جامعة بيرزيت
BIRZEIT UNIVERSITY

Binary Trees, Expression Trees, and Binary Search Trees

Abdallah Karakra

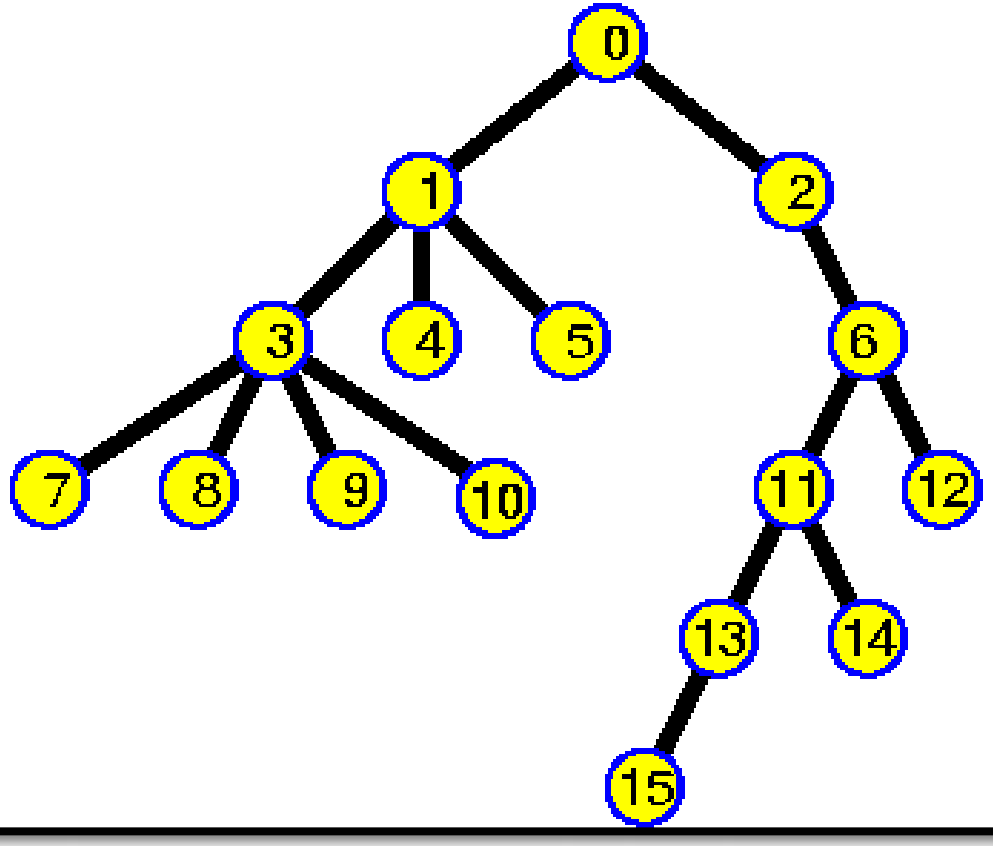
Computer Science Department

COMP242

Tuesday, April 16, 2024

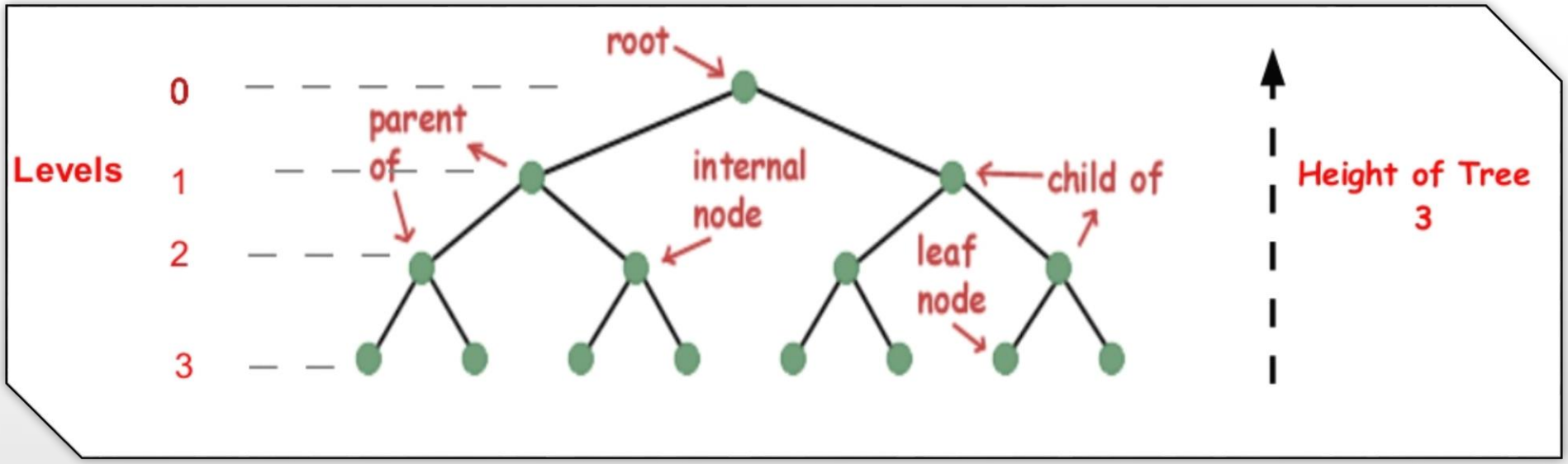
Recall

Tree Definitions



Tree has 16 nodes
Tree has degree 4
Tree has depth 5
Node 0 is the root
Node 1 is internal
Node 4 is a leaf
4 is a child of 1
1 is the parent of 4
0 is grandparent of 4
3, 4 and 5 are siblings

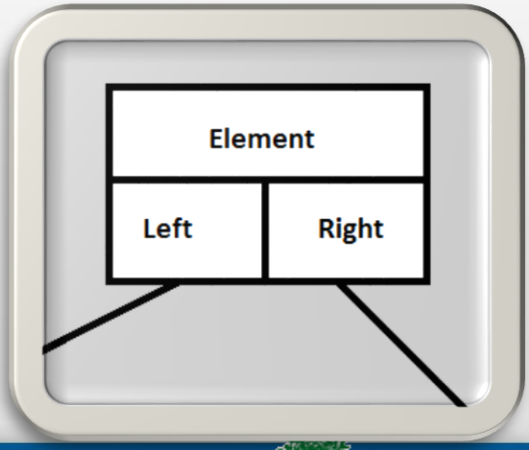
Recall


root
0
parent
of
Levels
1
internal
node
child of
2
leaf
node
3
Height of Tree
3

Binary Trees

- A binary tree is a tree in which no node can have more than two children(every node has at most 2 children).

- Each node has an element, a reference to a left child and a reference to a right child.



Full and Complete Binary Tree

There are two forms of binary tree:

1. **Full binary tree**: a binary tree T is full if each node is either **a leaf** or possesses **exactly two child nodes**.

