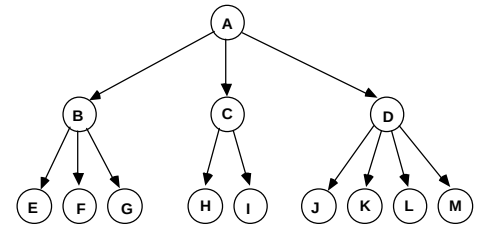
Δομές Δεδομένων & Ανάλυση Αλγορίθμων
3ο Εξάμηνο

- **Δέντρα**
- **Δυαδικά Δέντρα**
- **Δυαδικά Δέντρα Αναζήτησης (Binary Search Trees)**

Δημοσθένης Σταμάτης
http://www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

ΔΕΝΤΡΑ (TREES)

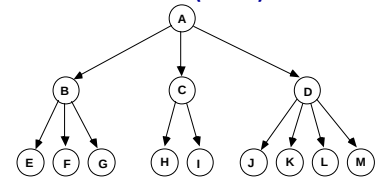


ΔΕΝΤΡΑ (TREES)

Ορισμός 1:
Δέντρο (*tree*) είναι ένα σύνολο **Τ** από **κόμβους (*nodes*)**, τέτοιο ώστε είτε:
(α) Το **Τ** είναι κενό ή
(β) Το **Τ** περιλαμβάνει ένα ξεχωριστό κόμβο, R, που ονομάζεται **ρίζα (*root*)** του **Τ** και οι υπόλοιποι **κόμβοι Τ - {R}** χωρίζονται σε μηδέν ή περισσότερα σύνολα κόμβων, **Τ1, Τ2, . . ., Τn**, που είναι ξένα μεταξύ τους και τα οποία είναι με τη σειρά τους δέντρα. Τα **Τ1, Τ2, . . ., Τn**, ονομάζονται υποδέντρα του **Τ**.

Ορισμός 2:
Δέντρο είναι μία συλλογή από στοιχεία, που ονομάζονται **κόμβοι**. Οι κόμβοι του δέντρου συνδέονται μεταξύ τους με τη βοήθεια **ακμών (*arcs*)** με βάση τους εξής κανόνες:
(α) Υπάρχει ένας και μόνον ένας κόμβος στον οποίο δεν καταλήγει καμία ακμή (**η Ρίζα {*Root*}** του δέντρου).
(β) Σε όλους τους υπόλοιπους κόμβους καταλήγει υποχρεωτικά μία και μόνο μία ακμή.

ΔΕΝΤΡΑ (TREES)



T = { A, B, C, D, E, F, G, H, I, J, K, L, M }

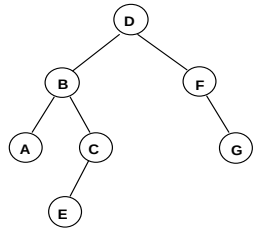
A: η ρίζα του Δέντρου

Το σύνολο **T - { A }** χωρίζεται σε τρία υποδέντρα:

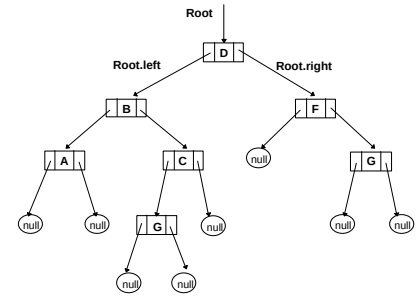
T1 = { B, E, F, G }, **T2 = { C, H, I }** και **T3 = { D, J, K, L, M }**

ΔΥΑΔΙΚΑ ΔΕΝΤΡΑ (BINARY TREES)

ΟΡΙΣΜΟΣ 3:
Δυαδικό δέντρο (*binary tree*) είναι ένα δέντρο του οποίου κάθε κόμβος έχει το πολύ δύο υποδέντρα. Τα υποδέντρα του δυαδικού δέντρου ονομάζονται αριστερό και δεξιό υποδέντρο αντίστοιχα.



