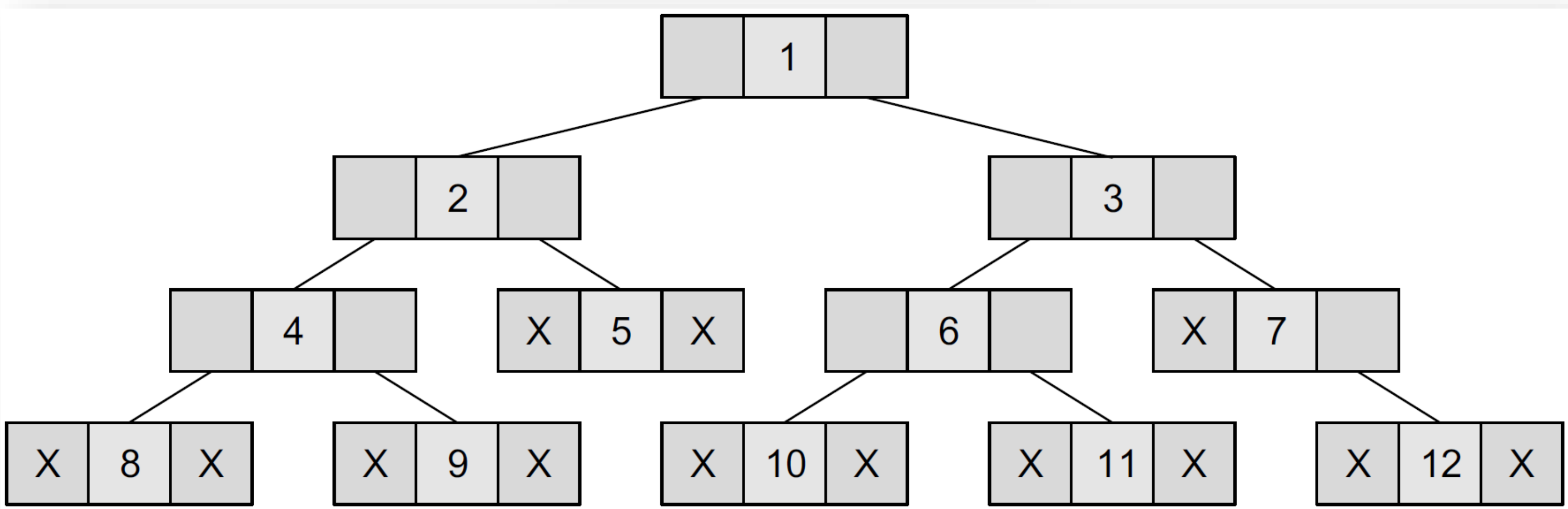
- A binary tree is said to be an extended binary tree (or a 2-tree) if each node in the tree has **either no child** or **exactly two children**
  - Nodes having two children are called **internal nodes** and nodes having no children are called **external nodes**



Binary Trees….. – LinkList Implementation

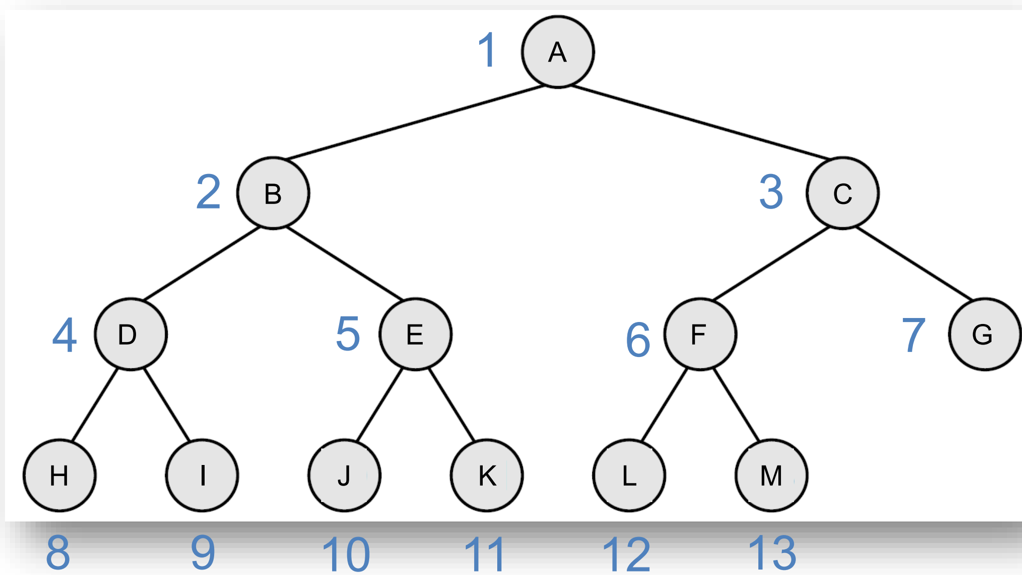
- In the linked representation of a binary tree, every node will have three parts: the data element, a pointer to the left node, and a pointer to the right node

```
struct node {
        struct node *left;
        int data;
        struct node *right;
};
```



