Southern Technical University
Technical Institute / Qurna
Dep. of Computer Systems Techniques

Second class
Subject : Data Structures
Lecturer : Israa Mahmood Hayder
Lecture no.18

هياكل البيانات اللاخطية(الاشجار)

(Trees)

- الاسبوع الثامن عشر -

الثامن عشر	* الأشجار. - أنواع الأشجار trees types. - طرق تمثيل الأشجار trees representation. - طرق استعراض الأشجار trees traversing methods.

B// Rationale (مبررات الوحدة) :-

The Tree is a non-linear linked list data structures, it is using the methods of pointers and linked lists for its implementation. Data structures organized as trees will prove valuable for a range of applications, especially for problems of information retrieval.

C// Central Ideas (الفكرة المركزية):-

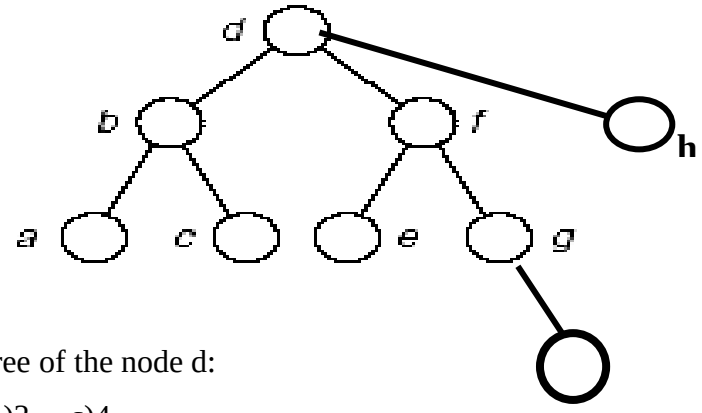
- Definitions and Examples on Trees
- Tree types (Normal & Binary)
- Complete binary tree
- Binary tree search
- Tree representation
- Traversing methods

D// Objectives (أهداف الوحدة):-

After studying this unit , the student will be able to:-

- Define the Tree and its elements
- Recognize the tree types and representation
- Create binary tree from a list.
- Traverse the tree using different methods

Circle the correct answer considering the shown graph:



1- Outdegree of the node d:

a) 2 b)3 c)4

2- The indegree of the node c :

a) 0 b)1 c)2

3- The tree is:

a) normal b)binary c)balanced

4- The simple path is:

a) d to g b)d to a c)d to b

5- The longest path is:

a) d to i b) f to i c) d to c

6- Adjacent set for vertex f:

a) {d,e,g} b) {e,g,i} c) {d,e,g,i}

7- Adjacent set for vertex b:

a) {a,c} b) {a,b,c} c) {a,c,d}

8- The degree of the tree:

a) 2 b) 3 c) 4

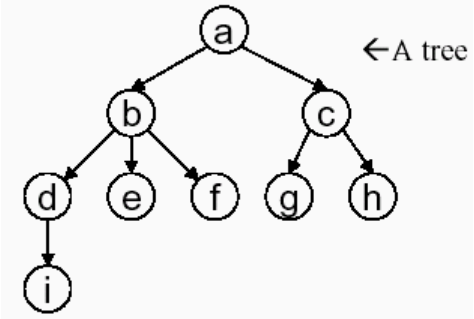
9- The number of levels is :

a) 2 b) 3 c) 4

10- The degree of the node d is :

a) 0 b)2 c) 3

1// Non-Linear Data Structures(Tree)



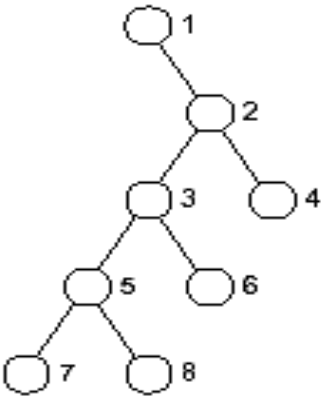
A Tree: Is set of related interconnected nodes in a hierarchical structure. It is a graph or diagraph that have no cycle.

