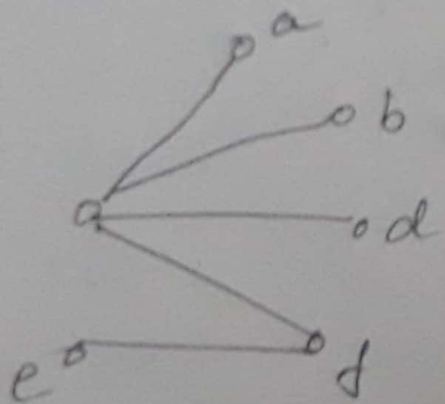
Trees

Tree doesnot have circuits.

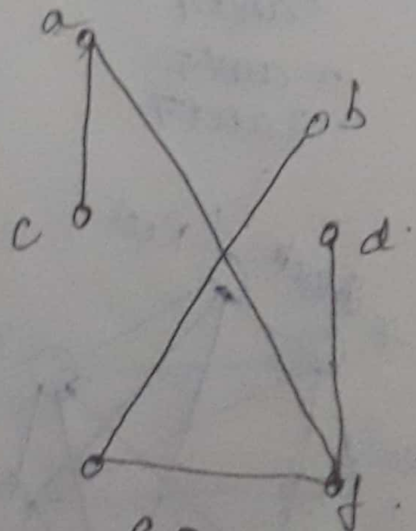
A connected graph which does not have a circuit.

- no multiple edges
- no self loops.
- no circuits.
- undirected.

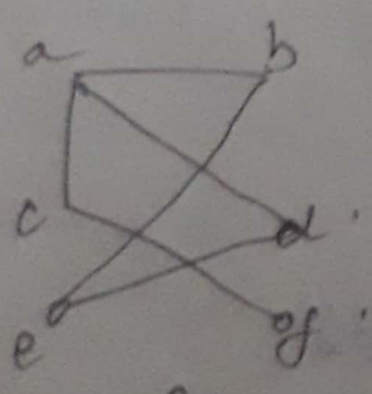
connected graph is when you start from one vertex you should be able to connect all verticies



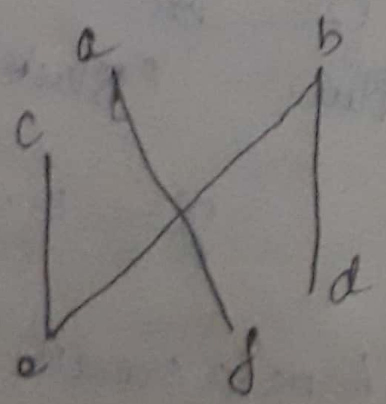
G1 Tree



G2 Tree



G3
It has circuit.
not tree



G4. It is not connected.
not tree

Root :- The vertex which has children is called root

The tree which hav root and which is directed is called rooted tree.

The direction is give away from the root

Terminologies.

→ Suppose that T is a rooted tree, If a vertex in T other than the root, the parent of v is unique vertex u such that there is a directed edge from u to v. When ~~you~~ u is the parent of v, v is called child of u.

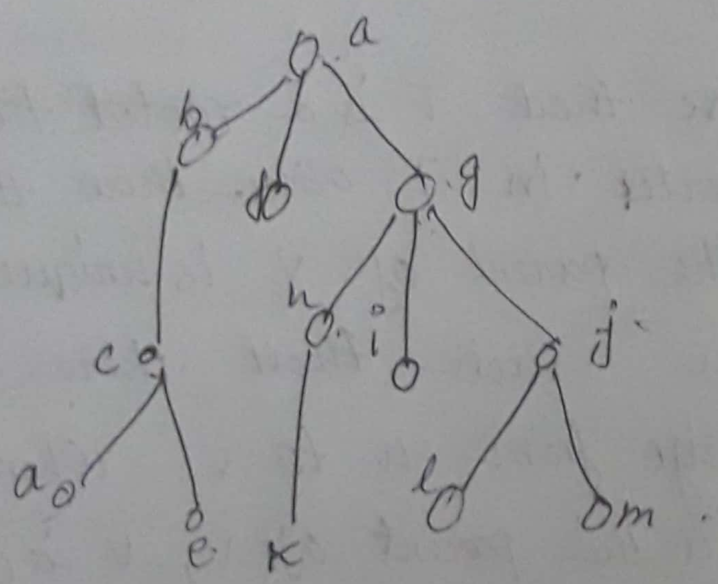
Vertices with the same parents are called siblings.

ancestors of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root

The descendant's of a vertex ~~to~~ v are the vertices that have v as an ancestor

→ A vertex of a tree is called leaf if it have no children.

→ Vertices that have children are called internal vertices.



→ internal vertices are - a, b, c, g, h, j

i) In the rooted tree T, i) find the parent of c, ii) the children of g, iii) The siblings of "h", iv) all ancestors of e, all decendants of b, all internal vertices and all leafs.

Sol:- i) b

ii) h, i, j

iii) i, j

iv) c, b, a

v) c, d, e.

vi) a, b, c, g, h j

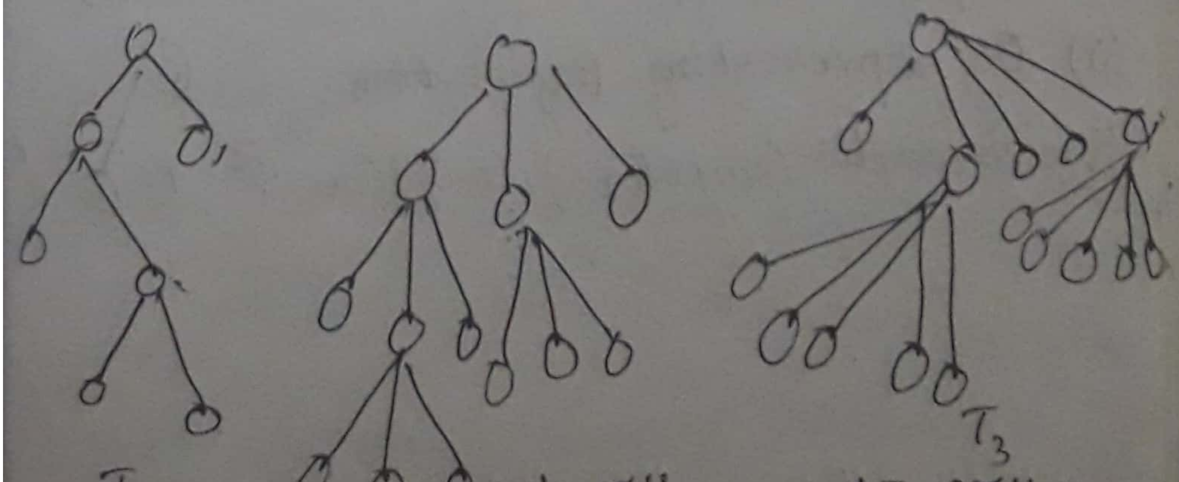
vii) d, e, f, k, i, l, m.

