-: Trees and Fundamental Circuits :-

→ **Definition:-** A Connected Graph without any Circuit is called a tree.

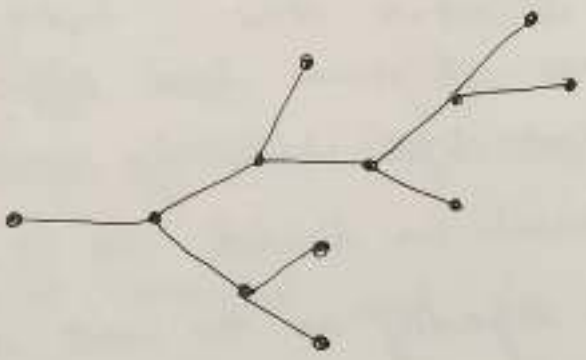


Fig 1. An example of a tree.

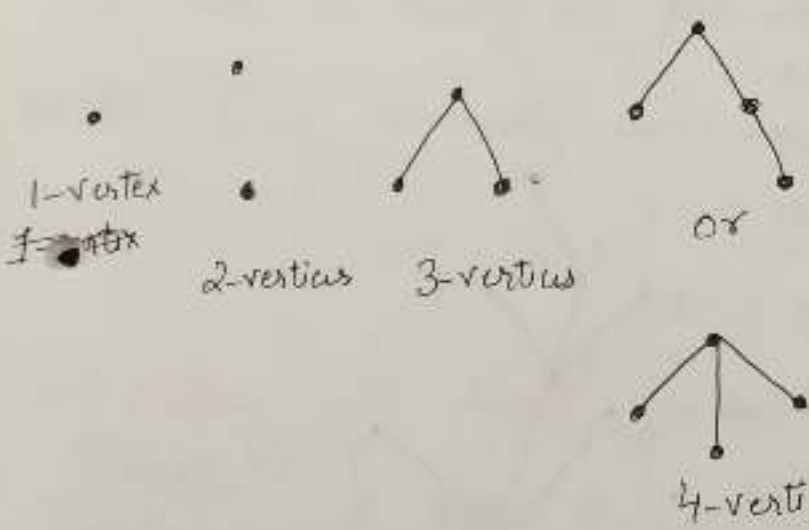


Fig 2. A Tree with one, two, three and four vertices.

→ **Note :-** Graph containing self-loops or parallel edges cannot be a tree. ∵ it forms a Circuit.

→ Examples of a tree :-

- Family tree
- A river with its tributaries and subtributaries
- The sorting of mail according to a tree (called decision tree or sorting tree).
  - All the mail arrives at some local office N. The most significant digit in the zip code is read at N_2 and the mail is divided into 10 piles $N_0, N_1, N_2, \ldots, N_9$ depending on the most significant digit.
  - Each pile is further divided into 10-piles according to the second most significant digit.
  - Repeating this till the mail is subdivided into 105 possible piles, each representing a unique five-digit zip code.

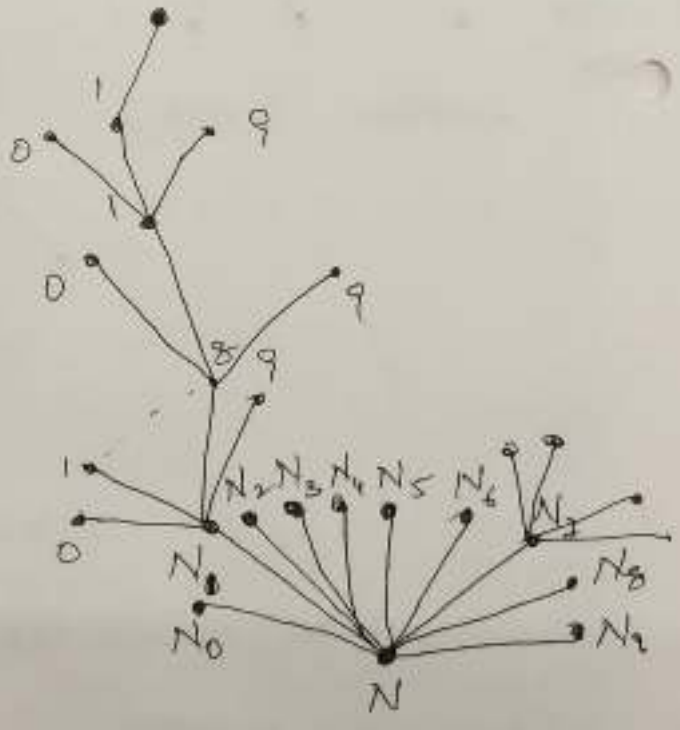


Fig-3 A decision tree for mail sorting.

Some properties of Trees:-

→ In this section we shall prove some properties of trees which can also be taken as alternative definitions for trees. First we list all of them below:-

A graph G with n-vertices is called a tree iff

(1) G is connected and circuitless; or (definition given before)

(2) G is connected and has $(n-1)$ edges, or

(3) G is circuitless and has $(n-1)$ edges, or

(4) there is exactly one path b/w every pair of vertices in G,

or (5) G is minimally connected graph. (we shall define minimally connected graph later).

→ Now we shall prove each of these one-by-one.

(Point 4).

Result 1:- A Graph G is called a tree iff there is one and only one path between every pair of vertices in the graph.

Proof:- **Given**: A Graph G is a tree

To prove:- there is one and only one path b/w every pair of vertices of G.

— Since G is a tree $\Rightarrow$ ~~every pair~~ G is connected
$\Rightarrow$ there is one path b/w every pair of vertices.

— If there are more than one paths b/w some pair (u, v) of vertices $\Rightarrow$ there is a closed path from u, $\Rightarrow$ there is a circuit in the graph $\longrightarrow\leftarrow$ as G is a tree. $\Rightarrow$ there cannot be more than one path b/w every pair of vertices.

Hence the result.

Converse :- Given:- There is one and only one path b/w every pair of vertices in G.

T.P G is a tree.

— Since there is path b/w every pair of vertices $\Rightarrow$ G is connected

— G doesnot have a circuit ∵ If there is a circuit then there is atleast one pair of vertices with two paths $\longrightarrow\leftarrow$ hence G has no circuit
And Hence G is a tree.

Result 2:- A tree with n-vertices has (n-1) edges.

Proof:- We shall prove this using induction on the no. of vertices.

	Tree	No. of edges.
for n=1	•	0 = (1-1)
n=2		1 = (2-1).
n=3		2 = (3-1).

- Let the result be true of no. of vertices less than n.
- Let G be a tree with n- vertices.
  - Since G is connected $\exists$ atleast one pair of vertices v_i, v_j connected by edge say e_k.
  - Also there is no other path b/w v_i & v_j.
- Remove e_k from G. This makes $G - \{e_k\}$ a disconnected graph. And we can divide this # into exactly ~~two~~ two components, say G_1 & G_2.
- Both G_1 & G_2 has no. of vertices less

than n. Also G_1 & G_2 will have no circuit as G was a tree.