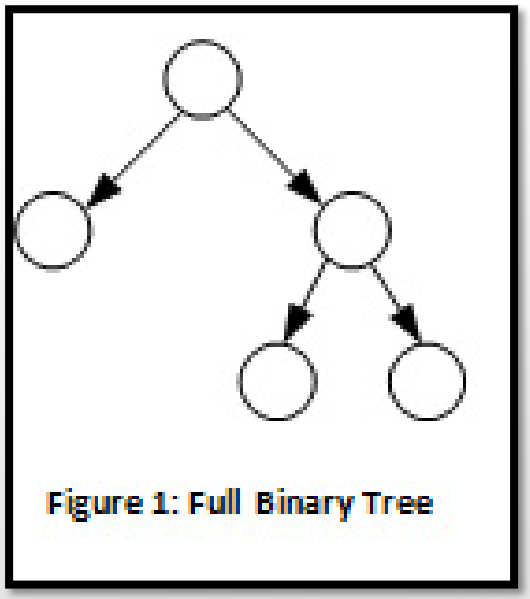
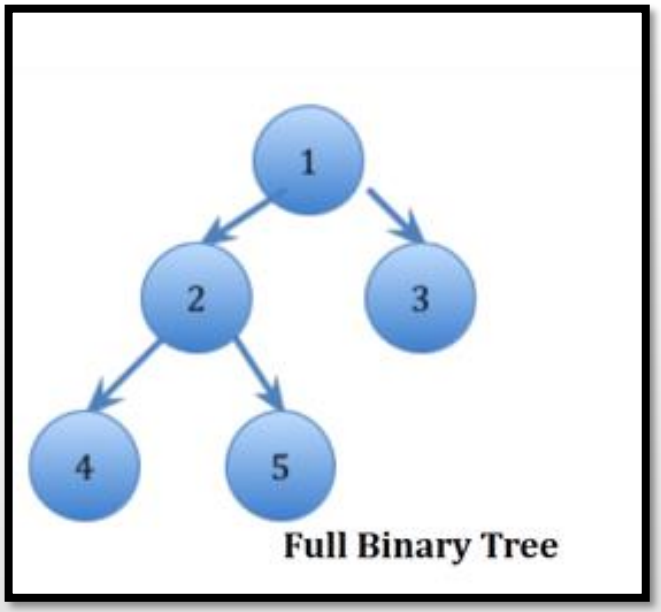
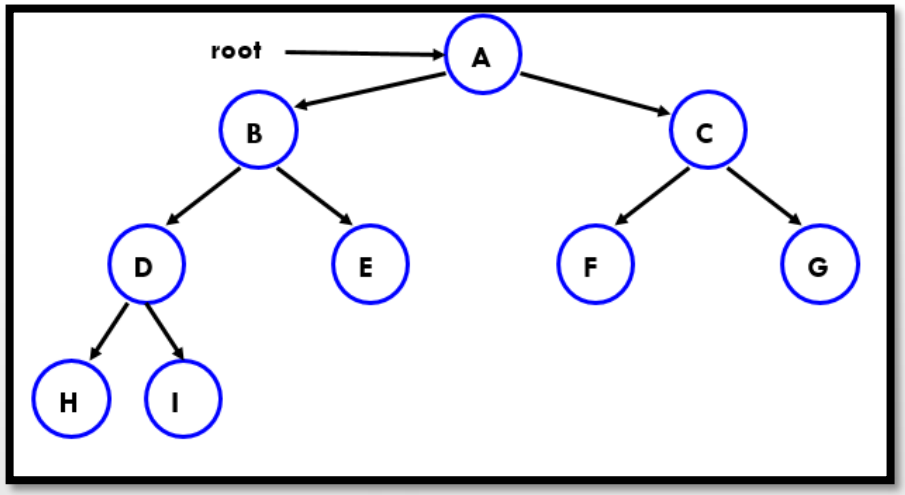


