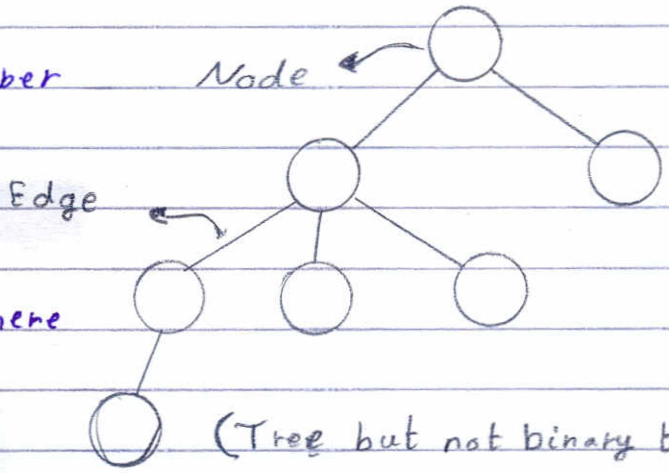
Binary Tree

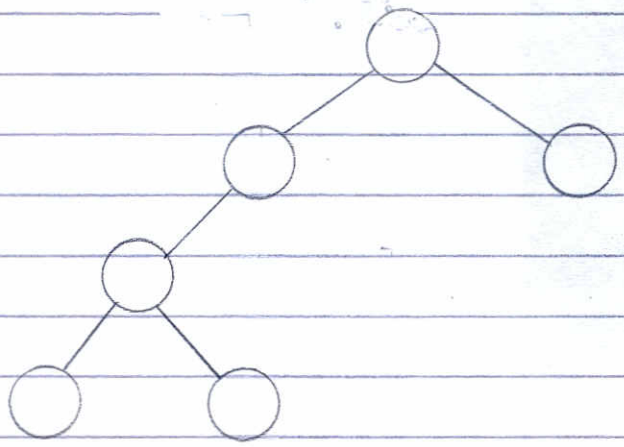
Tree: is a connected acyclic graph. It's hierarchical.

- Every node can have anynumber of subtrees.
- It canot be empty.
- for a tree with n nodes, there are (n-1) edges

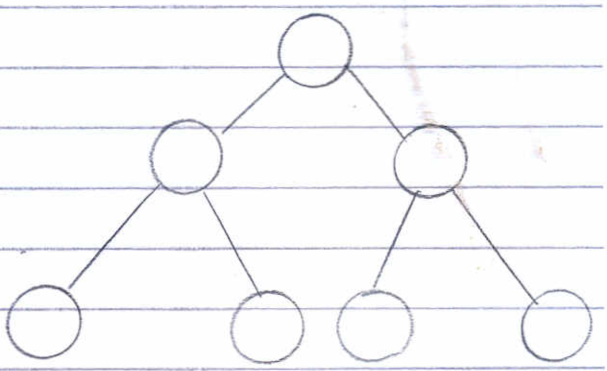


(Tree but not binary tree)

Binary Tree: It's a special case of tree.
It's a finite set of nodes, and it could bempty.
Every node has at most two sub trees.



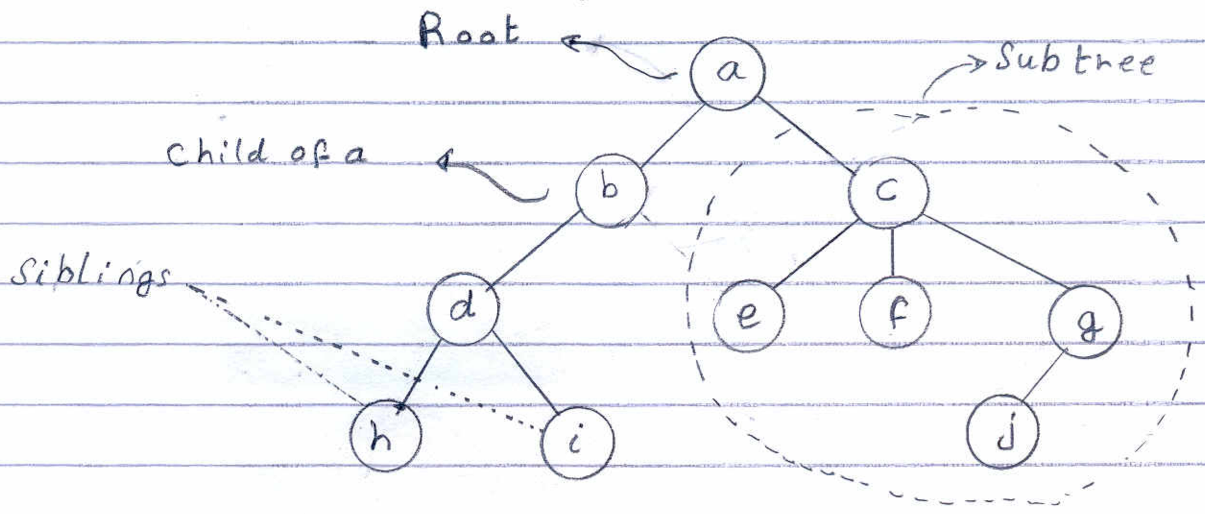
(Binary Tree)