M-ary Tree

A rooted tree is called M-ary Tree if every internal vertex has no more than m children.

The Tree is called a full M-ary Tree if every internal vertex has exactly m children

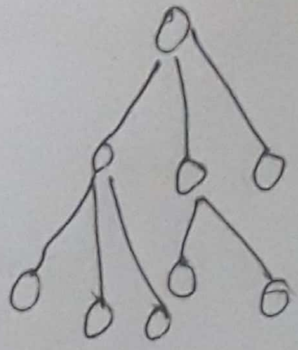
An m-ary tree m = 2 is called Binary Tree



T_1 full 2-ary Tree

full 3-ary T_2

T_3 full 5-ary.



T_4

~~not~~ m-ary tree.

· 3-ary tree but not full.

Ordered rooted treee.

When it has two children and roots are in a order. The ordered vertices are called left child and right child.

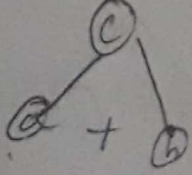
Applications of Trees (or) Trees as model

i) Saturated Hydrocarbons and trees.

ii) Compilers design → Ex:- $c = a+b$.

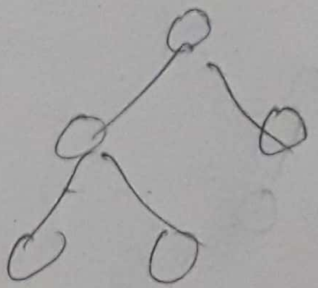
iii) ~~os~~ representing organizations.

iv) ~~drives~~dis Computer file systems.

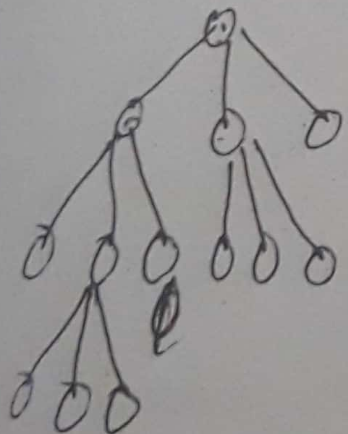


Properties of trees.

Theorem 1: A tree with n - vertices has n-1 edges.



Theorem 2:- A full m-ary tree with "i" internal vertices contains $n = mi+1$ vertices.



$m = 3$

full m-ary tree

full 3-ary tree

$i = 4$

$n = 4(3)+1$

$= 13$.

Theorem 3: A full m-ary tree with

1) with n vertices has $i = (n-1)/m$ internal vertic

leaves $l = [(m-1)n+1]/m$

ii) with i internal vertices.

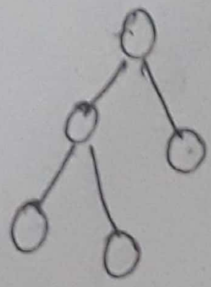
vertices $n = mi+1$

leaves $l = (m-1)i+1$

3) with leaves ~~has~~ l ~~~~

then $n = (ml - 1)/(m - 1)$

$i = (l - 1)/(m - 1)$



i) $n = 5$

$i = (5-1)/2$

$= 4/2 = 2$

$l = [(2-1)5+1]/2$

$= 6/3 = 3$

ii) $i = 2$

$n = 2(2)+1$

$= 5$

$l = (2-1)2+1$

$= 2+1$

$= 3$

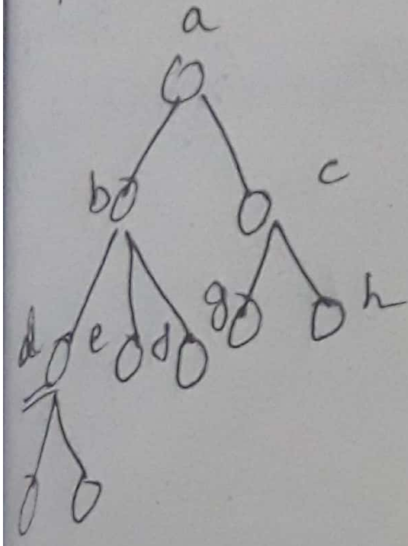
iii) $l = 3$

$n = (2(3)-1)/(2-1)$

$= 5$

$i = (3-1)/(2-1) = 2$

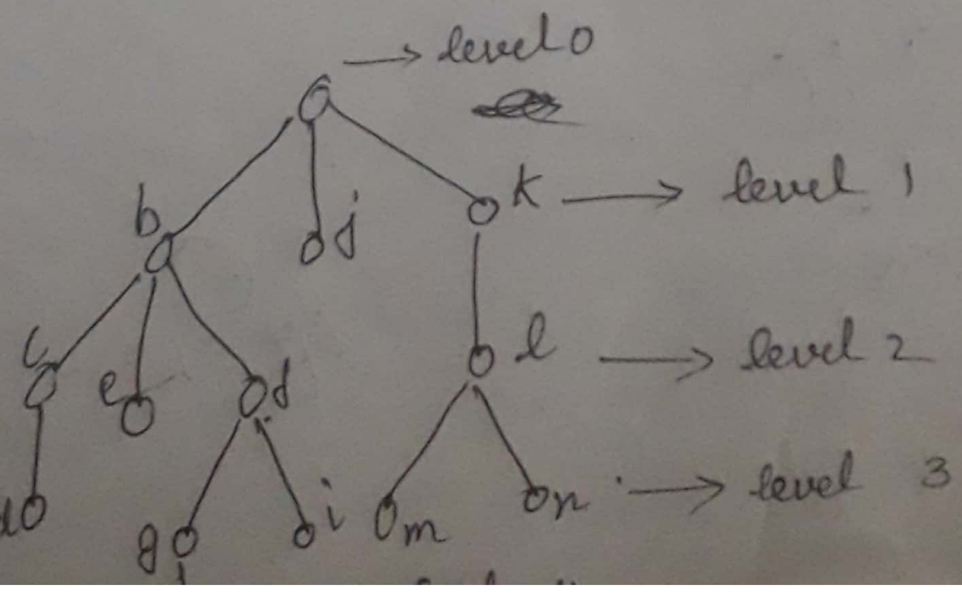
Level :- The level of a vertex in a rooted tree is length of the unique path from the root to this vertex.



level of d is 2.