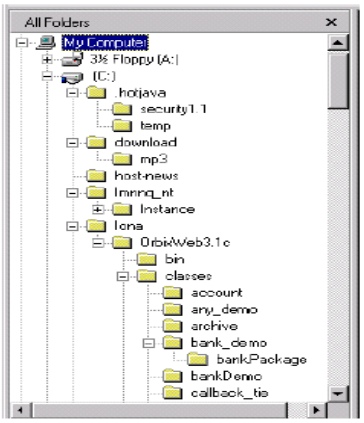
Trees examples:

Where have you seen a tree structure before?

Examples of trees:

- Directory tree
- Family tree
- Company organization chart
- Table of contents



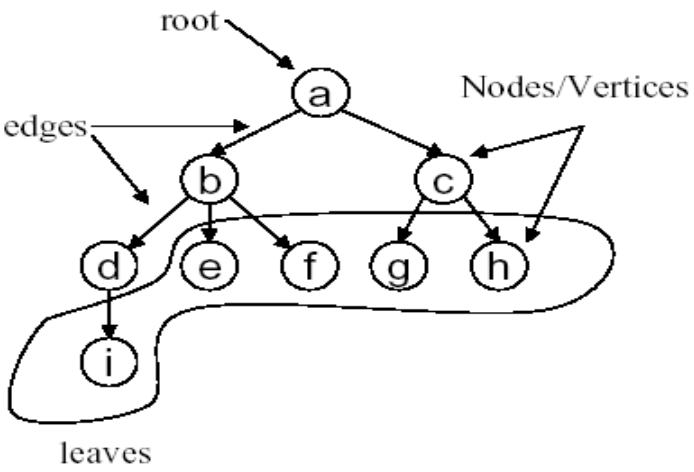


2// Definitions:

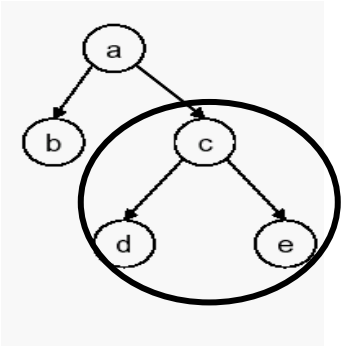
- The isolated node is also directed tree

- The first or top node in a tree is called the ***root*** node.

- An ***edge*** is a connection between two vertices (nodes)
- The relation between nodes is (***parent-child***) relation.
- ***Child*** of a node u: Any node reachable from u by 1 edge.
- ***Parent*** is the node closest to the root.
- All nodes except root have exactly one parent.
- A ***vertex*** (or ***node***) is a simple object that can have a name and can carry other associated information.



- Nodes with no children are called ***leaves*** or ***terminals***
- Node with branches called ***branch node***
- The terminals have outdegree=0.
- The indegree of the root=0.
- ***Depth*** of a node :
- Depth of root node is 0.
- Depth of any other node is 1 greater than depth of its parent.
- The ***level*** of any node is the length of its path from root (***depth***)
- A ***path*** in a tree is a list of distinct vertices in which successive vertices are connected by edges in the tree.
- example: {(a,c), (c, e)} is a path.
- ***Subtree*** : any node of a tree, with all of its branches.



