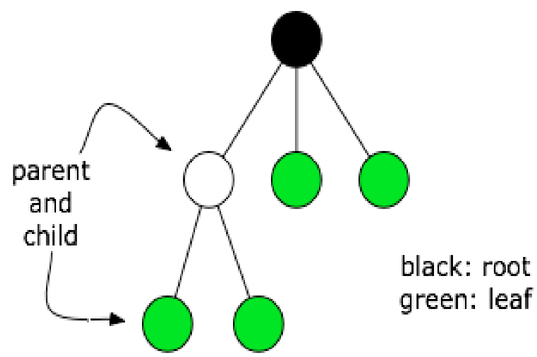
Data Structure

Lecture 7: The Tree

Asst. Instructor

Ali A. Al-Ani



UNIVERSITY OF DIYALA

Tree

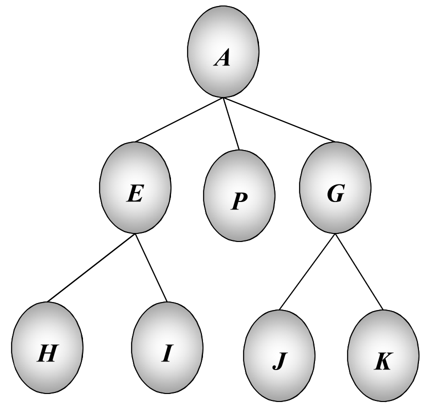
- *Linked lists usually provide greater flexibility than arrays, but they are linear structures and it is difficult to use them to organize a hierarchical representation of objects.*
- *Although stacks and queues reflect some hierarchy, they are limited to only one dimension. To overcome this limitation, we create a new data type called a* ***Tree****.*
- ***A Tree is a nonlinear data structure that models a hierarchical organization.***
- *Trees are common in computer science such as* ***Computer file systems are trees, prioritizing jobs, describing mathematical expressions and the syntactic elements of computer programs.***

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Tree

- *The characteristic features of the tree are that each element may have several successors (called its "**children**") and every element except one (called the "**root**") has a unique predecessor (called its "**parent**").*
- *A tree is usually visualized by placing elements inside ovals or rectangles, and by drawing the connections between parents and children with straight lines.*



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Tree

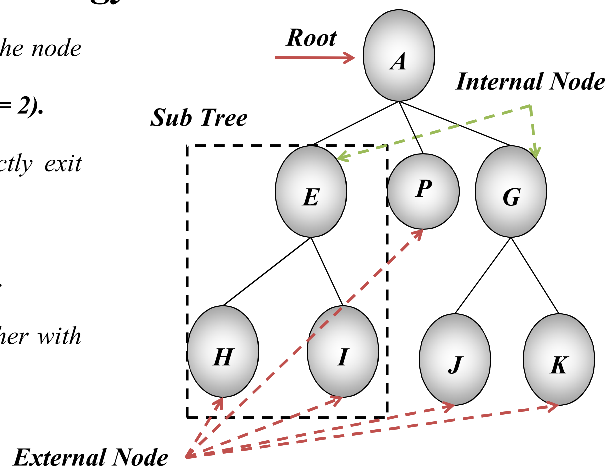
- *Formally, we define tree **T** to be a set of nodes storing elements in a parent-child relationship with the following properties:*
- *If **T** is nonempty, it has a special node, called the root of **T**, that has no parent.*
- *Each node **v** of **T** different from the root has a unique parent node **w**; every node with parent **w** is a child of **w**.*
- ***Note** that according to the definition, a tree can be **empty**, meaning that it doesn't have any nodes.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Tree: Terminology

1. ***Node Level***: *Is the number of the tracks between the node and the root of the tree.* ***Ex: (The level of node H = 2).***
2. ***Node Degree***: *Is the number of tracks that directly exit from the node* ***.Ex: (The degree of node E = 2).***
3. ***Tree Degree:*** *Is the highest node degree in the tree.*
4. ***A sub tree*** *in a tree is any node in the tree together with all of its descendants.*