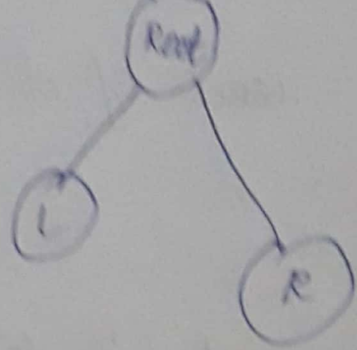
The level of the root is defined to be 0.

The height of rooted tree is the maximum of the levels of vertices. In order words, the height of rooted is length of the longest path from the root to the vertex.



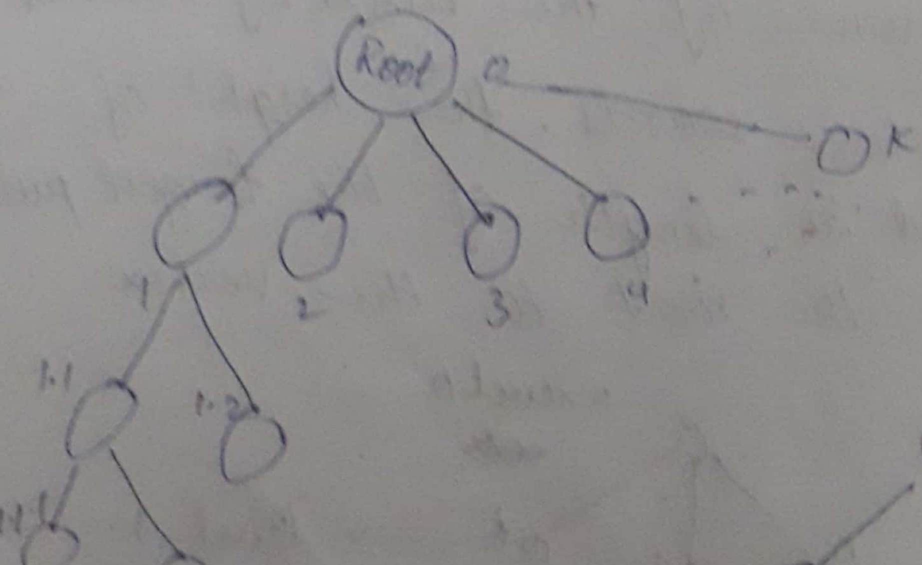
Tree Traversal

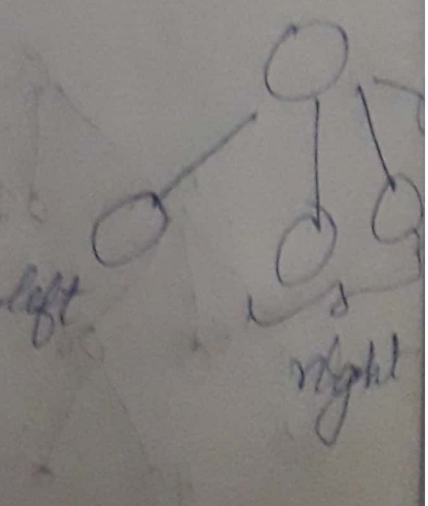
Ordered rooted tree : It a tree which has root and its children (right, left)



It is always left to right.

Universal address System





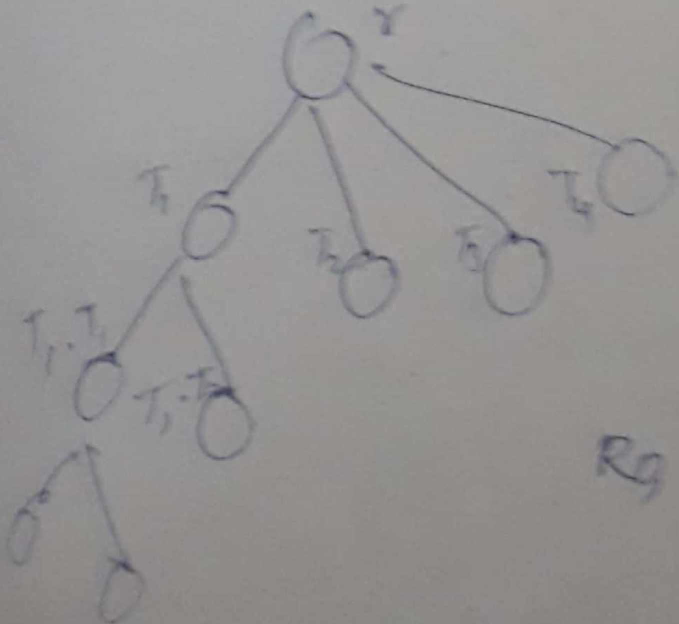
Traversal algorithm

→ procedure to visit all vertices of a tree is called traversal algorithm

most common are
Three types.

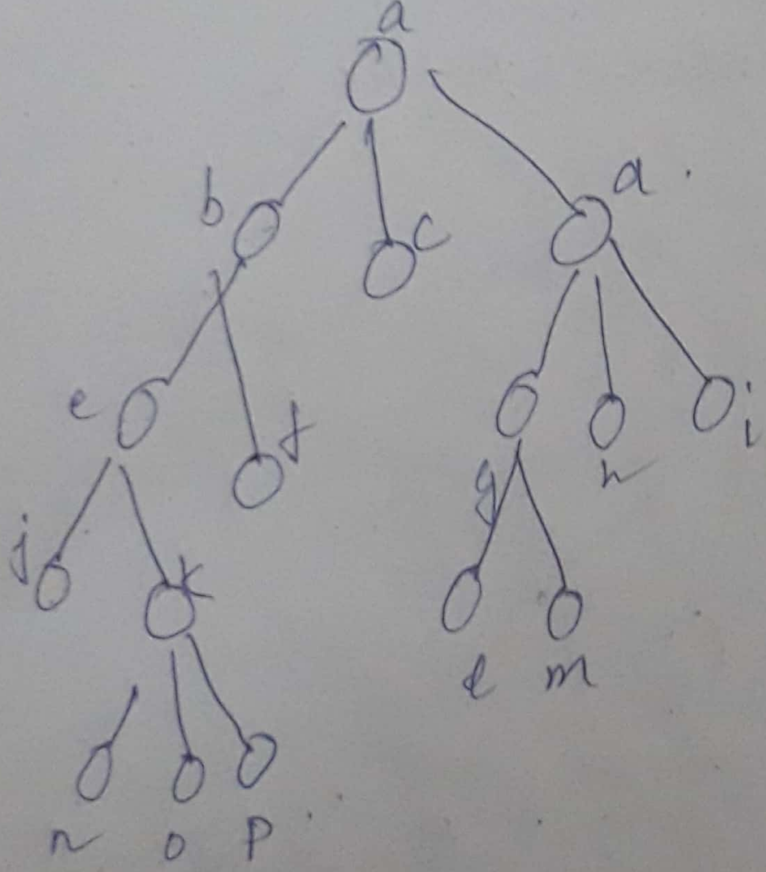
i) pre-order ⟶ Ro, L, R
ii) In-order L, Ro, R
iii) post-order L, R, Ro

→ preorder Traversal



Ro, L, R.