← Subtree

- The defining property of a ***tree*** is that there is one path connecting any two nodes.
- A disjoint set of trees is called a ***forest.*** If we delete and it edges connecting it with level 1, we obtain a set of disjoint tree called forest.
- There is exactly one path between the root and each of the other nodes in the tree
- Each node except the root has exactly one node above it in the tree, (i.e. it's ***parent)***, and we extend the family analogy talking of ***children***, ***siblings***, or ***grandparents***

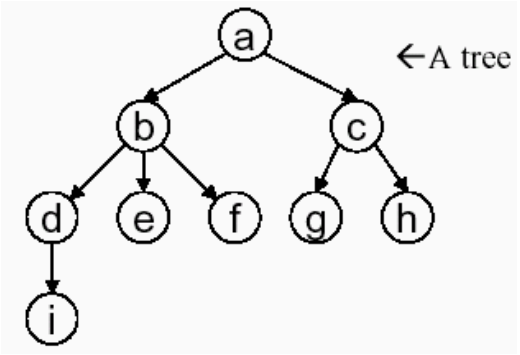
Example:

Level (0) →

Level (1) →

Level (2) →

Level (3) →



The level of the root=0, the level of the shown tree = 4 (0-3)

- The ***height*** of the tree is the length of the longest path from the root.
  Height of the shown tree=3
- The ***degree*** of the node c = the outdegree =2
- The degree of the tree = the highest degree in the tree =3.
- The ***degree*** of the node i =0
- The ***degree*** of the node a =2
- The ***degree*** of the node d =1

- Example path :((a,b) (b,d) (d,i))

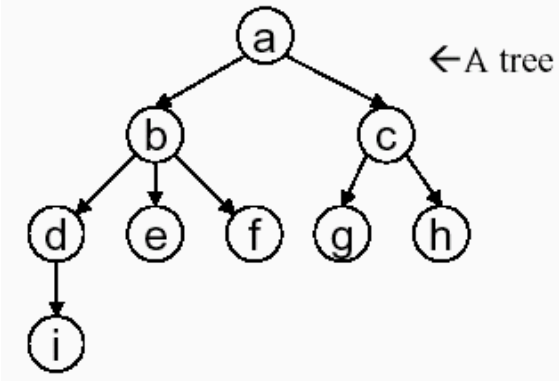
The forest: (when delete a)

1- {b,d,e,f,i}
2- {c,g,h}

Table of levels:

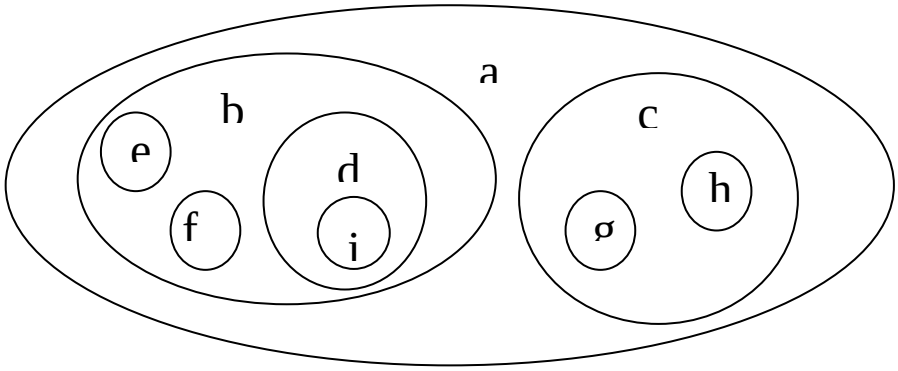
Level	nodes
1	a
2	b,c
3	d,e,f,g,h
4	i

3// Trees Representation:



a- Using Venn diagram:

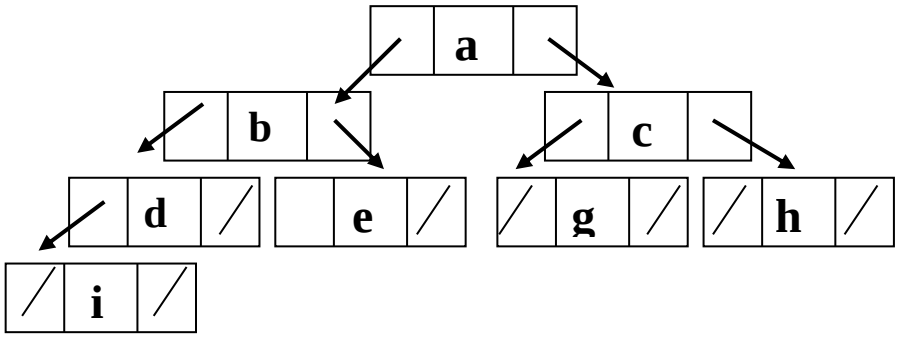
0



b- Using parentheses:

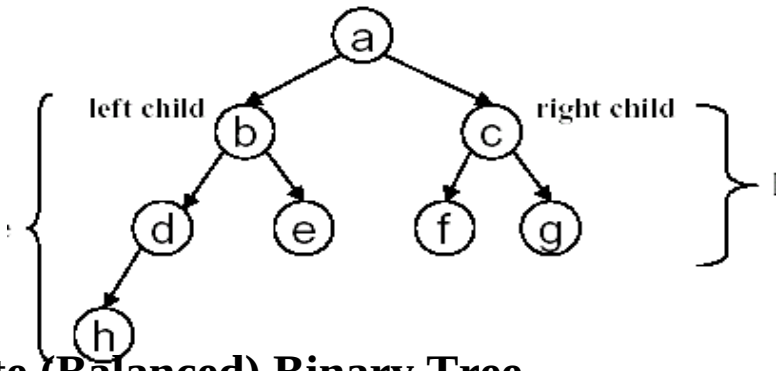
T= a(b(e) (f) (d(i)) c (g)(h))

c- Using linked lists:



4// Binary Trees

• A binary tree is a tree where each node has exactly zero, one or two children.
• i.e. each parent can have no more than 2 children.
• As with any Abstract Data Structure we can implement a binary tree in a number of ways, using arrays, strings, or structures and pointers

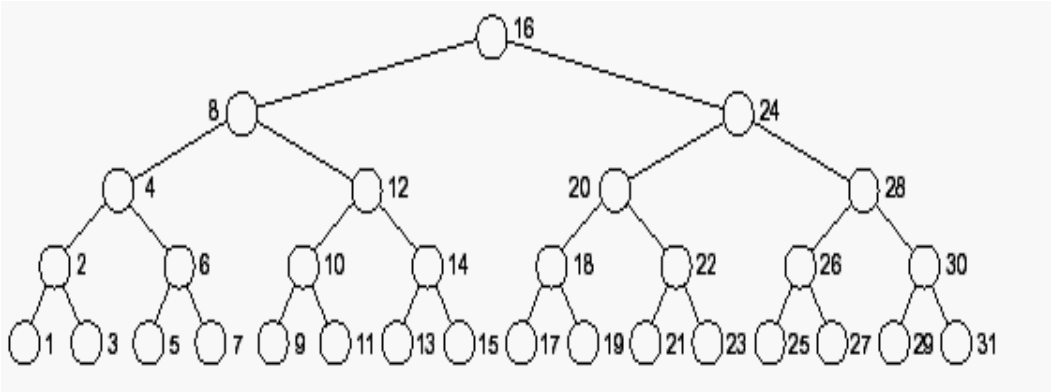


Left subtree

Right subtree

A- Complete (Balanced) Binary Tree

Is a binary tree that each node has exactly two childred or no children, and all terminals at the same level.



Balanced

In the balanced binary tree $n = 2^M - 1$

when **n** is number of nodes

m is number of levels

B- Numbering binary tree:

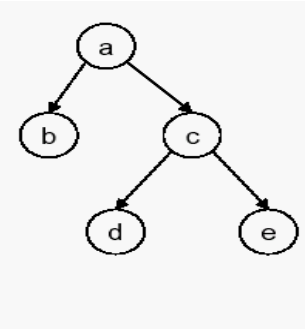
left node=n*2

right node= n*2+1

not balanced

(1*2)→

(2*2)→



← (1)