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree

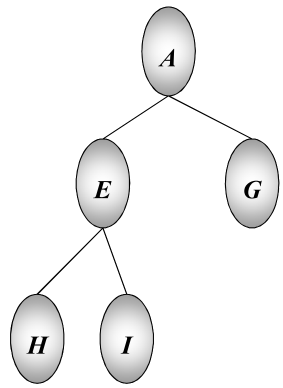
- *A tree where each node has a specific number of children appearing in a specific order is call a* ***multiway tree****. The simplest type of a multiway tree is the* ***binary tree****. Each node in a binary tree has exactly two children one of which is designated as a left child, and the other is designated as a right child. Then*
  1. ***Every node has at most two children.***
  2. ***Each child node is labeled as being either a left child or a right child.***
  3. ***A left child precedes a right child in the ordering of children of a node.***

Department of Computer Science
College of Science

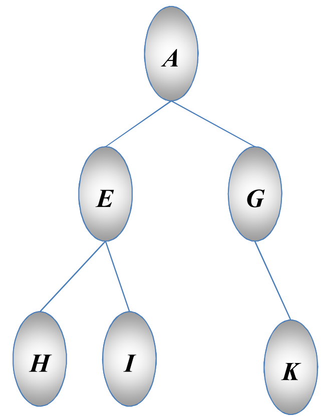
UNIVERSITY OF DIYALA

Binary Tree: Types

- *A binary tree is **proper** if each node has either **zero or two children**. Some people also refer to such trees as being **full binary trees**. Thus, in a **proper binary tree, every internal node has exactly two children**. A binary tree that is not proper is **improper**.*



Proper BT



Improper BT

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

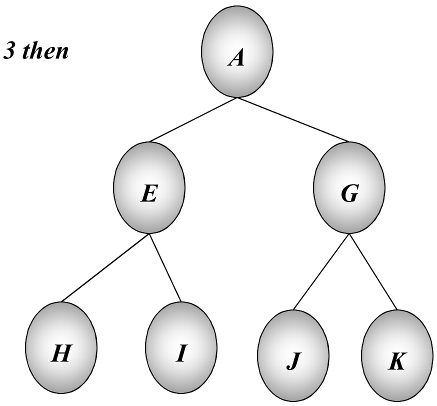
Binary Tree: Properties

- *Binary trees have several interesting properties dealing with relationships between their **heights** and **number of nodes**. In a binary tree, **level 0 has one node (root), level 1 has, two nodes** (the children of the root), **level 2 has, at most, four nodes, and so on.***
- ***In general**, level **L** has, at most, **(2^L)** nodes. We can see that the maximum number of nodes on the levels of a binary tree grows exponentially as we go down the tree. From this simple observation, we can derive the following properties relating the **height of a binary T** to its number of nodes.*
- ***No. of Node in the Binary Tree (T) = (2^(h+1) - 1)***
- *where : **h:** height of a binary Tree (T)*

Department of Computer Science
College of Science

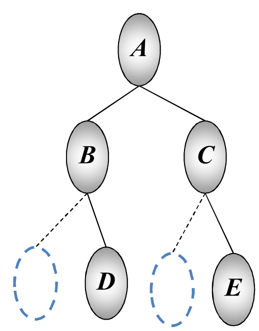
Binary Tree: Properties

- ***In the following figure the height of a binary Tree (T) is 3 then***
  - ***No. of Node in the Binary Tree (T) = ($2^{(h+1)}$ - 1)***
  - *No. of Node in the Binary Tree (T) = ($2^{(2+1)}$ - 1)*
  - *No. of Node in the Binary Tree (T) = 7*



Binary Tree: Representation

1. ***Linear representation of a binary tree utilizes one-dimensional*** *array of size* $\boldsymbol{2^{h+1}-1}$. *Consider the following tree.*



- *To represent this tree, we need an array of size* $\boldsymbol{2^{2+1}-1=7}$*, The tree is represented as follows* ***A[7]***.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
A	B	C	-	D	-	E

UNIVERSITY OF DIYALA

Binary Tree: Representation

- ***Advantages of linear representation:***
  1. *Simplicity.*
  2. *Given the location of the child (say, k), the location of the parent is easy to determine (k / 2).*
- ***Disadvantages of linear representation:***
  1. *Additions and deletions of nodes are inefficient, because of the data movements in the array.*
  2. *Space is wasted if the binary tree is not complete.*
  3. ***Note** that linear representation of a binary tree can be implemented by means of a linked list instead of an array. This way the above mentioned disadvantages of the linear representation will be resolved.*