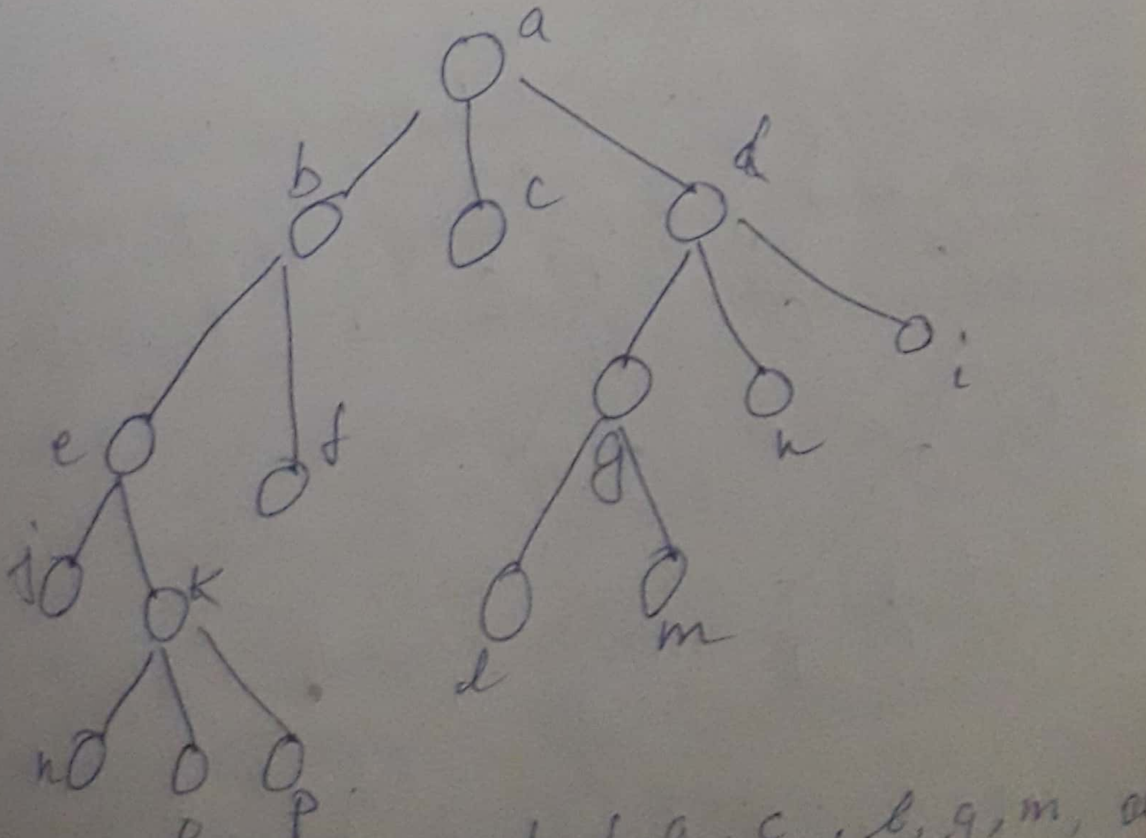
pre-order traversal :- let T be a ordered rooted tree with root r. If T consists only of r, then r is the pre-order traversal of T. Otherwise, suppose that $T_1, T_2 \cdots T_n$ are the sub trees at r from left to right in T. Pre order traversal begins by visiting r. It continues by traversing T_1 in pre-order, then T_2 in pre-order and so on until T_n is traversed in T.



a, b, e, j, k, n, o, p, f, c, a, g

~~h, i~~ l, m, h, i

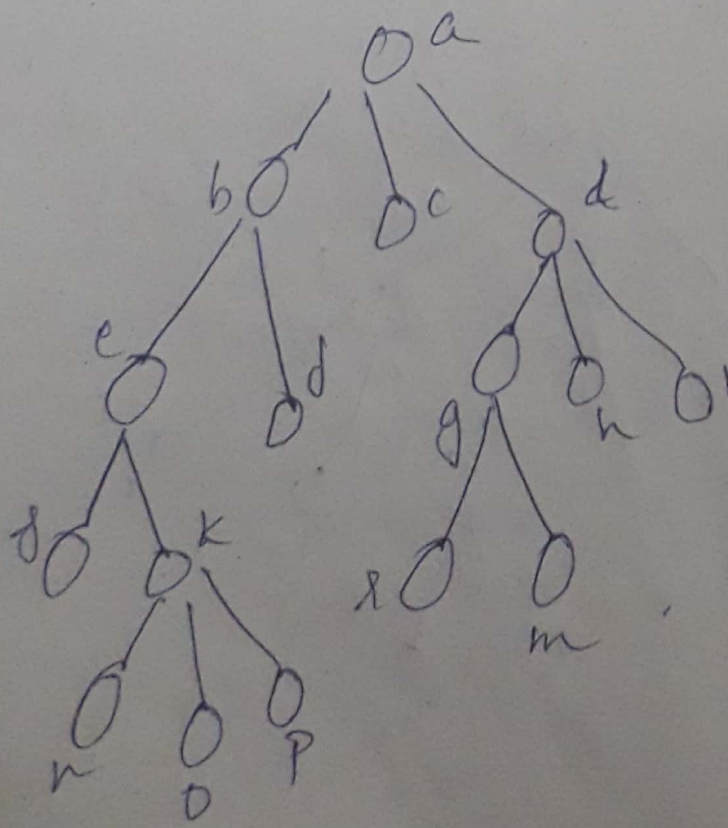
In-order :- Let T be an ordered rooted tree with root r. If T consists only of r then r is the In-order of traversal of T. Otherwise, suppose that $T_1, T_2 \ldots T_n$ are the sub trees at r from left to right. The In-order traversal begins by traversaling T_1 in In-order then visting r it continuous by traversing T_2 in In-order, then T_3 in In-order and finally T_n in inorder.