(Binary Tree)

كل نود تكون متصلة فقط بـ ٢ على الأكثر ، أي يمكن أن تتصل بواحدة فقط
أو لا تتصل على الإطلاق

Terminologies



Root : is the node at the heghest level of the hieracy.

Child : is an element that has a superior element.
(b is child of a)

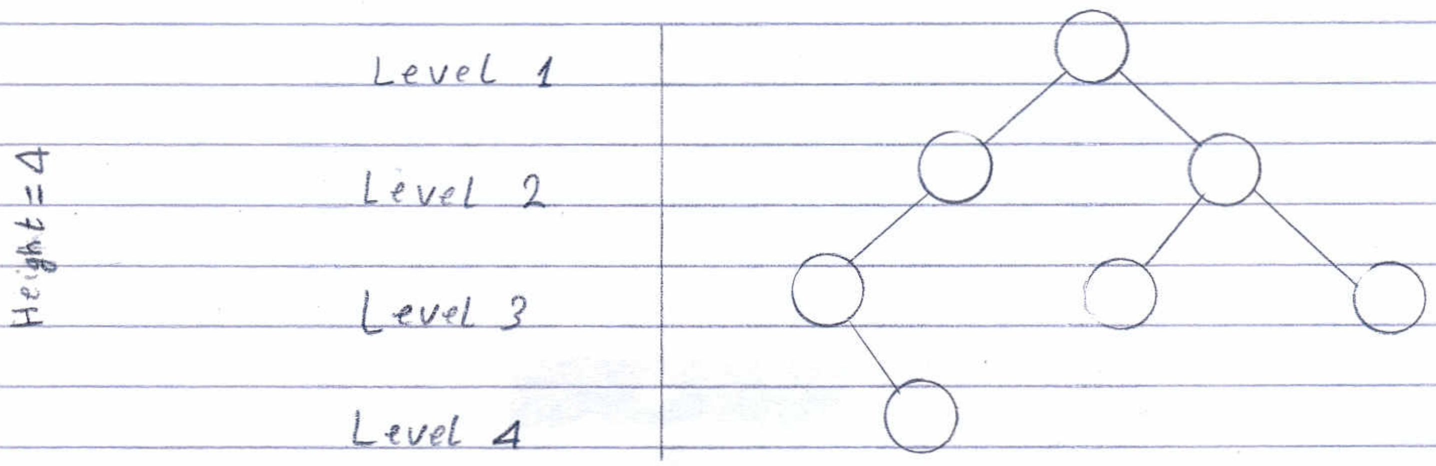
parent : is an element that has a subordinate element.
(g is Parent of j)

Siblings : are child elements with the same parent
(h , i are siblings cause they share the same parent d)

Descendants : الأحفاد
(Descendants of C are e , f , g , j)

Ancestors : الأجداد
(Ancestors of d are b , a)

Sub tree : is a smaller tree that is contained within the original tree.