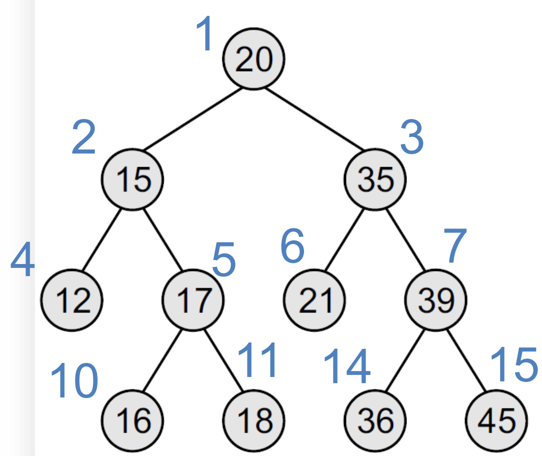
Binary Trees….. – Array Implementation.

- For a binary tree, we can number all of the nodes ordered
  - If K is a parent node, then its left child can be calculated as $2 \times K$ and its right child can be calculated as $2 \times K + 1$
    - The children of the node 4 are 8 and 9
  - The parent of the node K can be calculated as $\left\lfloor \frac{K}{2} \right\rfloor$
    - The parent of the node 5 is 2



Binary Trees….. – Array Implementation..