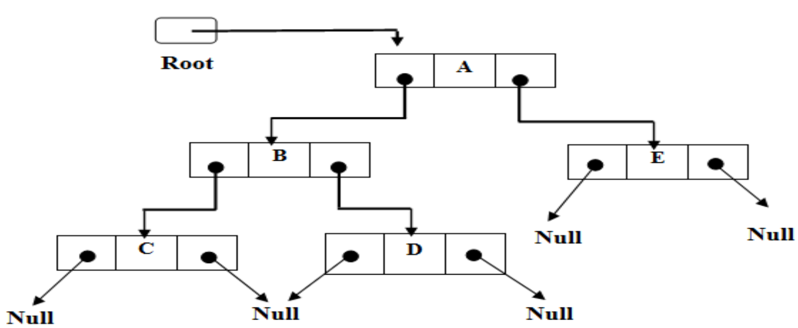
Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree: Representation

2. ***we will present an implementation of a binary tree T as a linked list by using the record structure in two ways .***

1. ***Using two pointers** one for the **left child** and one for the **right child.***

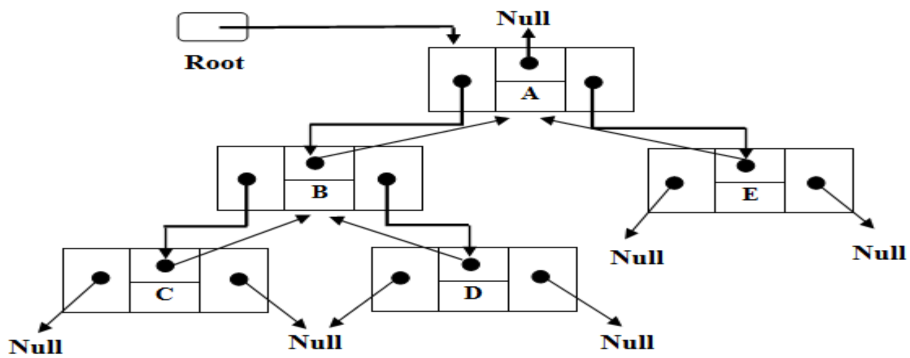


Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree: Representation

2. ***Using three pointers*** *one for the parent and two pointers one for the* ***left child*** *and one for the* ***right child***



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree: Traversals

- *Often we wish to process a binary tree by "****visiting****" each of its nodes, each time performing a specific action such as printing the contents of the node.*
- *Any process for visiting all of the nodes in some order is called a* ***Traversal****.*
- *Some applications do not require that the nodes be visited in any particular order as long as each node is visited precisely once.*
- *For other applications, nodes must be visited in an order that preserves some relationship. We will discuss three* ***different binary-tree traversals methods****.*

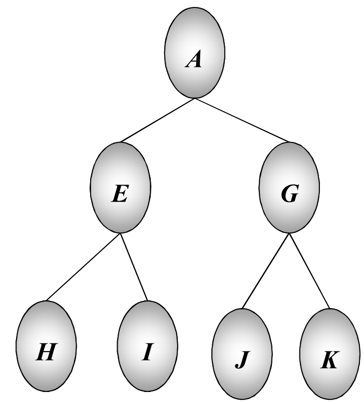
Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree: Traversals

1. ***Preorder Traversal of a Binary Tree***

- *In this traversals we visit any given node before we visit its children This is called a **Preorder Traversal**. For example, The preorder enumeration for the following tree is:*
- ***A E H I G J K***
- *The first node printed is the root. Then all nodes of the left sub tree are printed (in preorder) before any node of the right sub tree.*

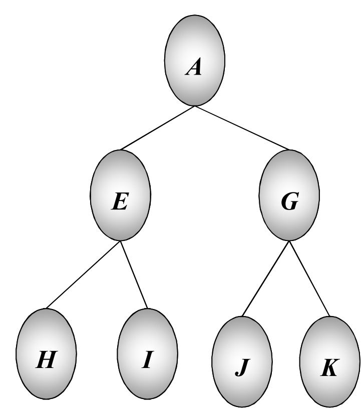


UNIVERSITY OF DIYALA

Binary Tree: Traversals

2. ***Postorder Traversal of a Binary Tree***

- *In this traversals we visit each node only after we visit its children (and their sub trees). The **Postorder** enumeration for the following tree is:* ***H I E J K G A***
- *The Postorder would be necessary if we wish to return all nodes in the tree to **free store**. We would like to delete the children of a node before deleting the node itself. But to do that requires that the children's be deleted first, and so on.*



