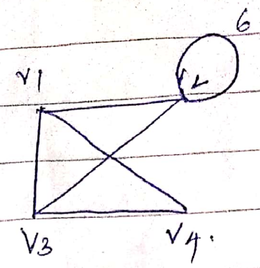
v_1 —e_1— v_2 e_1 is incident on v_1
" " " v_2.

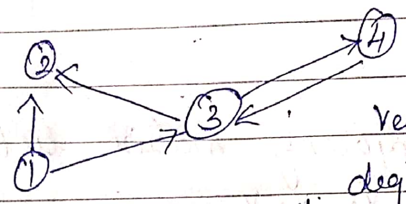
v_1 & v_2 are adjacent nodes.



vertex	v_1	v_2	v_3	v_4	
Degree	3	4	3	2	= 12.

$6 \times 2 = 12$.

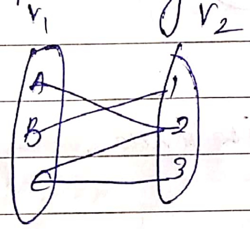
Hand shaking theorem

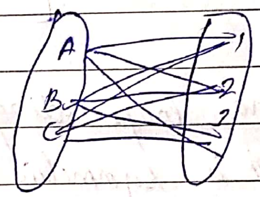


	Vertex	1	2	3	4	
	deg^+	2	0	2	1	= 5.
indegree	deg^-	0	2	2	1	= 5

$$\Sigma deg^-(v) = \Sigma deg^+(v) = e.$$

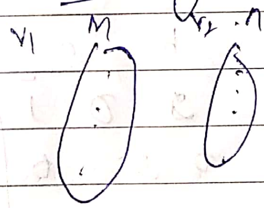
Bipartite graph. Complete Bipartiate graph

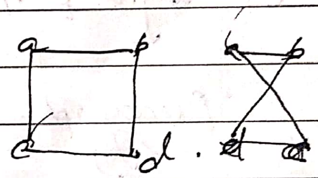




$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 1 & 0 \\ b & 1 & 0 & 0 & 1 \\ c & 1 & 0 & 0 & 1 \\ d & 0 & 1 & 1 & 0 \end{array}$$

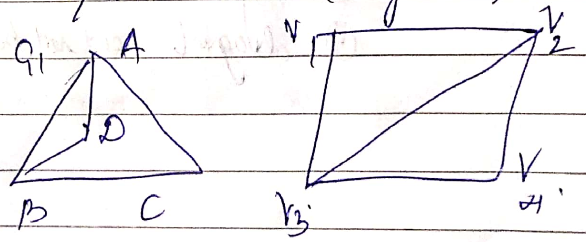
$K_{m,n}$ – general notation





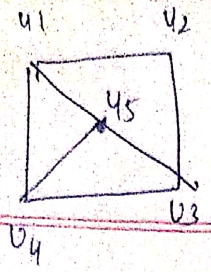
Two graph

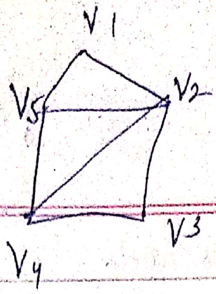
Isomorphic Graphs. G_1 & G_2 are isomorphic if ∃ a 1-1 correspondent b/w vertex set which preserves adjacency of the vertex. (degree is preserved).



	G_1 degree	G_2	
A	3	v_1	2
B	3	v_2	3
C	2	v_3	3
D	2	v_4	2.

G_1 & G_2 are isomorphic.





degre

u_1 : 3 v_1 : 2

u_2 : 2 v_2 : 4

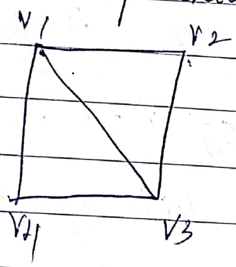
u_3 : 3 v_3 : 2

u_4 : 3 v_4 : 3

u_5 : 3 v_5 : 3

not isomorphic.

Matrix representation :- (Adjacency matrix - list vert

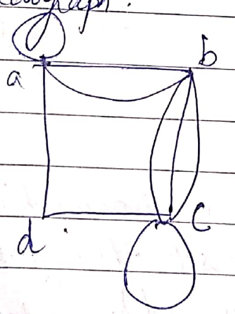


$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Simple graph

1. A_G has no loops diagonal entries is zero.
2. Simple graph is symmetric
3. degree of v_i is the number of 1's in the i^{th} row.

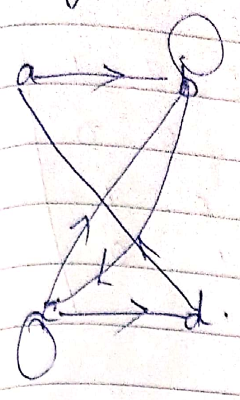
Pseudograph.



$$A_G : \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

1. Symmetric matrix.
2. diagonal need not be 1.

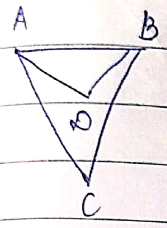
Multigraph direct graph.

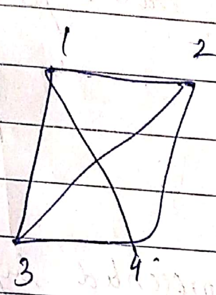


	a	b	c	d.
a	0	1	0	0
b	0	[illegible]	1	0
c	0	1	[illegible]	0
d	1	0	0	0

Not symmetric.

Check isomorphic.