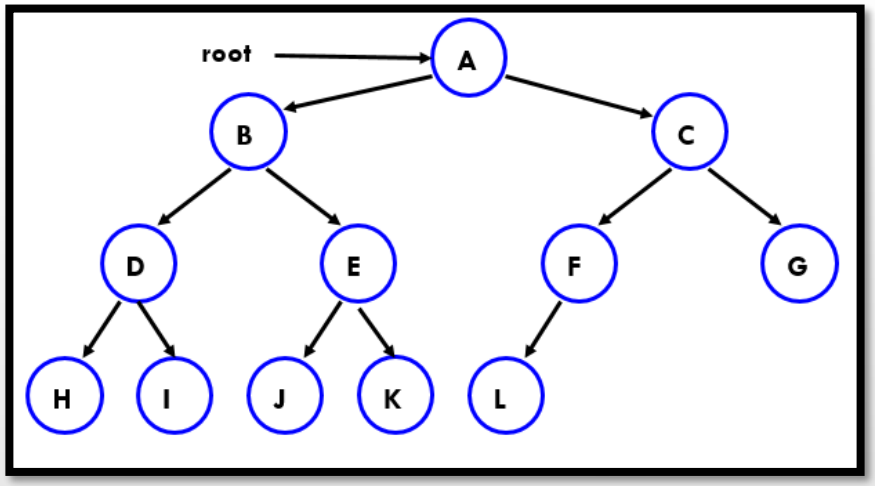
Figure 1: Full Binary Tree



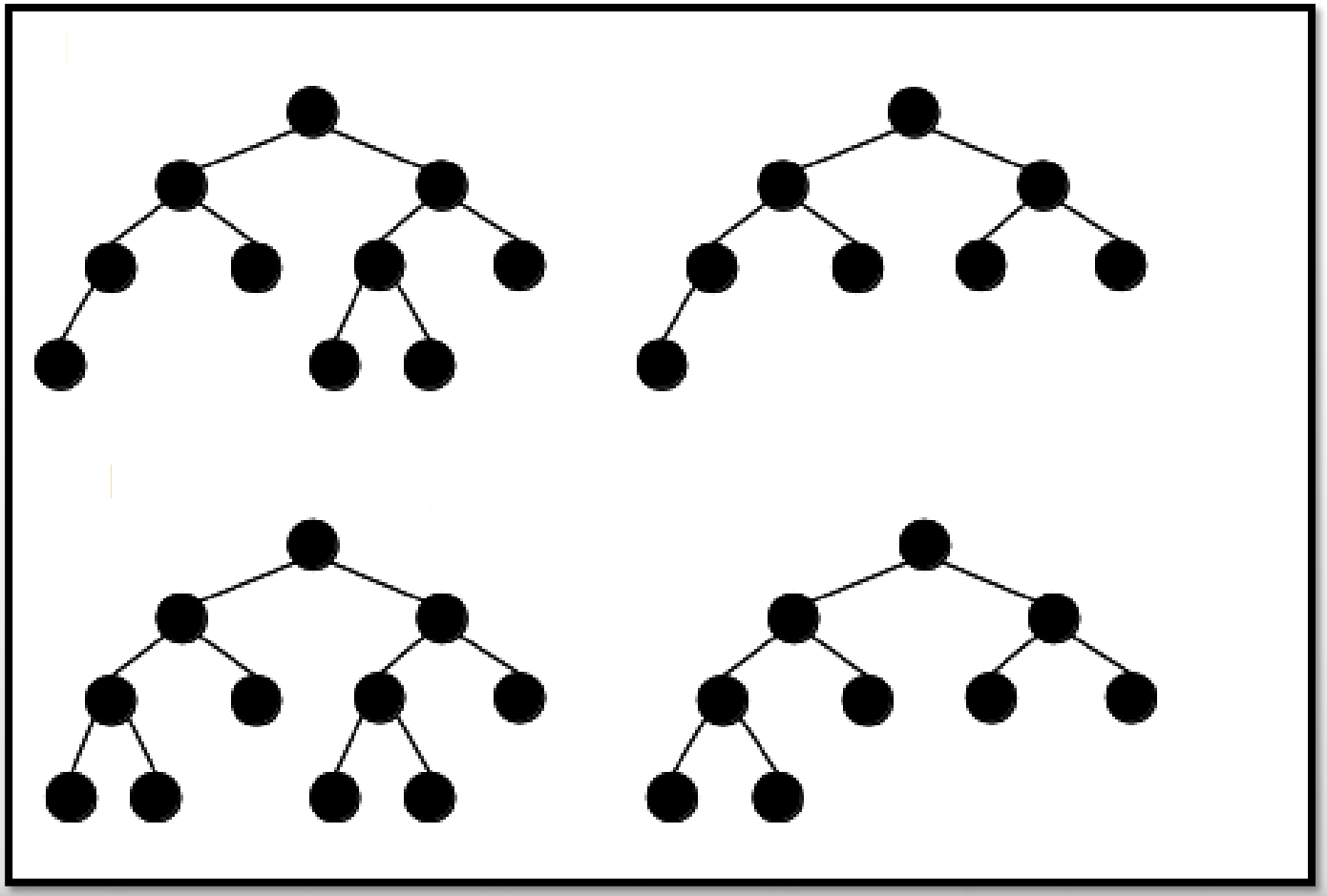
Full and Complete Binary Tree

2. A complete binary tree: is a binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right.

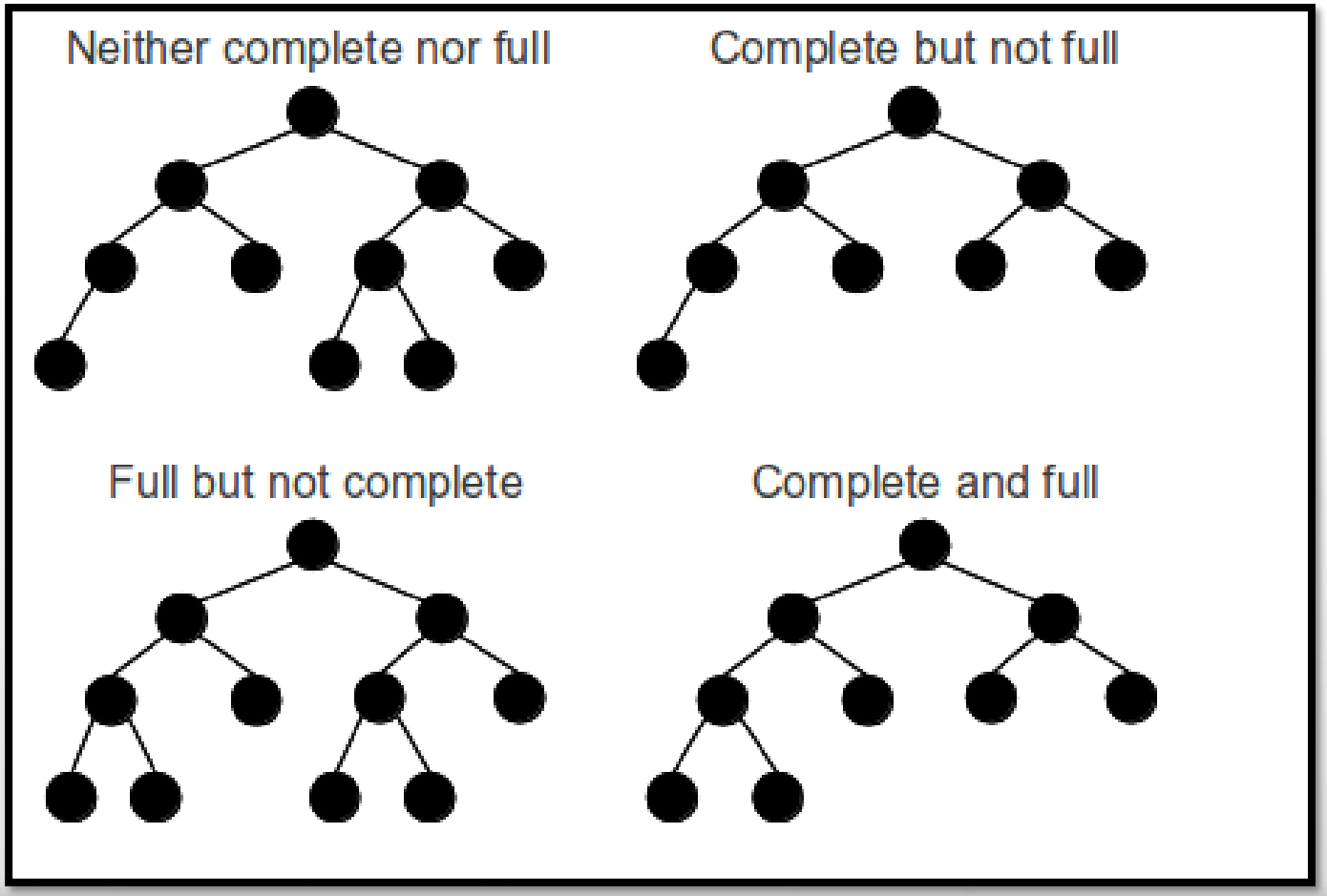




Full and Complete Binary Tree



Full and Complete Binary Tree

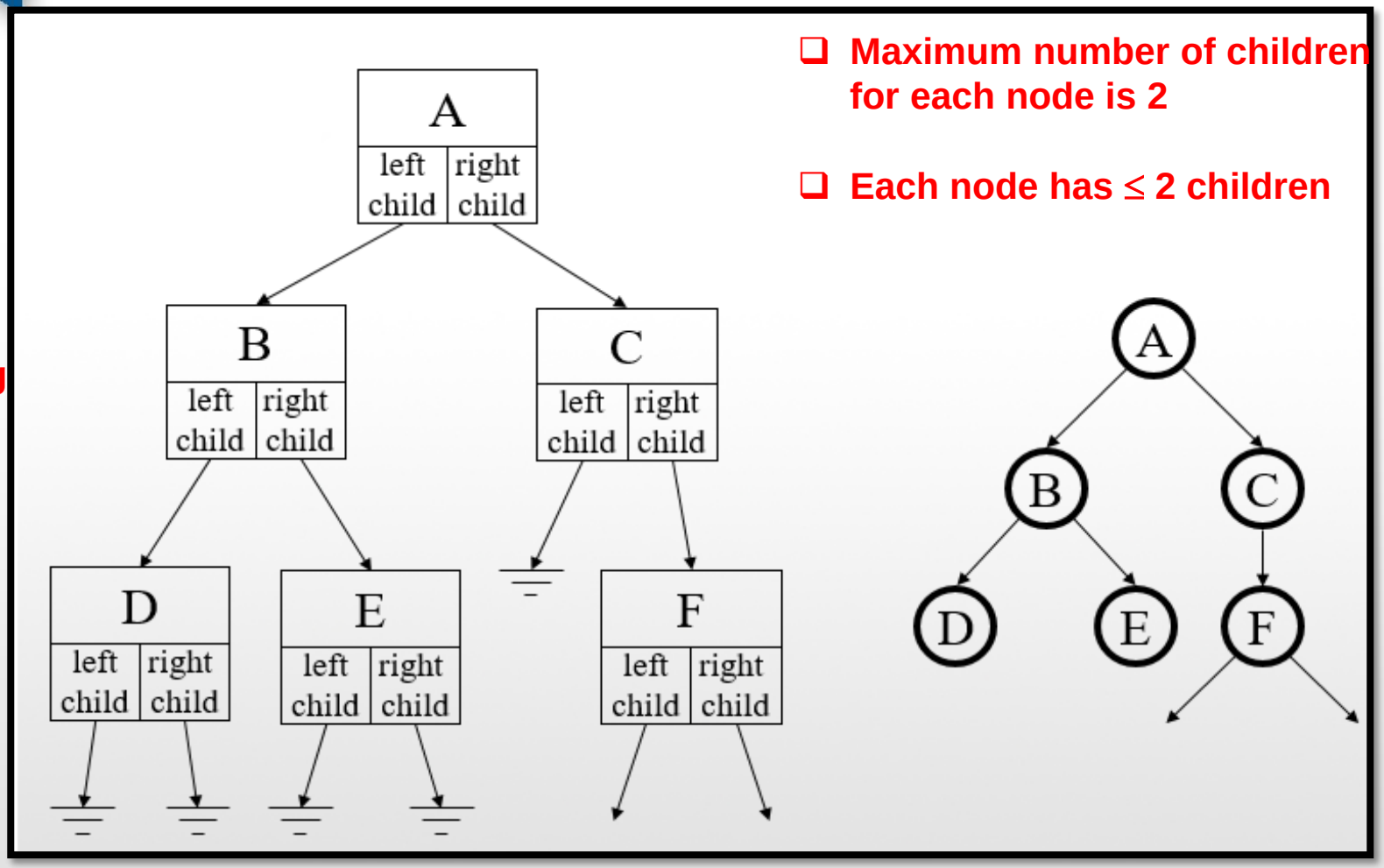


SELF STUDY

Binary Tree Representation

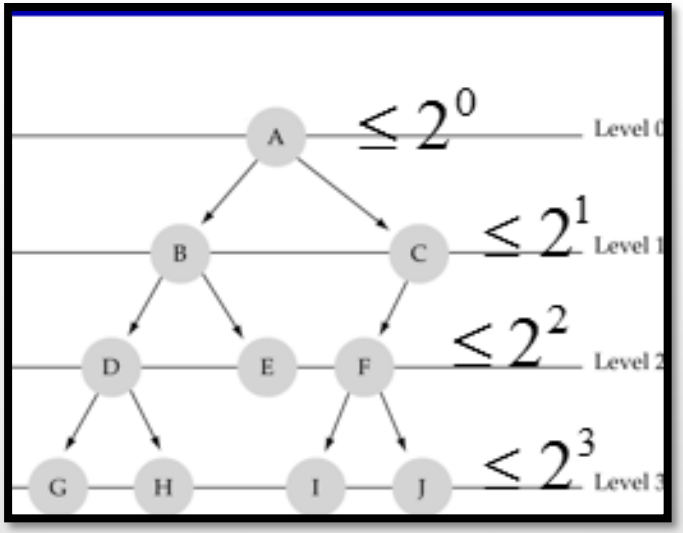
- **Maximum number of children for each node is 2**
- **Each node has ≤ 2 children**

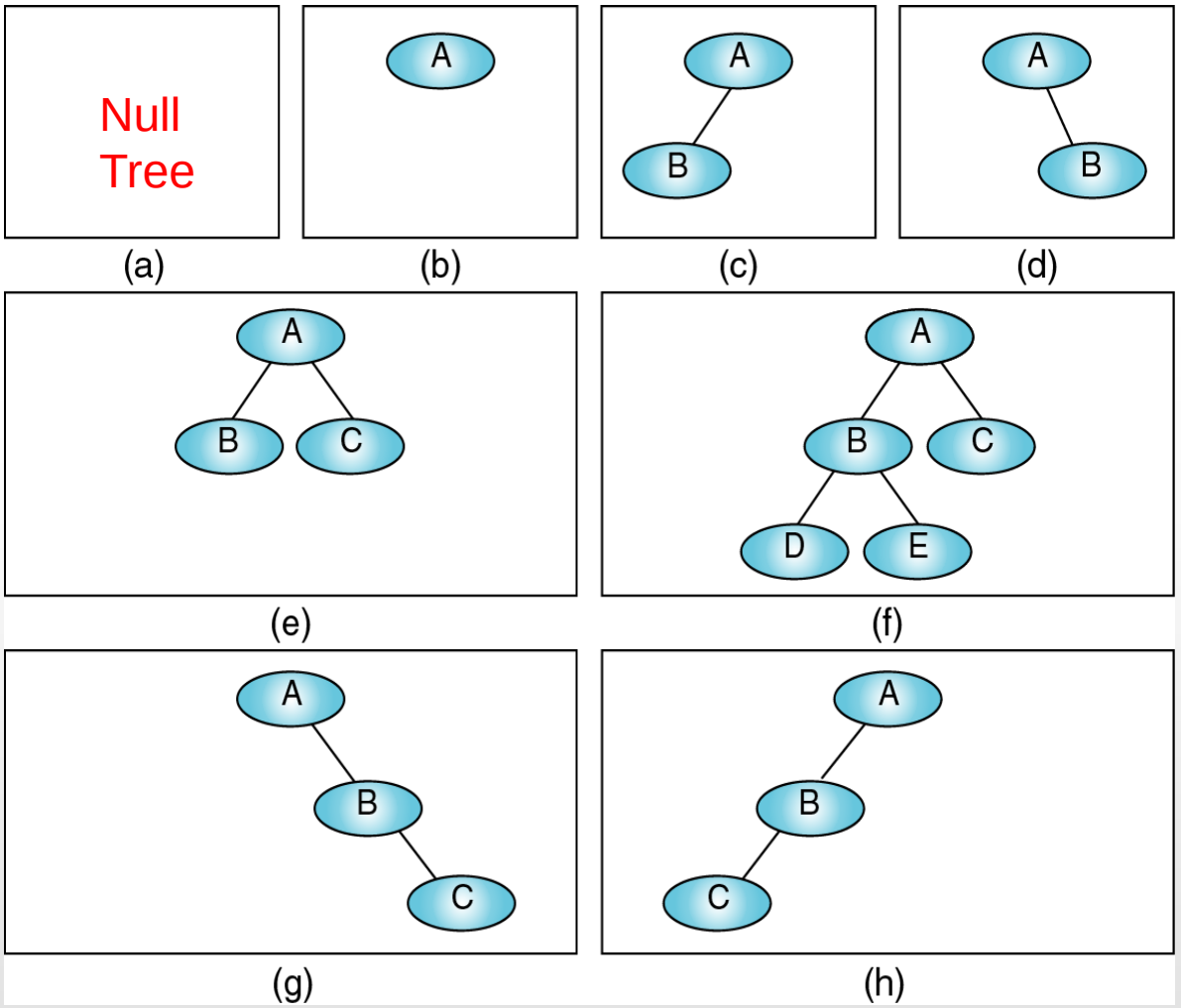
Each node is labeled as being either a left child or a right child



Binary Tree Representation

Max number of node in each level <= 2^L where L=0,1,2,…,L-1





Implementation : Binary Tree