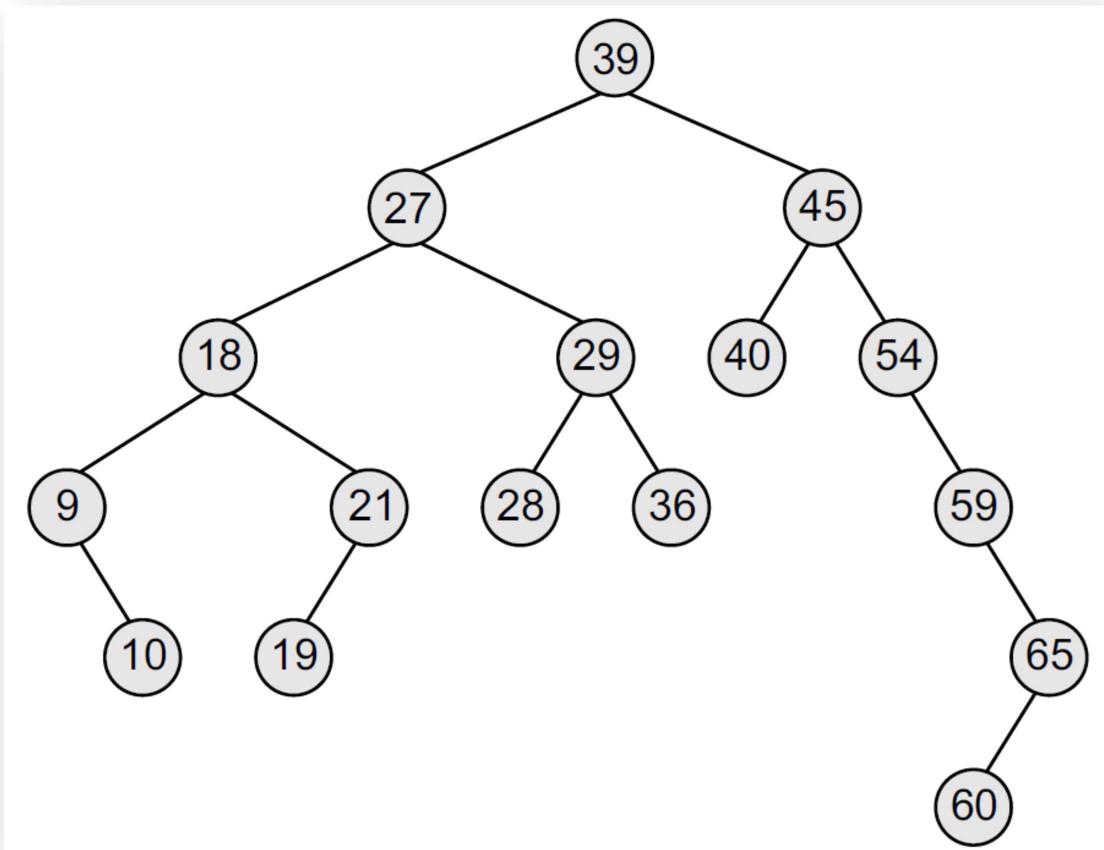
- Sequential representation of trees is done using single or one-dimensional arrays
  - Though it is the simplest technique for memory representation, it is inefficient as it requires a lot of memory space



1	20
2	15
3	35
4	12
5	17
6	21
7	39
8	
9	
10	16
11	18
12	
13	
14	36
15	45

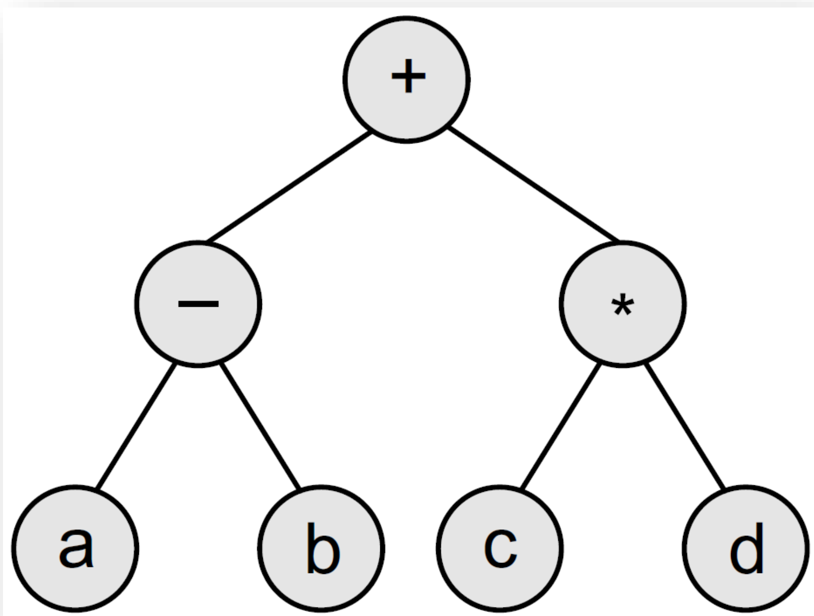
Binary Search Trees

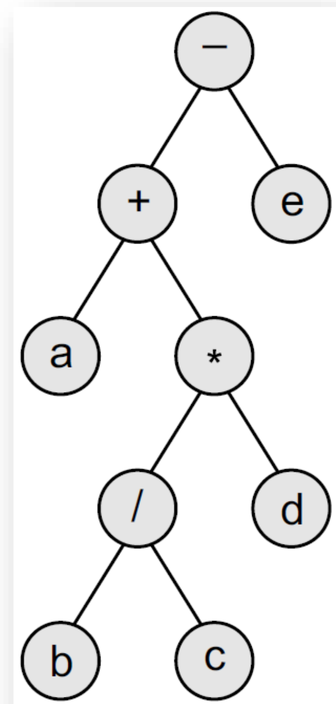
- A binary search tree, also known as an ordered binary tree, is a variant of binary tree in which the nodes are arranged in an order