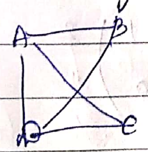


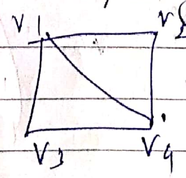
	G_1		G_2
A	3 →	1 →	3
B	3 →	2 →	3
C	2	3	3
D	2	4.	3.

not isomorphic.

Isomorphic & adjacency matrix.

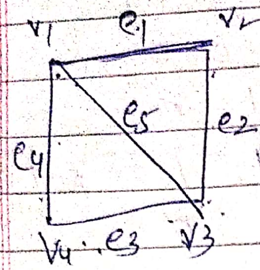
Establish the isomorphism of two graphs given below by considering their adjacency matrices.





under isomorphism degree preserved.

Incident Matrix



$$\begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \hline v_1 & 1 & 0 & 0 & 1 & 1 \\ v_2 & 1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 1 & 1 & 0 & 1 \\ v_4 & 0 & 0 & 1 & 1 & 0 \end{array}$$

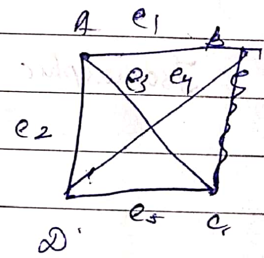
degree = entries 1.

(2)

Note: Each column has exactly 2 ones.
Rows with all zero will be isolated vertex
Rows with only one 1 is called pendent vertex.
$\deg(v_i)$ = number of 1's

Draw a graph represented by the foll. incident matrix.

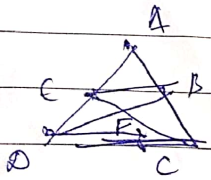
$$\begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \hline A & 1 & 1 & 1 & 0 & 0 \\ B & 1 & 0 & 0 & 1 & 0 \\ C & 0 & 0 & 1 & 0 & 1 \\ D & 0 & 1 & 0 & 1 & 1 \end{array}$$



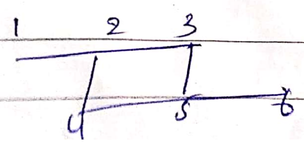
(1) Check whether G_1 & G_2 are Isomorphic.

$$AG = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{matrix} \deg \\ 1 \\ 1 \\ 2 \end{matrix} \qquad AG_2 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{matrix} 2 \\ 1 \\ 1 \end{matrix}$$

(1)



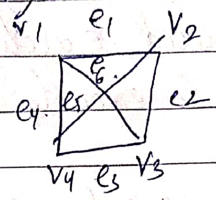
Euler circuit ABCDE degree even.
H-C.



EP <
EC <.

Paths, Cycles and Connectivity.

A path is a graph is a finite alternating sequence of vertices and edges begining and ending with vertices such that each edge is incident on the vertices preceding and following it.

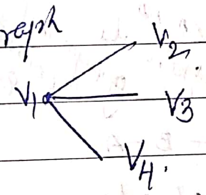


$v_1 e_1 v_2 e_5 v_4$.
$e_1 - e_5$
is a path.

lines connecting the vertices.

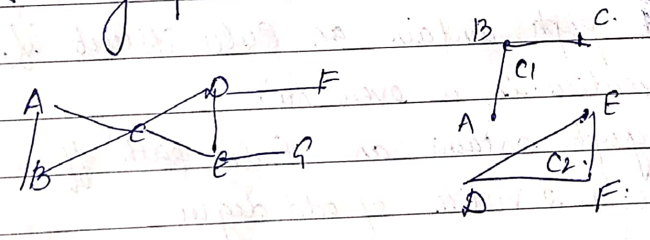
$v_1 e_1 v_2 e_5 v_4 e_4 v_1$ is a circuit or cycle.

A undirected graph is said to be a connected graph if their is a path btw every pair of distinct vertex of the graph



ab bc cd de ef fg
ac bd ce df eg
ad be cf dg
ae bf cg
af bg
ag.

A graph that is not connected is called disconnected.



Can have component.

Result The maximum no of edges in a simple disconnected graph G is $\frac{(n-k)(n-k+1)}{2}$.

Find max no of edges in a simple disconnected graph with 5 vertices and 2 components.

Ans. $n=5$ $k=2$..

Max no of edges $= \frac{(n-k)(n-k+1)}{2} = 6$.

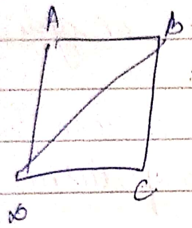
Eulerian and Hamiltonian Graph.

① A path of graph G is called Eulerian path if it includes every edge exactly once.

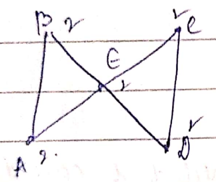
② A circuit (cycle / closed path) of a graph G is called Eulerian circuit if it includes each edge of G exactly once.

③ A graph containing an Eulerian circuit is called Eulerian graph.

eg



: Eulerian path: A→B → C → B.
No Euler circuit ∴ not euler graph.



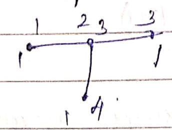
If the degree of every vertex is an even no then the graph will contain Euler circuit

A – E – C – D – E – B – A

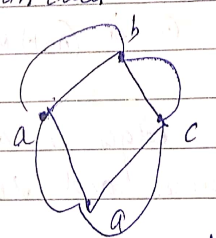
repeat vertex not edges.