Let no. of vertices in $G_1 = n_1 < n$

" " " " $G_2 = n_2 < n$

We have $n_1 + n_2 = n$.

by induction hypothesis no. of edges in $G_1 = n_1 - 1$, and no. of edges in G_2 is $n_2 - 1$

$\Rightarrow$ No. of edges in $G - \{e_k\} = G_1 \cup G_2$

$= n_1 + n_2 - 2$

$= n - 2$

$\Rightarrow$ No. of edges in $G = n - 2 + 1 = \underline{\underline{n-1}}$.

Result-3:- Any connected graph with n vertices and $(n-1)$ edges is a tree.

Proof:- Let G be not a tree.

$\#\#$ ie. G has circuits, let it be m in number.

$C_1, C_2, \longrightarrow, C_m$ (say).

— Remove one edge e_i from each of these C_i

$\Rightarrow G' = G - \{e_1, e_2, \ldots, e_m\}$ Now has no circuit with $(n-1) - m$ edges and n-vertices.

- Also G' is connected since G is connected and removing an edge from the circuit leaves it connected.

- Hence G' is a tree with n vertices and $(n-m-1)$ edges. By virtue of result 2, $m = 0$ hence G contains no circuit $\Rightarrow$ G is a tree.

Result 2 + 3 combined gives us definition 2 (or property 2).

→ Minimally connected Graph:- A connected graph is said to be minimally connected if removal of any one edge from it disconnects the graph.

→ Result 4:- A graph is a tree iff it is minimally connected.

Proof:- → If a graph G is minimally connected. T.P. that G is a tree.

ie we need to show that it doesnot have a circuit, Suppose it has a circuit.

- If we remove one edge from the circuit still the graph is connected, which is a contradiction to the definition of minimally connected. Hence there is no circuit. $\Rightarrow G$ is a tree.

Converse, Let G be a tree, T.P G is minimally connected.

Let G is not minimally connected

$\Rightarrow \exists e_i$ an edge in G s.t $G - \{e_i\}$ is connected.

$\because e_i$ is in some circuit

$\Rightarrow G$ is not a tree.

$\rightarrow\leftarrow$

Hence G is minimally connected.

→ Result 4 is the property 4 of the set of definitions given.

→ Result 5 :- A Graph G with n-vertices and $(n-1)$ edges is a tree $\Leftrightarrow$ it is Circuit less.

Proof:- If G is a tree $\Rightarrow$ It has no circuit which is one side of the result.

Converse:- Let G be a graph with n vertices and $(n-1)$ edges and is circuitless.

To prove:- G is a tree. It is sufficient to show that G is connected.

Suppose that it is not, ie Let G be disconnected.

In that that case G contains 2-or-more circuitless components, Let them be $G_1, G_2, \dots, G_m$

and $G = G_1 \cup G_2 \cup \dots \cup G_m$ $\rightarrow (1)$

Let the no. vertices in $G_1, G_2, \dots, G_m$ be $n_1, n_2, \dots, n_m$.

Hence we have $n_1 + n_2 + \dots + n_m = n$.

∵ No. of edges in $G_1 = n_1 - 1$, $G_2 = n_2 - 1 \dots G_m = n_m - 1$

∵ each of G_i is circuit-less and connected

$\Rightarrow G_i$ is a tree. $\Rightarrow$ each G_i has $n_i - 1$ edges.

∵ No. of edges in G will be

$(n_1-1) + (n_2-1) + \text{——} + (n_m-1) = n_1 + \text{——} + n_m - m$

$= n - m$

Also, no. of edges in G is $(n-1)$, given.

$\therefore$ (1) $\Rightarrow$

$(n-1) = (n-m) \Rightarrow m = 1$

$\rightarrow\leftarrow$

$\therefore$ there are two or more than 2 components

Hence G is connected $\Rightarrow$ G is a tree.

$\rightarrow$ This is point 3 of the list of alternate definition.

Pendant Vertices in a tree :-

- For a tree with n - vertices, we ~~sto~~ will have $(n-1)$ edges.
- Each edge contributes twice to the total degree of all the vertices $\Rightarrow$ Total degrees ie $2(n-1)$ to be divided among n - vertices.
- Since no vertex is isolated ($\because$ connected graph) ie it cannot have degree zero.

- Even
  If we start dividing these degrees equally among n-vertices, ~~there~~ will be atleast two vertices of degree 1 in a tree.
- This result is ofcourse for $n \geq 2$.

- Hence we have proved the following Thm!-

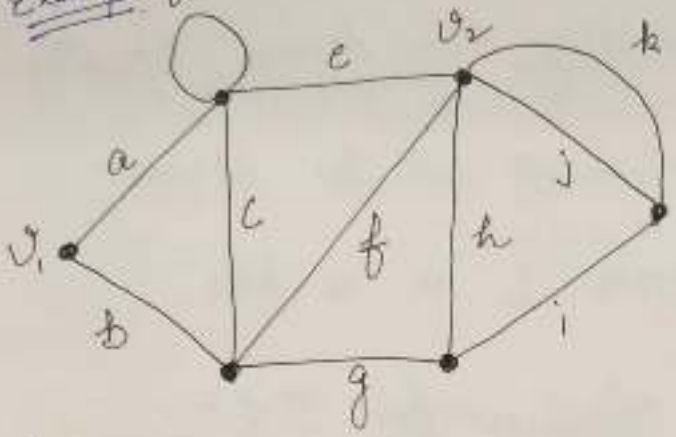
Theorem!- In any tree (with two or more vertices), there are atleast two pendant vertices.

It can also be shown that maximum no. of pendant vertices is $(n-1)$ for $n \geq 3$.

Distance and centers in a tree!-

→ Definition!- In a connected graph G, the distance $d(v_i, v_j)$ b/w two vertices v_i & v_j is the length of the shortest path (ie the no. of edges in the shortest path) b/w them.

Example 8



Find distance b/w vertices v_1 & v_2.

Solu

There are several paths b/w v_1 & v_2, namely,

	Path	~~distance~~ Length in case
Path 1	a – e	2
Path 2	a – c – f	3
Path 3	a – c – g – h	4
Path 4	b – c – e	3
Path 5	b – f	2

and several others ; minimum path length out of these is 2, $\therefore d(v_1, v_2) = 2$.

☆ In a tree there is only one path b/w any pair of vertices and hence that gives the distance b/w them.

→ Before we further discuss the distances, we define a metric:-

A function $f(x,y)$ of two variables is called a metric (or called distance in mathematical sense) if it satisfy the following conditions:-

(1) Nonnegativity:- $f(x,y) \geq 0$ $\forall$ x, y & $f(x,y) = 0$ iff $x = y$

(2) Symmetry:- $f(x,y) = f(y,x)$

(3) Triangle inequality:-
$f(x,y) \leq f(x,z) + f(z,y)$ for any z.

→ Theorem:- The distance b/w vertices of a connected graph is a metric.