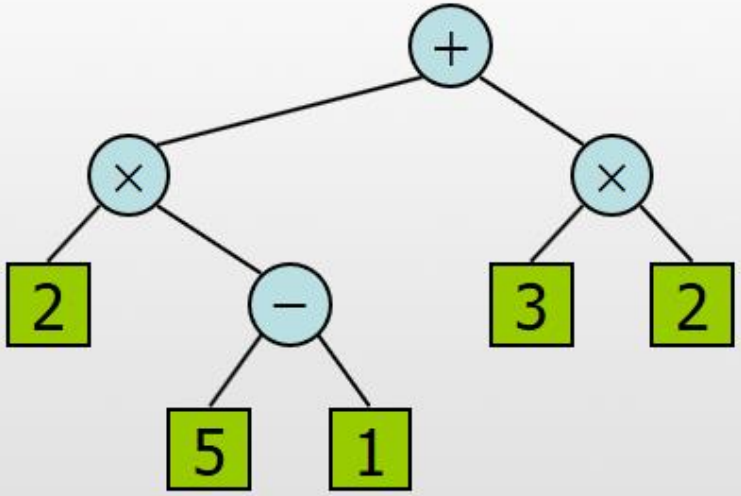
```
//Class Node for the Binary Tree
public class BinaryTreeNode {
   Object element;          //store data
   BinaryTreeNode left;    // left child
   BinaryTreeNode right; //right child

   public BinaryTreeNode(Object element){
      this(element, null, null);
    }

  public BinaryTreeNode(Object element,BinaryTreeNode left,BinaryTreeNode right)
   {
      this.element=element;
      this.left=left;
      this.right=right;
    }
}
```

Expression Trees

- A Binary Expression Tree is **A special kind of binary tree in which**

1. Each leaf is an operand (ex: constants, variables names,.).
2. The root and internal nodes are operators (ex: +, - ,*, ..etc).
3. Subtrees are subexpressions with the root being an operator.