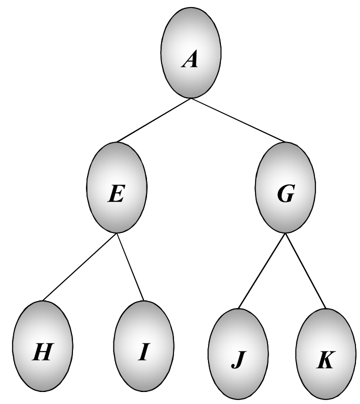
Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree: Traversals

3. ***Inorder Traversal of a Binary Tree***

- *In this traversals we first visits the left child (including its entire sub tree), then visits the node, and finally visits the right child (including its entire sub tree). This is called **a Inorder Traversal**. For example, The **Inorder** enumeration for the following tree is:*
- ***H E I A J G K***



UNIVERSITY OF DIYALA

Binary Tree: Applications

- ***There are a numbers of applications of Binary Tree such as:***

1. ***Arithmetic Expressions Representation .***
2. ***Decision Processes.***
3. ***Binary Search tree.***

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Binary Tree: Applications

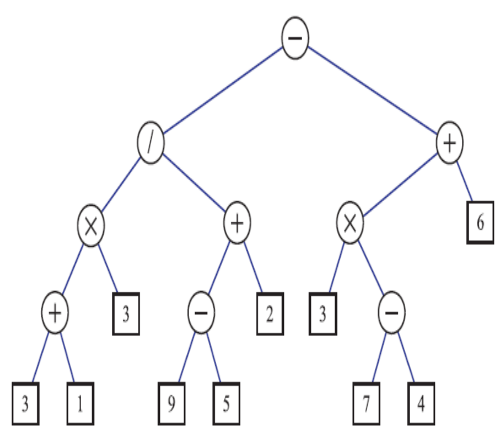
1. ***Representation Arithmetic Expressions:***

- *The Postorder traversal of a binary tree can be used to solve the expression evaluation problem. In this problem, we are given an arithmetic-expression tree, that is, a binary tree where :*
  1. ***If a node is external, then its value is that of its variable or constant.***
  2. ***If a node is internal, then its value is defined by applying its operation to the values of its children.***
- *An arithmetic expression can be represented by a tree whose external nodes are associated with* ***variables*** *or* ***constants****, and whose internal nodes are associated with one of the* ***operators .***

UNIVERSITY OF DIYALA

Binary Tree: Applications

- *The arithmetic-expression tree is a proper binary tree, since each of the operators* **+, −, ×,** *and / take exactly two operands. Of course, if we were to allow for unary operators, like negation* ***(−)****, as in* ***"−x,"*** *then we could have an* ***improper*** *binary tree.*
- *For Example: The following binary tree representing an arithmetic expression. This tree represents the expression* ***(((3+1)×3)/((9−5)+2))−((3×(7−4))+6)).***



Department of Computer Science
College of Science

Binary Tree: Applications

- *From traversal the above binary tree we will recognize the three forms of expressions:*
  1. ***Preorder Traversal** of the expression tree yields **prefix notation**;*
  2. ***Postorder Traversal** generates **postfix notation**;   **31+3*95-2+/374-*6+-***
  3. ***Inorder Traversal** yields conventional **infix notation**,   **3+1×3/9−5+2−3×7−4+6.***
- *Although without the parentheses necessary to clarify operator precedences.*
- ***H.W.: Draw the binary tree representation of the following arithmetic expressions:***
  1. ***(((5+2) ∗(2−1))/((2+9)+((7−2)−1)) ∗8),***
  2. ***R*T^2-M*(L-E/F),***
  3. ***a/b+c/d^2*(e-f) and***
  4. ***A:=b*c+(3-d*e)/f^4***