j, e, n, k, o, p, b, f, a, c, l, g, m, d, h, i

~~j, e, n, k, o,~~ p

Post-order traversal:- Let T be an ordered rooted tree with root r. If T consist only of r, then r is the post-order traversal of T. otherwise, suppose that $T_1, T_2, \ldots T_n$ are the sub trees at r from left to right. The post-order traversal begins by traversing T_1 in post order, T_2 in postorder then T_n in postorder and ends by visting the root.



j, n, o, p k, e, j, b, c, l, m, g, h, i, d, a

Infix, prefix and postfix notations

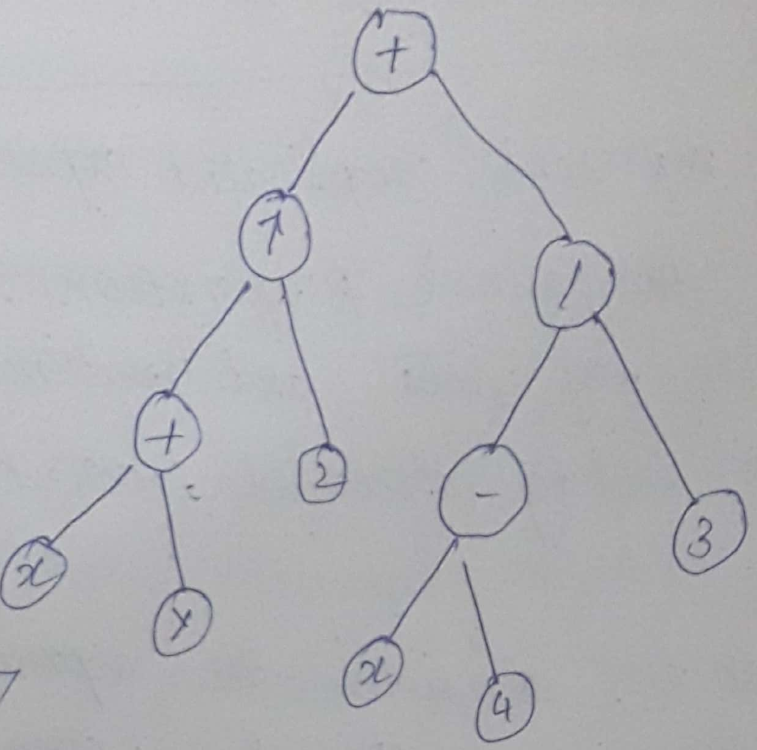
we can represent complicated expressions such as compound proportions, combination of sets and arithmetic expressions using ordered rooted trees.

for instance, consider the representation of a arithmetic operation ($+, -, *, /,$ and $\uparrow$) we use parathesis to indicate the order of operations.

An ordered rooted tree, can be used to represent such expressions where the, internal vertices represents the operations and the leaves represents variables or numbers. each operation operates on its left and right subtrees.

① what is the ordered rooted tree that represent the expression
$((x+y)\uparrow 2) + ((x-4)/3)$

Scanned with CamScanner



(2) What is the prefix form for this expression $((x+y)\uparrow 2)+((x-4)/3)$

$+ \uparrow + x\ y\ 2\ /\ -\ x\ 4\ 3$

In the prefix form of an expression a binary operator such as + preceeds its 2 operands. Hence we can evaluate an expression in prefix form right to left

(3) What is the value of prefix expressi

$+ - * 2\ 3\ 5\ / \uparrow 2\ 3\ 4.$

$/ 2^3\ 4$

$(2*3)\ 5$

$(2*3) - 5 + 2^3/4$ 2

$6 - 5$

$1 + 3 = 3$

step1 :- $/(2 \uparrow 3)\ 4$

$/(8)\ 4$

$\Rightarrow 8/4$

$= 2.$

step 2 :- $-(2 * 3)\ 5$

$-(6)\ 5$

$6 - 5$

$= 1$

$+\ 1\ 2.$

step3 : $1 + 2$

$= 3.$

What is the post fix form of the expression $((x+y)\uparrow 2) + ((x-4)/3)$

sol:-

$x, y, +\ 2, \uparrow\ x, 4\ -\ 3\ /\ +$

left to right.

What is the value for the post fix expression $7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +.$

~~$7\ (2*3)$~~

$7 - 6$ 1 $9/3$, $3 + 4$ 7

7 2 3 * − 4 ↑ 9 3 / + .

7 (2*3) −

7 (6) −

7 − 6

= 1 4 ↑ 9 3 / + .

1 ↑ 4

= 1 9 3 / + .

= 1 (9/3) + .

= 1 3 +

= 1 + 3 .

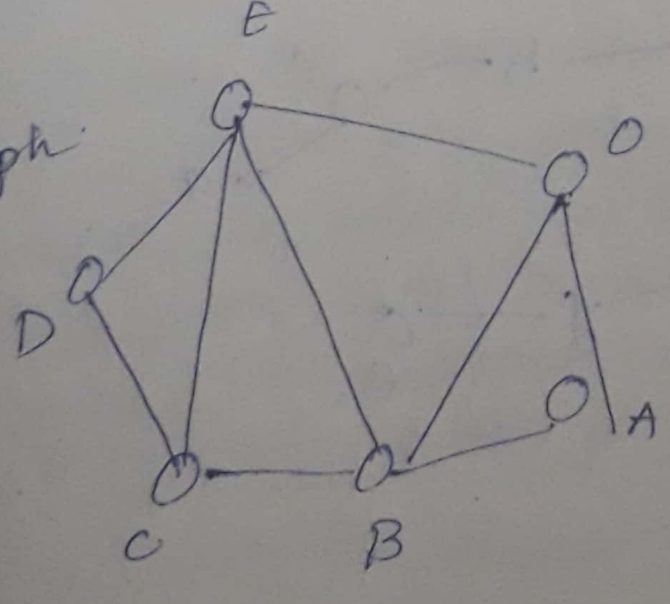
= 4

Spanning trees

myself
If n vertices in a Graph, then there should be n vertices in a tree and n-1 edges in tree (It should be connected tree)

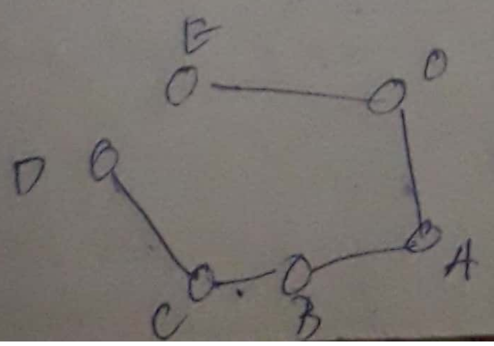
x mam
Let G be a simple graph. A spanning tree of G is a sub-graph of G {i.e is a tree containing very vertex of G.