Proof:- We need to show that the above three conditions are satisfied by the fn. $d(v_i, v_j)$ defined earlier.

(1) $d(v_i, v_j) \geq 0$ $\forall$ v_i, v_j as the path length is always non-negative.

$d(v_i, v_j) = 0 \iff v_i = v_j$ (as path length in a connected graph cannot be zero unless the two vertices are same).

(2) $d(v_i, v_j) = d(v_j, v_i)$

Obviously, as the shortest path b/w two vertices is same irrespective of the order.

(3) $d(v_i, v_j) \leq d(v_i, v_k) + d(v_k, v_j)$

$\forall$ vertex v_k of the graph

$\because$ $d(v_i, v_j)$ is the shortest distance b/w v_i & v_j, hence it can not be greater than any other path which goes through other vertices.

Hence $d(v_i, v_j)$ is metric.

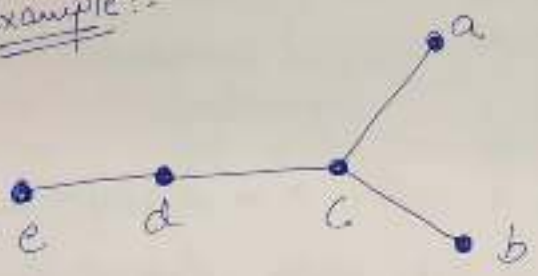
→ Eccentricity :- (also called associated number or Separation).

The eccentricity $E(v)$ of a vertex v in a graph G is the distance from v to the vertex farthest from v in G, i.e.

$$E(v) = \max_{v_i \in G} d(v, v_i)$$

→ Center of G :- A vertex with minimum eccentricity in graph G is called a center of G

Example:-



$E(a) = 3$, $E(c) = 2$, $E(e) = 3$.

$E(b) = 3$, $E(d) = 2$

c & d has minimum eccentricity

$\Rightarrow$ c & d are centers of the graph.

★ A graph can have more than one center.

— In a circuit (a graph that ~~is just~~ only contains a circuit), every vertex is a center.

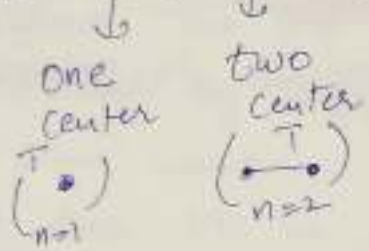
→ Theorem:- Every tree has either one or two centers.

Proof:- • In a tree the maximum distance, $\max d(v_i, v_j)$, from a given vertex v_i to any other vertex v_j occurs only when v_j is a pendant vertex.

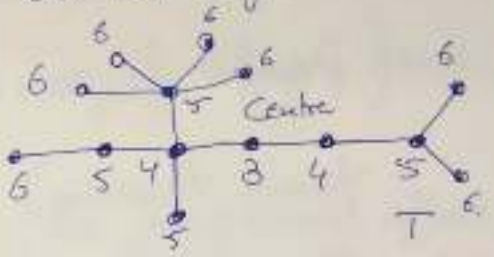
• Also a tree must have ~~more than~~ atleast two pendant vertices (where the no. of vertices in ~~a tree~~ a tree should be more than 2).

- Let T be a tree with n-vertices, $n \geq 3$.

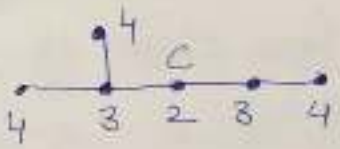
(for n=1, one center; n=2, two center — the theorem is trivial)



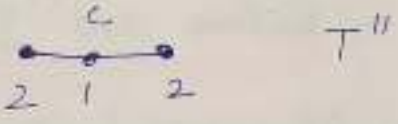
Example for the proof on the left:-

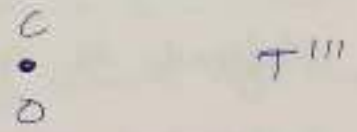


- Delete all the pendant vertices from T. The resulting graph T' is still a tree.



- Removal of all pendant vertices from T uniformly reduced the eccentricities of the remaining vertices (ie. vertices in T') by one.





Therefore, all vertices that T had as centers will still remain centers in T'.

- From T' we can again remove all pendant vertices and get another tree T''.
- We continue this process until there is left either a vertex (which is the center of T) or an edge (whose end vertices are the

two centers of T). Hence the theorem.

→ Corollary:- If a tree T has two centers, the two centers must be adjacent.

(Proof follows from the argument in previous theorem).

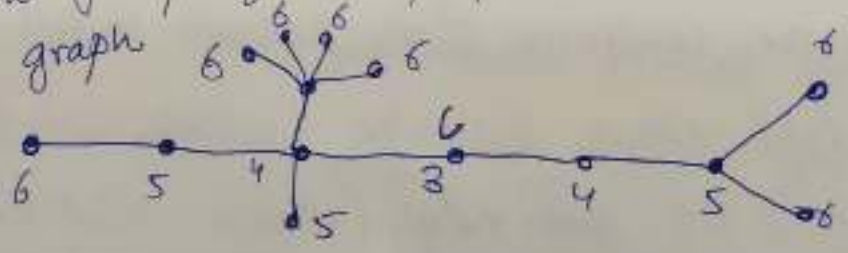
→ The Eccentricity of each vertex, E(v), represent how close v is to the farthest member of the group.

-: Radius and Diameter:-

- The eccentricity of a center in a tree is defined as the radius of the tree.
- The diameter of a tree T, is defined as the length of the longest path in T.

Example of tree:-

→ A Sociological Application:- Suppose that the communication b/w group of 14 people is represented by graph



when the vertices represent the persons and an edge represents the communication link b/w its two end vertices.

- Since the graph is connected, all members can be reached by any member, either directly or through some other member.
- Since tree is minimally connected, the group cannot afford to lose any of the ~~com~~ communication links.
- Vertex C should be the leader of the group, if closeness of communication were the criterion for leadership; (C is the center of tree).

→ Tree for representing data by Computer programmers:-

- Given a sequence of integers, no two of which are the same, find the largest monotonically increasing subsequence in it