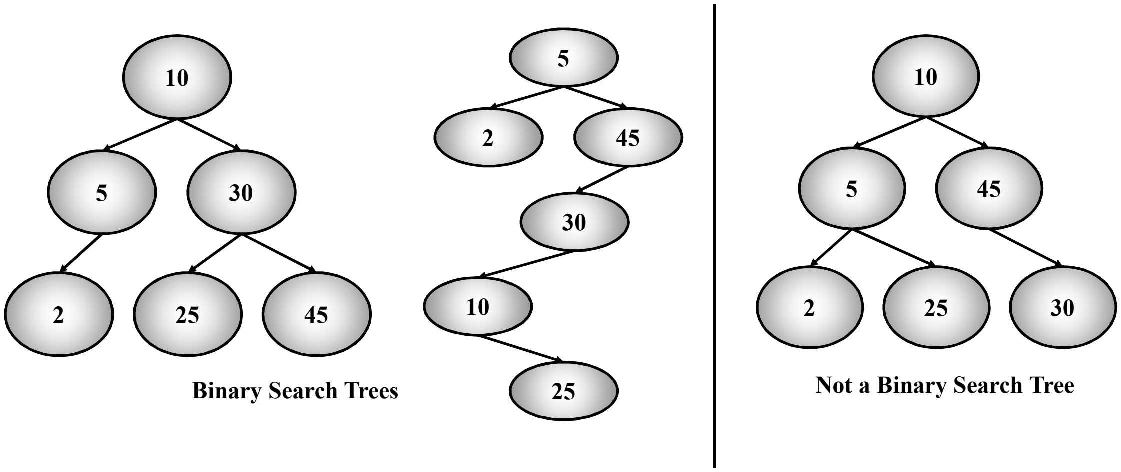
UNIVERSITY OF DIYALA

Binary Tree: Applications

2. ***Binary Search tree:** Binary Search Tree (BST) is a tree in which all the nodes follow the below mentioned properties: Let us assume that each node in the tree is assigned a key value, and assume that this is an integer.*
   1. ***The left sub-tree of a node has a key less than or equal to its parent node's key.***
   2. ***The right sub-tree of a node has a key greater than to its parent node's key.***

Department of Computer Science
College of Science

BT: Binary Search Tree



UNIVERSITY OF DIYALA

BT: Binary Search Tree

- *We can use a binary search tree **T** to locate an element with a certain value **x** by traversing down the tree **T**.*

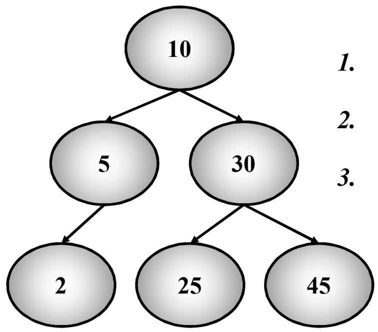
  1. *At each internal node we compare the value of the current node to our search element **x**.*
  2. *If the answer to the question is "smaller," then the search continues in the **left sub tree**.*
  3. *If the answer is "equal," then the search terminates **successfully**.*
  4. *If the answer is "greater," then the search continues in the **right sub tree**.*
  5. *Finally, if we reach an external node (which is empty), then the search terminates **unsuccessfully**.*