Expression Trees

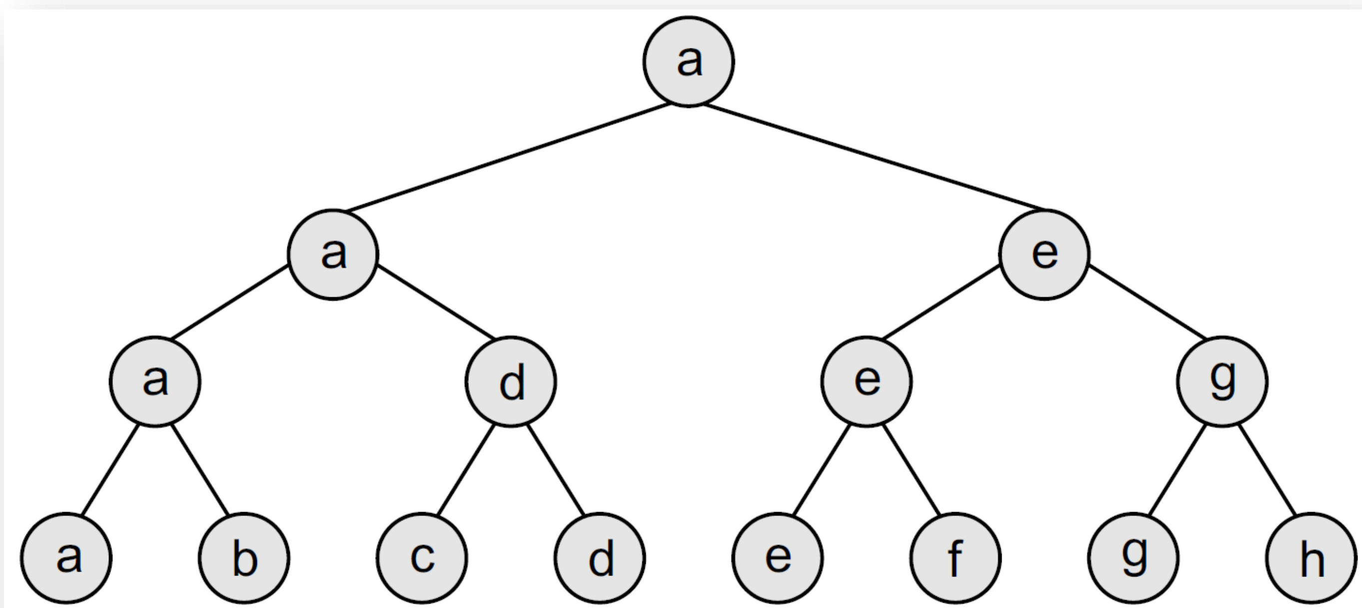
- Binary trees are widely used to store algebraic expressions
  - Given an algebraic expression $(a - b) + (c \times d)$
  - Given an expression $a + b \div c \times d - e$