Example:
expression tree for the expression
((2 × (5 − 1)) + (3 × 2))



Constructing an Expression Tree

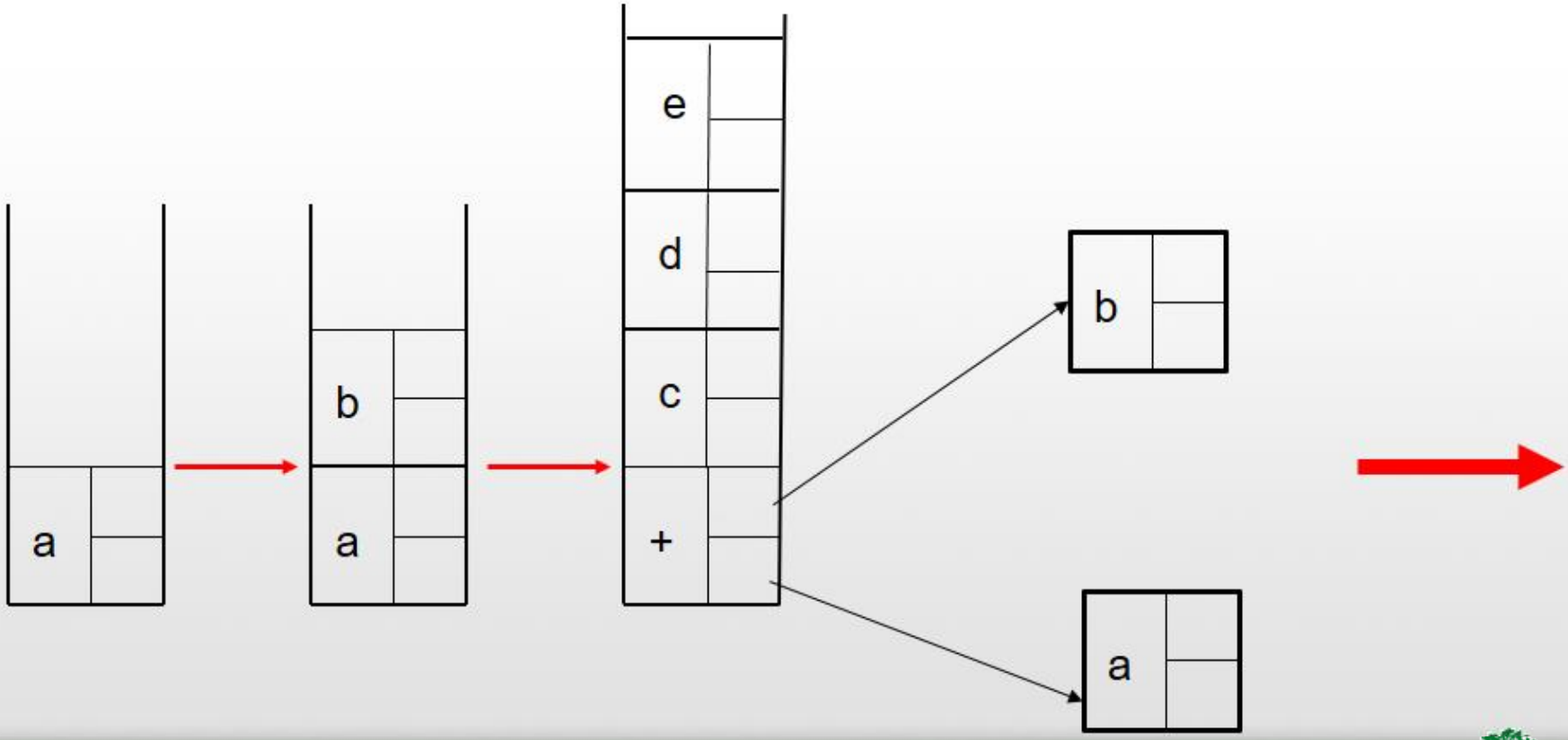
- The pseudo code algorithm to convert a valid postfix expression, containing binary operators, to an expression tree: (**We will use a stack to build an expression tree from postfix**)

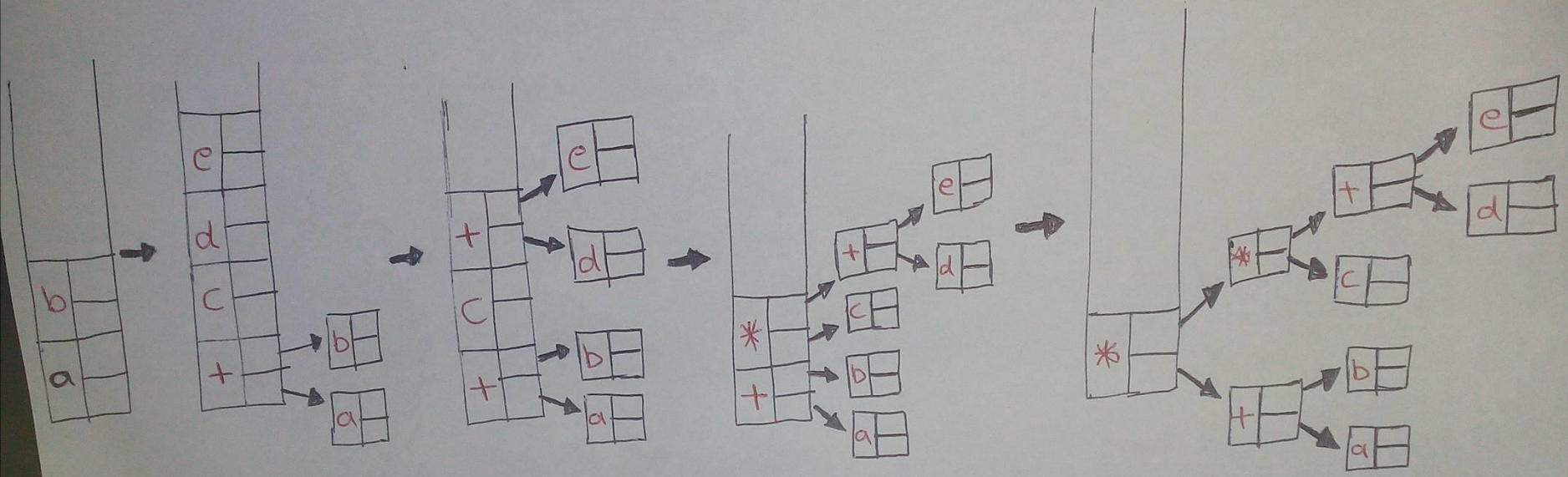
```
1  while(not the end of the expression)
2  {
3     if(the next symbol in the expression is an operand)
4     {
5       create a node for the operand ;
6       push the reference to the created node onto the stack ;
7     }
8    if(the next symbol in the expression is a binary operator)
9    {
10     create a node for the operator ;
11     pop from the stack a reference to an operand ;
12     make the operand the right subtree of the operator node ;
13     pop from the stack a reference to an operand ;
14     make the operand the left subtree of the operator node ;
15     push the reference to the operator node onto the stack ;
16   }
17 }
```

Constructing an Expression Tree

Example: Consider the expression (a + b) * (c * (d + e)).