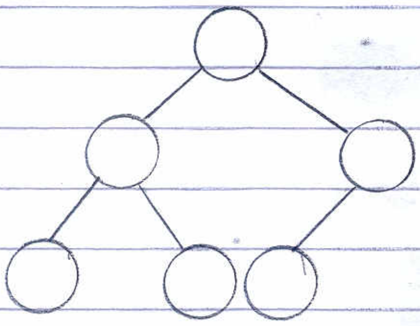


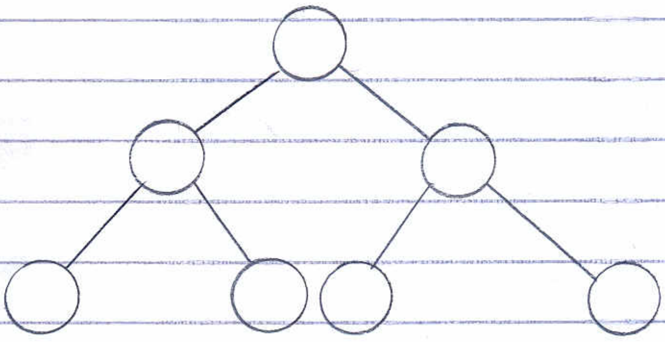
Height = maximum Level

Maximum number of nodes in a Level = $2^{Level - 1}$

The full binary tree

It's full binary tree if and only if the number of nodes in level equal to $2^{level - 1}$


(Binary tree but not full)


(full binary tree)

Full number of nodes = $2^{height} - 1$

↪ من قانون المتوالية العددية

Height = $\log_2 (n + 1)$

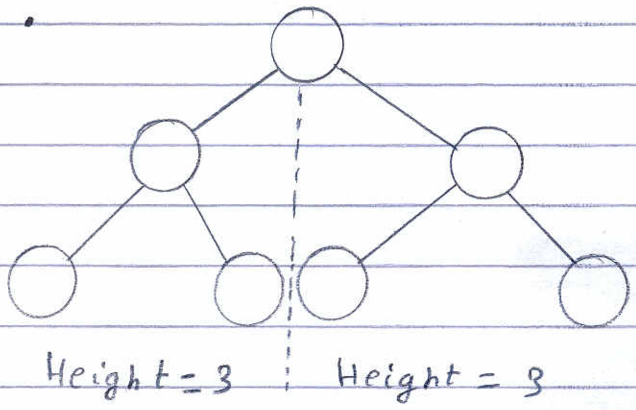
↪ Number of nodes

$n = 2^h - 1$

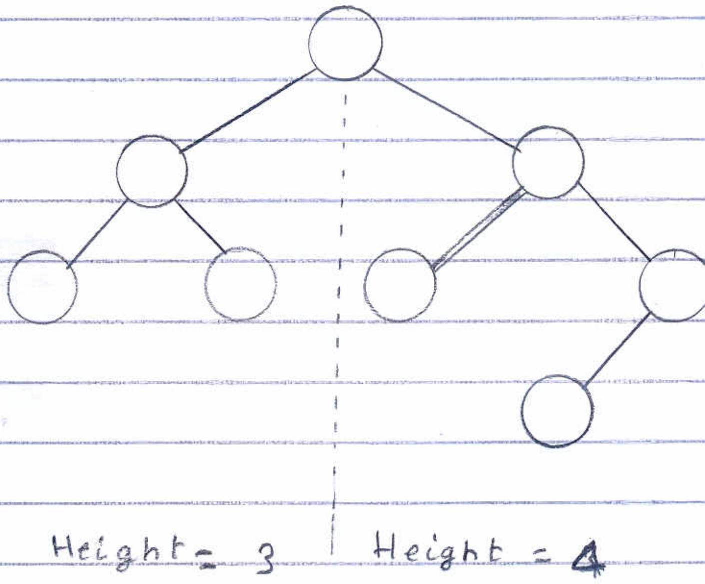
$2^h = (n + 1)$

$h = \log_2 (n+1)$

The balanced tree



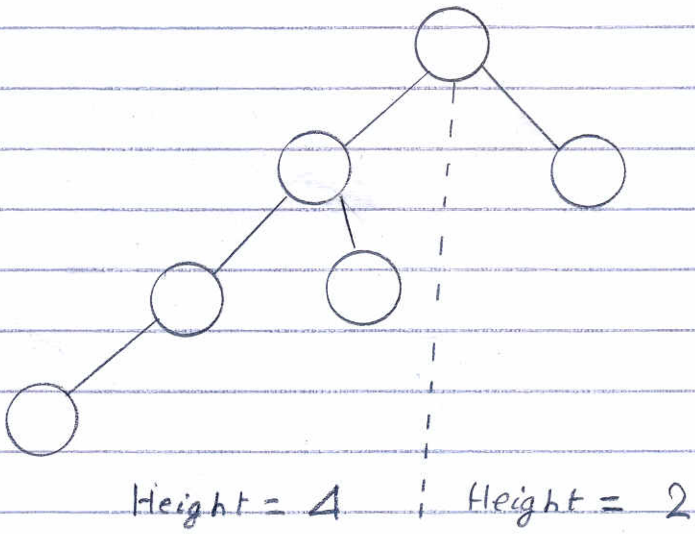
(Balanced tree)