A connected graph contain an Euler circuit iff each of its vertices is a even no.
A connected graph contains an Euler path iff it has exactly 2 vertices of odd degree.

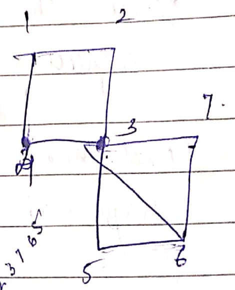
Prob. Identify Euler path Euler circuit



No Euler path.



abdabcda.
Euler path = Euler circuit



6736534123.
No Euler circuit

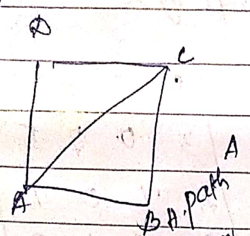
Hamiltonian path.

A path of a graph G is called a HP if it includes each vertex of G exactly only.

A circuit of a graph G is called a Hamiltonian circuit if it includes each vertex of G exactly once (closed H. path starting & end vertex are the same)

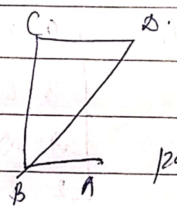
A graph containing a H. circuit is called Hamiltonian graph.

eg.



A → B → C → D
path

A-D-C-B-A
H cir

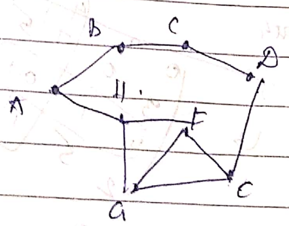


path A→B→C→D

A-B

No circuit.

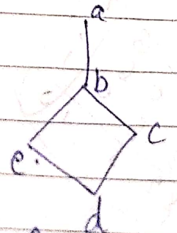
* From H.C we obtain H. edge path by deleting one edge.
* A H.C contains H.P but H.P does not contain H.C.
* A complete graph Kn will always have a Hamiltonian circuit
* A given graph may contain more than one H.C.



H.P A B C D E F G H A

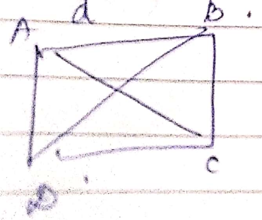
H C A B C D E F G H A.

(ii)



H. path A B C D E.

H. C A B C D E not H.C.

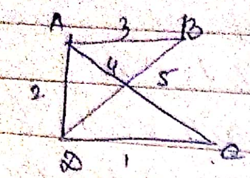


H.P A B C D

H.C A B C D A.

Shortest path

Using:



Weights on edges.

Weights edges.

$$W = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 3 & 4 & 2 \\ B & 3 & 0 & 0 & 5 \\ C & 4 & 0 & 0 & 1 \\ D & 2 & 5 & 1 & 0 \end{array}$$

leave diagonal
non diagonal entries replace by ∞.

$$L_0 = \begin{bmatrix} 0 & 3 & 4 & 2 \\ 3 & 0 & \infty & 5 \\ 4 & \infty & 0 & 1 \\ 2 & 5 & 1 & 0 \end{bmatrix} \qquad L_1 = \begin{bmatrix} 0 & 3 & 4 & 2 \\ 3 & 0 & 7 & 5 \\ 4 & 7 & 0 & 1 \\ 2 & 5 & 1 & 0 \end{bmatrix}$$

w.r.t 10 $L_0 \Rightarrow K_B = 1$

$$L_2 = \begin{bmatrix} 0 & 3 & 4 & 2 \\ 3 & 0 & 7 & 5 \\ 4 & 7 & 0 & 1 \\ 2 & 5 & 1 & 0 \end{bmatrix}$$

min(5, 5)

$$L_3 = \begin{bmatrix} 0 & 3 & 4 & 2 \\ 3 & 0 & 7 & 5 \\ 4 & 7 & 0 & 1 \\ 2 & 5 & 1 & 0 \end{bmatrix}$$

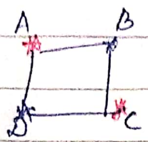
$$L_4 = \begin{bmatrix} 0 & 3 & 3 & 2 \\ 3 & 0 & 6 & 5 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 1 & 0 \end{bmatrix}$$

→ shortest path.

Graph Colouring

Defn.
A colouring of a simple graph G is the assignment of a colour to each vertex of the graph so that no 2 adjacent vertices are assigned the same colour.

Eg.

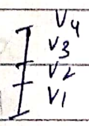


2 colours are required for proper colouring

Defn The smallest no of colours needed to produce proper colouring of a graph G is called chromatic number of G. denoted by $\chi(G)$

Defn. The no of ways of colouring a graph G with x or fewer colours is a f^n of x $P_G(x)$ called the chromatic polynomial of G

The smallest positive value of x for which $P_G(x) \neq 0$ is the chromatic number of G.



For any linear graph

$P_{Ln}(x) = x(x-1)^{n-1}$ = chro polynomial.

$\chi(G_n)$ = Chromatic number = 2.

For any complete graph k_n.

$x(x-1)(x-2)$.

$P_{kn}(x) = x(x-1) \ldots (x-(n-1))$ = chrom polyn.

$\chi_{kn} = n$. $\Rightarrow k_3 \Rightarrow \chi_{k_3} = 3$ $\quad k_4 \Rightarrow \chi_{k_4} = 4$.

Eg. Write the chromatic polynomial of k_3 find the no of ways of colouring k_3.