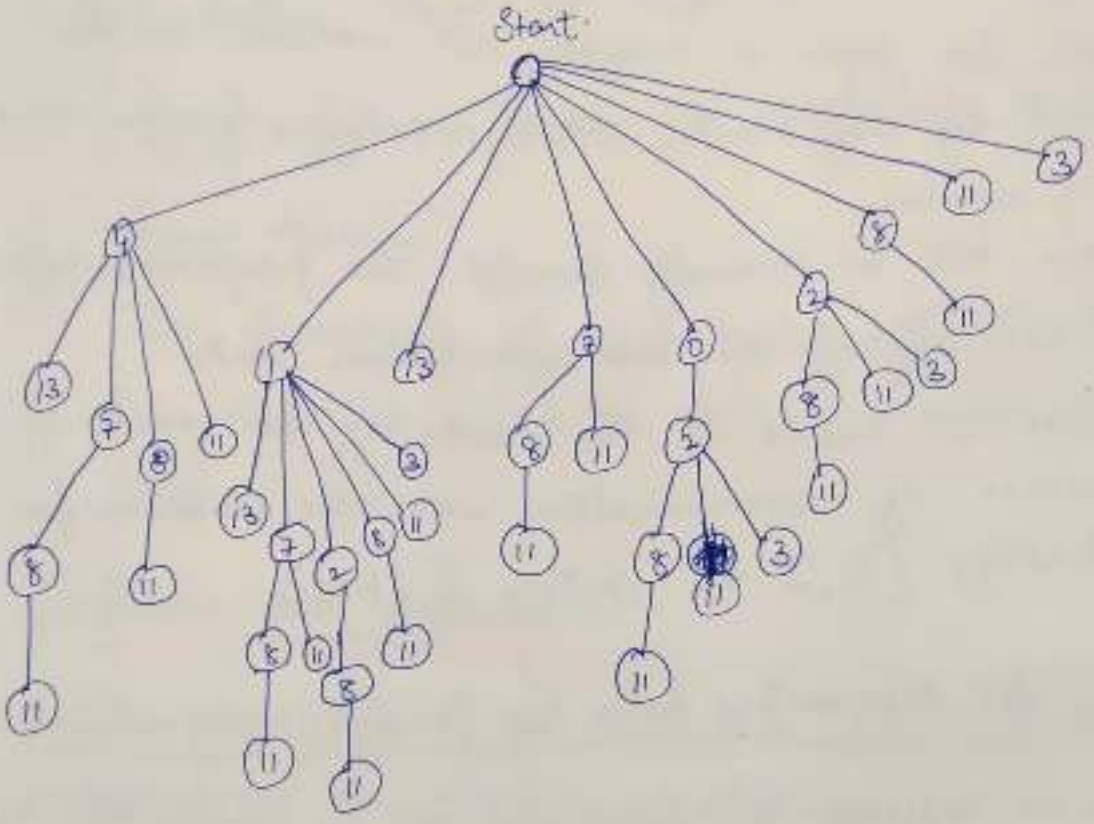
  Suppose the sequence is 4, 1, 13, 7, 0, 2, 8, 11, 3.

- This can be represented by a tree in which the vertices (except the start point) represent individual no. in the sequence.
- The path from the start vertex to a particular vertex v describes the monotonically increasing subsequence terminating in v.
- This sequence contains four longest monotonically

Increasing subsequences ie,

(4, 7, 8, 11), (1, 7, 8, 11), (1, 2, 8, 11) and (0, 2, 8, 11)

Each is of length 4 as shown below.



Rooted and Binary trees:-

→ A tree in which one vertex (called the root) is distinguished from all the others is called a ~~tot~~ rooted tree.

– In above example start vertex is the root.

–

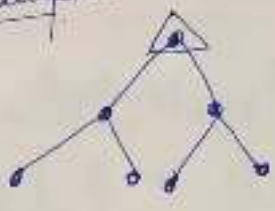
– Rooted tree with 4 vertices, Root is marked with a triangle.

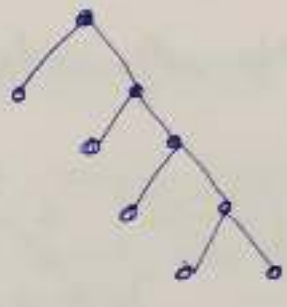
→ Binary Trees:-

Def.- A Binary tree is defined as a tree in which there is exactly one vertex of degree two, and each of the remaining vertices is of degree one or three.

- A binary tree will always have three or more vertices.
- Since, the vertex of degree two is distinct from all other vertices, this serves as the root and a binary tree is always rooted.

Example





Results:-

(1) The no. of vertices n in a binary tree is always odd.

Proof:- - There is one ~~vet~~ vertex of degree 2 (ie even vertices) and all other are of dg. 1 or 3 (ie odd vertices), there are (n-1) in no.

- No. of odd vertices is always even in any graph.

ie $n-1$ = even

$\Rightarrow$ $\boxed{n = \text{odd}}$

ie Binary tree will always have odd no of vertices.

(2) Find the no of pendant vertices in a binary tree?

Soln:- • Let p be the no. of pendant vertices.

Then, No. of vertices with degree 3 are $(n-p-1)$

• No. of edges in a tree with n vertices $= n-1$

• No. of edges in binary tree

$$= \frac{1}{2}\left[3(n-p-1) + p + 2\right]$$

(p → pendant vertices; $3(n-p-1)$ → deg. 3 vertices; 2 → one vertex of degree 2; $\frac{1}{2}$ → to find no. of edges)

(∵ No. of edges $= \frac{1}{2}$ total deg. of a graph)

$$\therefore \quad \frac{1}{2}\left[3n - 2p + 1\right] = n-1$$

$$\Rightarrow \quad 3n - 2p - 1 = 2n - 2.$$

$$2p = n+1 \quad \Rightarrow \quad \boxed{p = \frac{n+1}{2}}$$

(3) How many vertices of degree 3 are there in a binary tree?

Soln No. of pendant vertices $= \frac{n+1}{2}$

No. of vertices with deg. 3 $= (n-1) - \left(\frac{n+1}{2}\right)$

(∵ one vertex is of deg. 2)

$$= \frac{n}{2} - \frac{3}{2} = \frac{n-3}{2}.$$

→ A non-pendant vertex in a tree is called an Internal vertex.

(4) How many internal vertices are there in a binary tree?

Soln.

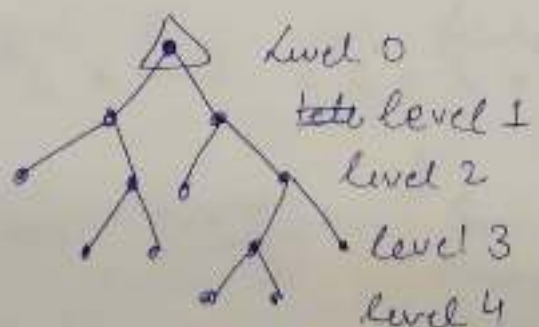
$$n - \left(\frac{n+1}{2}\right) = \left(\frac{n-1}{2}\right).$$

(n: Total; $\frac{n+1}{2}$: pendant vertices)

→ Levels in a binary tree:-

- A vertex v_i in a binary tree is said to be at level l_i if v_i is at distance l_i from the root.

→ Example of 13-vertex, 4 level binary tree.



→ • The most common application of binary trees is the search procedures, where each vertex of a binary tree represent a test with two outcomes, given by edges coming out of each vertex.

- Reaching a specified pendant vertex (the goal of the search) terminates the search.
- For such search procedure it is often important to construct a binary tree in which, for a given no. of vertices n, the vertex farthest from the root is as close to the root as possible.
- – Clearly, there can be only one vertex at level 0, ie root
  – Atmost 2 vertices at level 1, 4 at level 2 and so-on.