(Balanced tree)

Balanced tree: is aproperty that the heights of the left and right Subtrees differ at most by one level. $|H_{left} - H_{Right}| \leq 1$ → Height

So full tree, complete tree are also balanced tree.

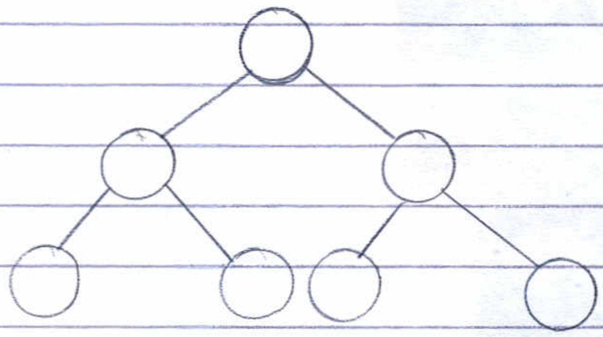


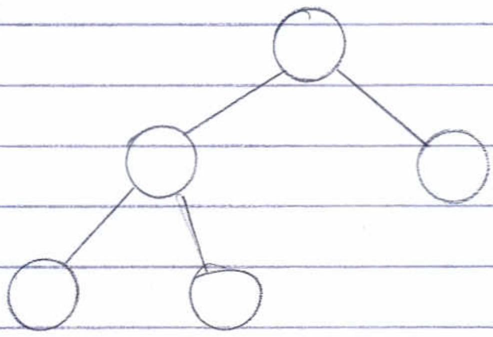
(Not balanced tree)

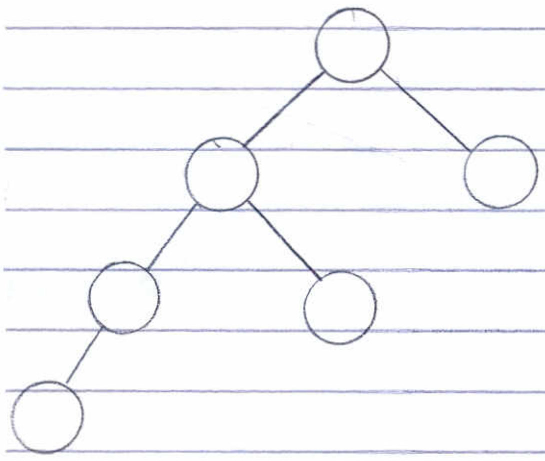
Complete Binary Tree

- It's abinary tree in which each level of the tree is completely filled

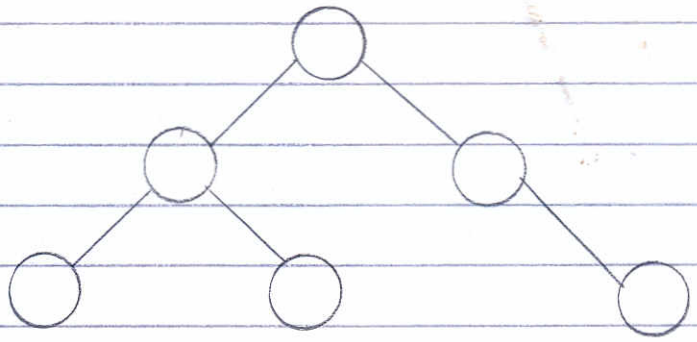
- But the bottom level could be incomplete, but the nodes should occupy from the leftmost positions.