$P_{k_3}(x) = x(x-1)(x-2)$ $\quad P_{k_3}(3) = 3(3-1)(3-2) = 6$.

Total ways of colouring k_3 with 3 colours $\chi_{kn} = 3$

For any discrete graph (with n vertices only)
$P_{U_n}(x) = x^n$.

$P_{U_n}(x) \neq 0$ for $x=1$ $\therefore \chi_{U_n}(x) = 1$.

For any disconnected graph with components $G_1, G_2, \ldots G_n$

$P_G(x) = P_{G_1}(x)\, P_{G_2}(x) \ldots P_{G_n}(x)$.

(6) Find number of proper colouring of G.

G_1 G_2.

$P_G(x) = P_{G_1}(x)\, P_{G_2}(x) = P_{K_3}(x)\, P_{K_3}(x)$.

$= \left(x(x-1)(x-2)\right)^2$.

$P_G(x) \neq 0$ for $x = 3$ $\therefore \chi(G) = 3$.

No of ways of proper colouring of G.
$= P_G(3) = \left(3(3-1)(3-2)\right)^2 = 6^2 = 36$

State the chromatic polynomial of complete graph K_4 & hence find its chromatic number.

Sol^n

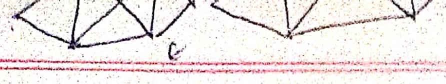
$P_{K_4}(x) = x(x-1)(x-2)(x-3)$

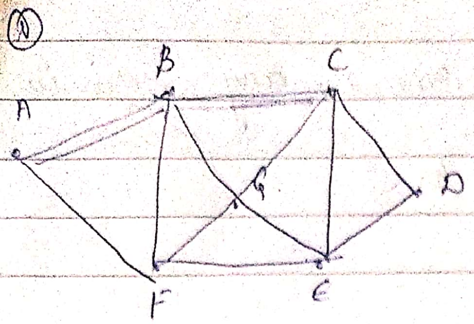
$P_{K_4}(x) \neq 0$ $x = 4$.

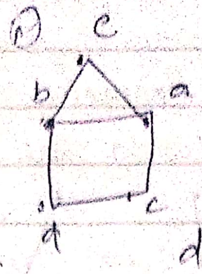
$\chi(K_4) = 4$.

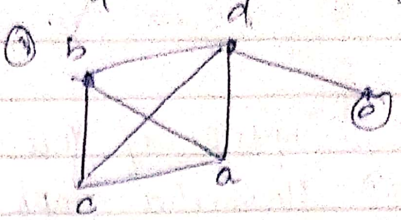
8) State the chromatic polynomial of linear graph L_4.

$P_{L_4}(x) = x(x-1)^{4-1} = x(x-1)^3$.









Euler circuit

Euler path : ~~ABC~~ FBAFCBGCDCF

Hamiltonian circuit

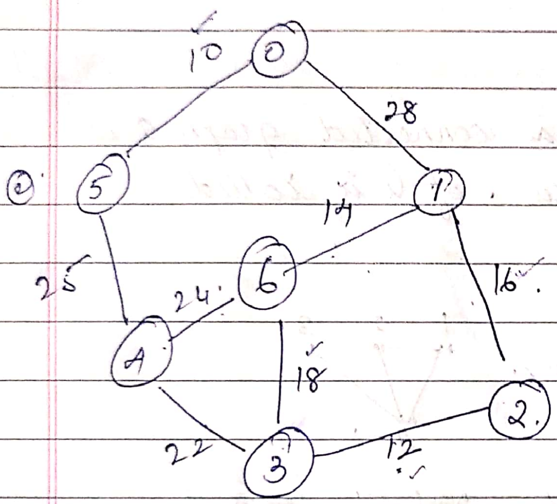
path : ABGCDEFA

② Euler circuit : ~~[illegible]~~

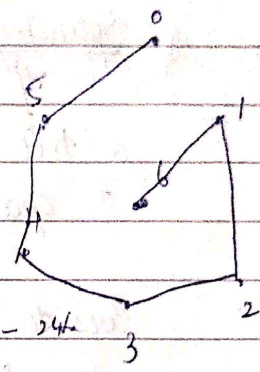
Hamilton C : ~~ABD~~ DEACDB

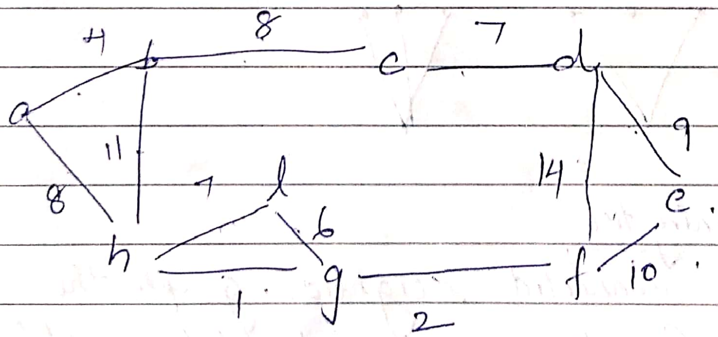
③ Euler circuit : No

Hamiltonian : No.