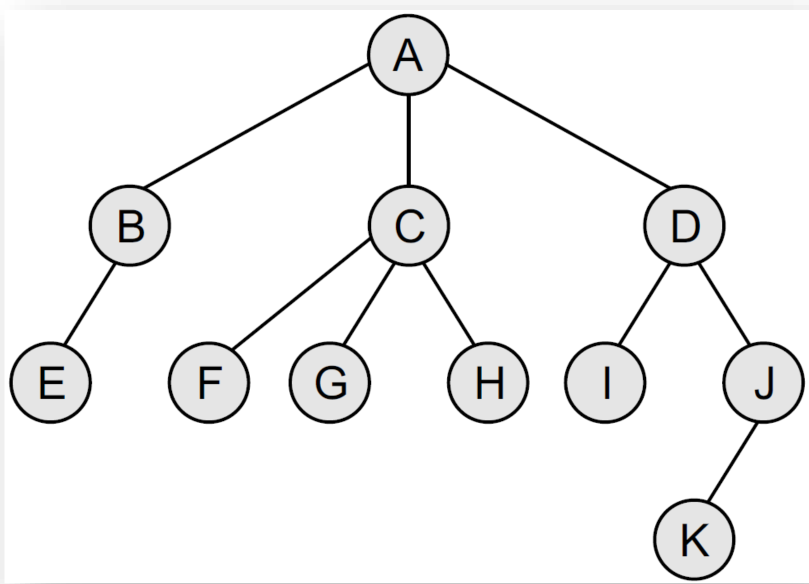
Tournament Trees

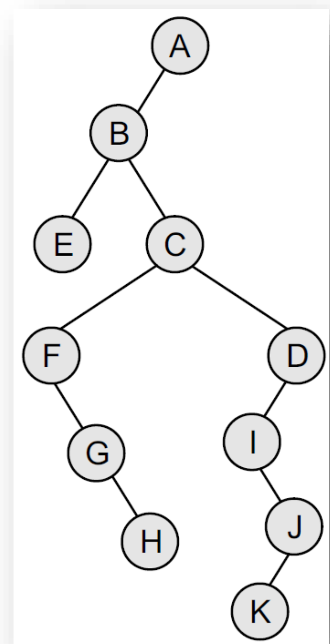
- In a tournament tree (also called a **selection tree**), each external node represents a candidate and each internal node represents the selected candidate by its children nodes
  - These tournament trees are also called **winner trees** because they are being used to record the winner at each level
    - We can also have a **loser tree** that records the loser at each level



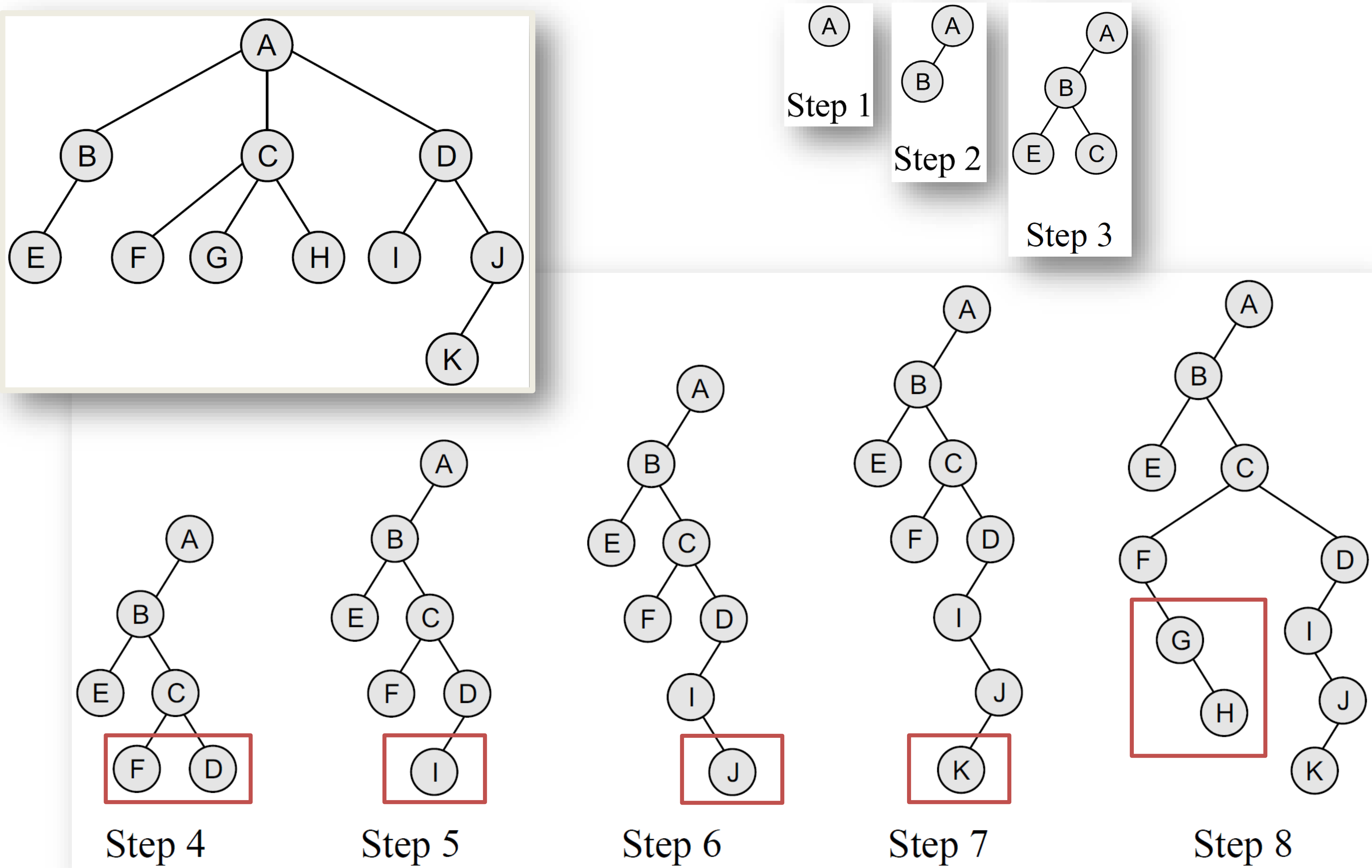
From a General Tree to a Binary Tree.