(Binary tree)


(Binary tree)


(Not binary tree)

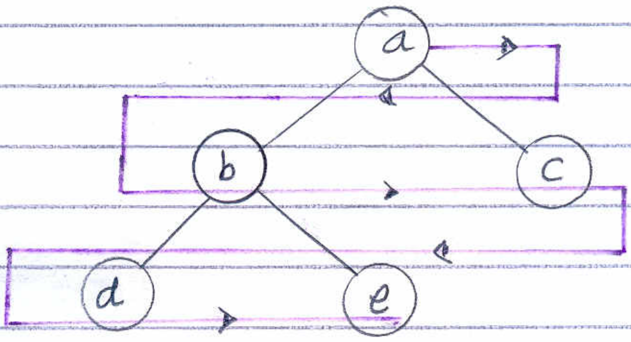
cause the height of the left sub tree is 4. and the right subtree is 2, so the difference is 2


(Not abinary tree)

*cause anode is missing ~~at~~ the bottom level, and there's anode after it.

Implementing a Complete binary tree as an array:

Level order



a	b	c	d	e

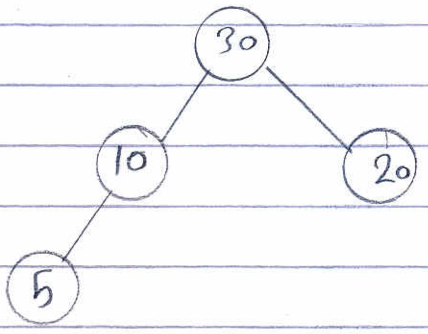
Array

Assuming that values are stored starting at subscript 0 then:

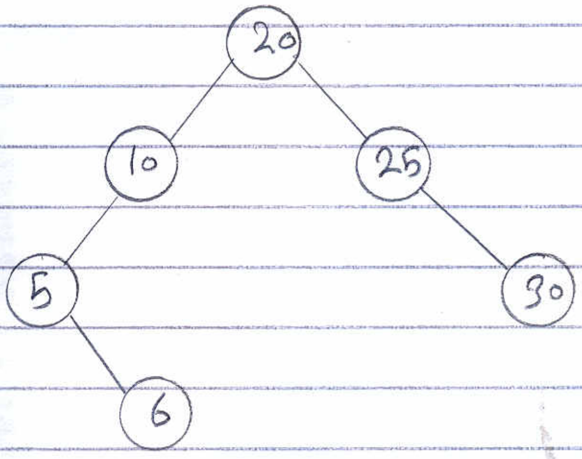
- Root of the complete binary tree is located at [0]
- Left child is located at [2 * parent index + 1]
- Right child is located at [2 * parent index + 2]
- Parent is located at either [child's index -1 / 2]

Binary Search tree

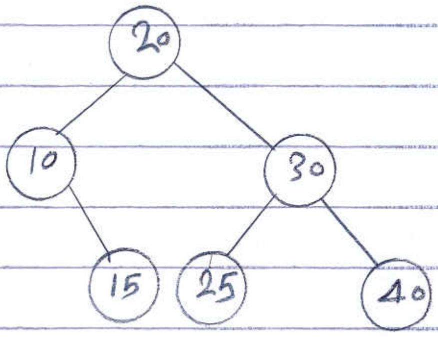
- It's a binary tree that stores key - data in its nodes
- No key duplication (each element is distinct)
- The elements in the left sub tree of the root are less than the root
- The elements in the right sub tree of the root are greater than the root
- The left and right subtrees of the root are also binary search trees



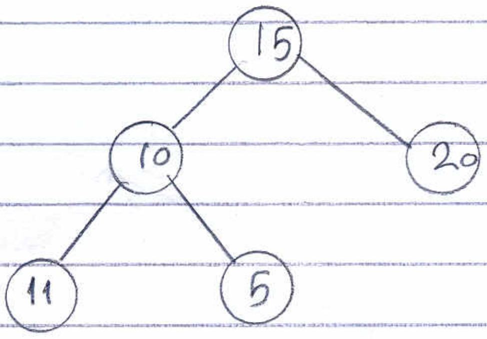
(Not Binary search tree)
cause 20 is less than 30....



(Binary search tree)



(Binary search tree)



(Not Binary search tree)

Tree Traversal

Pre-Order (Depth-first traversal)

Root - Left - Right

In - Order

Left - Root - Right

Post _order

Left - Right - Root

Level - order (Breadth - first traversal)

Top Level to Bottom starting from left to node to right.

Reverse - pre-order

Root - Right - Left

Reverse In - order

Right - Root - Left

Reverse post - order

Right - Left - Root