∴ max. no. of vertex possible in a k-level binary tree is

$$2^0 + 2^1 + 2^2 + \text{——} + 2^k \geq n.$$

$$\Rightarrow \quad 2^{k+1} - 1 \geq n.$$

→ The maximum level, l_{max} of any vertex in a binary tree is called the height of the tree.

- Minimum possible height of the binary tree is,

$$k = \min l_{max} \text{ when } 2^{k-1} - 1 = n$$

$$\Rightarrow 2^{\min l_{max}+1} = n+1$$

$$\Rightarrow \min l_{max} = \lceil \log_2(n+1) - 1 \rceil \rightarrow (*)$$

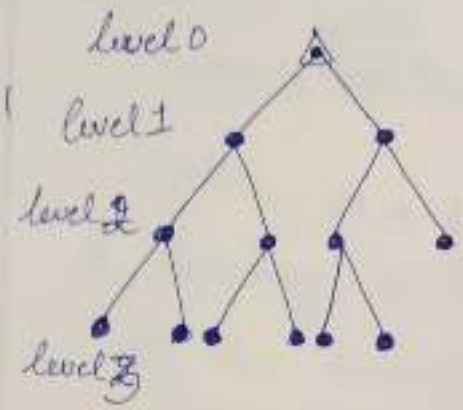
Where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. We take this in (*) as l_{max} should be an integer and even if the k^{th} level is not fully filled the height of the tree will be k.

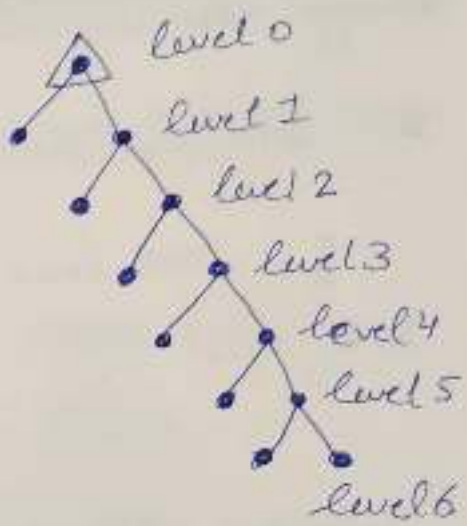
- On the other hand, to construct a binary tree for a given n s.t. the farthest vertex is as far as possible from the root, we must have exactly two vertices at each level, except at the 0 level. Thus for l_{max} will have maximum value s.t.

$$\max l_{max} = \frac{n-1}{2}$$

(Since n = odd for binary tree $\frac{(n-1)}{2}$ is an integer)

- For n=13 realising both these extremes we have





$$\min l_{max} = \lceil \log_2 (13+1) - 1 \rceil$$
$$= \lceil 2.8074 \rceil$$
$$= 3$$

$$\max l_{max} = \frac{13-1}{2}$$
$$= 6.$$

→ • In analysis of algorithms we are generally interested in computing the sum of the levels of all pendant vertices. This quantity is known as the path length (or ~~extenal~~ external path length) of a tree. Path length is defined as the sum of the path lengths from the root to all pendant vertices.

- The importance of the path length of a tree lies in the fact that this quantity is often directly related to the execution time of an algorithm

- Path length in previous example are :-
  - for first ~~tree~~ case (for 3-level tree).

$$3+3+3+3+3+3+2 = 20$$

  - for second case (for 6 level tree)

$$1+2+3+4+5+6+6 = 27$$

→ A tree with $2^{l_{max}} - 1$ vertices at level $l_{max} - 1$, yields the minimum path length for a given n (ie. $l_{max} - 1$ is fully filled).

→ Weighted path length :-

- In some applications, every pendant vertex v_j of a binary tree has associated with it a positive real number w_j. Given $w_1, w_2, \ldots, w_m$ the problem is to construct a binary tree (with m pendant vertices) that minimizes,

$$\sum w_j l_j$$

where l_j is the level of pendant vertex v_j, and the sum is taken over all pendant vertices.

Example :- A vending machine is to identify, by a sequence of tests, the coin that is put into the machine. Only 10 Rs, 5 Rs, 2 Rs, 1 Re coins can go through

the slot. Let us assume that the probability of a coin being a 10 Rs, 5 Rs, 2 Rs, 1 Re is 0.30, 0.5, 0.15, 0.05 respectively.

- Each test has effect of partioning partitioning the four types of coins into two complemantary test and asserting the unknown coin to be in one of the two sets.
- If the time taken for each test is same, what sequence of tests will minimize the expected time taken by the vending machine to identify the coin?

Soln:- Soln requires constructing a binary tree with 4 pendent vertices v_1, v_2, v_3 & v_4 and the corresponding weights are

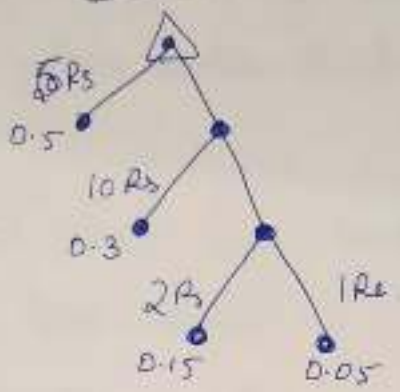
$w_1 = 0.30$, $w_2 = 0.5$, $w_3 = 0.15$, $w_4 = 0.05$

s.t the quantity $\sum w_i l_i$ is minimized.

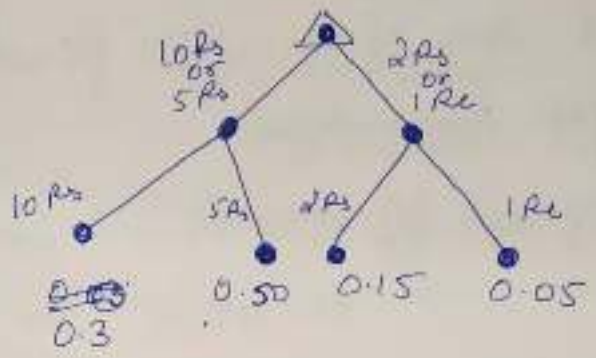
Following two solns. (trees) can be made (there are other possibilities also).

- Let t be the time at each step.

Soln 1.



Soln 2



Weighted Sum for Soln 1 =

$$0.5 \times t + 0.3 \times 2t + 0.15 \times 3t + 0.05 \times 3t$$

$$= 1.7t$$

Weighted Sum for Soln. 2 =

$$0.3 \times 2t + 0.5 \times 2t + 0.15 \times 2t + 0.05 \times 2t$$

$$= 2.0t$$

- For the two solns. above, first one takes less time as compared to second.

Counting Trees :-

Example :- Let $C_k H_{2k+2}$ be an hydrocarbon. the fact valency of C is 4 (ie carbon will have 4 atoms attached to it) and valency of H is 1. If C & H form a graph such that there

k - Carbon atoms and 2k+2 - H atoms forming vertices and each of the atom is linked to other using edges. ~~degree~~

- degree of C-atom is 4
" " H-atom is 1.