- The rules for converting a general tree to a binary tree are given below
  - **Rule 1**: Root of the binary tree = Root of the general tree
  - **Rule 2**: Left child of a node in the binary tree = Leftmost child of the node in the general tree
  - **Rule 3**: Right child of a node in the binary tree = Right sibling of the node in the general tree



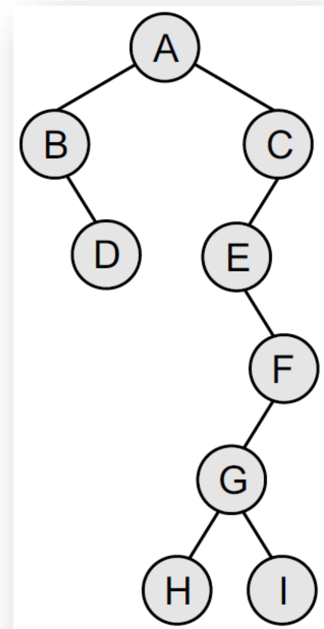


From a General Tree to a Binary Tree..



Traversing Binary Tree

- Traversing a binary tree is the process of visiting each node in the tree exactly once in a systematic way
  - There are different algorithms for tree traversals
    - Pre-order Traversal
      - *ABDCEFGHI*
    - Post-order Traversal
      - *DBHIGFECA*
    - In-order Traversal
      - *BDAEHGIFC*
    - Level-order Traversal
      - *ABCDEFGHI*