5 0	10
2 3	12
6 1	14
1 2	16
3 6	18
3 4	22 46 – 24
5 4	25
0 1	28





hg : 1

gf : 2

ab : 4

lg = 6

lh : 7

cd : 7

ah : 8

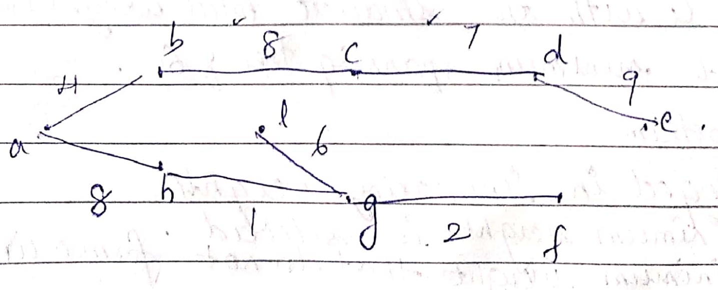
bc : 8

de : 9

ef : 10

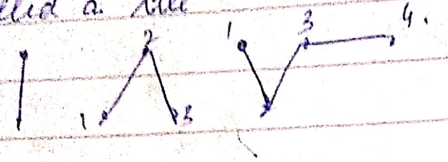
bh : 11

fd : 14



Tree.

A connected graph without any circuits is called a tree.



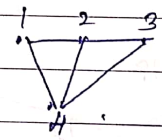
Properties of trees.

1. Undirected graph → tree iff there is a unique path btw every pair of vertices
2. A tree with n vertices has (n-1) edges
3. Any circuitless graph with 'n' vertices and (n-1) edges is a tree.

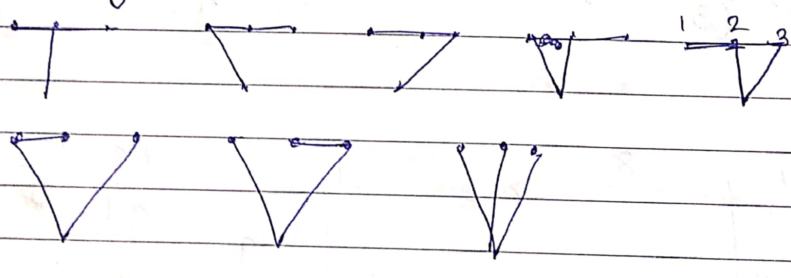
Spanning tree.

If a subgraph T of a connected graph G is a tree containing all vertices of G is called a spanning tree of G

Consider the connected graph.



Spanning trees of G.



Minimum spanning tree

If G is connected weighted graph the spanning tree of G with the smallest total weight (sum of wgts of edges) is called minimum spanning tree of G.

Kruskal's Algorithm

1. Edges arranged in increasing weights.
2. Edge with minimum weight is selected. Edges with minimum weight that do not form a circuit is added.

Defn Let G(V,E) be a graph with no multiple edges. Let $\{c_1, c_2, \dots c_n\}$ be any set of colours. A fn $f: V \to C$ is called colouring of G using n colours.

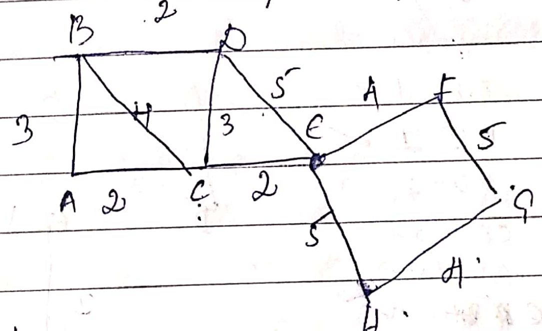
Kruskal's Algorithm:

① The edges of the given graph G are arranged in order of increasing weights.

② An edge G' with minimum weight is selected as an edge of required spanning tree.

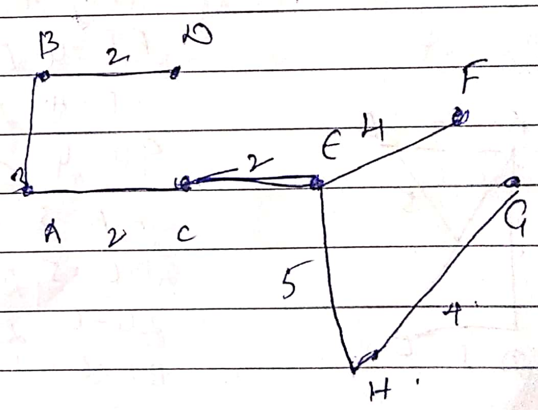
③ Edges with minimum weight that do not form a circuit are succesfully added.

i) Find minimal spanning tree using Kruskal's Algorithm



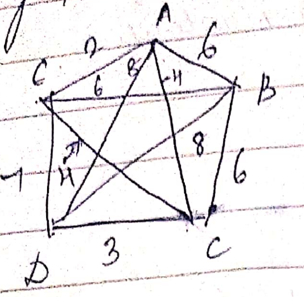
Ans:

~~AB~~ AC	2
BD	2
CE	2
AB	3
CD	3
BC	4
EF	4
HG	4
DE	5
EH	5
FG	5

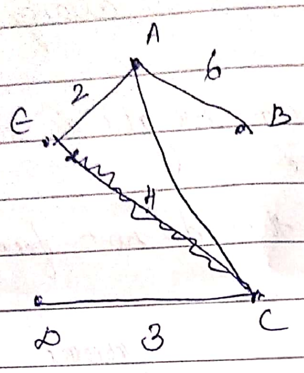


Minimum weight $= (2+3+2+2+4+5+4)$
$= 22$

2) Find the minimum spanning tree for the weighted graph using Kruskal's Algorithm.