Graph



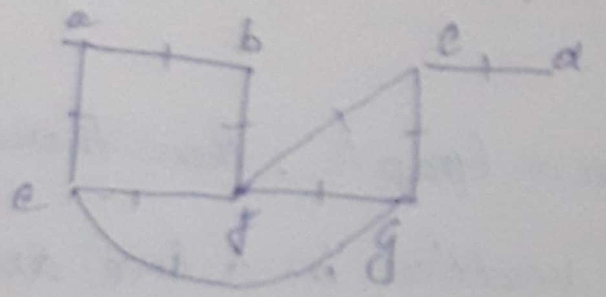
$V = 6$
$E - 9$

(G)



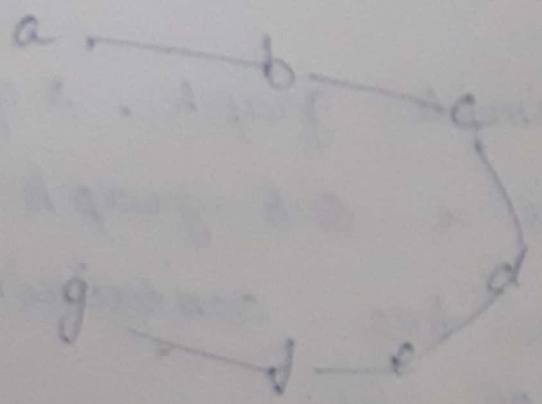
$V = 6$
$E = 5$

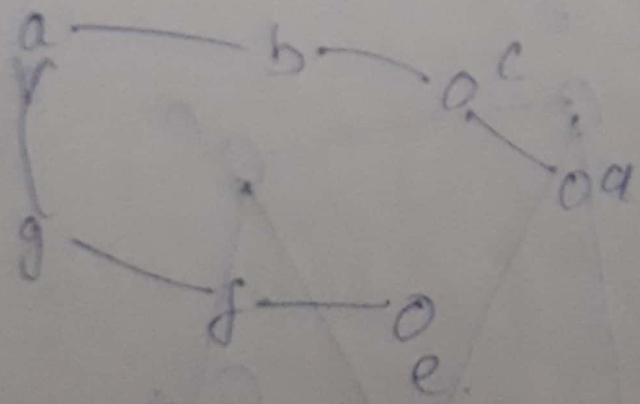
→ spanning tree



(G)

$V = 7$
$E = 9.$





depth first search

Instead constructing spanning trees by removing edges, spanning trees can be built up by adding edges.

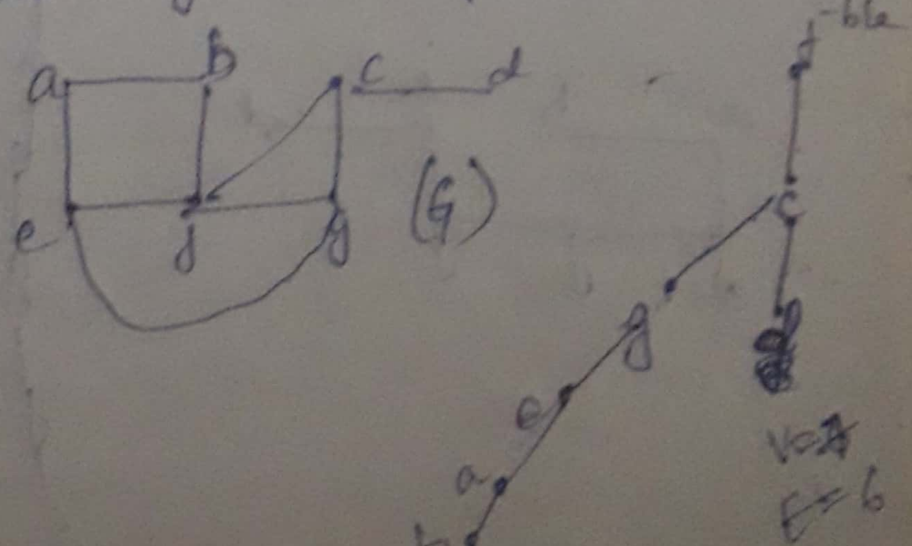
Two alogrithm based on this principal are

i) DFS
ii) BFS.

we can buit a spanning tree for a connected simple graph using DFS

· Arbitrarilly choose a vertex of the graph as a root. Form a path starting at this vertex by successively adding vertices and edges where each new edge is incident with last vertex in the path and a vertex not already in the path. continue adding vertices and edges to this path as long as possible

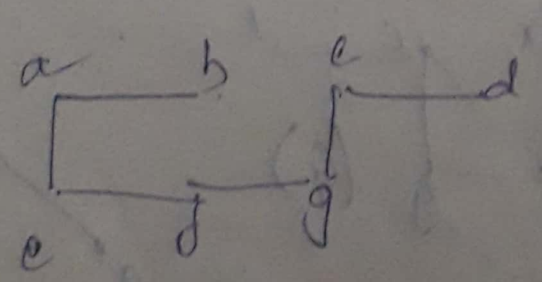
EX:-



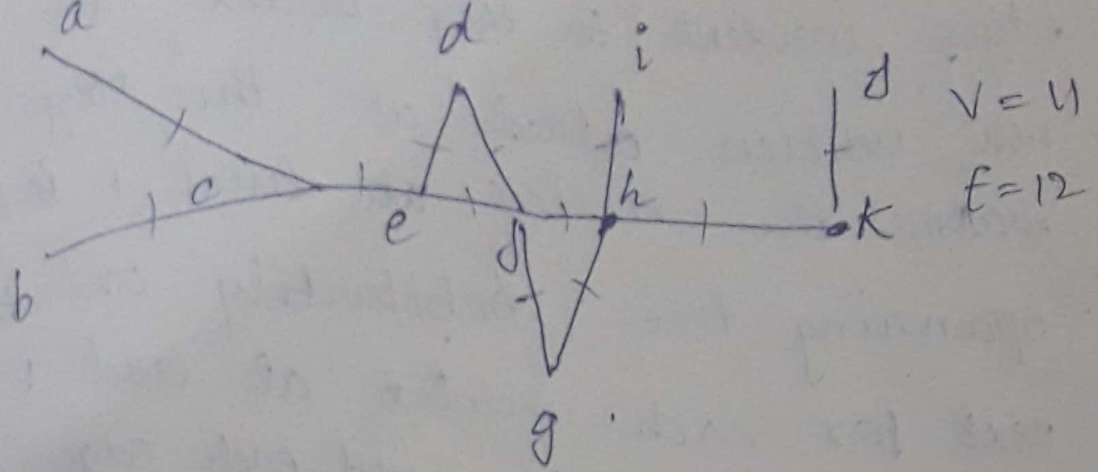
If the path goes through all vertices of graph. The tree consisting of this path, a spanning tree. However if the path doesnot go through all vertices, more vertices and edges may be added.