Department of Computer Science
College of Science

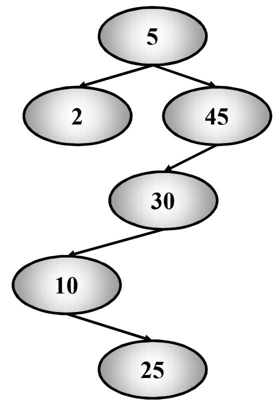
UNIVERSITY OF DIYALA

BT: Binary Search Tree

- *For Example: Find (root, 2)*



1. *10 > 2, left*
2. *5 > 2, left*
3. *2 = 2, found*

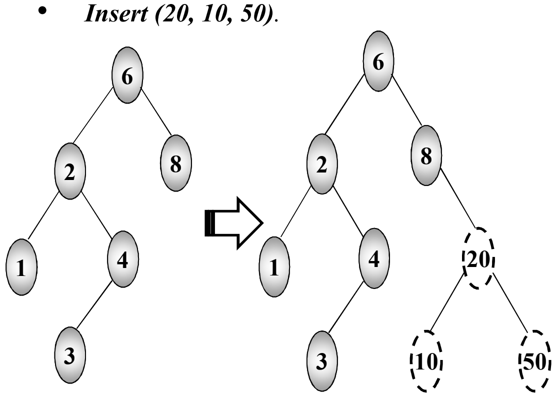


1. *5 > 2, left*
2. *2 = 2, found*

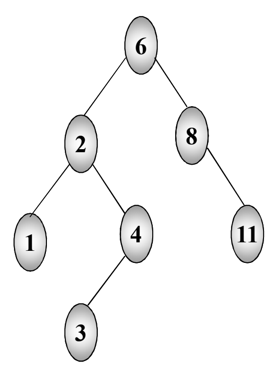
UNIVERSITY OF DIYALA

BST: Insert

- *Insert (20, 10, 50).*



- *Let's insert 6, 2, 4, 3, 1, 8 and 11 in an empty BST.*



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

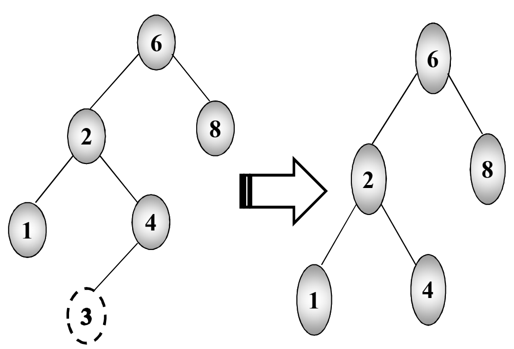
BST: Delete

- *First, find the item; then, delete it, but the Binary search tree property must be preserved!! We need to consider three different cases:*
    1. *Deleting a leaf :* ***Delete it immediately***
    2. *Deleting a node with only one child:* ***Adjust a pointer from the parent to bypass that node***
    1. *Deleting a node with two children:* ***replace the value of that node with the minimum element at the right sub tree. delete the minimum element has either no child or one child. So invoke case 1 or 2.***

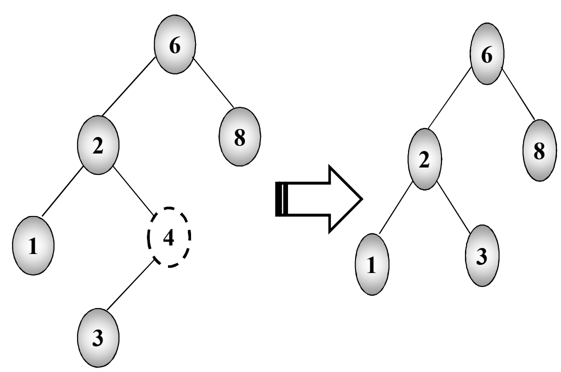
UNIVERSITY OF DIYALA

BST: Delete

1. *Deleting a leaf:* ***Delete node 3***



2. *Deleting a node with only one child :* ***Delete node 4***

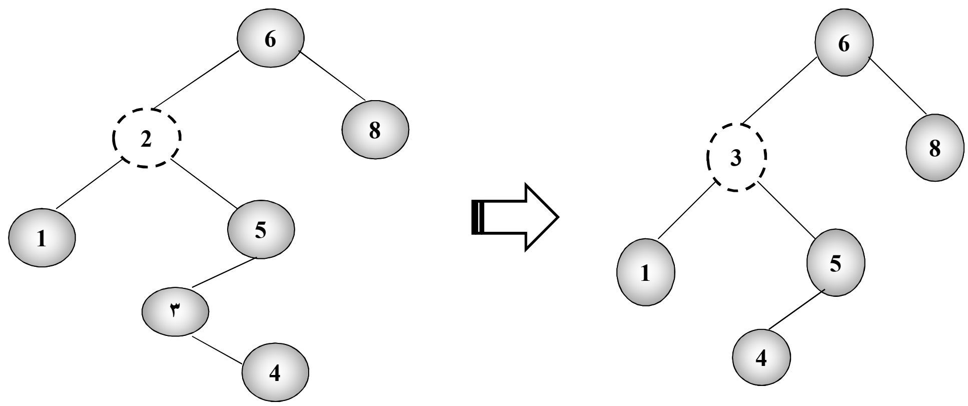


Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

BST: Delete

3. *Deleting a node with two children :* ***Delete node 2***



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

H.W.

1. ***Explain and write an algorithm to transfer the General Tree into Binary Tree.***
2. ***Explain how represent the Tree Structure by using the venn diagrams and nested parenthesis Mathematical concepts. give an example***
3. ***There are many other tree structure types, such as (AVL Tree, Spanning Tree, Heap, B-Tree) give an example for each.***

Department of Computer Science
College of Science

The End