← **(1*2+1)=3**

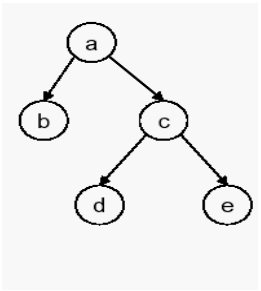
← **(2*3+1)=7**

C- Representing binary tree in memory:

not balanced

(1*2)→

(2*2)→



← (1)

← **(1*2+1)=3**

← **(2*3+1)=7**

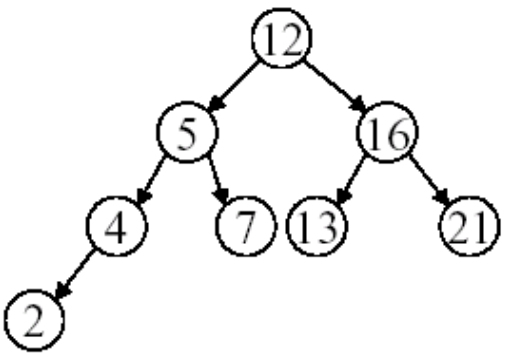
1	2	3	4	5	6	7							
a	**b**	**c**			**d**	**e**							

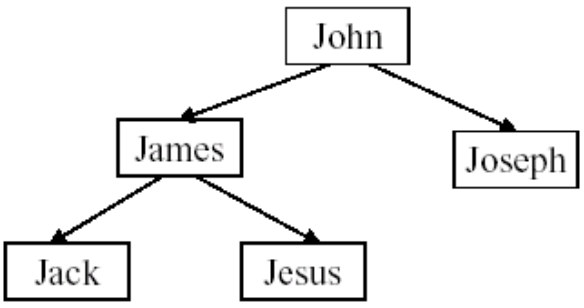
D- Binary Search Trees

• A binary tree which conforms to the following properties is called a

***binary search tree*.**

Ex: **Draw the BST of the list: 12,16,5,7,4,13,21,2**





BST for numbers **BST for names**

• Properties:

- Each value (key) in the tree exists at most once (i.e. no duplicates).

- The "greater-than" and "less-than" relations are well defined for the

data value.

- Sorting constraints:- for every node n :

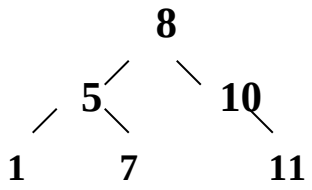
- All data in the left subtree of n is ***less than*** the data in the root of that subtree.
- All data in the right subtree of n is ***greater than*** the data in the root of that subtree.

Steps to build BST:

1- First number is considered as root

2- Next if was less it will be placed at the left, else it will placed at the right.

Example1: – Draw an ordered BST for the list **:** ***8, 10, 5, 1, 7, 11***



Quiz1: **Draw an ordered BST for the list :** ***80, 90, 55, 11, 70, 100***

5// Tree Traversal

• As with lists, we would like to be able to traverse through all
nodes in our tree.
• Problem: what order should the nodes be visited in?
- Top-down
- Left-to-right
- Bottom-up
• We may traverse a binary tree in 3 ways :- ***preorder, postorder, and inorder***

- Types of Traversal

• ***Preorder*** traversal (NLR):
- visit the node first and process.
- do preorder traversal of left subtree.
- do preorder traversal of right subtree.
- i.e. visits and processes each node in a tree BEFORE visiting and processing its children.

• ***Postorder*** traversal (LRN):
- do postorder traversal of left subtree.
- do postorder traversal of right subtree.
- visit the node last and process.
- i.e. visits and processes each node in the tree AFTER visiting and processing its children.