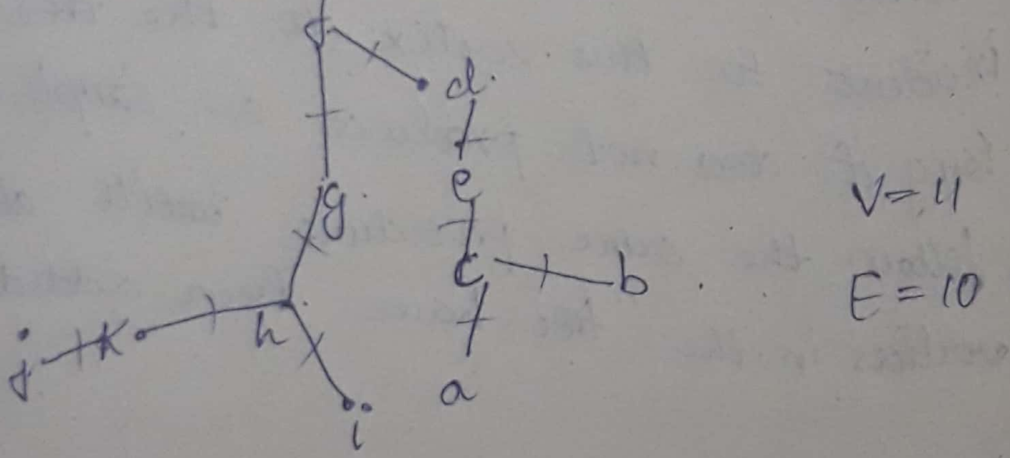
move back to the next to the last vertex in the path and if possible from a new path starting at this vertex passing through vertices that where not already visted.

DFS is also called back tracking because the algorithm returns to vertices perviously visited to add paths.



1) Use DFS to find spanning tree for the Graph G :





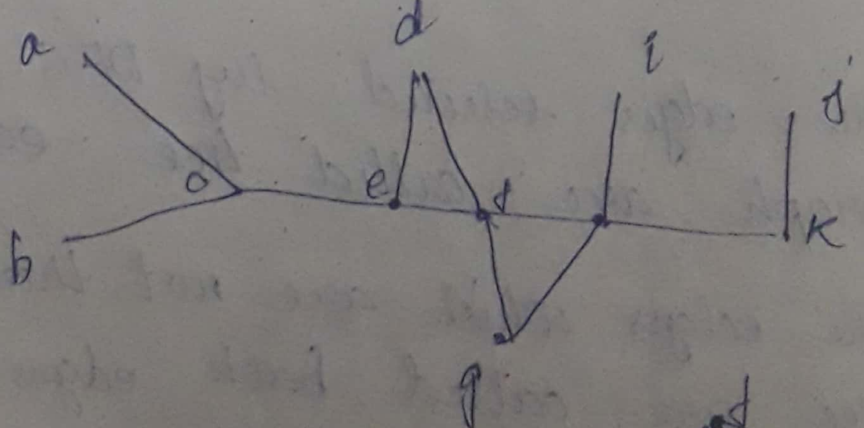
The edges selected by DFS of a graph are called tree - edges

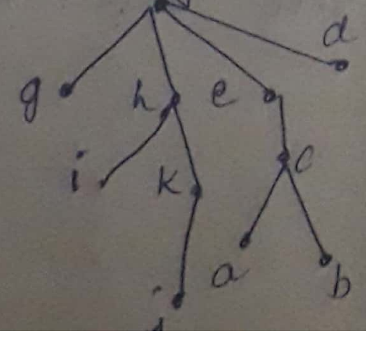
The edges which are not there in a tree are called back edges.

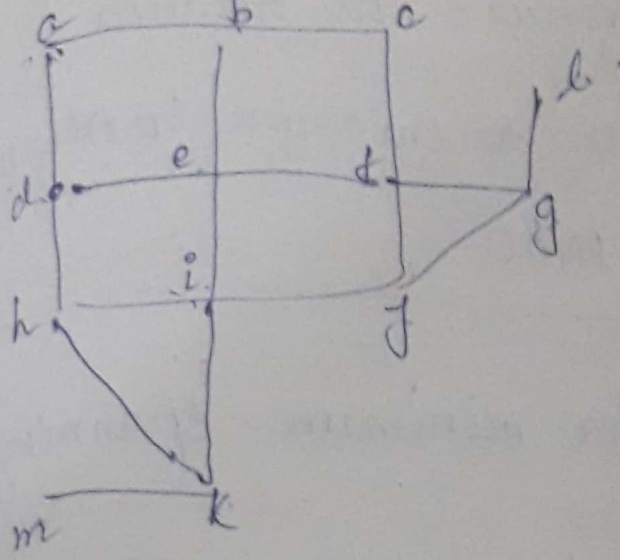
BFS

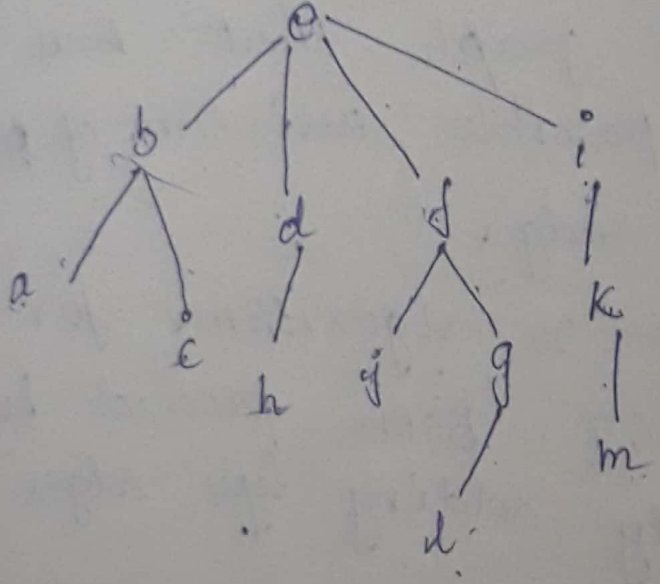
Arbitartily choose a root from the vertices of the graph then add all edges incident to this vertex. The new vertices added at this stage become the vertices at level 1 in the spanning tree. Arbitartily order them next for each vertex at level 1 visited in order add each edge incident to this vertex to the tree as long it does not produce a simple circuit follow the same procedure untill all the vertices in the tree have been added.