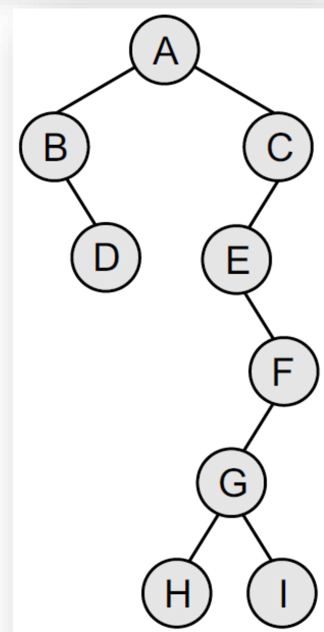


  - Different algorithms differ in the order in which the nodes are visited

In-order

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:          INORDER(TREE -> LEFT)
Step 3:          Write TREE -> DATA
Step 4:          INORDER(TREE -> RIGHT)
       [END OF LOOP]
Step 5: END
```

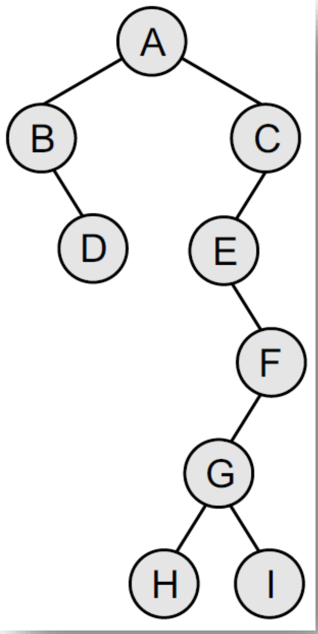
- In-order: BDAEHGIFC



Pre-order

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:             Write TREE –> DATA
Step 3:             PREORDER(TREE –> LEFT)
Step 4:             PREORDER(TREE –> RIGHT)
        [END OF LOOP]
Step 5: END
```