- Total No. of vertices :-

$$k + 2k + 2 = 3k + 2.$$

- Total no. of edges :-

$$\frac{1}{2} (\text{Sum of degrees})$$

$$= \frac{1}{2} (4k + 2k + 2) = 3k + 1$$

$$= (3k + 2) - 1$$

i.e. Atom Number of vertices minus One.

hence the hydrocarbon would be a tree.

- Alternative of the above ques :-

→ If we are given that hydrocarbon $C_k H_{2k+2}$ form a tree, find the no. of edges in it?

Soln k-vertices corresponding to carbon of
2k+2 - vertices " " hydrogen.

– Total no. of vertices = $k+2k+2 = 3k+2$

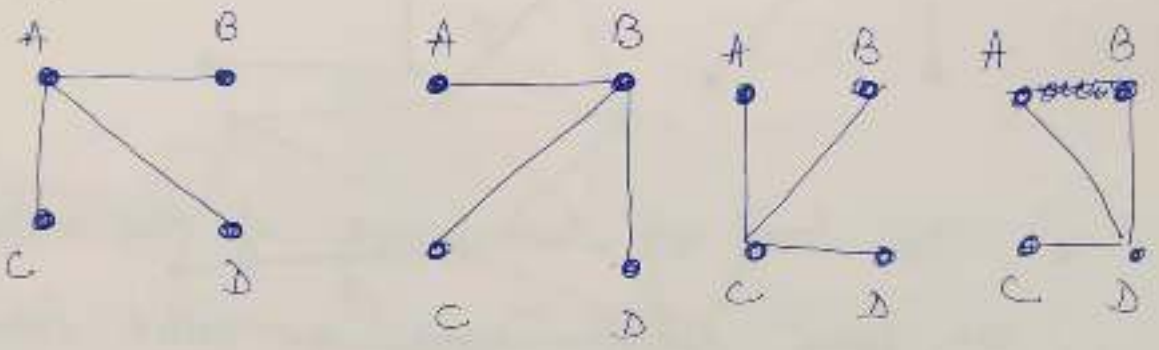
Since it is a tree no. of edges are $3k+2-1$

$= 3k+1$

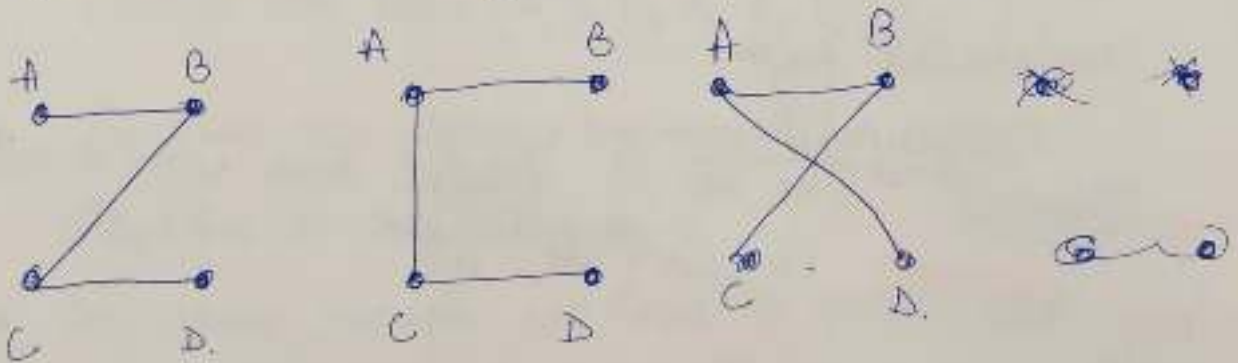
→ Next we have in how ~~to~~ many of such different trees exists.

→ Let us first answer this question with $n=4$ vertices.

- If we have 4 - labeled vertices



All the above four trees are different, Also



the above three are also different and we can make 9 - more in similar manner. In total there will be 16 of them.

- However the first 4 are isomorphic to each other and the other 12 will be isomorphic to each other. And in this way there are only two such possibilities which are non-isomorphic to each other. Such non-isomorphic ones would be

- The trees in which each of the vertices are given specific name are called labelled graphs and the one without labels are called unlabelled graph.

-: Cayley's Theorem :-

→ Theorem: The no. of labeled trees with n-vertices ($n \geq 2$) is n^{n-2}.

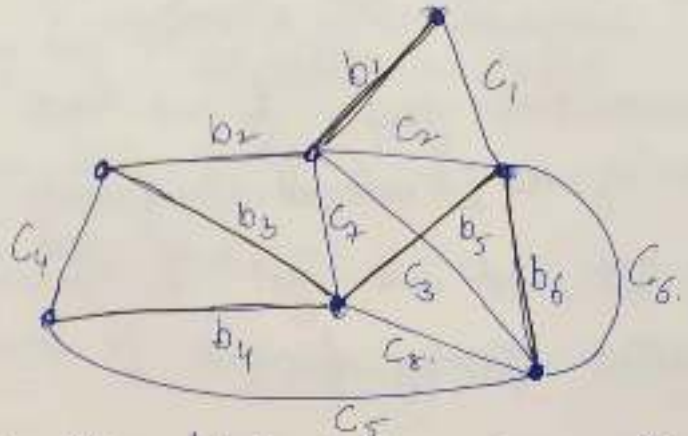
(only statement)

Example :- for $n = 4$,

There are $4^{4-2} = 4^2 = 16$ labeled graphs. This was earlier discussed for the case of hydrocarbons.

Spanning Trees

- So far we have discussed the tree and its properties when it occurs as a graph itself. Now we will discuss trees as a subgraph of another graph.
- A tree T is said to be a spanning tree of a connected graph G, if T is a subgraph of G and T contains all vertices of G.
- Example:-



Subgraph with black edges is a spanning tree. (labeled with edges $(b_1\ b_2\ b_3\ b_4\ b_5\ b_6)$).

- Sometimes the spanning tree is also called Skeleton or scaffolding of G.
- In above example we have 7 vertices and we need atleast 6-edges to make a tree of 7-vertices and adding even one more edge to it ~~will render~~ the subgraph will no more be a tree.

- Since spanning trees are the largest (with maximum no. of edges) trees among all trees in G, it is also quite appropriate to call a spanning tree a maximal tree subgraph or maximal tree of G.

- Spanning tree is defined only for a connected graph, because a tree is always connected and in a disconnected graph of n-vertices we cannot find a connected subgraph with n-vertices.

- For a disconnected graph if we have k-components, each of the k-connected components can have a spanning tree and we get k-spanning trees, collectively called spanning forest.

→ • Finding a spanning tree of a connected graph G:-

- If G has no circuit, it is its own spanning tree
- If G has a circuit, delete an edge from the circuit.
- If there are more circuits, repeat the operation till an edge from the last circuit is deleted — leaving a connected, circuit-free graph that contains all the vertices of G. Thus we have a spanning tree.

The above arguments can be used to prove the following thm:-

Theorem:- Every connected graph has at least one spanning tree.