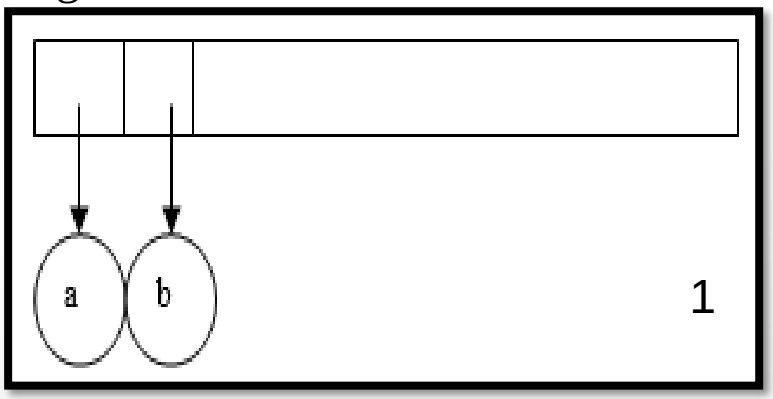
The postfix expression is: a b + c d e + * *

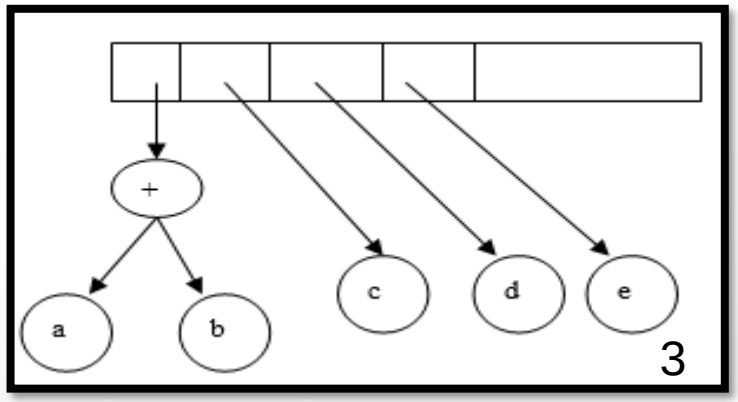


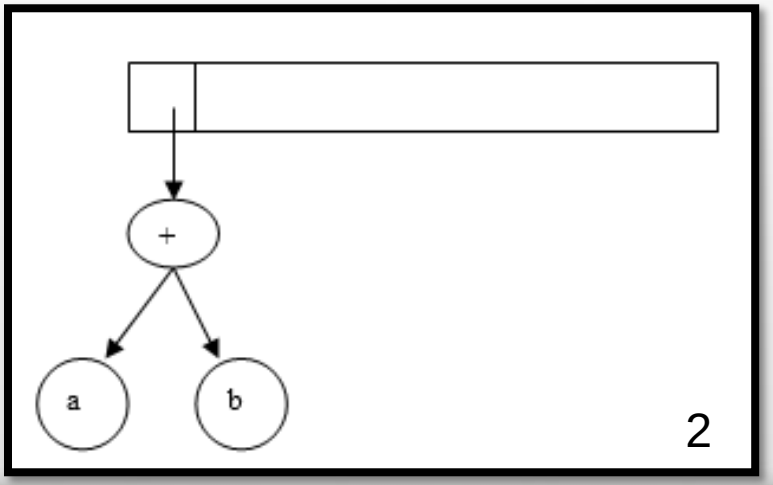

b
a
e
d
c
+
b
a
e
d
+
c
+
b
a
e
d
+
c
*
+
b
a
e
d
+
*
c
+
b
a

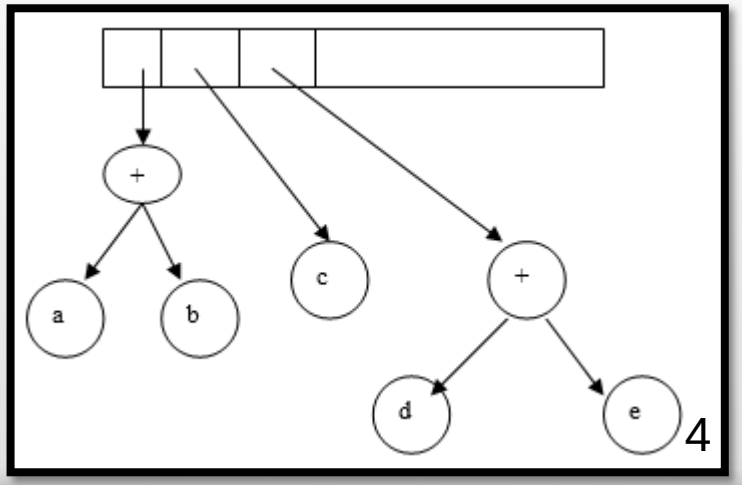
Constructing an Expression Tree

(I repeat the previous diagram. For convenience, the stack grow from left to right in the diagrams