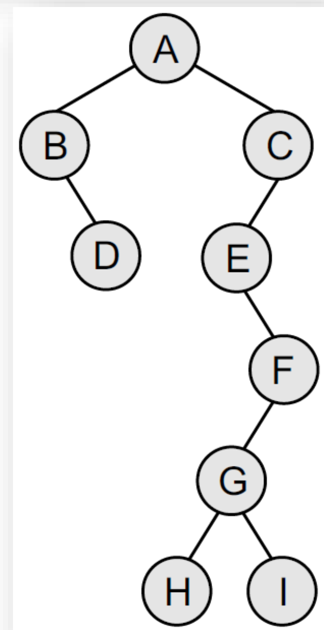
- Pre-order: ABDCEFGHI



Post-order

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:          POSTORDER(TREE -> LEFT)
Step 3:          POSTORDER(TREE -> RIGHT)
Step 4:          Write TREE -> DATA
       [END OF LOOP]
Step 5: END
```

- Post-order: DBHIGFECA



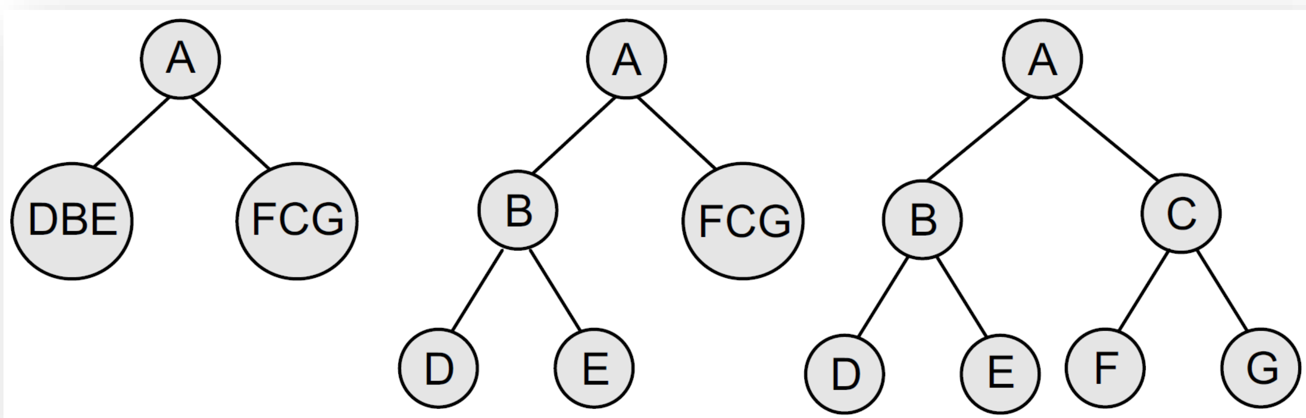
Constructing Binary Tree from Traversal.

- We can construct a binary tree if we are given at least two traversal results
  - In-order traversal
    - The in-order traversal result will be used to determine the left and the right child nodes
  - Either pre-order or post-order traversal
    - The pre-order/post-order can be used to determine the root node

Constructing Binary Tree from Traversal..

- Take in-order + pre-order for example
  - In-order: $D\ B\ E\ A\ F\ C\ G$
  - Pre-order: $A\ B\ D\ E\ C\ F\ G$

$D\ B\ E\ A\ F\ C\ G$	$D\ B\ E\ A\ F\ C\ G$	$D\ B\ E\ A\ F\ C\ G$
$A\ B\ D\ E\ C\ F\ G$	$A\ B\ D\ E\ C\ F\ G$	$A\ B\ D\ E\ C\ F\ G$



Constructing Binary Tree from Traversal…

- Take in-order + post-order for example
  - In-order: $D\ B\ H\ E\ I\ A\ F\ J\ C\ G$
  - Post-order: $D\ H\ I\ E\ B\ J\ F\ G\ C\ A$

$D\ B\ H\ E\ I\ A\ F\ J\ C\ G$

$D\ H\ I\ E\ B\ J\ F\ G\ C\ A$

$D\ B\ H\ E\ I\ A\ F\ J\ C\ G$

$D\ H\ I\ E\ B\ J\ F\ G\ C\ A$

$D\ B\ H\ E\ I\ A\ F\ J\ C\ G$

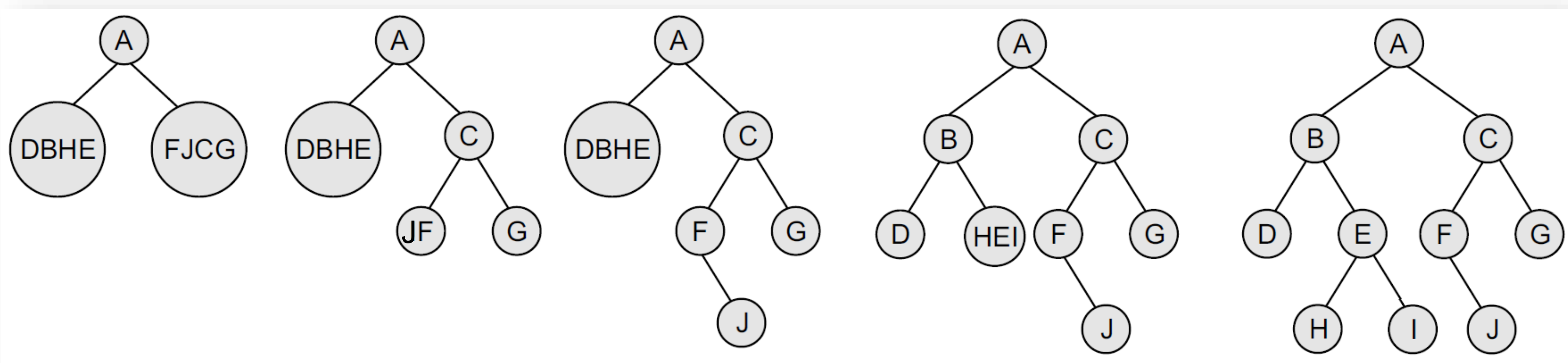
$D\ H\ I\ E\ B\ J\ F\ G\ C\ A$

$D\ B\ H\ E\ I\ A\ F\ J\ C\ G$

$D\ H\ I\ E\ B\ J\ F\ G\ C\ A$

$D\ B\ H\ E\ I\ A\ F\ J\ C\ G$

$D\ H\ I\ E\ B\ J\ F\ G\ C\ A$



Constructing Binary Tree from Traversal….

- Steps for constructing a binary tree from traversal sequences
    1. Use the pre-order/post-order sequence to determine the root node of the tree
    2. Elements on the left side of the root node in the in-order traversal sequence form the left sub-tree of the root node
    3. Similarly, elements on the right side of the root node in the in-order traversal sequence form the right sub-tree of the root node
    4. Recursively select each element from pre-order/post-order traversal sequence and create its left and right sub-trees from the in-order traversal sequence

Questions?

kychen@mail.ntust.edu.tw