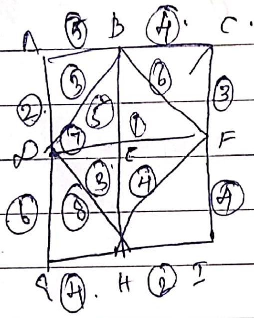


AC 2.
DC 3.
CE 4.
✗ AC 4.
AB 6.
✗ CB 6.
✗ BE 6.
✗ DE 7.
✗ AD = 8
✗ BD = 8.

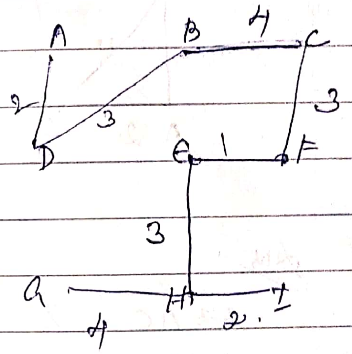


minimum weight is 15.

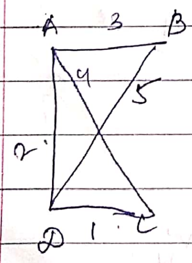
3)



EF 1 ✓
HI 2 ✓
AD = 2 ✓
CF = 3 ✓
BD = 3 ✓
EH = 3 ✓
BC = 4 ✓
FI = 4 ✗
HF = 4 ✗
GH = 4 ✓
AB = 5 ✗
BE = 5 ✗
DG = 6 ✗
BF = 6 ✗
DE = 7 ✗
DH = 8 ✗.



4)

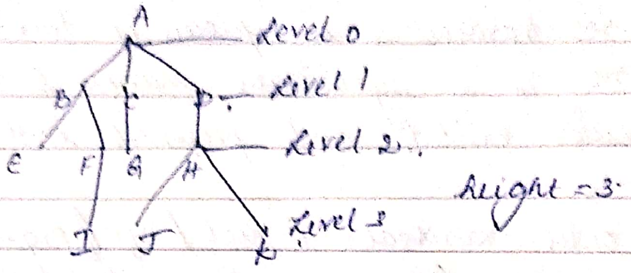


Rooted Tree, Binary Tree

① A vertex V_0 in a tree is said to be a root if ∃ unique path from V_0 to every other vertex in the tree

② The length of the path from the root of a rooted tree to any vertex V is called level of V.

③ The root is said to be at zero level.



The maximum level is called the height.

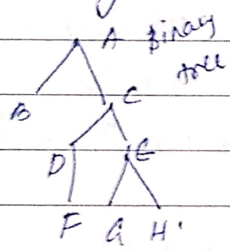
④ The vertices that are reachable from v through a single edge are called children of v.

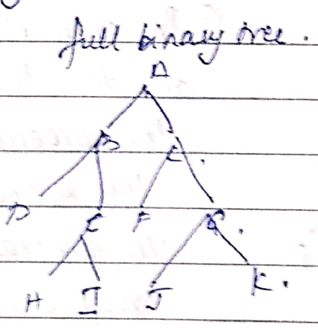
⑤ Every vertex that is reachable from a given vertex is called descendent of v.

⑥ If a vertex has no children v is called a leaf or terminal or pendent vertex.

⑦ Except the terminal vertex all others are called internal vertices

Binary tree: If every internal vertex of a rooted tree has exactly / atmost 2 children then the tree is called full binary tree / binary tree.





Properties

① The no of vertices of a full binary tree is odd and the no of pendent vertices = $\frac{n+1}{2}$

③ The minimum height of a n-vertex binary tree is equal to $\lceil \log_2 (n+1) - 1 \rceil$ where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Tree Traversal

A tree traversal is a process of traversing a tree in a systematic manner so that each vertex is ~~visite~~ visited and processed once

Pre order traversal. (Root / left / Right).
(Prefix or Polish form)

① Visit the root
② search the left subtree if exists.
③ search the right subtree.

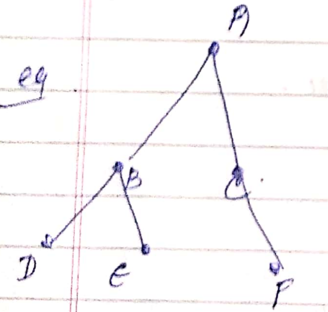
In order traversal (left / root / right).

① search ~~for~~ the left subtree if exists
② Visit the root.
③ Search the right subtree.

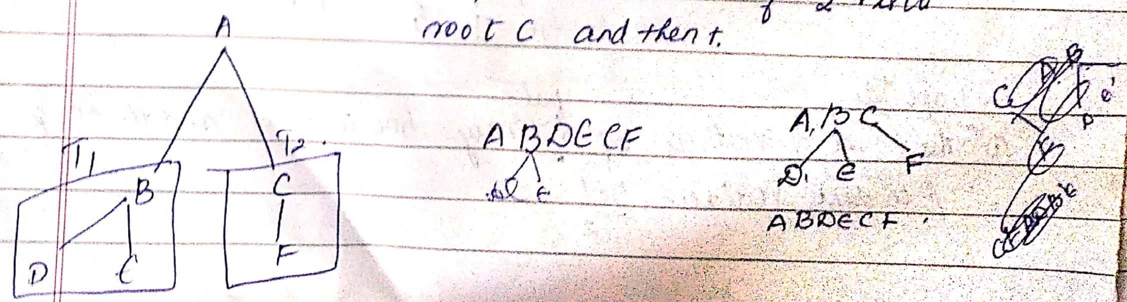
③ Post order traversal (left / right / root).