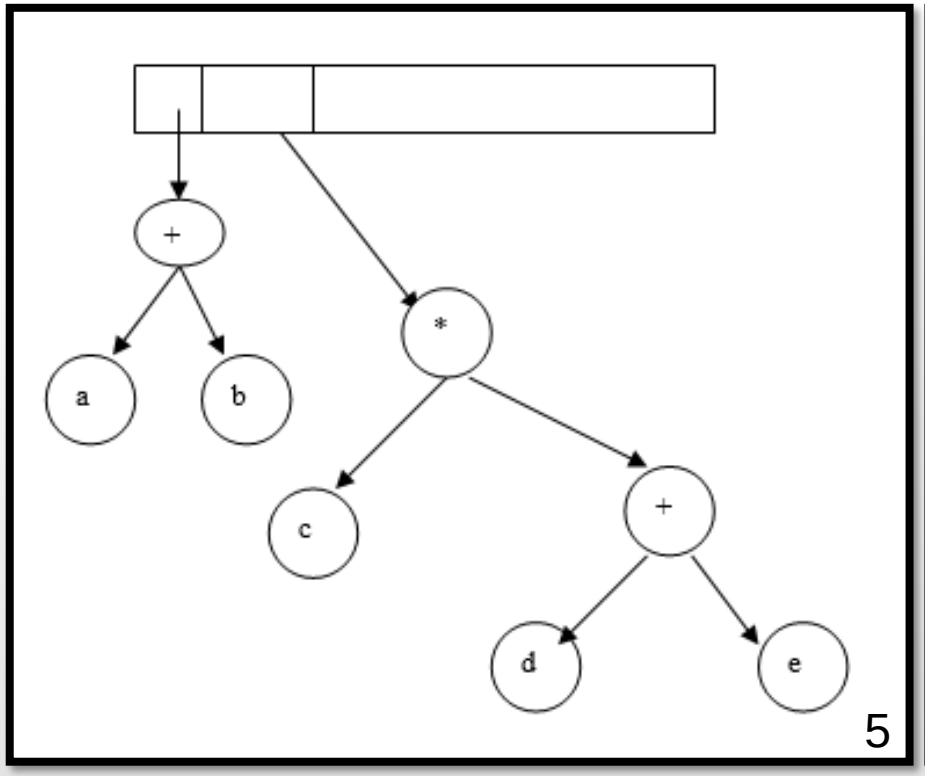




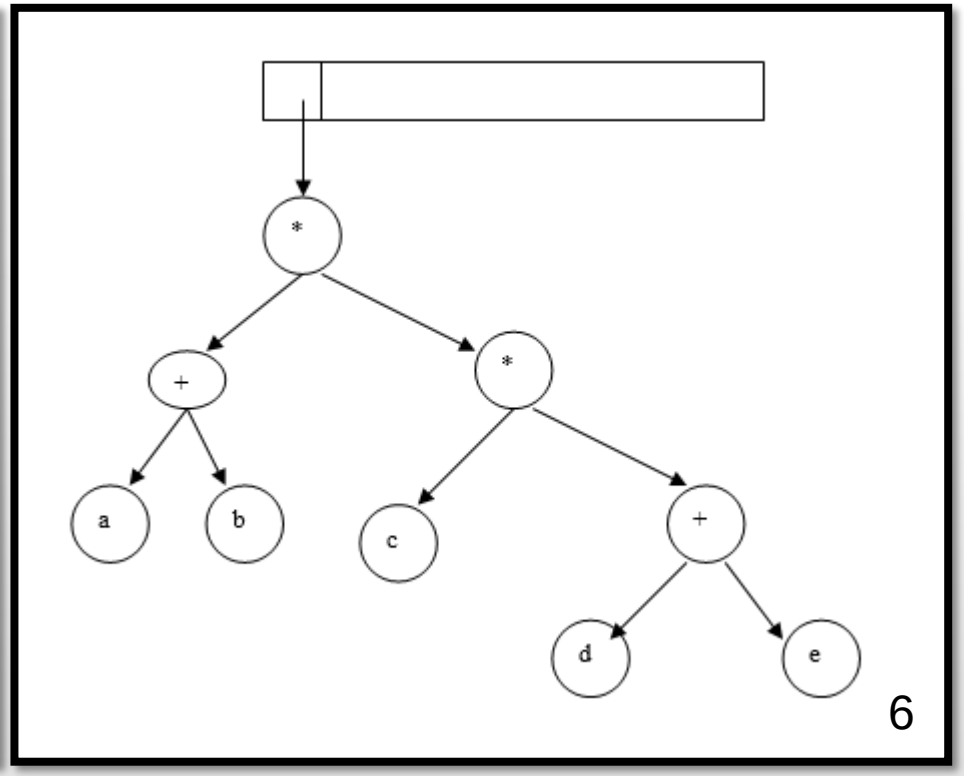




Constructing an Expression Tree



5



6

Examples

Give the prefix, infix and postfix expressions corresponding to the tree in the figure below.

Hint : prefix is preorder , infix is inorder, and postfix is postorder

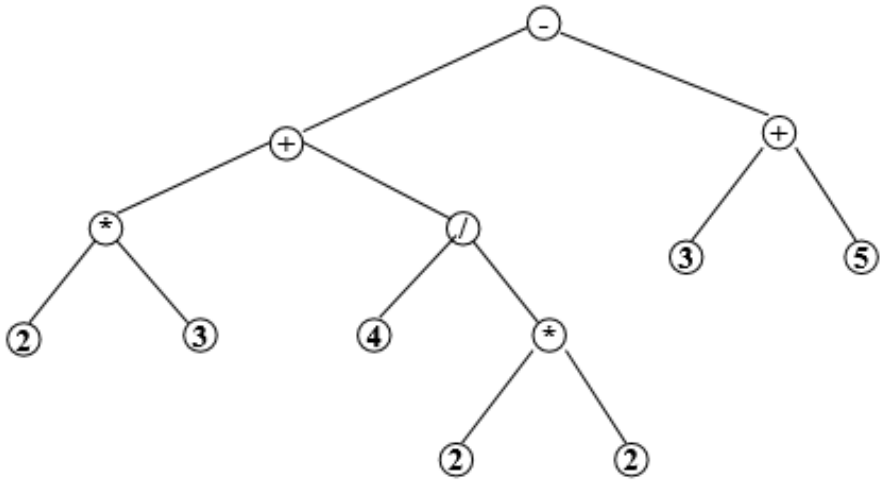


Figure 1: Expression Tree.

Solution:

prefix: - + * 2 3 / 4 * 2 2 + 3 5

infix: 2 * 3 + 4 / 2 * 2 - 3 + 5

postfix: 2 3 * 4 2 2 * / + 3 5 + -

A binary search tree

- **A binary search tree is a binary tree with a special property called the BST-property, which is given as follows:**

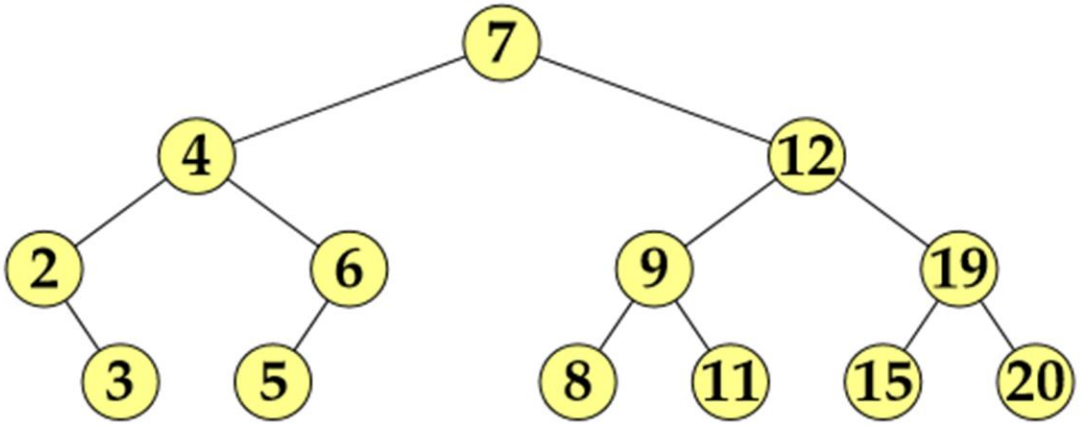
For all nodes x and y, if y belongs to the left subtree of x, then the key at y is less than or equal the key at x, and if y belongs to the right subtree of x, then the key at y is greater than or equal the key at x

Value (Left) <= Value (Root) < Value (Right)

Any one of them if duplication is allowed

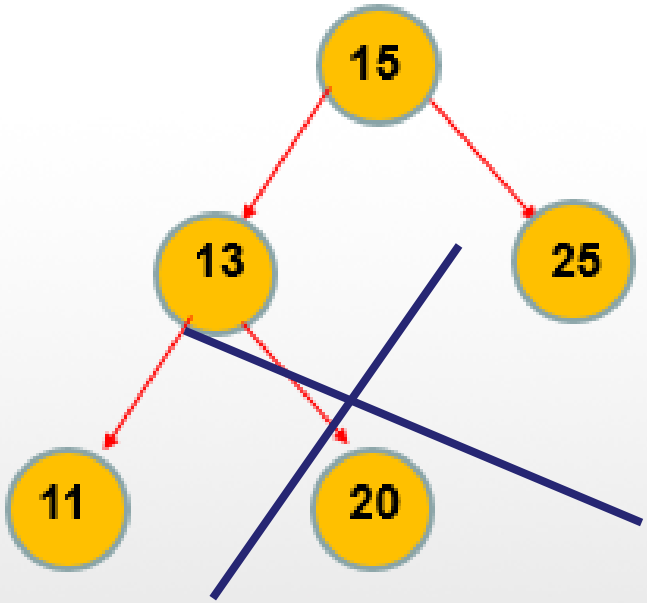
Value (Left) < Value (Root) <= Value (Right)