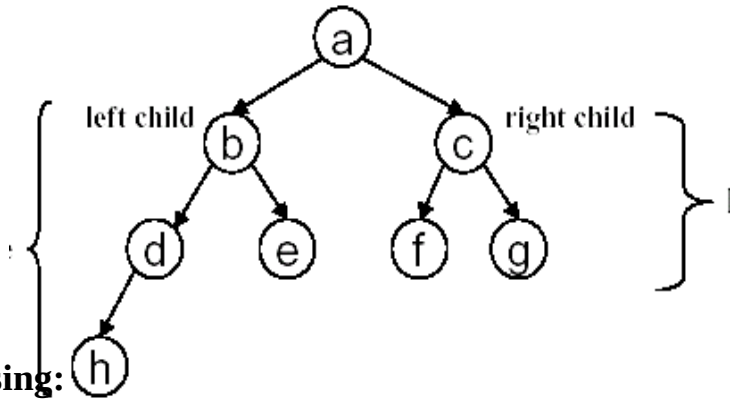
• ***Inorder*** traversal (LNR):
- do inorder traversal of left subtree.
- visit the node and process.
- do inorder traversal of right subtree.
- i.e. processes nodes in the tree in an ascending sorted order.

-Example of Tree Traversal

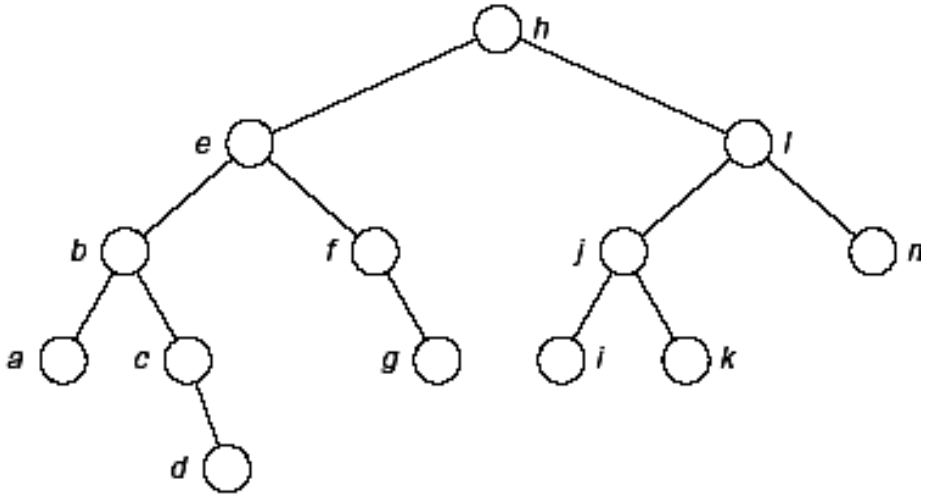
Preorder(NLR): a b d h e c f g

Inorder (LNR): h d b e a f c g

Postorder(LRN): h d e b f g c a



Quiz2: **Traverse the following tree using:**
1- inorder 2- postorder



Circle the correct answer considering the shown graph in Quiz2:

1- Outdegree of the node b:
a) 2 b)3 c)4

2- The indegree of the node c :
a) 0 b)1 c)2

3- The tree is:
a) normal b)binary c)balanced

4- The simple path is:
a) f to g b)h to a c)h to b

5- The longest path is:
a) h to g b) h to k c) h to d

6- Adjacent set for vertex f:
a) {e,f,g} b) {e,g} c) {g}

7- Adjacent set for vertex b:
a) {a,c} b) {a,b,c} c) {a,c,d}

8- The degree of the tree:
a) 2 b) 3 c) 4

9- The number of levels is :
a) 3 b) 4 c) 5

10- The degree of the tree is :
a) 4 b)2 c) 3

References:

1- Data Structures Demystified, by Jim Keogh and Ken Davidson, ISBN:0072253592, McGraw-Hill/Osborne © 2004

2- هياكل البيانات / الطبعة الثانية، تاليف د.عصام الصفار،اصدارات السفير للنشر/ بغداد، ٢٠٠١