→ • An edge in a spanning tree T is called a branch of T.

• An edge of G that is not in a given spanning tree T is called a chord.

→ Edges $b_1, b_2, b_3, b_4, b_5, b_6$ are branches of the spanning tree. And $c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$ are the chords of the spanning tree.

→ If set of all chords corresponding to the spanning tree T is denoted by $\tilde{T}$ then

$$T \cup \tilde{T} = G$$

Theorem:- With respect to any of its spanning trees, a connected graph of n-vertices and e edges has $(n-1)$ tree branches and $(e-n+1)$ chords.

Example:- For an electric circuit ~~for~~ with e elements (edges) and n-nodes (vertices), what is true.

minimum no. of elements, we must remove to eliminate all circuits in the network?

Soln:- The answer is $e-n+1$

→ For a k-connected component, Graph G with n-vertices, find the no. edges in spanning trees and no. of chords?

Soln: Let $n_1, n_2, \dots, n_k$ be the no. of vertices in each of k-components, then

$$n_1+n_2+\dots+n_k=n.$$

Also, for each of the connected component, the spanning tree will contain n_1-1 edges.

∴ total no. of edges in the spanning trees (forest) are:-

$$(n_1-1)+(n_2-1)+\dots+(n_k-1)$$

$$n_1+n_2+\dots+n_k-k$$

$$=n-k$$

- $n-k \geq 0$ ∴ no. of components cannot be greater than no. of vertices in the graph.

Let e be the no. of edges in G, then

No. of chords are $e-(n-k) = e-n+k$.

Also $e-(n-k) \geq 0$ ∴ no. of edges in spanning forest cannot be greater than the total no. of edges.

→ Rank and nullity :-

- Rank, r = $n-k$ = No. of branches in any spanning tree (or forest) of G.
- Nullity, $\mu = e-n+k$ = No. of chords in G.
- Rank + Nullity = No. of edges in G.
- Rank & nullity are independent of the way spanning trees are choosen.

→ Fundamental Circuits :-

→ • Adding an edge b/w any two vertices of a tree a circuit is created in the graph ∴ there is already one path b/w any two vertices of a tree.

- Also this will create only a unique circuit

∵ there was only one path b/w any two vertices earlier and adding one edge will create only one more path.

- Combining the above arguments we can prove the following theorem:-

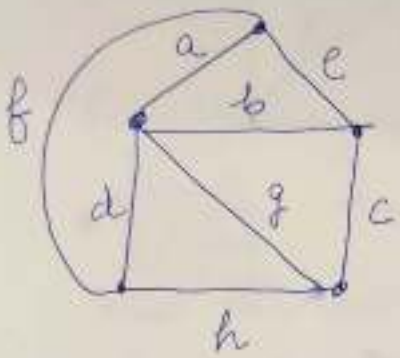
Theorem:- A connected graph G is a tree iff adding an edge b/w any two vertices in G creates exactly one circuit.

- Let T be a spanning tree of a connected graph G. Then adding one chord in T will create exactly one circuit. Such a circuit formed by adding one chord to a spanning tree is called a fundamental circuit.

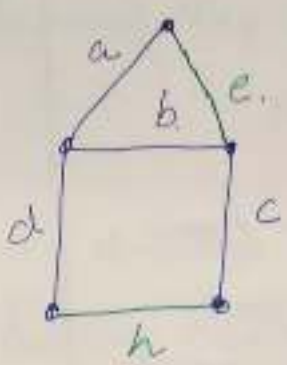
- How many fundamental circuit does a graph has?

Ans:- Exactly as the no. of chords
$= e - n + k = \mu$ (nullity of a graph).

Example:-



Graph G



Spanning tree, T
(a, b, c, d edges form a Spanning tree)

- Adding an edge 'e' gives a fundamental circuit (a, b, e)
- Adding another edge h, gives another fundamental circuit (b, c, h, d)
- however circuit (a, d, h, c, e) is not fundamental as it contains two chords.

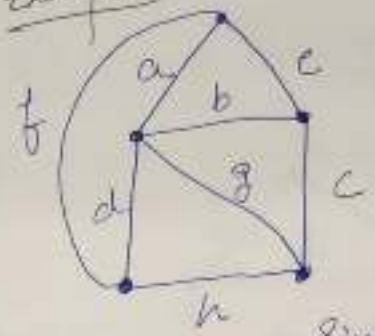
- Fundamental circuit is always defined w.r.t a given spanning tree, however the no. of fundamental circuits is independent of the spanning tree.

→ Finding all the spanning trees of a graph:-

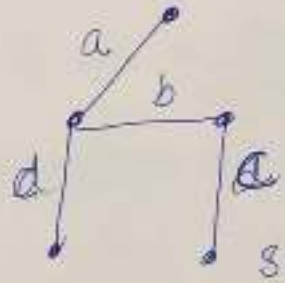
- To generate all the spanning trees start from a given spanning tree (Algorithms to find a spanning tree will be discussed later).

- Add a chord in the spanning tree, this will form a fundamental circuit.
- Remove any branch from the fundamental circuit just formed s.t. the ~~graph~~ subgraph remains connected will create a new spanning tree.
- This process is called Cyclic interchange or elemantary tree transformation

Example :-

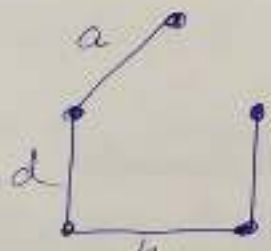


Single graph, G



Spanning tree T_1

- add the chord 'h', and remove the branch 'b'



Another spanning tree T_2

→ Spanning trees in a weighted graph :-

→ Weight of a spanning tree T of a graph G is defined as the sum of the weights of all the branches in T.

→ A spanning tree with the smallest weight in a weighted graph is called a shortest

Spanning tree or Shortest-distance spanning tree or minimal spanning tree

Example:-

- Suppose there are n-cities $v_1, v_2, \ldots, v_n$ and the cost of building road b/w v_i & v_j is c_{ij}, where the direct road can be build. The problem is to find the least expensive network that connects all n-cities together
- – The connected network must be a tree, s.t. that its weight is minimum.
  – Thus the problem is reduced to finding a shortest spanning tree in a connected weighted graph of n-vertices.

→ Theorem:- (only Statement) A spanning tree T (of a given weighted connected graph G) is a shortest spanning tree iff there exists no other spanning tree T', ~~of~~ formed by replacing one branch of T by a chord whose weight is smaller than that of T.