Ex:-



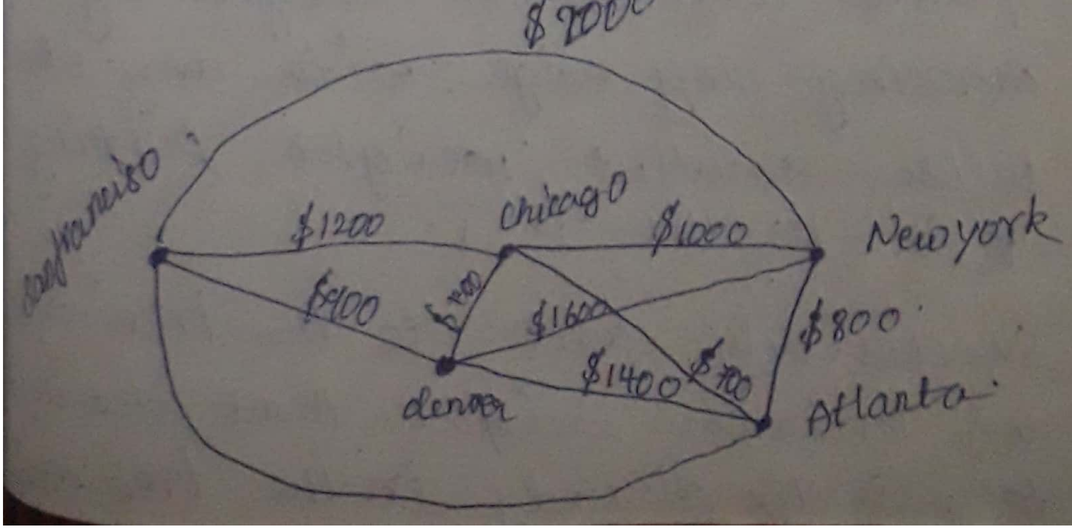






V= 13
E =12

Mininum Spanning tree / Mininum ~~host~~ cost S.T
(MST)



MST of sanfrancisco to Atlanta:

sanfrancisco to chicago to Atlanta

$ 1900

Algorithms for minimum spanning tree

A min spanning tree in connected weighted graph that has the smallest possible ~~wei~~ sum of weights on the edges.

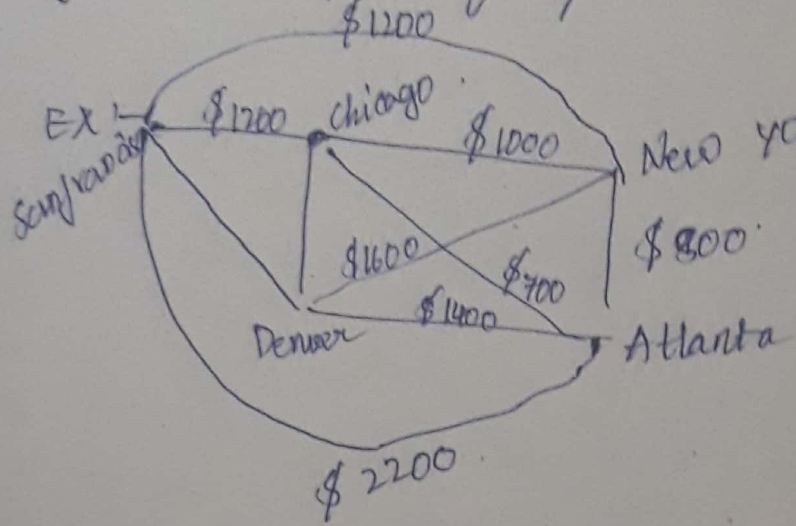
There are 2 algorithms for constructing MST. Both proceed by successively adding ~~fo~~ edges of smallest weight from those edges with the specified property that have not already been used.

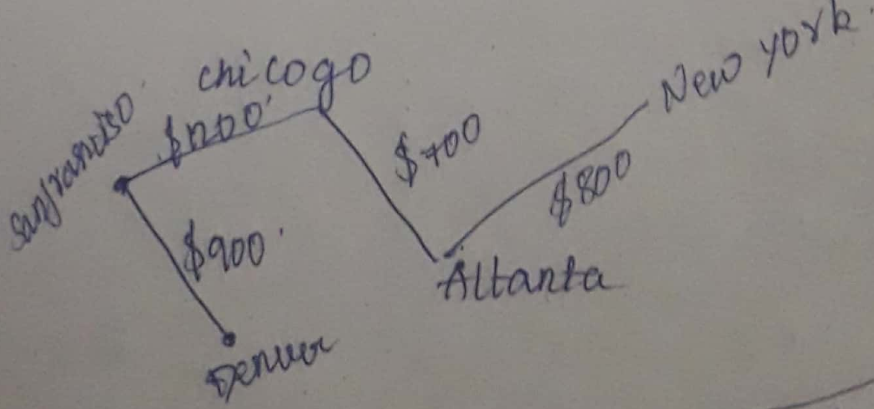
i) prince algorithm begin by choosing any edge with any edge with smallest weight putting it in to the S.T.

Successively add to the tree edges of minimum weight that are incident to a vertex already in the tree and not forming a simple circuit with those edges already in the tree. Stop when (n-1) edges have been added.

Note:- There may be more than one spanning tree for a given connected weighted simple graph.

Ex:-





$$\begin{array}{r} 1200 \\ +\ 800 \\ 900 \\ 700 \\ \hline 3600 \end{array}$$

chicogo to newyork weight is less but it closes the circuit so we should not take

choice	Edge	cost.
1.	{chicago, Atlanta}	$700
2.	{Atlanta, new york}	$800
3.	{chicago, Sanfrancisco}	$1200
4.	{Sanfranciso, Denver}	$900
		$3600