① search the left subtree if exists
② search the right subtree if exists
③ Visit the root.

eg



① Pre order : Visit root A first and then traverse T_1 & T_2.
The preorder search of T_1 visits root B then D and E.
The preorder search of T_2 visits root C and then F.

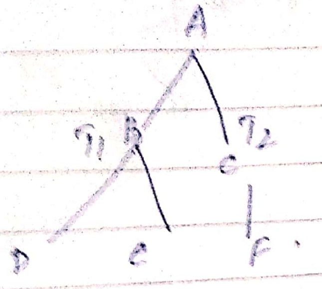


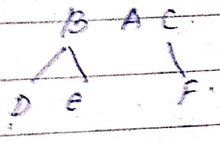
ABDECF

ABDECF.

L / Root / R.

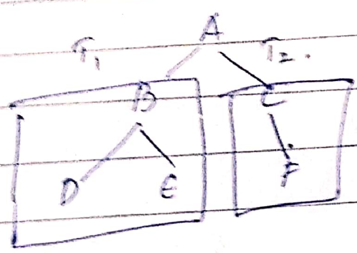
(1) Inorder: search T_1 first then visit root and finally T_2. Inorder search of T_1 visit D then root B & finally E. Inorder search of T_2 visit C and then F



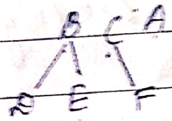


DBEACF.

(3) Post order:



left/right/root

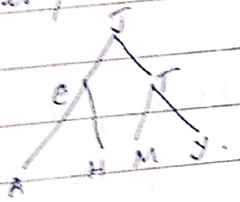


DEBFCA.

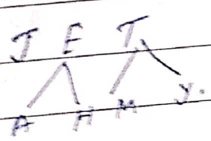
Visit T_1 then T_2 and finally root. The preorder search of T_1 visit D, E and finally the root. The preorder search of T_2 visit F and C.

Problem: List the order in which the vertices of the foll tree is processed using (i) pre order (ii) in order iii) postorder traversal.

(1)

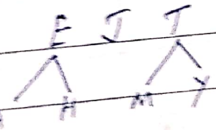


preorder (R/L/Right)



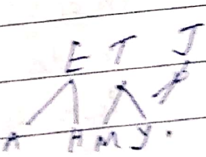
JEAHTMY.

Inorder. L/R/Right



AEHJMTY.

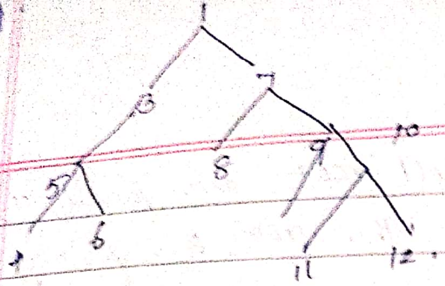
Postorder. L/Right/Root

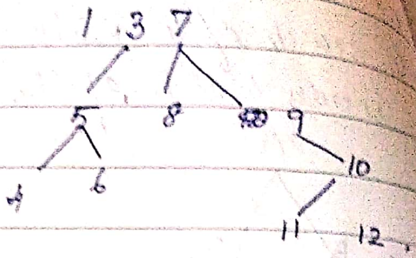


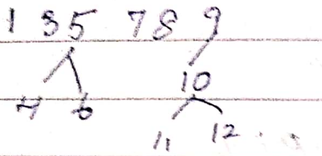
AHEMYTJ

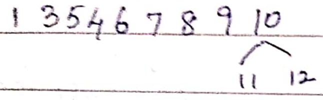
Pre order

R/L R



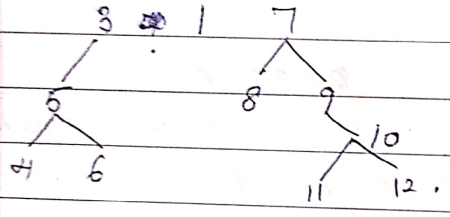


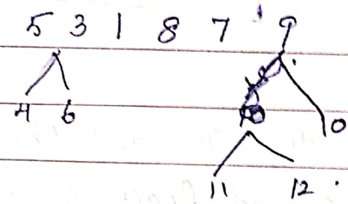


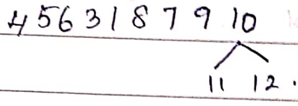


1 3 5 4 6 7 8 9 10 11 12 .

Inorder L/Root Right

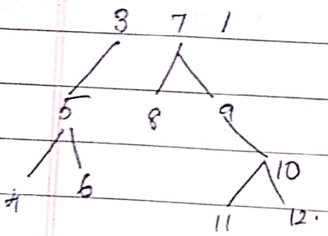


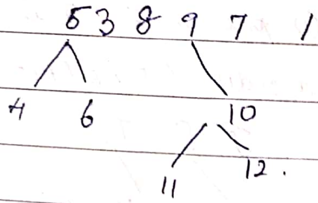


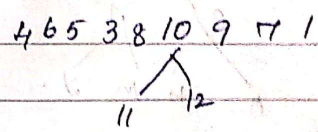


4 5 6 3 1 8 7 9 11 10 12 .

Post order L/R/Root .

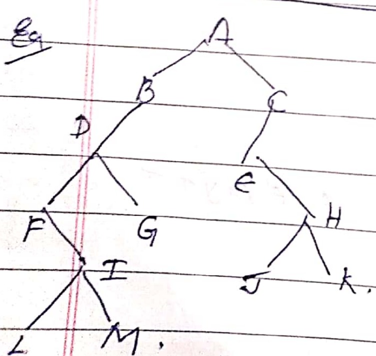






4 6 5 3 8 11 12 10 9 7 1 .

