Unit-III

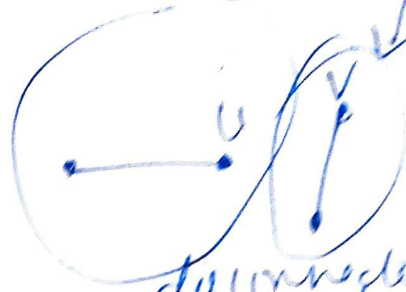
Tree - It is a kind of graph which is ~~connect~~ connected and acyclic.

Connected graph - There exists a path between any 2 vertices of a graph.

Components

Ex

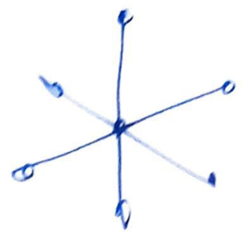
Connected graph

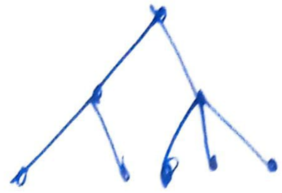


disconnected graph with 2 components (we can not go from u to v)

Acyclic graph - graph without cycle.

ex. of Tree:





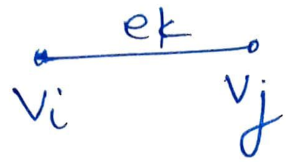
Thm A Tree with p vertices has $(p-1)$ edges

Pf. Induction on p.

Thm is True for $n=1,2,3,$ ~~4~~

Assume the Thm is True for $p-1$ vertices.

~~✱~~ Consider a tree T with p vertices in T,
Consider e_k an edge with v_i & v_j as end-vertices



According to Thm. (there exists only one path between v_i ~~& v_j and that is e_k~~ any 2 vertices in a tree), there is only one path between v_i & v_j & that is e_k.

$\Rightarrow$ deletion of e_k will disconnect the graph
$T-e_k$ has exactly 2 components & each is a Tree. Both the trees contain one edge less than the no of vertices

$\therefore$ $T-e_k$ has $p-2$ edges & p vertices

$\therefore$ T has exactly $p-1$ edges

<u>Thm</u> G is a Tree $\Leftrightarrow$ every 2 distinct vertices of G are connected by a unique path of G.

<u>Pf.</u> G is a Tree $\Rightarrow$ G is connected & acyclic

$\Rightarrow$ every 2 vertices are joined by at least 1 path

Suppose if u & v are joined by 2 different paths $\Rightarrow \exists$ a cycle

— Contradiction ($\because$ G is a Tree)

$\therefore \exists$ a unique path between any 2 vertices in G

<u>Converse</u> In G, every 2 distinct vertices of G are connected by a unique path of G

$\Rightarrow$ G is Connected

If G has a cycle containing vertices u & v then $\exists$ at-least 2 different paths

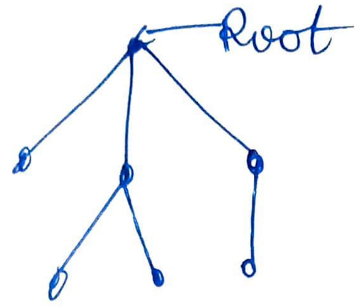
$\Rightarrow$ Contradiction ($\because \exists$ unique path)

$\Rightarrow$ G is acyclic

$\Rightarrow$ G is Tree

HP.

Rooted Tree - A fixed vertex called Root & all the vertices & edges below it.



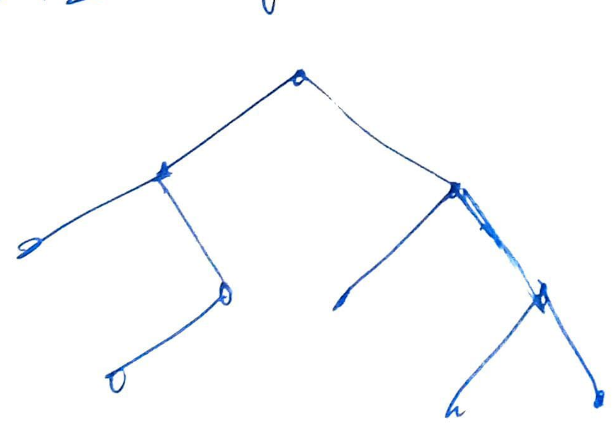
Binary Tree - A Binary Tree is defined as a finite set of element called nodes (vertices) set.

(1) T is empty (Called Null Tree / Empty Tree)

OR

(2) T contains a distinguished vertex R called Root of T & remaining nodes (vertices) of T form an ordered pair of disjoint binary Trees T_1 & T_2.

T_1 & T_2 : left & right subtrees of R.



Binary Tree

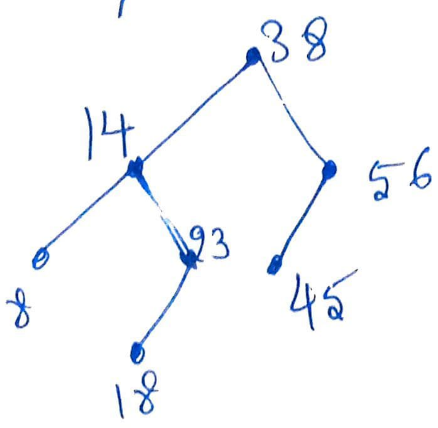
Binary Search Tree - A Binary tree T is called a Binary Search Tree if each node N of T has the following property: The value of N > every value in the left subtree of N and is < every value in the right subtree of N.

When duplicate values are allowed then each node N has the following properties:

1) N > M for every node M in a left subtree of N.

2) N ≤ M for every node M in a right subtree of N.

ex Find / insert 20 in given Binary Tree T.



ITEM = 20

Sol.

1. Compare ITEM = 20 with Root R = 38

   20 < 38 - go to left child of 38 which is (node) M.

2. Compare 20 with 14

   20 > 14 - go to Right child of 14, which is 23

3. Compare 20 with 23.

   20 < 23 - go to left child of 23 which is 18

4. Compare 20 with 18

   20 > 18 & 18 has no right child

   So insert 20 as right child of 18.