Αναπαράσταση Δυαδικού Δέντρου με τη βοήθεια συνδεδεμένης δομής

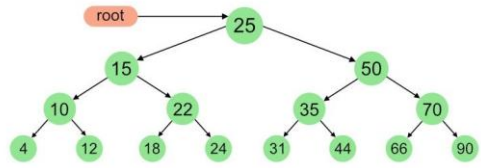


Διέλευση Δυαδικού Δέντρου

Η ενθεματική διέλευση επισκέπτεται τους κόμβους με την παρακάτω σειρά:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

Η προθεματική διέλευση επισκέπτεται τους κόμβους με την παρακάτω σειρά:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

Η επιθεματική διέλευση επισκέπτεται τους κόμβους με την παρακάτω σειρά:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25



Αναπαράσταση Δυαδικού Δέντρου με τη βοήθεια συνδεδεμένης δομής // class TreeNode

```
public class TreeNode
{
    TreeNode left;
    int item;
    TreeNode right;

    public TreeNode(int data) {
        item = data;
        left = right = null;
    }
        ……..
}
```


Αναπαράσταση Δυαδικού Δέντρου με τη βοήθεια συνδεδεμένης δομής // class TreeNode

```
public int getNodeData( ){
    return item;
}
public TreeNode getLeftNode( ) {
    return left;
}
public TreeNode getRightNode( ) {
    return right;
}
```


Αναπαράσταση Δυαδικού Δέντρου με τη βοήθεια συνδεδεμένης δομής // class TreeNode

```
public void setNodeData(int data ){
    item = data;
}
public void setLeftNode(TreeNode node ){
    left = node;
}
public void setRightNode(TreeNode node){
    right = node;
}
```


Αναπαράσταση Δυαδικού Δέντρου με τη βοήθεια συνδεδεμένης δομής // class BSTree

```
public class BSTree {
    private TreeNode root;
    public BSTree( ) {
        root = null;
    }
    public boolean isEmpty( ) {
        return (root == null);
    }
    public void insertElement(int data){
        if (isEmpty( )) root = new TreeNode(data);
        else  insertNode(data,root);
    }  . . .
```

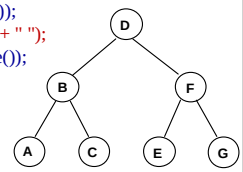

Αναπαράσταση Δυαδικού Δέντρου με τη βοήθεια συνδεδεμένης δομής // class BSTree

```
private void insertNode(int data, TreeNode node) {
    if (data < node.getNodeData()) {
        if (node.getLeftNode() == null)
            node.setLeftNode(new TreeNode(data));
        else insertNode(data,node.getLeftNode());
    }
    else {
        if (node.getRightNode() == null)
            node.setRightNode(new TreeNode(data));
        else insertNode(data,node.getRightNode());
    }
```

Ενθεματική Διέλευση (Inorder Traversal) από τους κόμβους ενός Δυαδικού Δέντρου

```
public void inOrderTraversal( ) {
    inOrder(root);
}
private void inOrder(TreeNode node) {
    if (node == null) return;
    inOrder(node.getLeftNode());
    System.out.print(node.item + " ");
    inOrder(node.getRightNode());
}
```

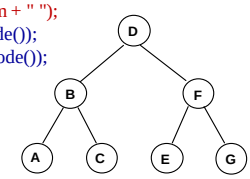
A B C D E F G



Προθεματική Διέλευση (Preorder Traversal) από τους κόμβους ενός Δυαδικού Δέντρου

```
public void preOrderTraversal( ) {
    preOrder(root);
}
private void preOrder(TreeNode node) {
    if (node == null) return;
    System.out.print(node.item + " ");
    preOrder(node.getLeftNode());
    preOrder(node.getRightNode());
}
```

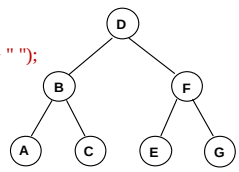
D B A C F E G



Επιθεματική/Μεταθεματική Διέλευση (Postorder Traversal) από τους κόμβους ενός Δυαδικού Δέντρου

```
public void postOrderTraversal( ) {
    postOrder(root);
}
private void postOrder(TreeNode node) {
    if (node == null) return;
    postOrder(node.left);
    postOrder(node.right);
    System.out.print(node.item + " ");
}
```

A C B E G F D



Δυαδικά Δέντρα

```
public int countNodes( ) {
    return countNodes(root);
}

private int countNodes(TreeNode node) {
    if ( node == null ) return 0;
    else return
        countNodes(node.getLeftNode()) +
        countNodes(node.getRightNode()) +1;
}
```


Δυαδικά Δέντρα

```
public int countLeafs( ) {
    return countLeafs(root);
}
private int countLeafs(TreeNode node) {
    if ( node == null ) return 0;
    else {
        int count = 0;
        count += countLeafs(node.getLeftNode());
        count += countLeafs(node.getRightNode());
        if ((node.getLeftNode() == null) &&
            (node.getRightNode() == null)) count++;
     return count;
}
```