Eg



Pre order : A B D F I L M C E H J K

Inorder : F L I M D G B A E J H K C .

Postorder : L M I F G D B J K H E C A .

Expression Tree.

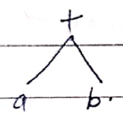
variable operators.

Binary Trees can be used to represent algebraic expression as such representation facilitate the computer evaluation of expression.

* The terminal vertices (leaves) are labelled with number or variables.
* The internal vertices are labelled with operations such as addition (+), subtraction (-), multiplication (*), division (/) and exponentiation (↑) $5 \uparrow 3$ 5^3.

Prefix, Postfix, Infix.

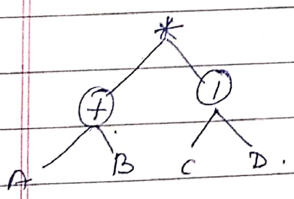
R/L/Right L/Root/Right Left, Right/Root.

eg 

prefix + ab
Infix (a+b) - usual mathematical expression
Postfix ab+

Not
Prefix { Rxy R: operator x, y no's or variable.
Infix { xRy
{ xyR. => answer will be same.

Note: To avoid ambiguity in infix notation we include a pair of parenthesis.

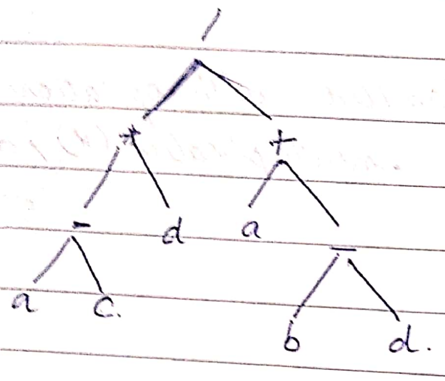


Infix. ((A + B) * (C / D)) (left root Right)

Prefix. * + AB / CD.
Postfix. AB + CD / *.

① Represent the expression $((a-c)*d) / (a+(b-d))$ as a binary tree and postfix and prefix notation.

Infix L/Root/Right.



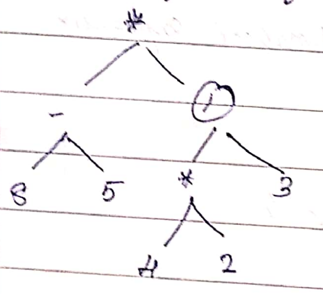
Root/L/R.

Prefix. / * - a c d + a -

Postfix. L/Right/root.

ac-d*abd-+/.

② Write prefix, infix, postfix expression and evaluate the expression.



Prefix * -85/+423

Infix ((8-5)*((4+2)/3))

postfix 85-42+3/*

(1) Evaluate $* - 8 5 / + 4 2 8$. operator.
Prefix: Rxy from right.

$* - 8 5 / 6 3$.

$* - 8 5 2$.

$* 3 2$ $\boxed{6/3 \text{ Quotient } 2}$

Ans 6.

(2) Postfix. xyR from left.

$8 5 - 4 2 + 3 / *$.

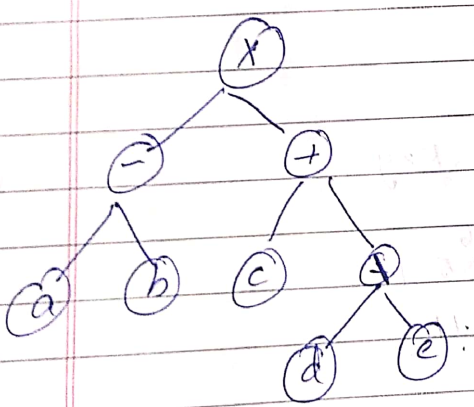
$3 4 2 + 3 / *$.

$3 6 3 / *$.

$3 2 *$.

6.

(3) Construct the tree of algebraic expression $(a-b) \times (c+(d/e))$. write prefix postfix forms.
evaluate $a=6, b=4$
$c=5, d=2, e=2$.



root/left/right
prefix : $\times - a b + c \setminus d e$
l/right/root)
post : $a b - c d e \setminus + \times$.

prefix $\times - 6 4 + 5 \setminus 2 2$

$\times - 6 4 + 5 1$

$\times - 6 4 6$

$\times 2 6 = 12$.

postfix $a b - c d e \setminus + \times$.

$6 4 - 5 2 2 \setminus + \times$.

$2 5 2 2 \setminus + \times$.

$2 5 1 + \times$

$2 6 \times$

12

Evaluate $+ - \uparrow 3 2 \uparrow 2 3 / 8 - 4 2$ in prefix.
Read from right

$+ - \uparrow 3 2 \uparrow 2 3 / 8 2$.

$+ - \uparrow 3 2 \uparrow 2 3 4$.

$+ - \uparrow 3 2 8 4$

$+ - 9 8 4$

$+ 1 4$

5.

Evaluate $(7\ 2 -) 3 + 2 3 2 + - 1 3 - * /$
xyR.

$5\ 3 + 2 3 2 + - 1 3 - * /$.

$8\ 2 3 2 + - 1 3 - * /$

$8\ 2\ 5 - 1 3 - * /$.

$8 (-3)\ 1 3 - * /$.

$8 (-3) (-2) * /$

$= 8\ 6\ /$

$= 8/6 = 4/3$.