A binary search tree

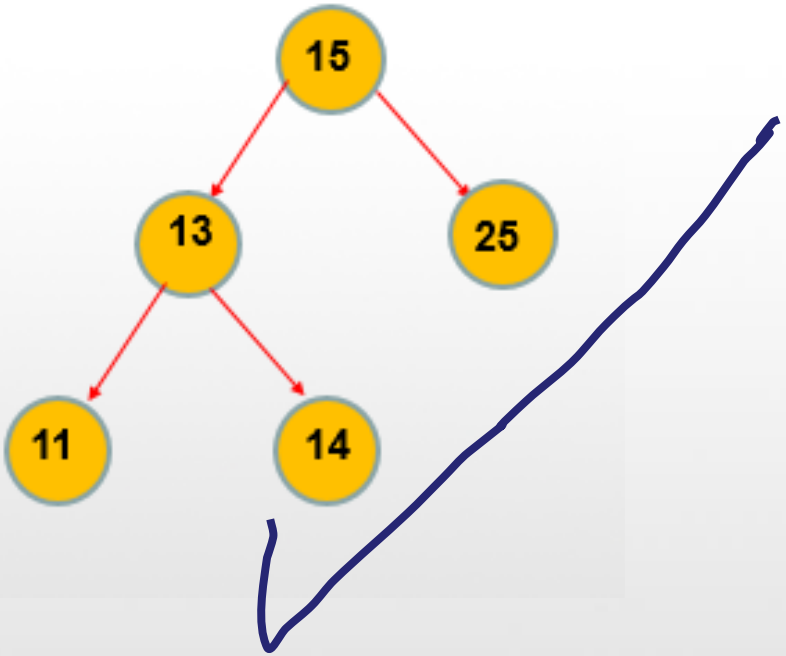


A binary search tree

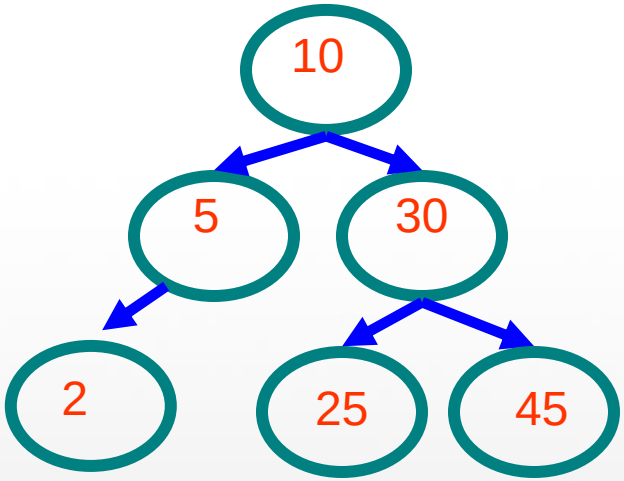
- Which of the following is a binary search tree ?

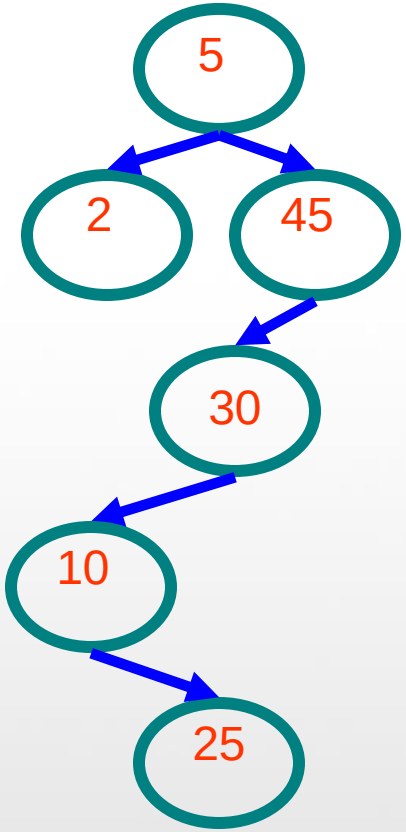


20 15

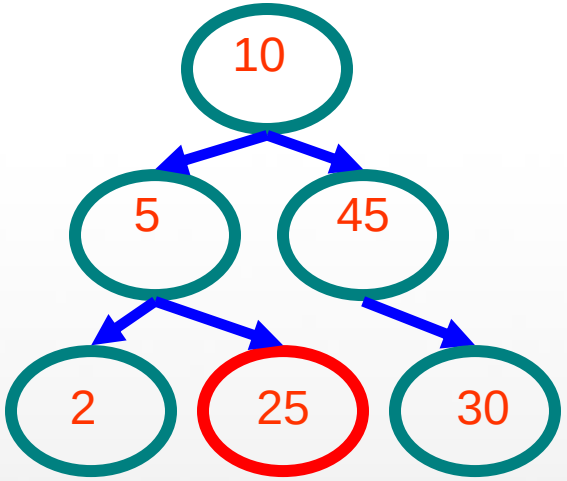


A binary search tree



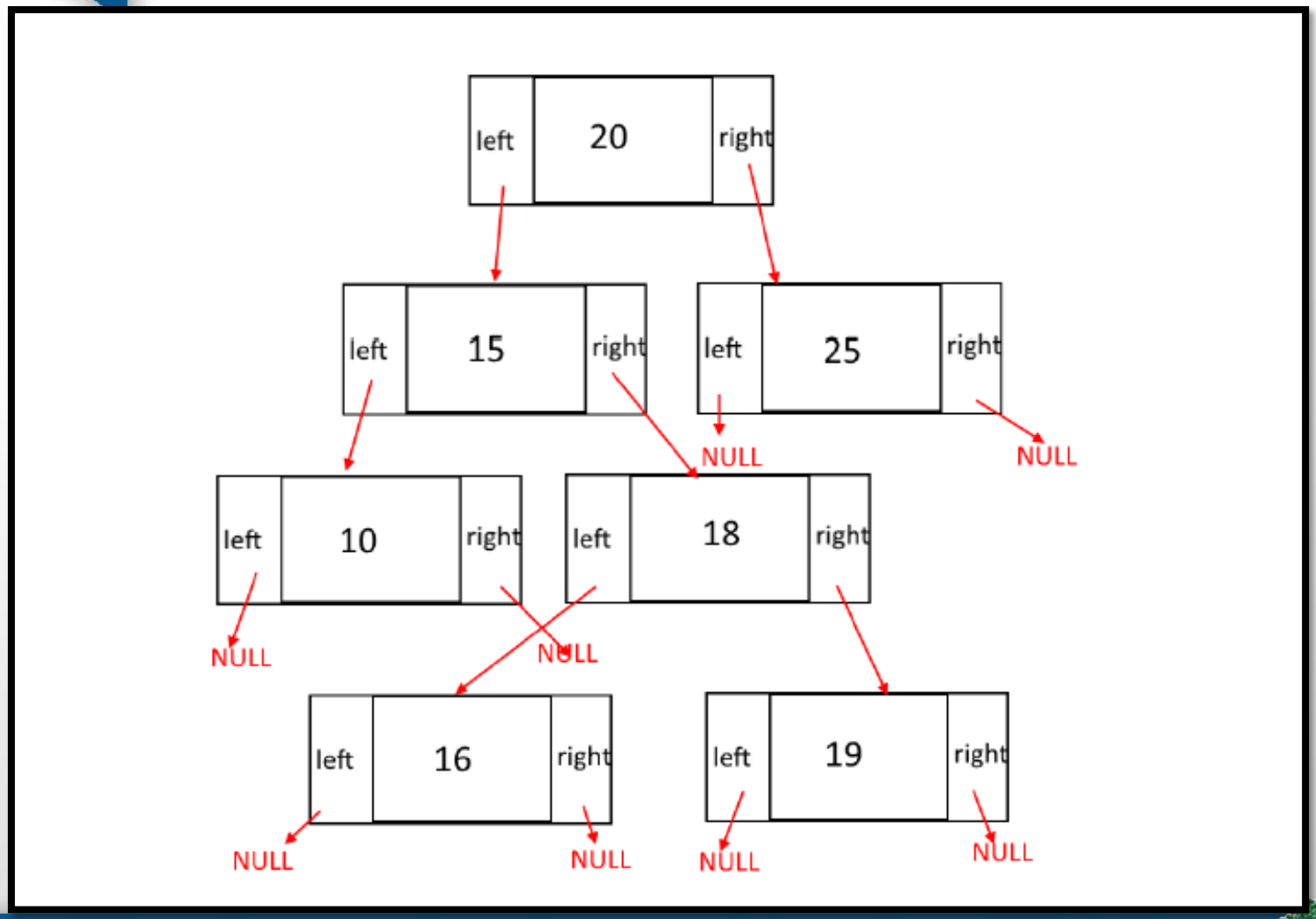


Binary search trees



Non-binary search tree

A binary search tree: Coding



Implementation : Binary Search Tree

```
//Class Node for the Binary Search Tree
public class BSTNode {
//for objects replace int to Object and modify the code
   int element;
   BSTNode left;
   BSTNode right;

   public BSTNode (int element){
      this (element,null,null);
   }
   public BSTNode (int element,BSTNode left,BSTNode right){
      this.element=element;
      this.left=left;
      this.right=right;
   }

}
```