→ Algorithm for Shortest Spanning Tree :-

- Prim's Algorithm :-

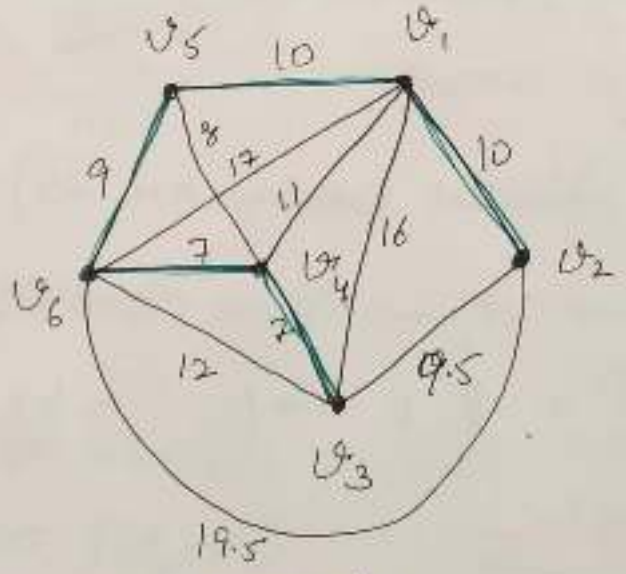
Steps :-

- Begin by choosing an edge with smallest weight, put it in the spanning tree
- look for minimum weight edge ~~which~~ is joining the adjacent vertices of the previously added vertex.
- Proceed to the next adjacent vertex with minimum weight, uptill there are $(n-1)$ ~~edges~~ edges in the tree, (where n is the no. of vertices in the graph G), such that no ~~vertex is repeated in the simple~~ circuit is formed.

Example :-

→ To check for adjacent vertices it is always better to make the corresponding weight matrix of the Graph G. Weight matrix (W) will have have w_{ij} as the weight of edge b/w v_i & v_j If there is no edge b/w v_i & v_j then weight is ∞ (∵ We need to find the minimum weighted tree).

Example



Find the minimum spanning tree using prim's algorithm.

Soln Weight matrix:-

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	∞	10	16	11	10	17
v_2	10	∞	9.5	∞	∞	19.5
v_3	16	9.5	∞	7	∞	12
v_4	11	∞	7	∞	8	7
v_5	10	∞	∞	8	∞	9
v_6	17	19.5	12	7	9	∞

- minimum weight is 7, choose any one
  Say we choose edge b/w v_3 & v_4.
- Then from v_6 we choose minimum weight ~~of~~ edge v_6 (v_3 is also of same weight but that edge is already in the tree)
- v_6 we take the min. weight going to v_5 (s.t. no circuit is made).
- Then to v_1 & from there to v_2, without forming circuit.

Hence the path (shortest spanning tree is)

$$v_3 \to v_4 \to v_6 \to v_5 \to v_1 \to v_2$$

Weight = $7 + 7 + 9 + 10 + 10 = 43$

→ Kruskal's Algorithm:-

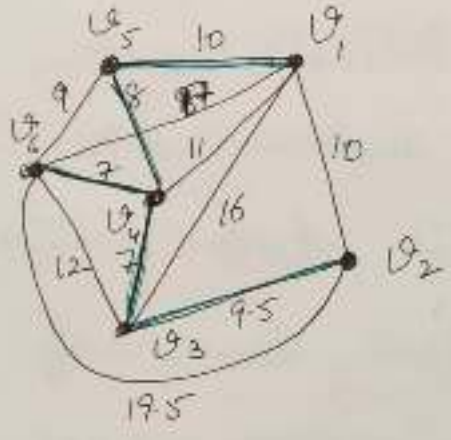
Steps:-

- Arrange all the vertices in a non-decreasing sequence of weights.
- Then from all the non-remaining edges edges select another smallest edge that makes

no circuit with the previously selected edges.

- Continue until $(n-1)$ edges have been selected and these edges constitute the desired shortest spanning tree.

Example (for the same example solved earlier)



- Non-decreasing seq. of weights of edges

edge	weight
(v_4, v_6)	7
(v_4, v_3)	7
(v_5, v_4)	8
(v_5, v_6)	9
(v_3, v_2)	9.5
(v_5, v_1)	10
(v_1, v_2)	10
(v_1, v_4)	11
(v_3, v_1)	12
(v_1, v_3)	16
(v_1, v_6)	17
(v_2, v_6)	19.5

- minimum weight = 7 for (v_4, v_6),
- Next (v_4, v_3) with weight 7,
- Next (v_4, v_5) with weight 8,
- (v_5, v_6) makes circuit So leave it
- Next are (v_3, v_2) (weight 9.5) and (v_5, v_1) (weight 10).

- Spanning tree is shown by green in the graph

- Total weight = 7+7+8+9.5+10

= ~~375~~ 41.5

☆ It can be noticed from example (and easier to argue in general also) that the Kruskal's algorithm is better as it doesnot constrain us to check for next minima in adjacent vertices.

Degree-Constrained Shortest Spanning Tree:-

- Some practical problems may require that degree of every vertex in a spanning tree are restricted. & Note that above algorithms to donot have any such restriction and

$1 \leq d(v_i) \leq n-1$ for every vertex v_i.

- Example:- flight scheduling problem:-

No. of flights from a given terminal are restricted to say 3 at any given time

ie $d(v_i) \leq 3$ for $\forall v_i$

- Such minimal spanning trees are called degree constrained ~~Shortest~~ shortest spanning trees.

- In General the problem can be stated as:-

Given a weighted connected graph G, find the shortest spanning tree T in G s.t.

$$d(v_i) \leq k \quad \forall \quad v_i \text{ in } T$$

ie the degree is restricted to k.

- If $k=2$, in fact the problem reduces to finding the shortest Hamiltonian path (travelling Salesman problem, without the salesman returning to his base, is a well known example of this).

- Such problem for $k=2$ are discussed earlier also. (for unweighted graphs)

- No efficient method of finding an arbitrary degree-constrained shortest spanning tree is there till now, and this is an open problem.

List of formulas:-

① In a simple graph of n-vertices & k components there can be atmost $\frac{(n-k)(n-k+1)}{2}$ edges.

② In a connected graph G with exactly 2k odd vertices, ∃ k edge-disjoint subgraphs s.t. they together contain all edges of G & that each is a unicursal.

③ A graph containing m-edges can be decomposed in $2^{m-1}-1$ different ways into pairs of disjoint subgraphs.

④ The length of a Hamiltonian path (if it exists) in a connected graph of n-vertices is (n-1)

⑤ In a complete graph with n-vertices there are $(n-1)/2$ edge disjoint Hamiltonian circuits, if $n \geq 3$ is an odd no.

⑥ In a binary tree with n-vertices, have $p = \frac{n+1}{2}$ pendant vertices

⑦ max. level l_{max} in any binary tree with n-vertices lies b/w

$$\lceil \log_2(n+1) - 1 \rceil \leq l_{max} \leq \frac{n-1}{2}$$

↓ min l_{max} ↓ max l_{max}

where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

l_{max} is also the height of the binary tree.

⑧ The no. of labeled trees with n-vertices ($n \geq 2$) is n^{n-2}.

⑨ A Graph G is an Hamiltonian graph if

(i) G is simple, & every vertex has degree atleast $n/2$

or

(ii) for every pair of k non-adjacent vertices (u, v) we have $d(u) + d(v) \geq n$

or

(iii) iff $cl(G)$ is Hamiltonian.